ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES

IP PARIS

THÈSE DE DOCTORAT

de l'École nationale des ponts et chaussées

Stochastic optimization for complex systems with renewable energy

Optimisation stochastique de systèmes complexes composés d'énergies renouvelables.

École doctorale N°531, Mathématiques, Sciences et Technologies de l'Information et de la Communication

Mathématiques appliquées

Thèse préparée à METRON et au laboratoire CERMICS, École des Ponts

Thèse soutenue le 30 janvier 2025, par

Zoé FORNIER

Composition du jury:

Bernardo PAGNONCELLI Pr, Skema Business School	*Rapporteur*
Alexander STREET Pr, Pontifical Catholic University of Rio de Janeiro	*Rapporteur*
Céline GICQUEL Pr, Université Paris Saclay	*Examinatrice*
Merve BODUR Pr, University of Edinburgh	*Examinatrice*
Bernardo Freitas Paulo DA COSTA Pr, Fundação Getulio Vargas	*Co-encadrant de thèse*
Pierre PINSON Pr, Imperial College of London	*Examinateur*
Frédéric MEUNIER Pr, École des Ponts	*Président du jury*
Vincent LECLÈRE Pr, École des Ponts	*Directeur de thèse*

Résumé

Dans cette thèse, nous nous intéressons à l'intégration d'aspects énergétiques dans les problèmes de planification de la production. D'une part, nous traitons la modélisation de systèmes complexes sous incertitude et développons des méthodes de résolution adaptées. D'autre part, nous explorons des approches pour traduire le concept d'équité mathématiquement.

Dans la première partie, nous proposons un modèle qui vise à optimiser conjointement la planification de la production d'une usine et la gestion de son approvisionnement en énergie, en présence de sources d'énergie renouvelables et de batteries. Le problème est formulé comme un Problème Stochastic Multiétapes Linéaire en variables continues et Binaires (PSMLB). Sa résolution est numériquement difficile en raison de sa taille, de la présence incertitudes et des contraintes d'intégrité. Nous explorons diverses stratégies de résolution (MPC, SDP) puis développons une heuristique qui utilise l'algorithme SDDP.

En deuxième partie, nous élaborons une nouvelle méthodologie pour résoudre des PSMLB. Cette méthodologie s'inscrit dans un cadre abstrait, qui repose sur la structure des arbres de scénarios, et dont le but est de coordonner SDDP et le Branch-and-Bound (BB). Cela conduit à un algorithme efficace et exact pour résoudre les PSMLB. Plus précisément, l'algorithme résout itérativement des relaxations partielles du problème qui reposent sur la structure de sous-arbres, où l'intégrité est maintenue, les portions élaguées étant approximées à l'aide de SDDP. Cela permet de trouver un compromis entre qualité de la solution et temps de calcul. Enfin, nous proposons des heuristiques inspirées des techniques BB pour faire grandir le sous-arbre.

Enfin, la troisième partie se concentre sur l'équité dans un contexte d'agrégation des prosumers (ou grands consommateurs) sur les marchés de l'électricité. Nous examinons différentes manières de trouver une allocation des bénéfices équitables et qui garantissent que chaque participant bénéficie de l'agrégation. Nous concevons des modèles stochastiques multi-périodes intégrant l'équité par construction grâce à des contraintes d'acceptabilité et des opérateurs d'équité. Notre approche met l'accent sur une prise de décision équitable dans des environnements dynamiques et incertains.

Abstract

In this thesis, we investigate the impact of integrating energy considerations into production planning problems. On one side, we focus on modeling complex systems under uncertainty. On the other side, we explore approaches to translate the concept of fairness in mathematical models.

The first part focuses on jointly optimizing production planning and energy supply management in an industrial setting with renewable energy and storage systems. Formulated as a multistage stochastic problem with continuous and binary variables, it presents challenges due to its size, uncertainties, and integer constraints. We explore various solution strategies comparing their performance across different setups.

The second part introduces a new methodology to solve Multistage Stochastic mixed-binary Linear Program (MSbLP), with stagewise independent noises. First, we develop an abstract framework, relying on the scenario tree structure, that aims to coordinate SDDP– that solves the continuous relaxation efficiently– and Branch-and-Bound (BB) procedures– that deal with binary variables. This leads to an exact algorithm to solve MSbLP, relying on partial relaxations of the problem. The algorithm consists of iteratively growing a small subtree on which the integrality constraints are maintained. The rest of the problem, continuously relaxed, is represented through value functions obtained via SDDP. This allows to balance between solution accuracy and computational feasibility. We discuss heuristics inspired by BB techniques to grow this subtree and evaluate their effectiveness in various industrial-inspired problem settings.

The third part shifts focus to fairness in aggregating prosumers in electricity markets. We investigate fair resource allocation strategies that ensure each participant benefits from aggregation. Using acceptability constraints and fairness operators, we design multi-period stochastic models that incorporate fairness by construction. Unlike traditional financial compensation mechanisms, our approach emphasizes fair decision-making in dynamic, uncertain environments, tested on both deterministic and stochastic models.

Remerciements

Arrive enfin le moment de remercier toutes celles et ceux qui ont contribué à rendre ces trois dernières années inoubliables. La tâche est ardue : traduire en mots l'immense gratitude que j'éprouve. Voici donc une tentative empreinte d'une utilisation abusive du mot **merci**, pardonnez ce manque d'inspiration.

Tout d'abord, je remercie l'ensemble du jury pour leur présence et examen de ma soutenance, et en particulier je remercie Bernardo Pagnoncelli et Alexandre Street d'avoir rapporté cette thèse: vos retours ont été précieux. J'aimerais aussi remercier l'ensemble des chercheurs, dont les conseils, remarques et idées, ont été essentiels pour la rédaction de ce manuscrit. En première ligne, Vincent bien sûr, merci pour ton soutien indéfectible, ta patience et surtout ta bienveillance. Tu mériterais une page entière, mais pour reprendre tes mots,"for the sake of conciseness", merci d'avoir fait de cette thèse une aventure enrichissante et particulièrement heureuse pour moi. Merci à Bernardo, qui, sans en posséder le titre officiel, fut un co-encadrant de thèse enthousiaste et empathique. Merci d'avoir si efficacement partagé ta passion pour les mathématiques. Merci également à Merve pour ton accompagnement cette dernière année et de ne jamais être à court d'idées. Enfin, merci à Pierre de m'avoir accueillie à Londres et pour nos échanges toujours stimulants.

Je tiens aussi à remercier l'équipe de METRON: merci de m'avoir encadrée et permis de découvrir le monde de l'entreprise tout en me laissant une grande liberté dans mes recherches. Merci aux Metroners, qui ont su transformer notre lieu de travail en un espace chaleureux. Merci aux sportifs du midi et à la joyeuse équipe pour les souvenirs éclatants que je garde des afterworks et des karaokes multiples où j'ai perdu la voix.

Merci également à Isabelle, Stéphanie, et l'ensemble du personnel de l'École des Ponts pour leur accompagnement au fil des démarches administratives. Merci à Frédéric et Jean Philippe de m'avoir fait confiance en me permettant d'enseigner. Enfin, merci Olivier pour tes cours passionnants qui m'ont donné le goût de la recherche opérationnelle, je serais sans eux sûrement perdue dans les méandres de l'IA.

Si on m'avait dit un jour que le nom de "Champs-sur-Marne" évoquerait une quelconque émotion en moi, j'aurais bien ri. C'est pourtant le cœur serré que je songe au départ, et il m'est impossible de rendre justice à tous mes camarades du CERMICS qui ont participé à rendre la vie si douce. Merci Emanuele pour ton énergie débordante. Merci Hélène d'être bien plus divertissante que Grey's Anatomy. Merci Anton pour ton autodérision hilarante. Merci Solal de toujours ralentir pour moi. Merci Alfred d'être un shot de positivité. Merci Raian de me faire douter des légumes. Merci Seta d'être mon bureau des plaintes. Merci Paul de rager avec autant d'humour. Merci Kacem d'être toujours joyeux. Merci Albéric de m'avoir fait découvrir The Do. Merci Maël de m'avoir presque donné envie de faire de la géométrie. Merci Camila, Roberta et Rutger de réveiller la pâtissière en moi. Plus globalement, merci à tous ceux du 2ème, nouveaux et anciens,

pour votre manque de sérieux occasionnel. Je n'oublie pas ceux des contrées obscures du 3ème, restez comme vous êtes: incompréhensibles mais fort sympathiques. Il me faut enfin évoquer la team piscine : merci d'avoir joué un rôle essentiel dans le maintien de ma santé mentale tout au long de la thèse.

Ces trois années furent aussi l'occasion de profiter pleinement de la vie parisienne. Pour cela, je dois remercier tous mes proches, avec qui les soirées au bar, au théâtre, au cinéma, les picnics quand il fait beau, les musées quand il fait moche, les vacances un peu partout, ont rempli mes soirées et week-ends de joie et de rires. Alors de manière un peu succincte, car il est temps de conclure, merci: à mes amies de longue date, Julie, Yasmine et Laura; à mes camarades de Pasteur, de Saint Dominique et de Télécom qui testent les eaux troubles du monde du travail avant moi; à Anton et Maël d'avoir égayé mon isolement pendant l'année covid-master; à Camille d'être une source sans fin de bons plans; à Justine et Guillaume pour une coloc inoubliable; à Ludo pour ton soutien, en particulier pendant la rédaction.

Enfin, merci à ma famille, dont l'appui a toujours été d'une valeur inestimable, et en particulier, merci à mes parents à qui je dois tout.

Contents

Part I

Introduction

Chapter 1

Introduction (en français)

Contents

1.1 Conscience énergétique dans l'industrie

Cette thèse est réalisée en collaboration avec METRON, une CleanTech française spécialisée dans la performance énergétique. METRON accompagne les groupes industriels et tertiaires dans l'optimisation de leur consommation énergétique et la réduction de leurs émissions de carbone. L'objectif de ce travail est d'améliorer notre compréhension des impacts des micro-réseaux dans un contexte de production industrielle, et ainsi accompagner la transition énergétique de l'industrie.

1.1.1 METRON: une entreprise CleanTech française

METRON[1], fondée en 2013, offre une solution digitale centrale pour la visualisation, le suivi, l'optimisation et la modélisation IA de la stratégie de performance énergétique.

[1] https://www.metron.energy/

Les principales fonctionnalités de la plateforme, illustrées dans la Figure 1.1[2], incluent la gestion de l'acquisition sécurisée des données, ainsi que le suivi de la performance énergétique pour visualiser, mesurer et comparer la consommation énergétique. La plateforme propose également des analyses avancées pour optimiser l'utilisation de l'énergie et détecter les facteurs d'inefficacité. Enfin, elle fournit des outils de gestion en temps réel des coûts énergétiques et de l'impact carbone, aidant ainsi les organisations à prévoir leurs budgets et à gérer leurs stratégies de durabilité.

Figure 1.1: Fonctionnalités de la plateforme METRON

Cette thèse s'inscrit dans les projets d'optimisation énergétique de METRON, et examine en particulier l'impact des investissements dans les micro-réseaux sur les groupes industriels. Les aspects clés étudiés incluent les défis opérationnels de la gestion d'un micro-réseau, son influence sur les pratiques de production existantes et les implications vis à vis de l'accès aux marchés de l'énergie.

1.1.2 Investir dans les micro-réseaux : quels enjeux?

Le réchauffement climatique affecte considérablement la résilience des systèmes énergétiques et nous incite à repenser notre façon de consommer l'énergie. Pour atteindre l'objectif de limiter la hausse des températures en dessous de 2°C fixé par l'Accord de Paris de 2016 [COP16], il est nécessaire de se tourner vers des sources d'énergie plus durables. Malgré des progrès notables, avec les énergies renouvelables atteignant $14,6\%$ de la consommation énergétique primaire mondiale et représentant près de 30% de la production mondiale d'électricité en 2023 (voir [Ins24]), ces efforts sont loin d'être suffisants pour atteindre les objectifs climatiques fixés.

Un afflux d'investissements est orienté vers les technologies durables, telles que les énergies renouvelables et le stockage d'énergie. Selon le rapport annuel de l'AIE [IEA24], les investissements dans l'énergie verte devraient représenter les deux tiers des investissements énergétiques totaux en 2024. La technologie photovoltaïque (PV) est en tête de ces investissements, surpassant toutes les autres technologies de production d'électricité. Cependant, l'intermittence des énergies

[2] https://www.metron.energy/solution/

renouvelables nécessite leur intégration à des systèmes de stockage d'énergie et l'amélioration des infrastructures de réseau électrique.

Le secteur industriel en particulier doit réduire ses émissions de carbone et atténuer les risques de dépendance vis-à-vis de systèmes énergétiques centralisés. Pour cela, les industriels peuvent investir dans la génération d'énergie renouvelable couplée à des systèmes de stockage d'énergie, tels que les micro-réseaux. Le ministère américain de l'Énergie définit un micro-réseau comme "un groupe de charges interconnectées et de ressources énergétiques distribuées dans des limites électriques clairement définies qui agit comme une entité contrôlable unique par rapport au réseau." Les micro-réseaux apportent de la flexibilité, notamment en cas de pénurie d'énergie, car ils peuvent fonctionner indépendamment du réseau principal. Cependant, leur utilisation entraîne des complexités opérationnelles. Parallèlement, il est crucial d'améliorer l'efficacité énergétique grâce à des technologies avancées ou des pratiques de production modernisées.

Le défi majeur est de trouver un équilibre entre les objectifs environnementaux et la viabilité économique. En effet, des technologies comme les micro-réseaux, bien que prometteuses, peuvent s'avérer extrêmement coûteuses sans soutien gouvernemental. Par exemple, une évaluation économique des micro-réseaux basés sur les énergies renouvelables [Wan+20] met en évidence le besoin de politiques plus efficaces pour encourager les investissements. Simultanément, la pression pour réduire les émissions carbone augmentent, avec des mesures telles que les taxes carbone. Le mécanisme d'ajustement carbone aux frontières de l'Union européenne CBAM vise, par exemple, à taxer les produits importés en fonction de leur empreinte carbone.

Les entreprises industrielles bénéficieraient d'une expertise pour comprendre les défis liés à la gestion de l'énergie. Toutefois, malgré le besoin d'efficacité énergétique, de nombreux groupes industriels ne sont pas encore entièrement numérisés et restent peu informés sur les outils d'optimisation et de science des données disponibles. Cette thèse vise à améliorer notre compréhension de l'intégration des micro-réseaux dans les environnements industriels et à explorer des stratégies pour optimiser leur utilisation.

Si le secteur industriel doit adapter sa consommation énergétique pour atteindre les objectifs environnementaux, les marchés d'énergie doivent aussi s'adapter afin d'intégrer une production énergétique décentralisée.

1.1.3 Vers un modèle de marché de l'énergie centré sur le consommateur

Avant toute chose, nous proposons un aperçu de l'industrie de l'électricité, basé sur l'ouvrage [KS04]. Un système électrique connecte des moyens de production et de consommation d'électricité qui impliquent divers acteurs: les gestionnaires de réseaux (qui gèrent les infrastuctures de transport et de distribution), les producteurs d'énergie, les fournisseurs d'énergie (qui achètent de l'énergie sur le marché de gros), les consommateurs (qui doivent acheter via des fournisseurs ou peuvent accéder au marché de gros selon leur taille), et les régulateurs de marché.

Pendant plus d'un siècle, de la fin du XIXe siècle jusqu'aux années 1980, le marché de l'électricité fonctionnait verticalement et était fortement réglementé : un même acteur contrôlait à la fois la production, le transport et la distribution d'énergie aux consommateurs finaux. En conséquence, les consommateurs n'avaient pas d'autre choix que d'acheter leur électricité auprès du fournisseur local, en situation de monopole. À partir des années 1980, la libéralisation du marché de la production de l'électricité conduit à une nouvelle organisation de marché de l'énergie.

Actuellement, les marchés de l'énergie se divisent en trois grandes catégories : le marché de capacité, qui garantit une capacité de production suffisante ; le marché de l'énergie, qui optimise la planification des échanges ; et le marché des services auxiliaires, qui soutient la stabilité du

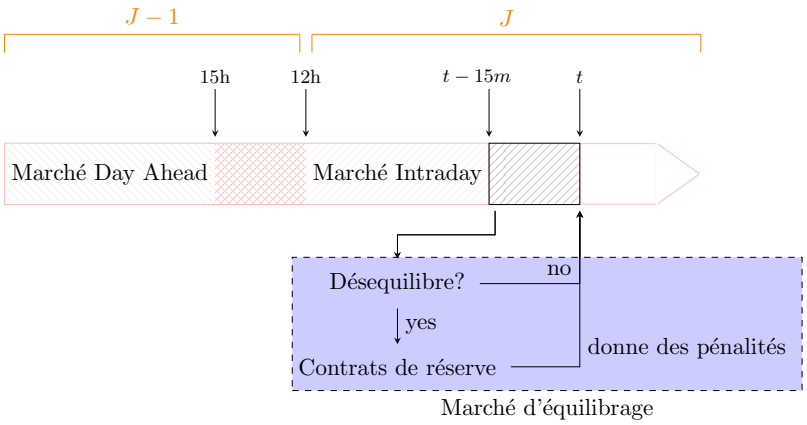


Figure 1.2: Illustration de la chronologie des marchés de l'énergie, à l'exclusion du marché à terme où les contrats sont établis des mois ou des années à l'avance.

système électrique. En particulier, le marché de l'énergie interagit avec les acteurs énergétiques selon différents horizons temporels (voir Figure 1.2). Le *marché à terme* comprend des contrats à long terme pour la couverture des prix et la gestion des risques; le *marché* spot *(ou* day-ahead*)* concerne les échanges d'énergie un jour à l'avance; le *marché intraday* permet des ajustements jusqu'à une heure avant la production; et le *marché d'équilibrage* assure la stabilité du système en temps réel. Plus précisément, les contrats de réserve sont activés sur le marché d'équilibrage pour maintenir la stabilité du système et les participants qui perturbent l'équilibre du réseau sont pénalisés. Dans cette thèse, nous nous concentrerons sur les marchés day-ahead et d'équilibrage.

Malgré la libéralisation des marchés d'énergie, ceux-ci restent en partie centralisés, ce qui empêche les petits consommateurs de négocier directement le prix de l'énergie. Avec l'émergence des *prosumers* (des acteurs à la fois consommateurs et producteurs d'énergie), certains marchés évoluent progressivement vers un modèle plus décentralisé, où les *prosumers* jouent un rôle plus actif. Un marché de l'énergie décentralisé, intégrant des *prosumers* intelligents, doit être conçu pour gérer la complexité des différents services et acteurs qui peuvent changer de rôle. Plusieurs conceptions de marché répondant à ces défis sont discutées dans l'article [PS16].

En particulier, certains *prosumers* peuvent souhaiter former une communauté et collaborer, un processus pouvant être facilité par des entreprises de petite ou moyenne taille jouant le rôle d'agrégateurs. Dans ce cas, les agrégateurs ont la responsabilité d'opérer le réseau efficacement, d'optimiser les flux énergétiques, et de maintenir l'équité dans la répartition des bénéfices parmi les participants. Intégrer la notion d'équité dans les modèles mathématiques couramment utilisés dans ces domaines pose des défis significatifs. La subjectivité inhérente à l'équité nécessite une approche élaborée et réfléchie. Cette problématique est examinée en profondeur dans la Partie III de cette thèse.

1.1.4 Vers de nouvelles pratiques avec la RO

Pour comprendre et trouver des solutions aux problèmes que nous nous posons, nous utilisons des outils de Recherche Opérationnelle (RO). Pour définir la RO, nous nous appuyons sur la présentation qui en est faite par la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF) [ROA] : "La Recherche Opérationnelle (RO) est la discipline des méthodes scientifiques utilisables pour élaborer de méilleures décisions. Elle permet de rationaliser, de simuler et d'optimiser l'architecture et le fonctionnement des systèmes de production ou d'organisation." Cette définition, volontairement large, comprend un grand nombre de méthodes

et d'applications. La RO vise à sélectionner et mettre en oeuvre des solutions efficaces au sein de systèmes complexes, et est présente dans tous les grands secteurs industriels tels que les transports, l'énergie, la production et les télécommunications.

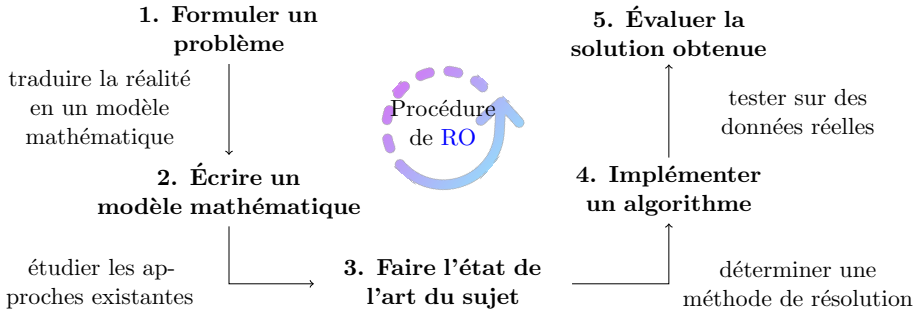


Figure 1.3: La méthodologie en RO.

La RO comprend une grande variété d'outils, allant des théories abstraites (comme la théorie des graphes, la théorie de la complexité ou la théorie des polyhèdres) aux algorithmes pratiques (par exemple l'algorithme du simplexe pour la programmation linéaire, l'algorithme de Dijkstra pour le plus court chemin ou la méthodologie de branch-and-bound pour la programmation linéaire en nombres entiers). Une simplification de la méthodologie de la RO en un processus structuré en cinq étapes est illustrée dans la Figure 1.3.

Tout d'abord, un problème du monde réel est formulé et traduit en un modèle mathématique. Après une étude approfondie de la littérature académique, le problème est classé et les approches de solution existantes sont identifiées. Une méthode appropriée est ensuite sélectionnée ou adaptée pour trouver une solution au problème initial. Celle-ci est testée et analysée pour évaluer son efficacité. Si la solution n'est pas satisfaisante, le processus est repris, permettant des ajustements dans les étapes de modélisation ou d'implémentation pour affiner les hypothèses ou améliorer les performances jusqu'à l'obtention d'un résultat satisfaisant.

Le principal défi de la RO est de trouver le juste équilibre entre une solution approximative, basée sur des modèles simplifiés, et une solution de haute qualité réalisable en pratique et satisfaisante pour les décideurs. Un exemple notable d'application de la RO est le problème du routage de véhicules, utilisé par des entreprises comme Renault ou la SNCF, qui vise à optimiser les itinéraires d'une flotte de véhicules pour desservir un ensemble de clients. Un autre grand succès de la RO est le problème de la coordination d'un système de barrages hydroliques: le système électrique brésilien est opéré quotidiennement à l'aide d'algorithmes d'optimisation stochastique.

1.2 Modèles Mathématiques

Nous présentons ici quelques modèles de RO qui permettent d'analyser l'impact de l'investissement dans un *micro-réseau* électrique sur les problèmes de production industrielle. Nous commençons par les problèmes de production, où l'utilisation de variables binaires est nécessaire pour représenter diverses contraintes techniques. Ensuite, nous abordons les problèmes d'approvisionnement en énergie, qui sont intrinsèquement stochastiques *i.e.,* aléatoires, en raison des énergies renouvelables. L'intégration de ces deux problématiques donne naissance à un modèle complexe, combinant variables entières et incertitudes. Enfin, nous examinons les défis de modélisation associés à l'agrégation de plusieurs entités, dans le but de générer des bénéfices collectifs.

1.2.1 Problèmes de production : le rôle majeur des variables binaires

L'optimisation de la gestion des stocks et de la planification des opérations est un défi majeur dans un environnement de production. Cette section présente un modèle générique de planification, illustrant certaines des contraintes spécifiques qu'on peut rencontrer.

Soit une usine disposant de I machines et de J produits, stockés dans des entrepôts à capacité limitée. L'objectif est d'établir un plan de production détaillé pour cette usine sur un horizon donné (par exemple une journée, une semaine, etc.). Pour cela, nous divisons l'horizon en intervalles de temps discrets, appelés *étapes*, indexés par $t \in [T]$ où T représente le nombre total d'étapes. Le plan de production indique pour chaque étape les quantités de produits à fabriquer et à stocker afin de satisfaire une demande (les commandes entrantes). L'objectif est de minimiser les coûts de l'usine, qui incluent les coûts de production et de stockage, ce qui mène au modèle :

$$\min_{u,b,s} \quad \sum_{t=1}^{T} f_t^{ij}(u_t^{ij}, b_t^{ij}, s_t^j) \tag{1.1a}$$

$$\text{s.t.} \quad s_t^j = s_{t-1}^j + \sum_i u_t^{ij} - d_t^j \qquad \forall t, j \tag{1.1b}$$

$$\underline{s}^j \le s_t^j \le \overline{s}^j \qquad \forall t, j \tag{1.1c}$$

$$\underline{u}^{ij}\, b_t^{ij} \le u_t^{ij} \le \overline{u}^{ij}\, b_t^{ij} \qquad \forall t, i, j \tag{1.1d}$$

$$\{u_t^{ij}\}_{i,j} \in \mathfrak{U}_t \qquad \forall t \tag{1.1e}$$

$$\{b_t^{ij}\}_{i,j} \in \mathfrak{B}_t \cap \{0,1\}^{I \times J} \qquad \forall t, \tag{1.1f}$$

où u_t^{ij} représente la quantité de produit j produite sur la machine i à l'étape t, b_t^{ij} est une variable binaire qui vaut 1 si $u_t^{ij} > 0$ et 0 autrement, et s_t^j modélise la quantité de produit j qui est stockée dans l'entrepôt à la fin de l'étape t. L'objectif (1.1a) consiste à minimiser la somme des coûts sur l'horizon, où f_t^{ij} est une fonction qui modélise les coûts associés aux variables u_t^{ij}, b_t^{ij}, et s_t^j. Pour des raisons de faisabilité, f est généralement une approximation linéaire ou convexe de la fonction de coûts réelle. Les contraintes (1.1b) décrive des dynamiques classiques de stock: le niveau de stock à l'étape précédente détermine le niveau de stock à l'étape t, ajusté en fonction de la production et de la satisfaction de la demande. De plus, le stockage est limité (1.1c). La production de j sur une machine est discontinue: en effet, si j est produit sur la machine i, la quantité produite doit être comprise entre $\underline{u}^{ij}$ et $\overline{u}^{ij}$, sinon elle vaut 0. Les variables binaires sont essentielles pour modéliser cette discontinuité (1.1d). En effet, si $b_t^{ij} = 0$, ce qui veut dire qu'on ne produit pas, alors $0 \le u_t^{ij} \le 0$; sinon, $\underline{u}_t^{ij} \le u_t^{ij} \le \overline{u}_t^{ij}$. Enfin, on réunit l'ensemble des contraintes imposées par les procédures de production dans les ensembles d'acceptabilité $\mathfrak{U}_t$ (1.1e) et $\mathfrak{B}_t$ (1.1f).

Dans les problèmes de production, les variables binaires sont un outil efficace pour modéliser les contraintes physiques liées au fonctionnement des machines (voir les contraintes (1.1d)). En général, une machine i ne peut produire qu'un seul produit à la fois, ce qui se modélise par:

$$b_t^{ij} = 1 \implies b_t^{ij'} = 0 \quad \forall j' \neq j.$$

On peut ré-écrire cette condition comme contrainte qui sera inclue dans $\mathfrak{B}_t$,

$$\sum_j b_t^{ij} \le 1. \tag{1.2}$$

Un autre exemple, nommé *contraintes de ressources partagées* ou *d'incompatibilité*, représente l'interdiction de produire les produits j et j' simultanément (à la même étape). Ces contraintes

découlent souvent de préférences opérationnelles ou de limitations du personnel. Pour modéliser cette incompatibilité de j et j', on utilise l'inégalité:

$$\max_i b_t^{ij} + \max_i b_t^{ij'} \leq 1. \tag{1.3}$$

Ici, $\max_i b_t^{ij}$ vaut 0 si j n'est pas produit à l'étape t et 1 sinon. Alors, la contrainte garantit qu'un seul des produits j ou j' est produit à l'étape t. Dans le cas contraire, cela entraînerait l'inégalité invalide $2 \leq 1$.

La variable s_t^j est également appelée *variable d'état* car elle décrit, en partie, l'état de l'usine à l'étape t, en fonction des décisions prises précédemment. Dans le problème décrit, l'état de l'usine est entièrement représenté par les niveaux de stock de chaque produit. Cependant, si le problème devient plus complexes, il pourrait être nécessaire d'introduire des variables supplémentaires pour décrire le système de l'usine à un instant donné. Supposons, par exemple, que certaines machines ne doivent pas être allumées plus de L^i fois sur l'ensemble de la période considérée, de $t = 1$ à T. Ces contraintes, appelées *contraintes de compteurs*, nécessite les nouvelles variables d'état c_t^i qui comptent le nombre de fois que la machine i a été allumée à t. Alors, on les modélise avec les équations suivantes.

$$c_t^i = c_{t-1}^i + \mathbb{1}_{\sum_j b_t^{ij} - \sum_j b_{t-1}^{ij} = 1} \qquad \forall t, i \tag{1.4a}$$

$$0 \leq c_t^i \leq L^i \qquad \forall t, i. \tag{1.4b}$$

Ici, $\sum_j b_t^{ij}$ vaut 1 si la machine i est allumée à l'étape t (rappelons que la somme est inférieure à 1, voir (1.2)), et 0 sinon. Alors, le compteur c_t^i est égal au compteur précédent c_{t-1}^i plus 1 si la machine est allumé à t, ce qui amè,e aux dynamiques (1.4a). Enfin, on s'assure que la limite d'allumage L^i n'est pas dépassée avec les contraintes (1.4b).

Afin d'atténuer l'usure des machines, il peut être exigé qu'une machine i reste en fonctionnement (*resp.* à l'arrêt) pendant un certain nombre d'étapes consécutives. Pour ces *contraintes de temps de fonctionnement/arrêt minimum*, des variables binaires supplémentaires sont nécessaires: up_t^i (*resp.* down_t^i) vaut 1 si la machine i est mise en marche (*resp.* arrêtée) à l'étape t, et 0 sinon. Enfin, ces nouvelles contraintes sont modélisées par

$$\text{up}_t^i - \text{down}_t^i = \sum_j b_t^{ij} - \sum_j b_{t-1}^{ij} \qquad \forall t, i \tag{1.5a}$$

$$\text{up}_t^i + \text{down}_t^i \leq 1 \qquad \forall t, i \tag{1.5b}$$

$$\sum_{\tau=t-M^i}^{t} \text{up}_t^i \leq \sum_j b_t^{ij} \qquad \forall t > M^i, i \tag{1.5c}$$

$$\sum_{\tau=t-m^i}^{t} \text{down}_t^i \leq 1 - \sum_j b_t^{ij} \qquad \forall t > m^i, i, \tag{1.5d}$$

où les équations (1.5a) et (1.5b) guarantissent que up_t^i et down_t^i aient les bonnes valeurs. Plus précisément, si la machine i est allumée à l'étape t, alors $\text{up}_t^i - \text{down}_t^i = 1$, et comme down_t^i est binaire, il suit que $\text{up}_t^i = 1$. De la même manière, si la machine i est éteinte à l'étape t, on doit avoir $\text{down}_t^i = 1$. Quand la machine i reste soit allumée, soit éteinte à l'étape t, $\text{up}_t^i - \text{down}_t^i = 0$, et la contrainte (1.5b) guarantit que up_t^i et down_t^i valent 0. Enfin, pour décrire l'état de l'usine à l'étape t, il nous faut plus d'information sur les M^i (*resp.* m^i) dernieres étapes: $\{\text{up}_\tau\}_{\tau \in [t-M^i, t]}$ et $\{\text{down}_\tau\}_{\tau \in [t-m^i, t]}$. Dans cet exemple, le rôle crucial des variables binaires pour modéliser correctement le fonctionnement de l'usine est mis en évidence.

Si le problème est linéaire, autrement dit toutes les contraintes et l'objectif du problème sont des fonctions linéaires des variables, il fait partie de la classe des Programme Linéaire en variables Mixtes (PLM). Tant qu'ils sont de taille raisonnable, les PLM peuvent être résolus efficacement avec des solvers comme Gurobi ou HiGHS. Dans le cas inverse, cela peut être dû aux variables binaires qui complexifient trop le problème, auquel cas on considère naturellement la relaxation continue du problème. Toutefois, réparer une solution relâchée d'un PLM peut s'avérer aussi compliqué que de trouver une solution au problème initial.

La production d'une usine génère une demande en énergie qui est en général satisfaite en achetant de l'énergie auprès d'un fournisseur. Cependant, dans le cas où l'usine investit dans un *micro-réseau* c'est-à-dire un système de génération et de stockage d'énergie, la question qui se pose est celle de l'exploitation optimal dumicro-réseau. Dans l'intention de répondre à cette problématique, nous présentons dans la section suivante un modèle qui optimise la gestion d'un micro-réseau. Dans un premier temps, on réduit la demande énergétique du micro-réseau à un paramètre fixé.

1.2.2 Problème de gestion d'un micro-réseau: optimisation sous incertitudes

Dans un contexte énergétique, les problèmes que l'on rencontre sont soumis à de nombreuses incertitudes, en particulier si on considère la génération d'énergies renouvelables qui sont intrinsèquement imprévisibles. Si on peut prédire en partie le comportement des renouvelables, par exemple l'énergie solaire qui suit des cycles jour-nuit, il reste une partie d'aléas. De la même manière, il est difficile de prédire les prix de l'énergie, volatiles et multi-factoriels, ou la demande en énergie. Ces incertitudes affectent à la fois les contraintes et l'objectif des modèles liés à l'énergie, et ont ainsi un impact sur les solutions obtenues en RO. Pour obtenir des solutions adaptées, nous nous tournons vers des théories d'optimisation sous incertitudes.

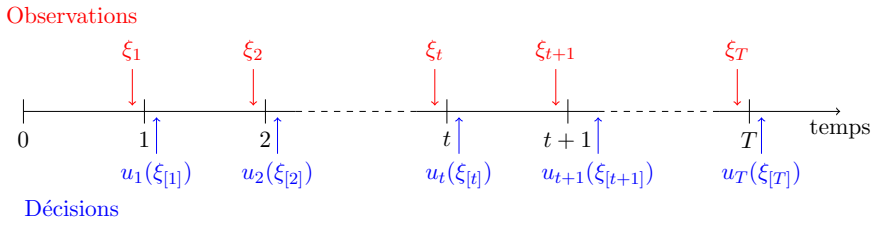


Figure 1.4: Coordination entre les observations et la prise de décision au cours du temps: à chaque étape t, on observe l'aléa ξ_t, pui on prend la décision $u_t(\xi_{[t]})$, fonction de l'information observée jusque là $\xi_{[t]} = \{\xi_1, \dots, \xi_t\}$.

On considère un problème multi-étapes, où des décisions (continues ou binaires) sont prises séquentiellement à chaque étape. Dans cette thèse, nous considérons une structure d'information dîte *Hasard-Décision* où les incertitudes sont observées avant la prise de décision à chaque étape, voir Figure 1.4. La structure inverse, *Décision-Hasard*, existe, ainsi que d'autres structures plus complexes comme *Hasard-Décision-Hasard*. On trouve dans la litérature plusieurs manières d'incorporer les incertitudes à la modélisation du problème qui dépendent principalement de ce que l'on sait de ces incertitudes. Par exemple, dans le cas où on connaît un *ensemble d'incertitudes* qui les contient, on peut avoir recours à l'*optimisation robuste* qui consiste à optimiser le problème dans le *pire cas*. Plus précisément, il s'agit de sélectionner une solution en supposant que l'aléa que l'on observera est le pire scénario pour cette décision. Nous nous plaçons dans le paragdime de l'*optimisation stochastique*, qui modélise les incertitudes comme des variables aléatoires dont on connaît la distribution.

S'il peut paraître optimiste de supposer qu'on connaît la distribution des paramètres énergétiques, les avances récentes en collecte et analyse de la donnée justifie cette approche. Par exemple, la Commission Européenne met a disposition les données[3] nécessaires pour estimer la puissance solaire perçue par des panneaux photovoltaiques depuis 2005 partout en Europe. De plus, on peut accéder aux prix de l'énergie sur différents marchés (comme le day-ahead ou intraday) sur le site web de EPEX Spot[4]. L'avancée de la recherche sur les algorithms de prédiction, par exemple à l'aide du machine learning, couplé avec l'abondance de données dont on dispose, permet d'estimer les lois de probabilité que suivent ces paramètres énergétiques.

Le problème qui nous intéresse est d'opérer un micro-réseau composé de plusieurs unités de génération d'énergie $g \in \mathcal{G}$ et d'un Système de Stockage d'Énergie (SSE)– ou batterie– dans lequel on peut stocker de l'énergie. Les unités de génération d'énergie renouvelable, $g \in \mathcal{G}_{re} \subset \mathcal{G}$, comme les panneaux solaires ou les éoliennes, ne sont pas contrôlables: l'énergie générée dépend uniquement de conditions externes. En revanche, les unités non-renouvelables, $g \in \mathcal{G} \setminus \mathcal{G}_{re}$, peuvent être contrôlées de manière analogue aux machine des productions modélisées dans la Section 1.2.1: on peut ajuster leur niveau de production tant que des contraintes physiques sont respectées. Le micro-réseau doit répondre à une demande en énergie. Il est aussi connecté au réseau principal, ce qui lui permet d'acheter de l'énergie sur les marchés day-ahead (DA) et d'équilibrage (B). Le problème de gestion du micro-réseau consiste à déterminer les flux d'énergie qui permettent de satisfaire la demande en énergie sur un laps de temps donné. Plus concrètement, à chaque étape $t \in [T]$, on doit décider quelle quantité d'énergie produire sur chaque unité non-renouvelable, l'énergie à charger ou décharger du SSE, et l'énergie à acheter sur les différents marchés d'énergie. Dans ce problème, les principales sources d'incertitude sont l'énergie disponible via les unités renouvelables, la demande en énergie et les prix de l'énergie. On les modélise comme des variables aléatoires discrètes dont on connaît la distribution, ce qui mène au programme stochastique linéaire multiétapes:

$$\min_{\mathrm{SOC},q,\phi} \quad \mathbb{E}\left[\sum_{t=1}^{T} \boldsymbol{p}_t^{DA} q_t^{DA} + \boldsymbol{p}_t^{B} q_t^{B})\right] \tag{1.6a}$$

$$\text{s.t} \quad \sum_{g\in\mathcal{G}_{re}} \boldsymbol{q}_t^g + \sum_{g\in\mathcal{G}\setminus\mathcal{G}_{re}} q_t^g - \phi_t^+ + \phi_t^- + q_t^{DA} + q_t^B \le \boldsymbol{q}_t^{\text{load}} \qquad \forall t \tag{1.6b}$$

$$\mathrm{SOC}_t = \mathrm{SOC}_{t-1} + \eta\phi_t^+ - \frac{1}{\eta}\phi_t^- \qquad \forall t \tag{1.6c}$$

$$\underline{\mathrm{SOC}} \le \mathrm{SOC}_t \le \overline{\mathrm{SOC}} \qquad \forall t \tag{1.6d}$$

$$(q_t^{DA}, q_t^B) \in \mathcal{M}_t \qquad \forall t \tag{1.6e}$$

$$(\phi_t^+, \phi_t^-, (q_t^g)_{g\in\mathcal{G}\setminus\mathcal{G}_{re}}) \in \mathcal{E}_t \qquad \forall t \tag{1.6f}$$

$$\sigma(\boldsymbol{\xi_t}) \subset \sigma(\boldsymbol{\xi_1}, \dots, \boldsymbol{\xi_{t-1}}) \qquad \forall t. \tag{1.6g}$$

A l'étape t, q_t^{DA} (*resp.* q_t^B) représente l'énergie achetée au marché day-ahead (*resp.* d'équilibrage); q_t^g l'énergie générée par l'unité non-renouvelable $g \in \mathcal{G} \setminus \mathcal{G}_{re}$; ϕ_t^+ (resp. ϕ_t^-) l'énergie chargée (resp. déchargée) dans la batterie; et SOC_t l'énergie contenus dans la SSE. Les incertitudes de l'étape t sont contenues dans le vecteur aléatoire $\boldsymbol{\xi_t} := (\boldsymbol{p}_t^{DA}, \boldsymbol{p}_t^B, (\boldsymbol{q}_t^g)_{g\in\mathcal{G}_{re}}, \boldsymbol{q}_t^{\text{load}})$ où $\boldsymbol{p}_t^{DA}$ (*resp.* $\boldsymbol{p}_t^B$) est le prix aléatoire du marché day-ahead (*resp.* d'équilibrage); $\boldsymbol{q}_t^g$ la génération alétoire de l'unité renouvelabale $g \in \mathcal{G}_{re}$; et $\boldsymbol{q}_t^{\text{load}}$ la demande aléatoire en énergie que le micro-réseau doit satisfaire. La fonction objectif (1.6a) est de minimiser la somme des coûts d'énergie en moyenne. La principale contrainte (1.6b) du problème est d'assurer l'équilibre énergétique: la

[3] https://re.jrc.ec.europa.eu/pvg_tools/en/
[4] https://www.epexspot.com/en

demande énergétique doit être satisfaite grâce à la production énergétique, les opérations du SSE et les achats énergétiques. La quantité d'énergie contenus dans le SSE est bornée (1.6d) et suit des dynamiques de stock classiques (1.6c). Les variables de décisions sont sujettes à des contraintes d'acceptabilité (1.6e) et (1.6f), qui généralisent de nombreuses contraintes techniques. Enfin, on modélise la structure d'information de l'aléa à l'étape t avec les contraintes de *non-anticipativité* (1.6g): les décisions sont prises sans connaître les aléas à partir de $t+1$, seulement leur distribution.

Le problème (1.6) est linéaire et continu (toutes les variables sont continues). Il est courant en optimisation stochastique de supposer que les variables aléatoires $\boldsymbol{\xi}_{[T]}$ sont indépendantes étape par étape *i.e.,* $\boldsymbol{\xi_t}$ et $\boldsymbol{\xi_{t'}}$ sont des variables indépendantes pour tous $t \neq t'$. En général, cette hypothèse n'est pas vérifiée dans la réalité, mais elle permet d'élaborer des algorithmes efficaces qui reposent sur les principes de Programmation Dynamique (PD). Dans le cas particulier où le problème et indépendant étape par étape et que la fonction objective est convexe, le problème est un Programme Stochastique Linéaire Multiétapes qui peut être résolu grâce à des algorithmes de génération de coupes.

Historiquement, le problème de planification de la production et celui de la gestion d'un micro-réseau sont étudiés et résolus séparemment. Toutefois, résoudre ces problèmes simultanément est essentiel pour accompagner les groupes industriels à investir dans des micro-réseaux. A ces fins, nous combinons les deux problèmes, en simplifiant le lien qui les unit à la demande en énergie. Celle-ci est dorénavant modélisée comme résultat des décisions de production de l'usine plutôt que comme une variable aléatoire. Nous présentons dans la section suivante un problème d'optimisation de planification couplée de production et d'énergie.

1.2.3 Un modèle générique pour la planification couplée de production et d'énergie

On présente maintenant modèle cadre générique qui vise à déterminer simultanément un planning de production avec les opérations d'un micro-réseau. On lie les Problèmes (1.1) et (1.6) à l'étape t à travers la demande en énergie q_t^{load} qui est fonction des décisions de production u et b. On modélise le problème comme un Programme Stochastique Linéaire Multiétapes à variables mixtes (à la fois entières et continues):

$$\min_{\mathrm{SOC},q,\phi,u,b,s} \mathbb{E}\left[\sum_{t=1}^{T} \boldsymbol{p}_t^{DA} q_t^{DA} + \boldsymbol{p}_t^{B} q_t^{B} + f_t^{ij}(u_t^{ij}, b_t^{ij}, s_t^{j})\right] \tag{1.7a}$$

$$\text{s.t} \quad \sum_{g\in\mathcal{G}_{re}} \boldsymbol{q}_t^{g} + \sum_{g\in\mathcal{G}_{nre}} q_t^{g} - \phi_t^{+} + \phi_t^{-} + q_t^{DA} + q_t^{B} \leq q_t^{\text{load}} \qquad \forall t \tag{1.7b}$$

$$q_t^{load} = g_t(u_t^{ij}, b_t^{ij}) \qquad \forall t. \tag{1.7c}$$

$$(1.1b) \text{ to } (1.1f)$$

$$(1.6b) \text{ to } (1.6g)$$

Dans le Problème (1.2.3), on trouve toutes les variables des Problèmes (1.1) et (1.6) et les contraintes qui leurs sont associées: le contraintes de production (1.1b) - (1.1f) et celles d'énergie (1.6b) - (1.6g). Le lien entre les deux problèmes est modélisé à travers la contrainte d'équilibre énergétique (1.7b), où q_t^{load} est fonction des variables $\{u_t^{ij}, b_t^{ij}\}$. Plus précisément, on considère une fonction g_t qui calcule la consommation énergétique q_t^{load} (1.7c) nécessaire au planning de production à l'étape t. A nouveau, pour des raisons de calculabilité, on suppose g linéaire. Enfin, la fonction objectif à minimiser est la somme des coûts de production (1.1a) et d'énergie (1.6a).

Si le problème contient un grand nombre de variables, la combinaison des incertitudes avec des variables entières le rend particulièrement difficile à résoudre. S'il existe des algorithmes théoriques exacte, ils convergent lentement. Dans Part II, nous proposons une méthode alternative pour résoudre ces problèmes difficiles.

Remark 1 (Extension de l'état)**.** *Comme dans la Section 1.2.1, en fonction du problème on peut étendre l'état du système pour inclure de l'information supplémentaire. Par exemple, dans le cas des contraintes de fonctionnement/arrêt minimum, on aura aussi besoin des variables de mise en marche/d'arrêt* $\{\text{up}_\tau^i\}_{\tau\in[t-M^i,t]}$ *and* $\{\text{down}_\tau^i\}_{\tau\in[t-m^i,t]}$ *pour décrire l'état du système à l'étape* t*.*

1.2.4 Agréger des entités indépendantes

Dans la section précédente, nous avons introduit le problème de planification couplée de production et d'énergie pour une usine unique. Dans certains cas, un décideur peut avoir à superviser seul plusieurs usines, ce qui revient à résoudre plusieurs variantes du Problème (1.7). Par exemple, si une entreprise possède plusieurs sites de production, il est naturel de vouloir coordonner la production et le stockage des différents sites pour minimiser les coûts globaux. Dans un autre contexte, une entreprise externe peut proposer comme service d'agréger des *prosumers* sur les marchés d'énergie. Certains marchés d'énergie ne sont accessibles qu'à partir d'une certaine taille (un volume d'énergie à vendre ou à acheter suffisant). Alors, des *prosumers* indépendants peuvent trouver un intérêt économique à se regrouper pour accéder à ces marchés, par exemple pour acheter des blocs d'énergie à un tarif préférentiel à se partager ensuite. Dans cette situation, il est avantageux qu'un seul décideur, qui connaisse les besoins de chacun, soit en charge d'optimiser les problèmes d'approvisionnement en énergie de ces différents *prosumers* collectivement plutôt qu'individuellement. Cela mène à la résolution de plusieurs problèmes (1.7) couplés par des décisions communes, ce qui peut être modéliser comme un très large problème stochastique.

Dans la Section 1.2.3, nous avons vu comment combiner les deux problèmes de planification de production et de gestion d'un micro-réseau: il s'agit essentiellement de juxtaposer les variables et contraintes des deux problèmes, et ceux-ci sont liés via les variables q_t^{load} et la contrainte d'équilibre énergétique. L'objectif est de minimiser les coûts globaux de l'usine, ce qui correspond à la somme des coûts de production et d'énergie. Nous pouvons procéder de la même manière pour modéliser l'agrégation de plusieurs entités. Toutefois, à la différence du problème traité dans la Section 1.2.3, les entités agrégées peuvent avoir des intérêts contradictoires. En effet, chaque entité a pour objectif de minimiser ses propres coûts, pas ceux de l'agrégateur. Le rôle de l'agrégateur externe est de trouver le juste équilibre entre optimiser les coûts globaux et garantir une répartition équitable des bénéfices entre les entités.

Traduire le concept d'équité en mathématiques n'est pas évident : il n'y a pas de définition universelle de l'équité et on peut en trouver des interprétations différentes voire contradictoires. Nous examinons différentes approches pour modéliser l'équité dans la Part III, en particulier dans un contexte d'aggrégation de *prosumers*.

1.3 Outils et défis mathématiques

Cette section est une introduction à l'optimisation stochastique, l'approche choisie pour résoudre le Problème (1.7). Plus précisément, nous examinons les méthodes de décomposition qui s'applique à la résolution d'un Problème Stochastic Multiétapes Linéaire en variables continues et Binaires (PSMLB). Tout d'abord, nous donnons une formulation générique d'un PSMLB ainsi qu'une formulation déterministe équivalente. Puis, nous présentons des sous-problèmes qui permette de mettre en place une approche par Programmation Dynamique (PD) appuyée par les principes de

Bellman. Enfin, à l'aide des opérateurs de Bellman, nous pouvons développer des *politiques* pour résoudre le problème, c'est-à-dire une implémentation des décisions *ici et maintenant* optimales à prendre.

1.3.1 Formulation du problème

Tout d'abord, nous proposons une formulation abstraite et compacte du Problème (1.7). Dorénavant, le caractère **gras** désigne les variables aléatoires, et nous notons $[n] = \{1, \dots, n\}$ l'ensemble des entiers positifs non nuls jusqu'à n. On considère un problème multiétapes où l'ensemble des incertitudes est modélisé par une séquence de variables aléatoires exogènes $\boldsymbol{\xi}_{[\boldsymbol{T}]}$, aussi appelées aléas. On suppose que chaque aléa $\boldsymbol{\xi_t}$ suit une loi de probabilité à support discret, que l'on connaît, dans l'espace de probabilités $(\Omega, \mathcal{A}, \mathbb{P})$. Les variables de contrôle sont soit continues, soit binaires. On obtient la formulation générique d'un PSMLB:

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{b}} \quad \mathbb{E}\left[\sum_{t=1}^{T} L_t(\boldsymbol{x_{t-1}}, \boldsymbol{y_t}, \boldsymbol{b_t}, \boldsymbol{\xi_t})\right] \tag{1.8a}$$

$$\boldsymbol{x_t} = F_t(\boldsymbol{x_{t-1}}, \boldsymbol{y_t}, \boldsymbol{b_t}, \boldsymbol{\xi_t}) \subset \mathfrak{X}_t \qquad \forall t \geq 1 \tag{1.8b}$$

$$\boldsymbol{y_t} \in \mathfrak{Y}_t(\boldsymbol{x_{t-1}}, \boldsymbol{\xi_t}) \qquad \forall t \geq 1 \tag{1.8c}$$

$$\boldsymbol{b_t} \in \mathfrak{B}_t(\boldsymbol{x_{t-1}}, \boldsymbol{\xi_t}) \cap \{0,1\}^{n_b} \qquad \forall t \geq 1 \tag{1.8d}$$

$$\sigma(\boldsymbol{y_t}, \boldsymbol{b_t}) \subset \sigma(\boldsymbol{\xi_1}, \dots, \boldsymbol{\xi_t}) \qquad \forall t \geq 1 \tag{1.8e}$$

$$\boldsymbol{x_0} = x_{init}. \tag{1.8f}$$

Dans le Problème (1.8), $\boldsymbol{x}_{[\boldsymbol{T}]}$ est une séquence de variables aléatoire d'états qui décrivent l'état du système à chaque étape, et qui suit une dynamique linéaire (1.8b) dans l'espace de faisabilité $\mathfrak{X}_{[T]}$. Soit n_b, le nombre de variables binaires par étape, on note $\boldsymbol{y_t}$ (resp. $\boldsymbol{b_t}$) les variables de contrôle continues (resp. binaires) de l'étape t, qui doivent satisfaire des contraintes linéaires contenues dans les ensembles de contrainte $\mathfrak{Y}_t(\boldsymbol{x_{t-1}}, \boldsymbol{\xi_t})$ (1.8c) et $\mathfrak{B}_t(\boldsymbol{x_{t-1}}, \boldsymbol{\xi_t})$ (1.8d). En optimisation stochastique, les variables d'état et de contrôle sont des variables aléatoires qui doivent respecter les contraintes de non-anticipativité (1.8e). Ces dernières contrôlent la quantité d'information disponible à l'étape t : on observe l'aléa à t avant de prendre des décisions, sans savoir ce que sera l'aléa futur. En particulier, cette contrainte de mesurabilité implique que les variables d'état et de contrôle à t sont des fonctions (mesurables) de l'aléa passé ($\xi_{[t]}$). Enfin, l'objectif (1.8a) est de minimiser la somme des coûts en moyenne. On peut décomposer les coûts par étape : le coût instantané de l'étape t est une fonction linéaire L_t qui dépend des variables et de l'aléa à t, ainsi que de l'état du système à l'étape précédente $\boldsymbol{x_{t-1}}$.

Remark 2 (Hypothèses de linéarité)**.** *Les notations utilisées dans le Problème* (1.8) *proviennent d'un cadre plus général : l'hypothèse de linéarité n'est pas nécessaire pour la PD classique et la convexité suffit pour les algorithmes comme SDDP (présenté dans la suite de la section). Toutefois, afin de pouvoir utiliser les solvers des PLM, nous supposons dans cette thèse qu'à chaque étape, le problème instantané peut être formulé comme un PLM.*

Il n'est pas immédiat d'adapter les algorithmes d'optimisation classiques à des modèles contenant des variables aléatoires. Toutefois, si celles-ci ont un support discret, le problème peut être reformulé comme un PLM. À cette fin, nous commençons par construire un *arbre de scénario* qui encapsule l'entièreté des incertitudes du problème tout en conservant la structure d'information. Généralement, l'arbre de scénario $\mathcal{T}$ est défini comme la collection de tous les scénarios, où chaque scénario $\{\xi_t^{j_t}\}_{t\in[T]}$ représente une réalisation de $\boldsymbol{\xi}_{[\boldsymbol{T}]}$, ce dernier contenant toutes les étapes.

On représente un exemple d'arbre de scénario pour un problèmes avec 4 étapes dans la Figure 1.5, où les incertitudes qui nous intéresse sont les conditions météorologiques– en particulier, s'il

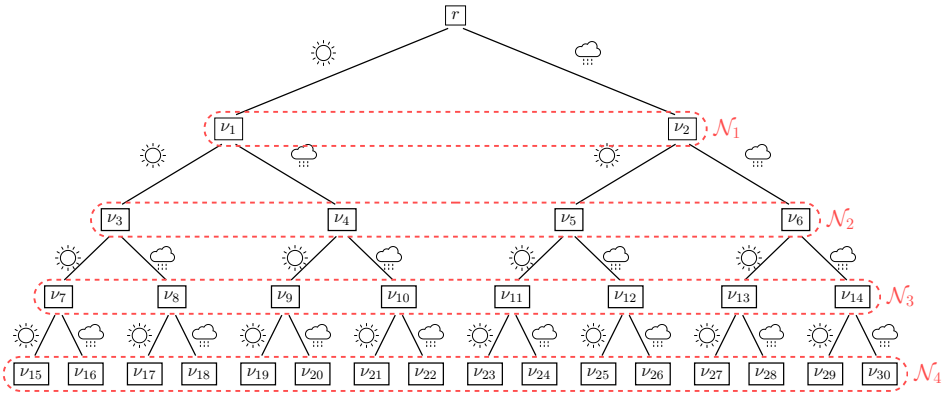


Figure 1.5: Exemple d'un arbre de scénario $\mathcal{T}$ où il y a deux possibilités à chaque étape: soit il fait beau, soit il pleut. Un scénario est alors une séquence contenant ce qu'il s'est passé au cours des 4 étapes. Par exemple, le scénario (🌧, 🌧, ☀, 🌧) est représenté dans l'arbre par le noeud ν_{28}.

pleut ou s'il fait beau à chaque étape. À partir de la racine de l'arbre r, les deux possibilités à l'étape $t = 1$ sont représentées par les noeuds ν_1, qui correspond au cas où il fait beau, et ν_2 s'il pleut. Puis, sachant qu'il a fait beau ou qu'il a plu à $t = 1$, il y a à nouveau deux cas possibles pour l'étape 2, ce qui aboutit à 4 noeuds $\{\nu_k\}_{k\in[3:6]}$ qui modélisent les scénarios possibles à $t = 2$. Notez que chaque *couche* de l'arbre correspond aux possibilités d'information disponible à une étape donnée du problème. De manière formelle, une couche $\mathcal{N}_t$ est définie par l'ensemble des noeuds de l'arbre $\mathcal{T}$ qui ont la profondeur t (à une distance t de la racine), et on note $a(\nu)$ le prédécesseur de ν dans l'arbre, aussi appelé parent de ν. Par exemple dans la Figure 1.5 ν_5 est le parent de ν_{11} et ν_{12}. Dans le Chapitre 4, on déroule une définition rigoureuse de l'arbre de scnéario appuyée par la théorie des graphes.

La structure de l'arbre de scénario permet de reformuler le Problème (1.8) comme un problème déterministe où les variables dépendent d'un noeud de l'arbre ν au lieu d'une étape t, et les dynamiques (1.8b) lient maintenant un noeud ν à son parent $a(\nu)$ au lieu d'une étape t à l'étape précédente $t-1$. On note π_ν la probabilité d'être dans le noeud ν qui représente le scénario $\xi_{[t]}$ *i.e.,* le produit des probabilités conditionnelles du chemin qui relie r à ν dans l'arbre de scénario

$$\pi_\nu = \mathbb{P}(\boldsymbol{\xi}_t = \xi_t, a(\nu)) = \mathbb{P}(\boldsymbol{\xi}_t = \xi_t | \, a(\nu)\,)\pi_{a(\nu)},$$

avec $\pi_r = 1$. Enfin, on reformule le problème comme un PLM, appelé *formulation extensive* ou *équivalent déterministe* du Problème (1.8):

$$\min_{x,y,b} \quad \sum_{t=1}^{T} \sum_{\nu\in\mathcal{N}_t} \pi_\nu L_t(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \tag{1.9a}$$

$$x_\nu = F_t(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \subset \mathfrak{X}_\nu \qquad \forall t \geq 1, \forall \nu \in \mathcal{N}_t \tag{1.9b}$$

$$y_\nu \in \mathfrak{Y}_\nu(x_{a(\nu)}, \xi_\nu) \qquad \forall t \geq 1, \forall \nu \in \mathcal{N}_t \tag{1.9c}$$

$$b_\nu \in \mathfrak{B}_\nu(x_{a(\nu)}, \xi_\nu) \cap \{0,1\}^{n_b} \qquad \forall t \geq 1, \forall \nu \in \mathcal{N}_t. \tag{1.9d}$$

$$x_r = x_{init} \tag{1.9e}$$

Le Problème (1.9) est équivalent au Problème (1.8): l'espérance des coûts (1.8a) se réécrit comme une somme, comme $\boldsymbol{\xi}_{[T]}$ sont à support fini; les contraintes de non-anticipativité (1.8e) sont

assurées par la structure de l'arbre de scénario. On rappelle que toutes les contraintes (les dynamiques F_ν et les ensembles de faisabilité $\mathfrak{X}_\nu, \mathfrak{Y}_\nu, \mathfrak{B}_\nu$), ainsi que les fonctions de coût L_t sont linéaires. Ainsi, le Problème (1.9) est bien un PLM. S'il existe des algorithmes efficaces pour résoudre les PLM, la taille du problème formulé ici est telle que le problème ne peut être résolu avec des solvers si l'horizon du problème dépasse quelques unités.

1.3.2 Programmation Dynamique et Approximations

Les PSMLB sont connus pour être difficiles à résoudre, même en ayant recours à des approximations. Toutefois, sous certaines hypothèses Markoviennes, on peut décomposer le problème à $T-$étapes en une séquence de sous-problèmes à $\tau-$étapes paramétrisés. Cette décomposition est particulièrement adaptée à une approche par Programmation Dynamique (PD) qui consiste à décomposer un problème complexe en des sous-problèmes plus petit, à partir desquels on peut retrouver une solution du problème original.

Nous supposons dorénavant que les aléas sont indépendants étapes par étapes, autrement dit $(\boldsymbol{\xi_1}, \dots, \boldsymbol{\xi_T})$ est une séquence de variables aléatoires indépendentes une à une. Alors, si notre problème est formulé comme dans (2.8), la théorie de la programmation dynamique dit qu'à une étape donnée t, les seules informations nécessaires pour prendre des décisions optimales *maintenant* sont l'*état* du système x_{t-1} et l'aléa observé à t. Nous définissons donc une séquence de sous-problèmes qui dépendent uniquement de l'état entrant du système x:

$$V_t(x) := \min_{\boldsymbol{x_{[t:T]}}, \boldsymbol{y_{[t:T]}}, \boldsymbol{b_{[t:T]}}} \mathbb{E}\left[\sum_{\tau=t}^{T} L_\tau(\boldsymbol{x_{\tau-1}}, \boldsymbol{y_\tau}, \boldsymbol{b_\tau}, \boldsymbol{\xi_\tau})\right] \tag{1.10a}$$

$$\boldsymbol{x_\tau} = F_\tau(\boldsymbol{x_{\tau-1}}, \boldsymbol{y_\tau}, \boldsymbol{b_\tau}, \boldsymbol{\xi_\tau}) \subset \mathfrak{X}_\tau \qquad \forall \tau \geq t \tag{1.10b}$$

$$\boldsymbol{y_\tau} \in \mathfrak{Y}_\tau(\boldsymbol{x_{\tau-1}}, \boldsymbol{\xi_\tau}) \subset \mathbb{R}^{n_u} \qquad \forall \tau \geq t \tag{1.10c}$$

$$\boldsymbol{b_\tau} \in \mathfrak{B}_\tau(\boldsymbol{x_{\tau-1}}, \boldsymbol{\xi_\tau}) \cap \{0,1\}^{n_b} \qquad \forall \tau \geq t \tag{1.10d}$$

$$\sigma(\boldsymbol{y_\tau}, \boldsymbol{b_\tau}) \subset \sigma(\boldsymbol{\xi_1}, \dots, \boldsymbol{\xi_\tau}) \qquad \forall \tau \geq t \tag{1.10e}$$

$$\boldsymbol{x_{t-1}} = x. \tag{1.10f}$$

Les fonctions $\{V_t\}_{t\in[T]}$ sont appelées fonctions de *Bellman*. $V_t(x)$ représente le coût optimal *à partir de maintenant* (de l'étape t) si l'état initial du système est x. En appliquant les principes de PD de Bellman, on obtient la récursion :

$$\hat{V}_t(x, \xi) = \min_{y,y,b} L_t(x, y, b, \xi) + V_{t+1}(z) \tag{1.11a}$$

$$z = F_t(x, y, b, \xi) \subset \mathfrak{X}_t \tag{1.11b}$$

$$y \in \mathfrak{Y}_t(x, \xi) \tag{1.11c}$$

$$b \in \mathfrak{B}_t(x, \xi) \cap \{0,1\}^{n_b} \tag{1.11d}$$

$$V_t(x) = \mathbb{E}\,[\,\hat{V}_t(x, \boldsymbol{\xi_t})\,]. \tag{1.11e}$$

Alors, on peut récursivement calculer l'ensemble des fonctions de Bellman, jusqu'à obtenir $V_1(x_{init})$, qui est la valeur optimale du Problème (1.8). En pratique, ces fonctions sont difficiles à calculer, et on peut au mieux les approximer. En effet, il y a peu de cas où on connaît l'expression explicite de $V_t(x)$. Si l'état du système est continu, pour obtenir une approximation de $V_t(x)$, on doit calculer $V_t(x^i)$ sur un nombre fini de points $\{x^i\}_{i\in[I]}$ dans l'espace des états; alors on étend la définition de $V_t(x)$ à tout point x par interpolation sur les valeurs $\{V_t(x^i)\}_{i\in[I]}$.

Dans l'intention de développer une méthodologie adaptée à n'importe quelle approximation $\mathcal{R}$ des fonctions de Bellman, on présente les *opérateurs arrière de Bellman* qui estiment le coût

optimal de t à T à partir de l'approximation des fonctions de Bellman à $t+1$*i.e.,*

$$\hat{\mathcal{B}}_t(\mathcal{R})(x,\xi) = \min_{z,y,b} L_t(x,y,b,\xi) + \mathcal{R}(z) \tag{1.12a}$$
$$z = F_t(x,y,b,\xi) \subset \mathfrak{X}_t \tag{1.12b}$$
$$y \in \mathfrak{Y}_t(x,\xi) \tag{1.12c}$$
$$b \in \mathfrak{B}_t(x,\xi) \cap \{0,1\}^{n_b} \tag{1.12d}$$
$$\mathcal{B}_t(\mathcal{R})(x) = \mathbb{E}\,[\,\hat{\mathcal{B}}_t(\mathcal{R})(x,\boldsymbol{\xi})\,]. \tag{1.12e}$$

De manière analogue, on définit les *opérateurs avant de Bellman* qui détermine la décision à prendre à l'étape t à partir de l'état courant x, de l'aléa et de l'approximation des fonctions de Bellman:

$$\hat{\mathcal{F}}_t(\mathcal{R})(x,\xi) \in \arg\min_{z,y,b} L_t(x,y,b,\xi) + \mathcal{R}(z) \tag{1.13a}$$
$$z = F_t(x,y,b,\xi) \subset \mathfrak{X}_t \tag{1.13b}$$
$$y \in \mathfrak{Y}_t(x,\xi) \tag{1.13c}$$
$$b \in \mathfrak{B}_t(x,\xi) \cap \{0,1\}^{n_b}. \tag{1.13d}$$

Les opérateurs avant construisent la politique optimale induite par une séquence d'approximations $\{\mathcal{R}_t\}_{t\in[T]}$, où *politique* est une règle de décision qui donne les décisions *ici-et-maintenant* optimales en fonction des coûts futurs estimés $\mathcal{R}_t$. En pratique, à une étape donnée t où l'état du système actuel est x_{t-1}, on observe l'aléa ξ_t, et on calcule les décisions optimales localement à prendre, et le nouvel état du système induit x_t en calculant $\hat{\mathcal{F}}_t(\mathcal{R}_t)(x_{t-1},\xi_t)$. On peut ainsi obtenir une infinité de politiques différentes qui dépendent des approximations faites. En particulier, la politique optimale du problème est obtenue à partir des vraies fonctions de Bellman V_t: $\hat{\mathcal{F}}_t(V_t)(x_t,\xi)$.

1.3.3 Algorithmes de Programmation Dynamique

Dans cette section, nous présentons formellement deux algorithmes qui reposent sur les principes de PD pour résoudre les problèmes stochastiques multiétapes: Stochastic Dynamic Programming (SDP) et Stochastic Dual Dynamic Programming (SDDP). Ces algorithmes sont détaillés dans les Chapitres (3) et (5).

Tout d'abord, SDP est un algorithme flexible qui requiert peu d'hypothèses sur la structure du problème et ainsi permet de résoudre une large classe de problèmes (dont des problèmes non linéaires et avec variables entières). Le principe de l'algorithme (décrit dans Algorithm 1) est de calculer chaque fonction de Bellman récursivement, en commençant par l'étape finale T, où $V_T \equiv K$ (on suppose $K = 0$ par simplicité). Alors, en utilisant les opérateurs de Bellman (1.12), on calcule $\hat{V}_{T-1}(x,\xi) = \hat{\mathcal{B}}_t(V_T)(x,\xi)$ pour chaque état possible x et aléa $\xi \in \text{supp}(\boldsymbol{\xi_{T-1}})$. Pour calculer $\hat{V}_{T-1}(x,\xi)$, on a besoin de la valeur $V_T(y)$ pour chaque état sortant potentiel y. Cela nécessite de discrétiser l'espace des états et de faire une interpolation sur ces valeurs pour avoir une définition approximée de $V_T(y)$ pour n'importe quel y. Une fois qu'on a obtenu $V_{T-1}(y)$, on itère le procédé pour calculer $V_{T-2}(y)$, et ainsi de suite.

Algorithm 1: Stochastic Dynamic Programming

Data: Paramètres du problème, grille de discrétisation $\mathbb{X}$, interpolateur
Result: stratégie et valeur optimale;
$V_T \equiv 0$; $V_t \equiv 0$;
for $t : T-1 \to 0$ **do**
  for $x \in \mathbb{X}_{t-1}$ **do**
    $V_t(x) = \mathcal{B}_t(V_{t+1})(x)$; `// Résoudre` $|\text{supp}(\boldsymbol{\xi_t})|$ `Problèmes` (1.12)
  Étendre la définition de V_t à partir de $\mathbb{X}_{t-1}$ to $\mathfrak{X}_{t-1}$ par interpolation.

Pour résoudre le Problème (1.8) avec SDP, il faut résoudre $\prod_{t=1}^{T} |\mathbb{X}_t|.|\text{supp}(\boldsymbol{\xi_t})|$ problèmes d'optimisation, ce qui dépasse rapidement les capacités de calcul des ordinateurs. Par exemple, dans un problème de production avec J produits, si on discrétise les niveaux de stock (*resp.* les aléas) avec seulement 10 (*resp.* 5) valeurs, SDP résout $T \times 5 \times 10^J$ problèmes d'optimisation. En pratique, SDP peut être implémenté tant que la dimension de l'état ne dépasse pas 5. De plus, si on discrétise l'espace d'état et qu'on calcule les fonctions de Bellman par interpolation, il est difficile de préserver des bornes et des guaranties de convergence.

Algorithm 2: Stochastic Dual Dynamic Programming

$\underline{V}_t^0 \equiv -\infty$ pour $t \in [T]$, ;
for $k \in \mathbb{N}$ **do**
  `/* Phase avant: calcul d'une trajectoire */`
  Fixer $x_0^k = x_{init}$;
  for $t = 1 \to T-1$ **do**
    Tirer aléatoirement $\xi_t^k \in \text{supp}(\boldsymbol{\xi_t})$;
    $x_t^k = \mathcal{F}_t(\underline{V}_{t+1}^{k-1})(x_{t-1}^k, \xi_t^k)$;
  `/* Phase arrière: amélioration des approximations */`
  Fixer $\underline{V}_T^k \equiv 0$;
  for $t = T-1 \to 1$ **do**
    for $\xi \in \Xi_t$ **do**
      Résoudre $\dot{\mathcal{B}}_t(\underline{V}_{t+1}^k)(x_{t-1}^k, \xi)$ pour $\dot{\alpha}_\xi$ and $\dot{\beta}_\xi$;
    Calculer $\alpha_t^k := \sum_{\xi \in \Xi_t} p_\xi \dot{\alpha}_{t,\xi}^k$ and, $\beta_t^k := \sum_{\xi \in \Xi_t} p_\xi \dot{\beta}_{t,\xi}^k$;
    Mettre à jour $\underline{V}_t^k := \max\left(\underline{V}_t^{k-1}, \langle \alpha_t^k, \cdot \rangle + \beta_t^k\right)$;

Alternativement, SDDP nécessite plus d'hypothèses sur la structure du problème que SDP mais permet de résoudre des problèmes de dimension bien plus grande que SDP. Si on suppose que les sous-problèmes (1.10) sont convexes, $V_{[T]}$ peut être approximé par des coupes linéaires calculées à l'aide de la théorie de la dualité. Le point central de SDDP est d'affiner itérativement une sous-approximation de chaque fonction de Bellman V_t au "bon endroit". Plus précisément, une itération de SDDP consiste en deux phases (voir Algorithm 2): une *phase avant*, dans laquelle on calcule une trajectoire $x_{[T]}^k$ localement optimale à l'aide des opérateurs avant (1.13) en fonction de l'approximation courante $V_{[T]}^{k-1}$ dont on dispose ; puis une *phase arrière* où ces approximations $V_{[T]}^{k-1}$ sont améliorées par l'ajout de nouvelles coupes calculées en $x_{[T]}^k$. Par convexité, ces coupes donnent de l'information sur l'ensemble de l'espace d'état, pas juste dans un voisinage de $x_{[T]}^k$. Enfin, il faut noter que, pour une discrétisation donnée $\mathbb{X}_t$, SDP tourne

pendant un laps temps donné : on ne dispose d'aucune information intermédiaire si l'algorithme est stoppée prématurément et donner du temps supplémentaire n'assure pas l'obtention de meilleurs résultats. En opposition, SDDP affine progressivement les approximations des fonctions de Bellman. Ainsi, si on stoppe l'algorithme entre deux itérations, on a une borne inférieure valide et une politique induite du problème initial : plus l'algorithme tourne longtemps, meilleure est cette borne inférieure.

	SDP		SDDP	
Hypothèse d'indépendance	Oui	👎	Oui	👎
Support fini des bruits	Oui	👎	Oui	👎
Hypothèses structurelles	Non	👍	Oui	👎
Contrôles discrets	Oui	👍	Non	👎
Discrétisation de l'état	Oui	👎	Non	👍
Résultats intermédiaires	Non	👎	Oui	👍
Dimension de l'état maximale	≈ 5	👎	≈ 30	👍

Table 1.1: Résumé des hypothèses et limites de calcul de SDP et SDDP

SDDP est utilisé en pratique pour résoudre des problèmes réels complexes. Par exemple, le système hydraulique du Brésil est opéré à l'aide de SDDP. Toutefois, SDDP n'est pas adapté à la résolution de PSMLB : avec les variables binaires on perd la convexité du problème. Un axe de recherche principal de cette thèse est d'explorer différentes façons de résoudre les MSbLP. Enfin, on résume dans la Table 1.1 les principales hypothèses et réflexions sur les performances computationnelles de SDP et SDDP.

1.4 Contributions

Cette section résume les principales contributions et résultats de cette thèse.

Contribution 1 (Problème couplé de plannification de la production et d'énergie)**.** *On propose un modèle pour le problème de plannification couplée de production et d'approvisionnement en énergie comme un PSMLB. Ce modèle prend en compte les incertitudes liées au problèmes, les contraintes techniques liées à la production (en particulier des contraintes de ressources partagées) modélisées avec des variables binaires, et des variables d'investissement comme les achats sur le marché day-ahead. Pour résoudre ce problème, on développe une heuristique qui construit une politique en résolvant des problèmes stochastiques à $t-$étapes où la fonction de coût finale est donnée par l'approximation des fonctions de Bellman obtenue avec SDDP.*

Dans la Section 1.1, nous avons souligné le besoin croissant de résoudre des problèmes industriels en tenant compte de la consommation énergétique induite et de son approvisionnement. Cela a motivé le projet présenté dans le Chapter 3, qui a abouti à une publication dans Energy Systems [FLG24]. Dans le Chapter 3, après avoir discuté plusieurs stratégies de résolution pour les PSMLB à partir de méthodologies connues (comme Model Predictive Control (MPC), Stochastic Dynamic Programming (SDP) et SDDP), nous proposons des méthodes heuristiques utilisant les approximations des fonctions de Bellman fournies par SDDP. Plus précisément, nous proposons une variante des opérateurs avant (1.13), qui, au lieu de résoudre un problème $1-$étape à τ avec les approximations $\mathcal{R}_{\tau+1}$, résout un problème à $t-$étapes avec les approximations $\mathcal{R}_{\tau+t}$. Pour ces approximations $\mathcal{R}$, on utilise les coupes calculées par SDDP, obtenues au préalable en un temps raisonnable.

En testant ces différentes méthodologie sur un cas client de METRON, nous avons constaté que la

méthode la plus efficace pour résoudre le problème opérationnel (sans variables d'investissement) est MPC. On peut en conclure que dans ce contexte spécifique, le traitement des variables binaires s'avère plus crucial que la gestion des incertitudes. Cependant, avec l'ajout de variables stratégiques, il semble pertinent d'utiliser SDDP pour fixer les variables de première étape (qui impactent fortement les coûts opérationnels), puis d'appliquer MPC pour résoudre le problème opérationnel ainsi paramétré.

Contribution 2 (B&B pour résoudre les PSMLBs)**.** *Nous proposons une procédure de Branch-and-Bound (BB) exploitant la structure sous-jacente des arbres de scénarios pour résoudre les PSMLB, en s'appuyant sur les approximations relachées des fonctions de Bellman obtenues via SDDP. Nous développons un algorithme de résolution exacte, Algorithm 12, qui utilise des stratégies de branchement pour faire pousser un sous-arbre de l'arbre de scénarios.*

Le Chapter 4 se concentre sur l'élaboration d'un cadre abstrait pour aborder les PSMLB à l'aide d'une méthodologie de type BB. Nous commençons par définir formellement les arbres de scénarios et introduisons différentes structures de sous-arbres. Ensuite, nous présentons les *fonctions d'assignation*, qui fixent, conservent ou relachent les variables binaires, modifiant ainsi l'ensemble des solutions admissibles de notre problème. Ces fonctions d'assignation constituent l'objet fondamental sur lequel s'applique la méthode BB. Nous discutons ensuite des conditions sur les fonctions d'assignation permettant de tirer parti de SDDP pour résoudre certaines parties du problème. Ces conditions conduisent à un algorithme BB efficace, décrit dans l'Algorithm 12.

Plus spécifiquement, en considérant des sous-arbres particuliers obtenus en élagant des branches de l'arbre de scénarios, nous construisons une relaxation partielle du Problème (1.8) oùl'intégralité est relâchée sur les portions élaguées. À partir d'un sous-arbre vide, l'Algorithm 12 fait pousser itérativement un sous-arbre en utilisant des méthodes de BB et converge vers la valeur exacte du Problème (1.8). Nous présentons plusieurs approches pour faire pousser un sous-arbre efficacement (couche par couche, aléatoirement, en évaluant l'écart à l'intégralité ou par *strong-branching*). Cette méthodologie généralise en particulier l'approche heuristique proposée en Contribution 1.

Contribution 3 (Modéliser l'équité)**.** *Nous proposons un cadre et des outils pour intégrer l'équité dans des modèles mathématiques, en particulier dans le contexte de l'agrégation de prosumers. Notre approche repose sur le concept de* fairness-by-design*, où nous établissons un degré d'équité directement dans le modèle, plutôt que de recourir à des redistributions postérieures, comme en théorie des jeux. Plus précisément, nous suggérons de prendre en compte l'équité de deux manières : i) en choisissant la manière d'agréger les coûts des agents, ii) en imposant des contraintes d'acceptabilité qui garantissent que chaque agent tire un bénéfice de sa participation à la coalition par rapport à une situation individuelle. Nous discutons également des extensions aux cas dynamiques et stochastiques.*

Comme discuté en Section 1.2.4, il peut être nécessaire d'agréger plusieurs usines aux intérêts divergents, ce qui soulève la question de comment garantir un traitement équitable entre elles. Avant toute chose, nous donnons dans le Chapter 6 une perspéctive globale sur le concept d'équité, qui paraît intuitif mais pose des difficultés en modélisation. A partir d'une discussion conceptuelle, nous présentons différentes définitions de l'équité et leur traduction en modèles mathématiques, en soulignant l'importance de la modélisation et de l'analyse contextuelles pour choisir une approche appropriée.

Puis, dans le Chapter 7, nous élaborons un cadre de modélisation *fairness-by-design* destiné aux agrégateurs d'énergie. Plus précisément, nous présentons des opérateurs d'agrégation permettant de concilier l'efficacité (optimisation des coûts) et l'équité (équité des allocations) via la fonction objectif. Nous introduisons également des contraintes d'*acceptabilité* pour garantir que chaque

agent a un intérêt à participer à l'agrégation, en s'assurant que son coût au sein de l'agrégation est inférieur à son coût individuel.

Cependant, dans un contexte dynamique et/ou stochastique, le coût n'est plus une valeur unique mais un vecteur. Ainsi, la comparaison des coûts est effectuée selon un certain ordre. Nous analysons différentes options pour le cadre dynamique (long terme) – afin de s'assurer qu'un agent n'a pas intérêt à quitter l'agrégation avant la fin – et pour le cadre stochastique, en utilisant la théorie des ordres stochastiques.

Ce projet a conduit à un préprint [FLP24], actuellement en cours de révision dans Computational Management Science.

Chapter 2

Introduction

Contents

2.1 Energy awareness in industrial groups

This PhD is done in collaboration by METRON, a French CleanTech expert in energy performance. METRON helps industrial and tertiary groups to optimize energy consumption and reduce carbon emissions. The motivation behind this thesis is to accompany industrial groups through microgrid investments.

2.1.1 METRON: a French CleanTech company

METRON[1] was founded in 2013 and aims to digitally transform energy systems to support global decarbonization efforts. The company offers a comprehensive digital platform designed to visualize, monitor, optimize, and model energy performance strategies and decarbonization roadmaps using data analytics and artificial intelligence.

[1] https://www.metron.energy/

The main functionalities of the platform, illustrated in Figure 2.1[2], include data acquisition management for high-quality, secure data collection, and energy performance monitoring for visualizing, measuring, and benchmarking energy consumption. The platform also offers advanced analytics to optimize energy use and detect inefficiencies. Additionally, it tracks carbon impact and provides real-time energy cost management, helping organizations forecast budgets and manage sustainability strategies.

Figure 2.1: Functionalities of Metron's platform

This thesis is part of METRON's energy optimization projects. More specifically, the research examines the impact of microgrid investments on industrial groups. Key aspects under consideration include the operational challenges of managing a microgrid, its influence on existing production practices, and the implications for access to energy markets.

2.1.2 The stakes of investing in microgrids for industrial groups

Global warming is a pressing issue that significantly impacts the resilience of energy systems, leading us to question our approach to energy consumption. To achieve the goal of limiting temperature rise to below 2°C set in the 2016 Paris Agreement [COP16], we have to move towards more sustainable energy sources. Despite notable progress, with renewable energy reaching 14.6% of global primary energy consumption and contributing nearly 30% of global electricity generation in 2023 (see [Ins24]), much remains to be done.

A surge in investments is directed at clean energy technologies, such as renewables and energy storage. According to the IEA's annual report [IEA24], clean energy investments are projected to account for two-thirds of total energy investments in 2024. Solar photovoltaic (PV) technology leads these investments, surpassing all other electricity generation technologies. However, the intermittency of renewable energy calls for integration with energy storage and upgrades to grid infrastructure.

The industrial sector in particular must adapt by reducing carbon emissions and mitigating

[2] https://www.metron.energy/solution/

risks from centralized energy dependence. Key strategies include investing in renewable energy alongside energy storage systems, such as microgrids. Refer to [Sus17] for implemented examples. According to the US Department of Energy, microgrids are defined as "a group of interconnected loads and distributed energy resources within clearly defined electrical boundaries that acts as a single controllable entity with respect to the grid." Microgrids provide increased reliability, especially during energy shortages, as they can operate independently from the main grid. However, they also introduce operational complexities. Meanwhile, improving energy efficiency through advanced technologies or updated production practices is equally crucial.

Balancing environmental goals with economic viability is a significant challenge. Technologies like microgrids, though promising, can be prohibitively expensive without government support. For instance, an economic evaluation of renewable energy microgrids ([Wan+20]) underscores the need for more effective policies to encourage investment. Simultaneously, there is increasing pressure to reduce carbon emissions, with measures such as carbon taxes. For example, the EU's Carbon Border Adjustment Mechanism (CBAM) seeks to address this by taxing imported goods based on their carbon footprint.

Industrial companies would benefit from affordable solutions and expert guidance in navigating energy management and optimization challenges. Despite the critical need for energy efficiency, many industrial groups are not fully digitized and remain unaware of the available optimization and data science tools. This thesis aims to improve our understanding of how microgrids can be integrated into industrial settings and to explore strategies for optimizing their use.

Having examined how the industrial sector can adapt its energy consumption to align with environmental goals, we now discuss the necessary adaptation of the energy market to accommodate decentralized energy production.

2.1.3 Towards a consumer-centered energy market model

To enhance understanding, we begin by providing a brief overview of the electric power industry and refer to [KS04] for more details. A power system connects production and consumption means for electricity, involving various actors: grid operators (transmission system operators and distribution companies), energy producers, retailers (who purchase energy on the wholesale market), consumers (who must buy from retailers or can access the wholesale market depending on their size), regulators and market operators.

For over a century, from the late 19^{th} century to the 1980s, the electricity market functioned as a vertically integrated and heavily regulated system, meaning they oversaw energy generation, transmission and distribution to end-consumers. Consequently, consumers had no alternative but to purchase electricity from the local monopoly supplier. Starting in the 1980s, deregulation began to reshape the industry, leading to the emergence of various market organizations.

Nowadays, we categorize energy markets into three main types: the capacity market, which ensures sufficient generation capacity; the energy market, responsible for optimal scheduling and exchanges; and the ancillary service market, which supports power system operations. Our focus is on the energy market, which interacts with energy actors on various timelines (see Figure 2.2). The *futures market* involves long-term contracts for price hedging and risk management; the *day-ahead (or spot) market* trades energy for the next day; the *intraday market* allows for adjustments by trading up to an hour before production; and the *balancing market* ensures real-time system balance. More specifically, reserve contracts are activated in the balancing market to maintain system stability and participants compromising power balance are penalized. In this thesis, we concentrate on the day-ahead and balancing markets.

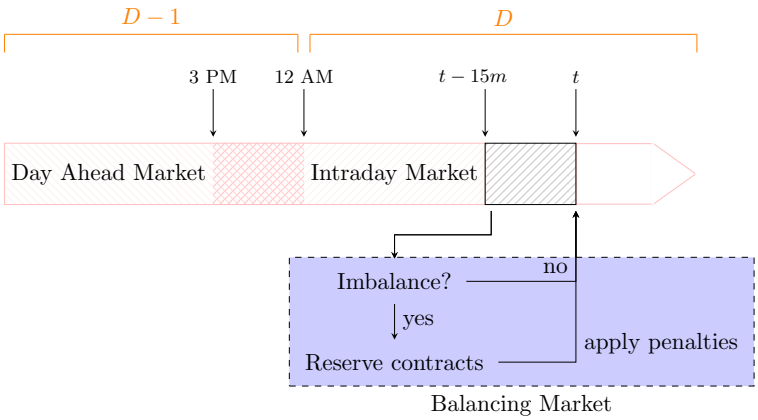


Figure 2.2: Illustration of the energy market timeline, excluding the futures market where contracts are established months or years in advance.

However, deregulated markets remain centralized in certain aspects, limiting small-end consumers from directly trading energy. With the rise of prosumers (both consumers and producers), some markets are gradually shifting toward a more decentralized model, where prosumers play a more active role. A decentralized energy market integrating smart prosumers must be designed to handle the complexity of diverse services and actors that can change roles. Several market designs addressing these challenges are discussed by Parag and Sovacool [PS16].

In particular, some prosumers may wish to form a community and collaborate, a process that can be facilitated by small or medium-scale companies acting as aggregators. In such case, aggregators are responsible for ensuring efficient operation, optimizing energy flows, and maintaining fairness in the distribution of benefits among participants. Incorporating fairness into mathematical models commonly used in these fields poses significant challenges. The inherent subjectivity of fairness necessitates a thoughtfully crafted approach. This issue is examined thoroughly in Part III of the thesis.

2.1.4 Operation research as support towards new practices

We now present an overview of Operational Research (OR), despite the absence of an official definition, as noted by the Association of European Operational Research Societies in their own presentation [EUR]. We define OR as a discipline that develops and applies analytical methods to improve decision-making. This definition is intentionally broad, encompassing diverse methods and applications. OR aims to facilitate the selection and implementation of effective solutions within complex systems, and has been used intensively in business, industry and government.

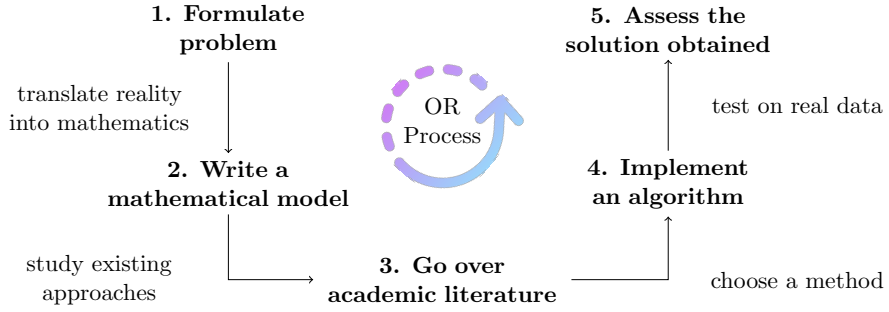


Figure 2.3: Operational research methodology

OR employs a variety of tools, ranging from abstract theories (*e.g.,* graph, complexity, or polyhedral theory) to practical algorithms (*e.g.,* simplex algorithm for linear programming, Dijkstra's algorithm for shortest path or branch-and-bound methodology for mixed integer programming). We can simplify the OR methodology into a structured five-step process, illustrated in Figure 2.3. First, a real-world problem is formulated and translated into a mathematical model. Next, relevant academic literature is reviewed to classify the problem and explore existing solutions. A suitable method is then selected and adapted if needed for implementation. The proposed solution is tested and analyzed to determine its effectiveness. If the solution falls short, the process is revisited, allowing for adjustments to the modeling or implementation stages to refine assumptions or improve performance until a satisfactory outcome is reached.

The main challenge lies in striking the right balance between an approximate solution, as models are always simplified to some extent, and a high-quality solution that is satisfactory for decision-makers. A notable example of OR application is in solving vehicle routing problems, used by companies like Renault or SNCF, which optimizes the routes over a fleet of vehicles serving a set of customers. Another major success is the hydrothermal coordination problem for the Brazilian power system, solved daily using multistage stochastic optimization algorithms.

2.2 Mathematical models

We present here OR models to discuss the impact of renewable energy investments on industrial problems. We start with production problems, underlying the need for binary variables to represent various technical constraints. Then, with in mind the potential investments of industrials in microgrids, we present energy procurement problems, which are inherently stochastic due to the presence of renewable energies, and where the focus is to optimize an energy system. This leads to a challenging model that integrates production scheduling and energy procurement. We conclude the section by discussing the modeling challenges of aggregating multiple entities for collective benefits.

2.2.1 Production problems: the need of binary variables

Optimizing inventory management and scheduling operations presents significant challenges in a production environment. This section presents a generic scheduling model, with some examples of specific constraints that may be encountered.

Consider a factory with I machines and J products that can be stored in warehouses with limited capacity. Our goal is to compute a detailed production plan for this factory for a given horizon (a day, a week, etc.). Concretely, we decompose the horizon into time slots (typically 15 minutes or an hour) that we call *stages* $t \in [T]$, T being the total number of stages. At each stage, the production plan indicates what, and in which quantities, to produce or store in order to satisfy demand (incoming orders). Our objective is to minimize the factory costs, which include

production and inventory costs. We formulate the following model

$$\begin{aligned}
\underset{u,b,s}{\text{Min}} \quad & \sum_{t=1}^{T} f_t^{ij}(u_t^{ij}, b_t^{ij}, s_t^j) && && \text{(2.1a)}\\
\text{s.t.} \quad & s_t^j = s_{t-1}^j + \sum_i u_t^{ij} - d_t^j && \forall t, j && \text{(2.1b)}\\
& \underline{s}^j \le s_t^j \le \overline{s}^j && \forall t, j && \text{(2.1c)}\\
& \underline{u}^{ij}\, b_t^{ij} \le u_t^{ij} \le \overline{u}^{ij}\, b_t^{ij} && \forall t, i, j && \text{(2.1d)}\\
& \{u_t^{ij}\}_{i,j} \in \mathfrak{U}_t && \forall t && \text{(2.1e)}\\
& \{b_t^{ij}\}_{i,j} \in \mathfrak{B}_t \cap \{0,1\}^{I\times J} && \forall t, && \text{(2.1f)}
\end{aligned}$$

where u_t^{ij} represents the quantity of product j produced on machine i at time t, b_t^{ij} is a binary variable that equals 1 if $u_t^{ij} > 0$ and 0 otherwise, and s_t^j models the quantity of product j that can be stored in a warehouse at time t. The objective (2.1a) aims to minimize the aggregated cost over all stages, where f_t^{ij} is a function of variables u_t^{ij}, b_t^{ij}, and s_t^j that models their associated costs. For tractability purposes, f is usually a linear or convex approximation of the real cost function. The constraints (2.1b) describe classical stock dynamics, where the previous stock level determines the level of stock at time t, adjusted for production and demand. Further, the stock is capacitated (2.1c). When a machine i is turned on to produce product j, the production must fall between bounds $\underline{u}^{ij}$ and $\overline{u}^{ij}$. If no production occurs, the output is naturally zero. Binary variables are essential to model this discontinuity in production bounds (2.1d). Specifically, if $b_t^{ij} = 0$, meaning we do not produce, then $0 \le u_t^{ij} \le 0$; otherwise, $\underline{u}_t^{ij} \le u_t^{ij} \le \overline{u}_t^{ij}$. Finally, we represent the constraints imposed by production processes and machinery through feasibility sets $\mathfrak{U}_t$ (2.1e) and $\mathfrak{B}_t$ (2.1f).

In production problems, binary variables are a convenient tool to model physical machinery constraints (see constraints (2.1d)), as seen in the following examples. Typically, a machine i can produce a single product at any given time, which can be modeled as:

$$b_t^{ij} = 1 \implies b_t^{ij'} = 0 \quad \forall j' \neq j,$$

which translates to the inequality to be included in $\mathfrak{B}_t$,

$$\sum_j b_t^{ij} \le 1. \tag{2.2}$$

Another example is *shared resources* or *incompatibility* constraints, which indicate that two products j and j' cannot be produced simultaneously. Such constraints often arise from operational preferences or staffing limitations. To model incompatibility between products j and j', we use the inequality

$$\max_i b_t^{ij} + \max_i b_t^{ij'} \le 1. \tag{2.3}$$

Here, $\max_i b_t^{ij}$ equals 0 if product j is not produced at stage t and 1 if it is. Consequently, this constraint ensures that either product j or j' can be produced at stage t, but not both.

We designate s_t^j as a *state variable*, representing the factory's state at a given time t and resulting from previous decisions. Without further specification of the scheduling problem, the only state variables are the stock levels of each product. However, additional constraints related to the factory's processes could require additional state variables. For example, for maintenance reasons,

some machines should not be turned on more than L^i times over the entire problem duration, from $t = 1$ to T. Those *counter* constraints introduce a new state variable c_t^i that counts the number of times machine i has been activated up to stage t. They can be modeled with the following equations

$$c_t^i = c_{t-1}^i + \mathbb{1}_{\sum_j b_t^{ij} - \sum_j b_{t-1}^{ij} = 1} \qquad \forall t, i \tag{2.4a}$$

$$0 \leq c_t^i \leq L^i \qquad \forall t, i. \tag{2.4b}$$

Here, $\sum_j b_t^{ij}$ equals 1 if machine i is turned on at stage t (where the sum cannot exceed 1, see (2.2)), and 0 otherwise. Thus, the count c_t^i equals the previous count c_{t-1}^i plus 1 if the machine is turned on at stage t, amounting to dynamics (2.4a). Finally, we ensure that the turning-on limit for machine i is not exceeded by applying constraint (2.4b).

Alternatively, to prevent excessive wear, it may be required that once a machine i is turned on (*resp.* off), it remains operational (*resp.* shut down) for a certain number of consecutive stages. For *minimum uptime/downtime* constraints, we need additional binary variables: up_t^i (*resp.* down_t^i) equals 1 if we turn on (*resp.* down) machine i at stage t, and 0 otherwise. Finally, we can model those new requirements with

$$\text{up}_t^i - \text{down}_t^i = \sum_j b_t^{ij} - \sum_j b_{t-1}^{ij} \qquad \forall t, i \tag{2.5a}$$

$$\text{up}_t^i + \text{down}_t^i \leq 1 \qquad \forall t, i \tag{2.5b}$$

$$\sum_{\tau=t-M^i}^{t} \text{up}_t^i \leq \sum_j b_t^{ij} \qquad \forall t > M^i, i \tag{2.5c}$$

$$\sum_{\tau=t-m^i}^{t} \text{down}_t^i \leq 1 - \sum_j b_t^{ij} \qquad \forall t > m^i, i, \tag{2.5d}$$

where equations (2.5a) and (2.5b) ensure that up_t^i and down_t^i have the correct values. Specifically, if machine i is activated at stage t, then $\text{up}_t^i - \text{down}_t^i = 1$, and since down_t^i is binary, it follows that $\text{up}_t^i = 1$. Similarly, if machine i is deactivated at stage t, we must have $\text{down}_t^i = 1$. When machine i remains either on or off at stage t, $\text{up}_t^i - \text{down}_t^i = 0$, and constraint (2.5b) ensures that both up_t^i and down_t^i are equal to 0. Then, to describe the state of the system at stage t, we need the starting (*resp.* shutting down) information of the M^i (*resp.* m^i) previous stages *i.e.,* $\{\text{up}_\tau\}_{\tau \in [t-M^i, t]}$ and $\{\text{down}_\tau\}_{\tau \in [t-m^i, t]}$. Once again, we see the critical role of binary variables in modeling crucial constraints in production problems.

If the problem is linear, it is classified as a Mixed-Integer Linear Program (MILP), which can be efficiently solved with a solver such as Gurobi or HiGHS, as long as it is reasonably sized. On the other hand, if binary variables make the problem too difficult to solve, one might consider relaxing them. However, repairing a non-binary solution might prove difficult.

Production generates an energy demand, typically met by purchasing energy from a supplier. However, if an industrial group invests in a microgrid *i.e.,* its own energy generation and storage, it must manage microgrid operations. In the following section, we introduce a model for energy procurement, where the production side is simplified to a single energy demand parameter.

2.2.2 Energy procurement problem: the need of considering uncertainty

In energy problems, many parameters are uncertain, especially if we consider renewable energy generation, which is inherently unpredictable. While part of their output, like solar power following

day-night cycles, can be forecast, some parts remain unpredictable. Even with improved weather prediction models, gaps exist between predicted and actual energy generation. Additionally, energy markets are highly volatile and influenced by multiple factors, making them difficult to predict. Energy demand can be unpredictable as well, with unexpected factory shutdowns or outages. These uncertainties affect both the constraints of our energy models (generation units and demand) and the objective (energy prices), ultimately impacting the quality of the solutions obtained. To address this, we consider optimization theories that take into account uncertainty in the decision-making process, allowing for more adaptable solutions.

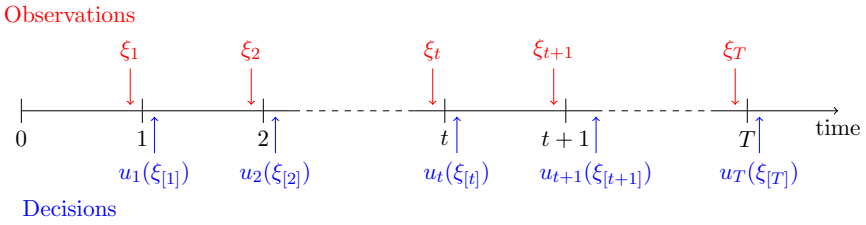


Figure 2.4: Coordination between decisions and the information available across time: at each stage t, we observe noise realization ξ_t and make an optimal decision $u_t(\xi_{[t]})$ function of past information $\xi_{[t]} = \{\xi_1, \dots, \xi_t\}$.

We consider a multistage problem, where decisions (either continuous or binary) can sequentially be made over stages in which uncertainties occur, see Figure 2.4. In this thesis, we always consider a *Hazard-Decision* information structure where uncertainties are observed before decision making at each stage. Alternatively, *Decision-Hazard* or even more complex structures such as *Hazard-Decision-Hazard* frameworks exist. There are different ways to handle uncertainties in the literature, depending on what is known. For example, if we know the uncertainties are contained in an *uncertainty set*, then the *robust optimization* approach consists of optimizing the problem against the *worst-case*. More precisely, it consists in selecting a solution assuming that the uncertainty realized will be the worst possible for this decision. This thesis focuses on the *stochastic optimization* approach, which models uncertainties using random variables with known distributions.

Although assuming the distributions of energy parameters are known may seem unrealistic, recent advances in data collection and analysis support this approach. For example, the European Commission provides a photovoltaic geographical information system[3] that allows users to download hourly irradiation data from 2005 for any location in Europe. Additionally, an online performance simulator for grid-connected photovoltaic systems is available. Energy prices for different markets (such as day-ahead and intraday) can also be accessed through the EPEX Spot website[4]. Despite the abundance of historical data, predicting these parameters remains difficult. Nevertheless, advanced forecasting algorithms, including machine learning, enable us to estimate their probability distributions.

We now consider a microgrid composed of multiple energy generation units $g \in \mathcal{G}$ and an Energy Storage System (ESS)– or battery– in which energy can be stored. Renewable units, $g \in \mathcal{G}_{re} \subset \mathcal{G}$, such as solar panels and wind turbines, are uncontrollable, as their output depends on external conditions. In contrast, non-renewable units, $g \in \mathcal{G} \setminus \mathcal{G}_{re}$, like fuel generators, can be controlled similarly to production machines in Section 2.2.1: they allow adjustable production levels, as long as physical constraints are satisfied. The microgrid has an energy demand– or load– to

[3] https://re.jrc.ec.europa.eu/pvg_tools/en/
[4] https://www.epexspot.com/en

satisfy, and it is connected to the main grid, allowing for energy purchases on the Day-Ahead (DA) and Balancing (B) markets. The energy procurement problem aims to determine how to operate the microgrid to satisfy energy demand over a horizon of time. More specifically, at each stage $t \in [T]$, we decide how much energy to produce (through non-renewable units), discharge, store and purchase from each market. In this problem, the main sources of uncertainties are the amount of energy the microgrid generates with renewable units, the energy load and the cost of purchasing energy from the grid. We model those uncertainties (renewable generation and energy prices) as finite discrete random variables of known probabilities, and obtain a multistage stochastic linear program:

$$\min_{\mathrm{SOC},q,\phi} \quad \mathbb{E}\Big[\sum_{t=1}^{T} \boldsymbol{p}_t^{DA} q_t^{DA} + \boldsymbol{p}_t^{B} q_t^{B})\Big] \tag{2.6a}$$

$$\text{s.t} \quad \sum_{g\in\mathcal{G}_{re}} \boldsymbol{q}_t^g + \sum_{g\in\mathcal{G}\setminus\mathcal{G}_{re}} q_t^g - \phi_t^+ + \phi_t^- + q_t^{DA} + q_t^B \le \boldsymbol{q}_t^{\text{load}} \qquad \forall t \tag{2.6b}$$

$$\mathrm{SOC}_t = \mathrm{SOC}_{t-1} + \eta\phi_t^+ - \frac{1}{\eta}\phi_t^- \qquad \forall t \tag{2.6c}$$

$$\underline{\mathrm{SOC}} \le \mathrm{SOC}_t \le \overline{\mathrm{SOC}} \qquad \forall t \tag{2.6d}$$

$$(q_t^{DA}, q_t^B) \in \mathcal{M}_t \qquad \forall t \tag{2.6e}$$

$$(\phi_t^+, \phi_t^-, (q_t^g)_{g\in\mathcal{G}\setminus\mathcal{G}_{re}}) \in \mathcal{E}_t \qquad \forall t \tag{2.6f}$$

$$\sigma(\boldsymbol{\xi}_t) \subset \sigma(\boldsymbol{\xi_1}, \ldots, \boldsymbol{\xi_{t-1}}) \qquad \forall t. \tag{2.6g}$$

At stage t, q_t^{DA} (*resp.* q_t^B) represents the energy purchased from the day-ahead (*resp.* balancing) market; q_t^g the energy generated by non-renewable unit $g \in \mathcal{G} \setminus \mathcal{G}_{re}$; ϕ_t^+ (resp. ϕ_t^-) the energy charged (resp. discharged) in the battery; and SOC_t the energy stored in the ESS. The uncertainties at stage t are modeled as random variables: $\boldsymbol{p}_t^{DA}$ (*resp.* $\boldsymbol{p}_t^B$) is the random day-ahead (*resp.* balancing) price at t; $\boldsymbol{q}_t^g$ the random generation of renewable unit $g \in \mathcal{G}_{re}$; and $\boldsymbol{q}_t^{\text{load}}$ the random energy load to be satisfies by the microgrid. For simplicity, we regroup the uncertainties at t in random vector $\boldsymbol{\xi_t} := (\boldsymbol{p}_t^{DA}, \boldsymbol{p}_t^B, (\boldsymbol{q}_t^g)_{g\in\mathcal{G}_{re}}, \boldsymbol{q}_t^{\text{load}})$. The objective (2.6a) is to minimize the expected sum of energy purchases from the day-ahead and balancing markets. The main constraint (2.6b) ensures energy balance: there is enough energy– through production, ESS discharge, or purchases– to satisfy the energy load. The energy stored in the ESS is bounded (2.6d) and follows classical stock dynamics (2.6c). Further, the decision variables are subject to feasibility constraints (2.6e) and (2.6f), that are to be specified. Finally, *non-anticipativity* constraint (2.6g) models the information available at stage t: we observe uncertainty at t before making decisions, knowing the past but not future random realizations (from $t+1$ onwards), only of their probability distributions.

The problem (2.6) is linear, meaning both the constraints and the objective function are linear in the decision variables; and continuous *i.e.,* all variables are continuous. A standard assumption in stochastic optimization is that random variables $\boldsymbol{\xi}_{[\boldsymbol{T}]}$ are stagewise independent *i.e.,* $\boldsymbol{\xi_t}$ and $\boldsymbol{\xi_{t'}}$ are independent for all $t \neq t'$. While this assumption is often unrealistic, it allows for the use of efficient algorithms based on Dynamic Programming (DP) Principles. In particular, under stagewise independence assumptions and if the objective is convex, the resulting Multistage Stochastic Linear Program (MSLP) can be efficiently solved using the cut-generation algorithm known as Stochastic Dual Dynamic Programming (SDDP).

Having described both the production scheduling and energy procurement problems, we now aim to merge them. The connection between the two problems can be reduced to the energy

load, determined by the factory's production decisions, rather than being modeled as a random variable. Traditionally solved separately, integrating these two become essential for industrial groups investing in microgrids. Indeed, joint optimization can significantly reduce costs through demand-side management and smart microgrid operations. The following section presents a generic joint production and energy planning model.

2.2.3 A generic model for joint production and energy planning

We now present a framework that considers production scheduling and energy procurement problems simultaneously. We link Problem (2.1) and Problem (2.6) at each stage t through the energy load q_t^{load} which is now a function of production decisions u and b. We can model the problem as a Multistage Stochastic mixed-Integer Linear Program (MSiLP), or more specifically as a Multistage Stochastic mixed-binary Linear Program (MSbLP) (we only consider binary variables):

$$\min_{\text{SOC},q,\phi,u,b,s} \mathbb{E}\left[\sum_{t=1}^{T} \boldsymbol{p}_t^{DA} q_t^{DA} + \boldsymbol{p}_t^{B} q_t^{B} + f_t^{ij}(u_t^{ij}, b_t^{ij}, s_t^{j})\right] \tag{2.7a}$$

$$\text{s.t} \quad \sum_{g\in\mathcal{G}_{re}} \boldsymbol{q}_t^{g} + \sum_{g\in\mathcal{G}_{nre}} q_t^{g} - \phi_t^{+} + \phi_t^{-} + q_t^{DA} + q_t^{B} \leq q_t^{\text{load}} \qquad \forall t \tag{2.7b}$$

$$q_t^{load} = g_t(u_t^{ij}, b_t^{ij}) \qquad \forall t. \tag{2.7c}$$

$$(2.1b) \text{ to } (2.1f)$$

$$(2.6b) \text{ to } (2.6g)$$

In Problem (2.7), we find variables from Problem (2.1) and Problem (2.6) and their associated constraints: production constraints (2.1b) - (2.1f) and energy procurement constraints (2.6b) - (2.6g). The link between both problems is modeled through the energy balance equation (2.7b), where q_t^{load} is now a function of production variables $\{u_t^{ij}, b_t^{ij}\}$. More specifically, we consider a function g_t that gives the energy consumption q_t^{load} depending on production decisions in (2.7c). Again, for tractability purposes, we assume g is linear in production variables. Finally, the objective is to minimize the sum of production costs(2.1a) and energy costs(2.6a).

A large MSbLP is known to be difficult to solve. Though there exist theoretical algorithms to find the optimal solution, they are slow to converge and impractical to solve real life problems. This motivates Part II, where we propose an alternative method.

Remark 3 (State expansion)**.** *As in Section 2.2.1, depending on the production constraints, we can expand the system's state with additional information. For example, with minimum uptime/downtime constraints, we also need starting and shutting down variables* $\{\text{up}_\tau^i\}_{\tau\in[t-M^i,t]}$ *and* $\{\text{down}_\tau^i\}_{\tau\in[t-m^i,t]}$ *to describe the state of the system at stage* t*.*

2.2.4 Aggregating entities

In the previous section, we introduced a joint production and energy planning problem for a single factory. In some cases, a single decision-maker has to manage multiple factories and thus solve different variants of Problems (2.7). This can come from a multi-site company or an external company aggregating prosumers. In the first case, a company with multiple production sites will naturally coordinate production and inventory across sites to minimize global costs. In the second case, aggregation is often motivated by economic benefits. For example, when a group of prosumers reaches a large enough size, it can access some specific electricity markets, such as buying energy blocks to be shared among the factories. In this context, it becomes

advantageous for a single decision maker, with insight into each prosumer's needs, to optimize these problems collectively rather than independently. This leads to the resolution of multiple Problems (2.7) coupled by some joint decision, resulting in a very large stochastic problem, which further emphasizes the need for efficient algorithms.

In Section 2.2.3, we saw how to merge two problems by constructing a new model that contains all the variables from each problem and retains most of their (identical) constraints. Some constraints are adjusted to link the variables (*e.g.,* the energy balance constraint (2.7b) connects the problem with new variable q_t^{load}), and new constraints or variables may be added (such as the energy consumption constraint (2.7c)). This same methodology applies when constructing the aggregated problem. However, a major difference lies in the objective: in Section 2.2.3, the goal was to minimize overall costs for a single factory, so when merging the two problems, we can simply sum their objectives. In the case of an external aggregator, the different prosumers might have different interests. Therefore, the model must ensure that the aggregator provides solutions that benefit each factory, incentivizing their participation. Indeed, each factory is mainly interested in its own cost, not the aggregated one. This introduces a multi-objective framework where the aggregator must balance minimizing overall costs with ensuring a fair allocation of benefits among the entities.

Fairness is challenging to model mathematically, as it lacks a clear definition and can evoke different interpretations. We explore how to define and model fairness in a prosumer aggregation context in Part III, specifically in multistage stochastic problems.

2.3 Mathematical tool and challenges

In this section, we introduce stochastic optimization, the approach chosen to solve Problem (2.7). Specifically, we discuss decomposition methods applicable to MSbLP. We start with a generic formulation of MSbLP, and then outline how to construct a deterministic equivalent for this problem. Next, we introduce subproblems that enable a Dynamic Programming (DP) approach based on Bellman's DP principles. Finally, we present Bellman operators, which allow us to develop *policies*– guidelines for optimal *here-and-now* decisions that incorporate their anticipated future impact.

2.3.1 Problem formulation

We start by presenting abstract formulations encompassing in particular Problem (2.7). From now on bold font is used to design random variables. Further we denote $[n] = \{1, \dots, n\}$ the set of integers up to n. We consider a multi-stage problem where all the uncertainty is modeled by a sequence of exogenous random variables $\boldsymbol{\xi}_{[T]}$, sometimes called noises. We assume that each noise $\boldsymbol{\xi}_t$ follows a discrete probability distribution, assumed to be known, in probability space $(\Omega, \mathcal{A}, \mathbb{P})$. The decisions are represented either by continuous or binary variables. The resulting

Multistage Stochastic mixed-binary Linear Program (MSbLP) reads

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{b}} \quad \mathbb{E}\left[\sum_{t=1}^{T} L_t(\boldsymbol{x_{t-1}}, \boldsymbol{y_t}, \boldsymbol{b_t}, \boldsymbol{\xi_t})\right] \tag{2.8a}$$

$$\boldsymbol{x_t} = F_t(\boldsymbol{x_{t-1}}, \boldsymbol{y_t}, \boldsymbol{b_t}, \boldsymbol{\xi_t}) \subset \mathfrak{X}_t \qquad \forall t \geq 1 \tag{2.8b}$$

$$\boldsymbol{y_t} \in \mathfrak{Y}_t(\boldsymbol{x_{t-1}}, \boldsymbol{\xi_t}) \qquad \forall t \geq 1 \tag{2.8c}$$

$$\boldsymbol{b_t} \in \mathfrak{B}_t(\boldsymbol{x_{t-1}}, \boldsymbol{\xi_t}) \cap \{0,1\}^{n_b} \qquad \forall t \geq 1 \tag{2.8d}$$

$$\sigma(\boldsymbol{y_t}, \boldsymbol{b_t}) \subset \sigma(\boldsymbol{\xi_1}, \dots, \boldsymbol{\xi_t}) \qquad \forall t \geq 1 \tag{2.8e}$$

$$\boldsymbol{x_0} = x_{init}. \tag{2.8f}$$

In Problem (2.8), $\boldsymbol{x_{[T]}}$ is a sequence of random state variables describing the state of the system, that follows linear dynamic equations (2.8b), in feasible sets $\mathfrak{X}_{[T]}$. Let n_b be the number of binary variables per stage, we denote $\boldsymbol{y_t}$ (resp. $\boldsymbol{b_t}$) the continuous (resp. binary) decision variables made at stage t, that must satisfy linear constraints represented by $\mathfrak{Y}_t(\boldsymbol{x_{t-1}}, \boldsymbol{\xi_t})$ (2.8c) and $\mathfrak{B}_t(\boldsymbol{x_{t-1}}, \boldsymbol{\xi_t})$ (2.8d). In stochastic optimization, state and decision variables are random variables and must respect non-anticipativity constraints (2.8e), which control the information available at stage t: we observe uncertainty at t before making decisions, with no knowledge of future random realizations (from $t+1$ onward). In particular, this measurability constraint implies that the states and control variable at stage t are (measurable) functions of the past noises ($\xi_{[t]}$). Finally, the objective (2.8a) is to minimize expected aggregated costs. Those costs are decomposed by stages: the instantaneous cost at stage t is L_t a linear function of the decisions and noise at t as well as the incoming state $\boldsymbol{x_{t-1}}$.

Remark 4 (Linearity assumptions)**.** *The notations used in Problem (2.8) come from a more generic framework, where linearity is not necessary for standard DP and convexity is enough for SDDP algorithms. However, to leverage the efficiency of MILP solvers, we assume in this thesis that, at each stage, the instantaneous problem is an MILP.*

Adapting standard optimization algorithms to models with random variables is not straightforward. However, if the random variables have discrete support, the problem can be reformulated as an MILP. To that end, we build a *scenario tree* that captures all the uncertainty in the problem while maintaining the information structure. Usually, the scenario tree $\mathcal{T}$ is defined as the collection of all scenarios, where each *scenario* $\{\xi_t^{j_t}\}_{t\in[T]}$ represents one realization of $\boldsymbol{\xi_{[T]}}$, since the latter contains "all times".

An example of a scenario tree with four stages is illustrated in Figure 2.5, where the uncertainty lies in the weather conditions—- specifically, whether it rains or is sunny at each stage. Starting from the root node r, the two possible outcomes at stage $t = 1$ are represented with nodes ν_1, corresponding to sunny weather, and ν_2 if it rains. Then, knowing the weather has been sunny or rainy at $t = 1$, we have again two possibilities for the second stage, resulting in 4 possible scenarios $\{\nu_k\}_{k\in[3:6]}$ at $t = 2$. Note that each *layer* of the tree corresponds to a stage t of Problem (2.8). Formally, a layer $\mathcal{N}_t$ is the set of nodes in $\mathcal{T}$ of depth t, and $a(\nu)$ is the parent of node ν (its predecessor in the tree). For example in Figure 2.5, ν_5 is the parent of ν_{11} and ν_{12}. In Chapter 4, we provide a rigorous definition of scenario trees using graph theory.

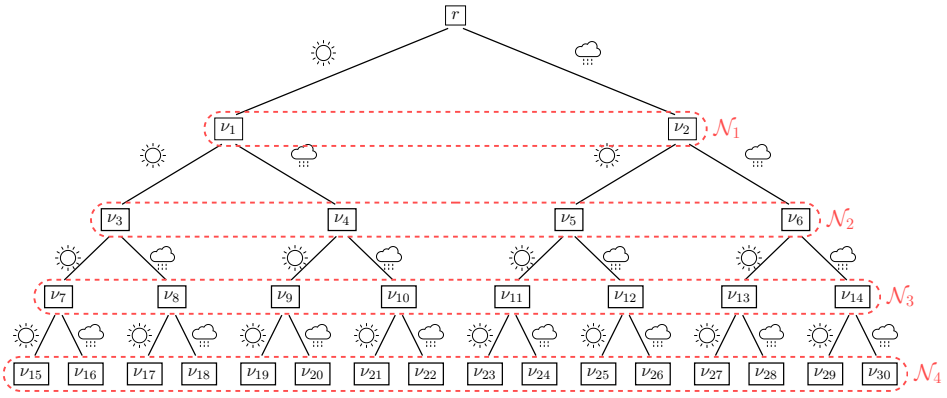


Figure 2.5: Example of a scenario tree $\mathcal{T}$ where there are two possible outcomes per stage: either the weather is sunny or it is rainy. A scenario is a sequence of what actually happened over the four stages. For example, a potential scenario is (, , ,) that corresponds to node ν_{28}.

Leveraging the scenario tree structure, we can reformulate Problem (2.8) as a large deterministic problem where (deterministic) variables depend on node ν instead of stage t, and the dynamics (2.8b) are between a node ν and its parent $a(\nu)$ instead of between stage t and stage $t+1$. We denote π_ν the probability of being in node ν which represent scenario $\xi_{[t]}$ *i.e.,* the product of conditional probabilities on the path linking r to ν in the scenario tree

$$\pi_\nu = \mathbb{P}(\boldsymbol{\xi}_t = \xi_t, a(\nu)) = \mathbb{P}(\boldsymbol{\xi}_t = \xi_t | \, a(\nu)\,)\pi_{a(\nu)},$$

with $\pi_r = 1$. We then obtain a MILP, called the *extensive formulation* or *deterministic equivalent* of Problem (2.8):

$$\min_{x,y,b} \quad \sum_{t=1}^{T} \sum_{\nu \in \mathcal{N}_t} \pi_\nu L_t(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \tag{2.9a}$$

$$x_\nu = F_t(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \subset \mathfrak{X}_\nu \qquad \forall t \geq 1, \forall \nu \in \mathcal{N}_t \tag{2.9b}$$

$$y_\nu \in \mathfrak{Y}_\nu(x_{a(\nu)}, \xi_\nu) \qquad \forall t \geq 1, \forall \nu \in \mathcal{N}_t \tag{2.9c}$$

$$b_\nu \in \mathfrak{B}_\nu(x_{a(\nu)}, \xi_\nu) \cap \{0,1\}^{n_b} \qquad \forall t \geq 1, \forall \nu \in \mathcal{N}_t. \tag{2.9d}$$

$$x_r = x_{init} \tag{2.9e}$$

Problem (2.9) is equivalent to Problem (2.8): the expectation of costs in Equation (2.8a) is rewritten as a sum, as $\boldsymbol{\xi}_{[T]}$ are finitely supported; the non-anticipativity constraints (2.8e) are ensured by the structure of the scenario tree. Recall that all constraints (dynamics F_ν and feasible sets $\mathfrak{X}_\nu, \mathfrak{Y}_\nu, \mathfrak{B}_\nu$) as well as the objective functions L_t are linear, thus Problem (2.9) is indeed an MILP. Even if there are efficient algorithms to solve MILP, the size of the problem is exponential in the number of stages, making it intractable to solve when T is more than a few units.

2.3.2 Dynamic Programming and Approximated problems

MSbLP are notoriously challenging to solve, even with approximation techniques. However, with certain Markovian assumptions, we can decompose them into parameterized subproblems, which can then be solved using a Dynamic Programming (DP) approach. DP is a decomposition method that consists in simplifying a complex problem by breaking it down into smaller subproblems,

which are then solved to construct a solution for the original problem. Multistage problems are well-suited to DP, as they naturally allow for a time-decomposition: from a $T-$stages problem, we can construct $\tau-$stages subproblems with $\tau < T$.

From now on, we assume that the noises are stage-wise independent, meaning that $(\boldsymbol{\xi_1}, \dots, \boldsymbol{\xi_T})$ is a sequence of stochastically independent random variables. Then, if our problem is of the form (2.8), the Dynamic Programming theory tell us that, at a given stage t, the only information needed to make optimal decisions *now* is the system's current *state* x_{t-1} and the noise observed at t. This allows us to define a sequence of subproblems that depend on an incoming state x:

$$V_t(x) := \min_{\boldsymbol{x_{[t:T]}}, \boldsymbol{y_{[t:T]}}, \boldsymbol{b_{[t:T]}}} \mathbb{E}\left[\sum_{\tau=t}^{T} L_\tau(\boldsymbol{x_{\tau-1}}, \boldsymbol{y_\tau}, \boldsymbol{b_\tau}, \boldsymbol{\xi_\tau})\right] \tag{2.10a}$$

$$\boldsymbol{x_\tau} = F_\tau(\boldsymbol{x_{\tau-1}}, \boldsymbol{y_\tau}, \boldsymbol{b_\tau}, \boldsymbol{\xi_\tau}) \subset \mathfrak{X}_\tau \qquad \forall \tau \geq t \tag{2.10b}$$

$$\boldsymbol{y_\tau} \in \mathfrak{Y}_\tau(\boldsymbol{x_{\tau-1}}, \boldsymbol{\xi_\tau}) \subset \mathbb{R}^{n_u} \qquad \forall \tau \geq t \tag{2.10c}$$

$$\boldsymbol{b_\tau} \in \mathfrak{B}_\tau(\boldsymbol{x_{\tau-1}}, \boldsymbol{\xi_\tau}) \cap \{0,1\}^{n_b} \qquad \forall \tau \geq t \tag{2.10d}$$

$$\sigma(\boldsymbol{y_\tau}, \boldsymbol{b_\tau}) \subset \sigma(\boldsymbol{\xi_1}, \dots, \boldsymbol{\xi_\tau}) \qquad \forall \tau \geq t \tag{2.10e}$$

$$\boldsymbol{x_{t-1}} = x. \tag{2.10f}$$

The functions $\{V_t\}_{t\in[T]}$ are called *cost-to-go* or *Bellman*'s functions. $V_t(x)$ represents the optimal cost that can be obtained *from now on* (here from stage t) if the initial state of the system is x. We can apply Bellman's dynamic programming principles and obtain the following recursion:

$$\hat{V}_t(x, \xi) = \min_{y,b} L_t(x, y, b, \xi) + V_{t+1}(z) \tag{2.11a}$$

$$z = F_t(x, y, b, \xi) \subset \mathfrak{X}_t \tag{2.11b}$$

$$y \in \mathfrak{Y}_t(x, \xi) \tag{2.11c}$$

$$b \in \mathfrak{B}_t(x, \xi) \cap \{0,1\}^{n_b} \tag{2.11d}$$

$$V_t(x) = \mathbb{E}\left[\, \hat{V}_t(x, \boldsymbol{\xi_t}) \,\right]. \tag{2.11e}$$

Then, we see that recursively, by computing all cost-to-go functions, we obtain $V_1(x_{init})$, the value of Problem (2.8). In practice, those functions are hard to compute, and most times can only be approximated. Indeed, if the state of the system is continuous, there are few cases where we can get an explicit expression for $V_t(x)$. To obtain an approximation of $V_t(x)$, we first compute the value $V_t(x^i)$ for a finite set of points $\{x^i\}_{i\in[I]}$ in the state space; then we get $V_t(x)$ for any x by interpolation on the values of the cost-to-go function $\{V_t(x^i)\}_{i\in[I]}$.

To account for all possible approximations $\mathcal{R}$ of cost-to-go functions, we introduce Bellman's *backward operator* that estimate cost-to-go at stage t from cost-to-go at stage $t+1$, *i.e.,*

$$\hat{\mathcal{B}}_t(\mathcal{R})(x, \xi) = \min_{z,y,b} L_t(x, y, b, \xi) + \mathcal{R}(z) \tag{2.12a}$$

$$z = F_t(x, y, b, \xi) \subset \mathfrak{X}_t \tag{2.12b}$$

$$y \in \mathfrak{Y}_t(x, \xi) \tag{2.12c}$$

$$b \in \mathfrak{B}_t(x, \xi) \cap \{0,1\}^{n_b} \tag{2.12d}$$

$$\mathcal{B}_t(\mathcal{R})(x) = \mathbb{E}\left[\, \hat{\mathcal{B}}_t(\mathcal{R})(x, \boldsymbol{\xi}) \,\right]. \tag{2.12e}$$

Conversely, we define *forward operators*, that determine the decision to make at stage t, given

the current state, noise realization and a future cost-to-go approximation:

$$\hat{\mathcal{F}}_t(\mathcal{R})(x,\xi) \in \arg\min_{z,y,b} L_t(x,y,b,\xi) + \mathcal{R}(z) \tag{2.13a}$$

$$z = F_t(x,y,b,\xi) \subset \mathfrak{X}_t \tag{2.13b}$$

$$y \in \mathfrak{Y}_t(x,\xi) \tag{2.13c}$$

$$b \in \mathfrak{B}_t(x,\xi) \cap \{0,1\}^{n_b}. \tag{2.13d}$$

Forward operators yield the induced *policy* of a sequence of approximated cost-to-go functions $\{\mathcal{R}_t\}_{t\in[T]}$, where a *policy* is a decision rule that provides the best *here-and-now* decisions x_t with estimated cost-to-go $\mathcal{R}_t$. In practice, at a given stage with current state x_{t-1}, we observe the realization of noise ξ_t, and we obtain the local optimal decisions to make and the new state x_t of the system by computing $\hat{\mathcal{F}}_t(\mathcal{R}_t)(x_{t-1},\xi_t)$. Then, we can obtain an infinite number of policies, depending on the approximations made. In particular, the optimal policy is obtained with the true cost-to-go functions V_t: $\hat{\mathcal{F}}_t(V_t)(x_t,\xi)$.

2.3.3 Dynamic Programming Algorithms

We now briefly present two algorithms relying on DP principles that solve stochastic multistage programs: Stochastic Dynamic Programming (SDP) and Stochastic Dual Dynamic Programming (SDDP). We elaborate on both algorithms in Chapters 3 and 5.

First, SDP is a flexible algorithm that needs few structural assumptions and solves a large class of problems, including mixed-integer or non-linear ones. The algorithm's core idea (see Algorithm 3) is to compute all cost-to-go functions recursively, starting from the final stage T, where $V_T \equiv K$ (we assume with $K = 0$ here for simplicity). Then, using Bellman's operators (2.12), we compute $\hat{V}_{T-1}(x,\xi) = \hat{\mathcal{B}}_t(V_T)(x,\xi)$ for each state x and noise realization $\xi \in \text{supp}(\boldsymbol{\xi_{T-1}})$. Note that to compute $\hat{V}_{T-1}(x,\xi)$, we need the value $V_T(y)$ for all potential outgoing state y. This requires discretizing the state space and interpolating points to derive $V_T(y)$ for any given y. Once we have $V_{T-1}(y)$, we repeat the process to compute $V_{T-2}(y)$, and so forth.

Algorithm 3: Stochastic Dynamic Programming

Data: Problem parameters, discretization grid $\mathbb{X}$, interpolator
Result: optimal strategy and value;

```
V_T ≡ 0 ; V_t ≡ 0;
for t : T − 1 → 0 do
    for x ∈ X_{t−1} do
        V_t(x) = B_t(V_{t+1})(x);        // Solve |supp(ξ_t)| problems (2.12)
    Extend the definition of V_t from the grid X_{t−1} to X_{t−1} by interpolation.
```

To solve Problem (2.8) using SDP, we have to solve $\prod_{t=1}^{T} |\mathbb{X}_t|.|\text{supp}(\boldsymbol{\xi_t})|$ optimization problems. This quickly becomes computationally intractable as the number of iterations grows exponentially with the state dimension. For example, in a production problem with J products, if we discretize each stock level (*resp.* noises) with only 10 (*resp.* 5) values, SDP solves $T \times 5 \times 10^J$ optimization problems. In practice, SDP can generally be considered for state dimensions up to 5. Further, as we need to discretize the state space and then interpolate the value function, it is challenging to maintain provable bounds and convergence guarantees.

In contrast, SDDP requires more structural assumptions than SDP but pushes beyond SDP's computational limits. Assuming the subproblems (2.10) are convex, $V_{[T]}$ can be approximated with linear cuts computed using duality theory. The crux of SDDP is to iteratively refine lower

bounds of each value function V_t at the "right places". More precisely, an SDDP iteration has two phases (see Algorithm 4): a *forward phase*, in which a system trajectory $x^k_{[T]}$ is sampled, using forward operators (2.13) based on the current cost-to-go approximations $V^{k-1}_{[T]}$; and a *backward phase* where these cost-to-go approximations $V^{k-1}_{[T]}$ are improved with new cuts computed at $x^k_{[T]}$. By convexity, these cuts give information everywhere, not just around $x^k_{[T]}$.

Finally, note that, for given discretization grids $\mathbb{X}_t$, SDP run for a fixed amount of time: we have no information if we stop early, and giving more time does not provide better results. On the other hand, SDDP progressively refine the cost-to-go approximations. Thus, whenever we stop the algorithm (between iterations), we have a valid lower bound and an induced policy. Giving more computation time provide a better lower bound.

Algorithm 4: Stochastic Dyal Dynamic Programming

$\underline{V}^0_t \equiv -\infty$ for $t \in [T]$, ;
for $k \in \mathbb{N}$ **do**
  `/* Forward phase: compute trajectory */`
  Set $x^k_0 = x_{init}$;
  for $t = 1 \to T-1$ **do**
    Randomly draw $\xi^k_t \in \text{supp}(\boldsymbol{\xi}_t)$;
    $x^k_t = \mathcal{F}_t(\underline{V}^{k-1}_{t+1})(x^k_{t-1}, \xi^k_t)$;
  `/* Backward phase: update approximations */`
  Set $\underline{V}^k_T \equiv 0$;
  for $t = T-1 \to 1$ **do**
    for $\xi \in \Xi_t$ **do**
      Solve $\dot{\mathcal{B}}_t(\underline{V}^k_{t+1})(x^k_{t-1}, \xi)$ for $\dot{\alpha}_\xi$ and $\dot{\beta}_\xi$;
    Compute $\alpha^k_t := \sum_{\xi \in \Xi_t} p_\xi \dot{\alpha}^k_{t,\xi}$ and, $\beta^k_t := \sum_{\xi \in \Xi_t} p_\xi \dot{\beta}^k_{t,\xi}$;
    Update $\underline{V}^k_t := \max\left(\underline{V}^{k-1}_t, \langle \alpha^k_t, \cdot \rangle + \beta^k_t\right)$;

SDDP is widely used to tackle real and large-scale industrial problems. For instance, it is implemented to manage the hydraulic system in Brazil. However, its convexity requirement disqualifies it from solving MSbLP, as convexity is lost with binary variables. The main purpose of this thesis is to explore new algorithms to solve MSbLP. Finally, we summarize in Table 2.1 the key assumptions and insights on computational performance are summarized of SDP and SDDP.

	DP		SDDP	
Independence assumption	Yes	👎	Yes	👎
Finitely supported noise	Yes	👎	Yes	👎
Structural assumptions	No	👍	Yes	👎
Discrete control	Yes	👍	No	👎
State discretization	Yes	👎	No	👍
Progressive results	No	👎	Yes	👍
Maximum state dimension	≈ 5	👎	≈ 30	👍

Table 2.1: Summary of SDP and SDDP's assumptions and computational limits

2.4 Contributions

This section gives an overview of the main contributions and results of this thesis, along with its layout.

Contribution 4 (Joint production and energy planning problem)**.** *We model a joint production and energy planning problem as an MSbLP. This model accounts for uncertainties, industrial constraints (in particular shared-resource constraints) modeled with binary variables, and "investment" variables, such as day-ahead purchases. We provide a heuristic method to solve an MSbLP that constructs a policy by solving $t-$multistage stochastic programs with final cost-to-go functions being approximated by SDDP cuts.*

We emphasized in Section 2.1.2 the rising need to solve industrial problems with an insight on energy procurement. This motivated the project presented in Chapter 3, which led to a publication [FLG24] in Energy Systems. In Chapter 3, after discussing multiple solution strategies to solve MSbLP based on known methodologies (such as Model Predictive Control (MPC), Stochastic Dynamic Programming (SDP) and SDDP), we propose heuristic methods that use the approximated cost-to-go functions given by SDDP. More precisely, we propose a variant of bellman forward operators (2.13), that instead of solving a $1-$stage problem at stage τ with approximations $\mathcal{R}_{\tau+1}$, solves a $t-$stage problem with approximations $\mathcal{R}_{\tau+t}$. Further, the cost-to-go approximations that we use are computed by SDDP's cuts, which we can obtain in a reasonable time beforehand.

In our tests on a METRON use case, we found that the operational problem (excluding investment variables) is best solved using MPC. Indeed, in this specific setting, addressing binary variables is more critical than dealing with uncertainties. However, with the addition of strategic variables, it seems that we should use SDDP to fix the first stage variables (that impact significantly operational costs) and then apply MPC to solve the resulting parameterized operational problem.

Contribution 5 (A B&B framework for solving MSbLP)**.** *We provide a Branch-and-Bound (BB) framework that relies on the underlying structure of scenario trees to solve MSbLP, leveraging the relaxed approximations obtained with SDDP. From this framework, we develop an exact algorithm, Algorithm 12, to solve MSbLP that relies on branching strategies to grow a subtree of the scenario tree.*

The focus of Chapter 4 is to provide an abstract framework to address MSbLP through BB methodology. Thus, we start by giving a formal definition of a scenario tree as well as introduce various subtrees' structures. Then, we introduce *assignation functions*, which fix, keep or relax the binary variables, thus modifying the feasible set of our problem. These assignation functions are the fundamental object on which we apply BB. We finally discuss conditions on the assignation functions under which we can leverage SDDP to solve some (part of) the problem. Those conditions lead us to an efficient BB algorithm described in Algorithm 12.

More specifically, considering particular subtrees that are constructed by pruning branches of the scenario tree, we construct a partial relaxation of Problem (2.8) by relaxing integrality on the pruned portions of the scenario tree. Starting from an empty subtree, Algorithm 12 iteratively grows the subtree with BB methods, and converges to the true value of Problem (2.8). We present multiple approaches (layer by layer, randomly, through integrality gap or strong branching) to grow the subtree efficiently. In particular, this framework generalizes the heuristic approach proposed in Contribution 4.

Contribution 6 (Modeling fairness)**.** *We present a framework and tools to accommodate fairness into mathematical models, in particular in the context of prosumer aggregation. In our framework,*

we advocate for fairness-by-design*, where we establish a degree of fairness directly within the model instead of relying on ex post redistribution, as would be usual in game theory. More specifically, we suggest to take fairness into account by i) choosing the way in which we aggregate the agents' cost, ii) enforcing acceptability constraints which guarantee that each agent is better off in the coalition than alone. Moreover, we discuss extensions to the dynamic and stochastic cases.*

As discussed in Section 2.2.4, we may need to aggregate multiple factories with diverging interests. This raises the question of how to guarantee fair treatment among them. We begin, in Chapter 6, by providing a global perspective on fairness, a concept that seems intuitive but is challenging to model. From conceptual discussions to practical applications, we present different definitions of fairness and their translation into mathematical models. We emphasize the importance of context-specific modeling and analysis when choosing a fairness approach.

From this discussion, we propose in Chapter 7 a fairness-by-design framework that can be used for energy aggregators. Specifically, we present aggregation operators to balance efficiency (cost optimization) with fairness (allocation optimization) through the objective function. We further introduce *acceptability* constraints to ensure that each agent has an interest in participating in the aggregation. This is modeled by ensuring that its cost in the aggregation is lower than by itself. However, in dynamic and/or stochastic settings, the cost is no longer a single value but an element of a larger vector space. Thus, the comparison between both costs is done according to some specific ordering. We discuss various options for the (long-term) dynamic setting – to ensure that an agent has no interest to leave the aggregation before the end; and for the stochastic setting – using stochastic order theory.

This project led to a preprint [FLP24] and is now under review in Computational Management Science.

Chapter 3

Joint production and energy supply planning of an industrial microgrid

Contents

In this chapter, we propose to model a joint production and energy supply planning problem as a Multistage Stochastic mixed-binary Linear Program (MSbLP). After discussing various known methodologies to solve the problem, we propose heuristic methods that leverage Stochastic Dual Dynamic Programming (SDDP) cost-to-go approximations. The content of this chapter was published in Energy Systems [FLG24].

3.1 Introduction

The latest Intergovernmental Panel on Climate Change (IPCC) warns us yet again about the consequences of climate change and incites governments, industries and citizens to change accordingly. The COP27, held in November 2022, set up a clear objective of securing global net-zero emissions by mid-century. Therefore, the industry, counting for one-fourth of global emissions (6th IPCC report), must take strong actions to reduce them. In this respect, the Clean Energy Ministerial Industrial Deep Decarbonisation Initiative (IDDI) calls for a change in the energy supply, as industries consume fuel massively to produce local energy, especially steel and cement production. To put things in perspective, renewable generation represented only 16.9% of electricity generation in the industrial sector in 2020, which is far less than its share in global electricity generation, up to 28% in 2020, according to the International Energy Agency (IEA), see their Tracking Industry 2021 report [Intc] and their Global Energy Review 2021 report [Inta]. For instance, microgrids are an alternative energy supply model. They are defined (see *e.g.,* [HPG18]) as a small-scale power grid that can operate independently or collaboratively with the power grid. Generally, they are made of Energy Storage System (ESS), renewable energy generation units (wind turbines, solar panels) and consumption units (factories, buildings, etc.). With recent technological advances, such energy systems are becoming more efficient and cheaper to install and operate. Moreover, some governments subsidize energy transition efforts, which encourages factories to invest in onsite renewable energy. For instance, The Fairfield, California brewery[1] has invested in a solar array and wind turbine which provide an average of 30% of its electricity needs. Another example is the French company E.Leclerc which equips some of its hypermarkets with solar generation.

In a recent review of energy sustainability in manufacturing systems [RM21], the authors point out that, in most papers, the problem of energy management is decoupled from manufacturing operations. However, they argue that this decoupling is not realistic as the two problems are interdependent, and suggest that research should be conducted on solving those problems jointly. In this paper, we address this issue by proposing a joint production and energy supply planning problem.

Unfortunately, incorporating renewable energies in the supply mix is challenging as they are intermittent, unpredictable and uncontrollable. To counteract these defects it is often suggested to add an ESS (we refer to [Geo+21] for an overview of the available ESS). Doing so allows transferring energy across time steps, making it controllable and compensating for intermittency. Unpredictability of the renewable production requires going from a deterministic formulation to a stochastic formulation. Indeed, a classical deterministic problem is often misleading and optimistic about the potential of the ESS. Unfortunately, multistage stochastic problems are known to be numerically challenging (see *e.g.,* [Sha06]). Starting from a standard scheduling industrial problem, we consider relying on an onsite microgrid to provide an alternative energy supply to the main grid. We obtain a mixed-integer multistage stochastic problem optimizing jointly the production planning and the energy supply management of an industrial facility with in advance and intraday energy purchases.

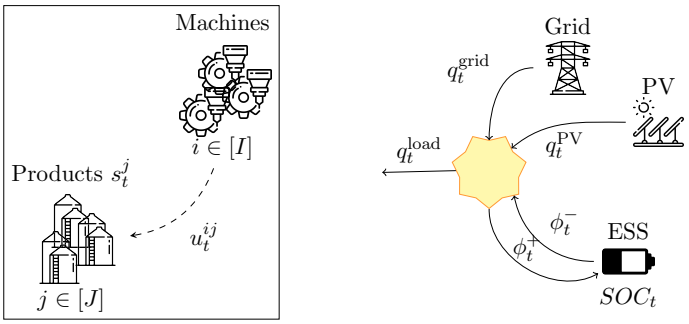
3.1.1 The industrial microgrid management problem

Building renewable energy production and storage management systems to supply an industrial facility is a complex task. One of the questions at hand is the financial rentability of such a system, which is not guaranteed. To incite the industry to invest in renewable energies, we need

[1] https://www.anheuser-busch.com/breweries/fairfield-ca

economic guarantees: see [Intb] and [Intd] for an overview of clean energy transition costs in 2021. The economic viability of a microgrid is based on controlling the investment costs and managing the microgrid efficiently. In this paper, we do not discuss the investment part but focus on the operational part.

We consider a facility with I machines that produce up to J types of products that can be stored (see Figure 3.1a). Our goal is to provide the facility with a joint production and energy supply planning, on a discrete horizon $t \in [T]$. The planning should minimize the total expected cost (economic, environmental and labor) while satisfying production targets and technical constraints.



(a) Factory Structure

(b) Energy exchanges

Figure 3.1: Industrial Management Problem

Depending on the facility at hand, many technical constraints need to be satisfied. We can classify them into three types. First, *physical constraints* are induced by the machines at hand. For example, most machines, such as grinders or plastic extruders, require warming up before being operational. Another straightforward example comes from the food industry, where machines need to be cleaned up to reconfigure the production line. Second, *process constraints* which correspond to precedence constraints mandating sequential execution of some tasks (usually called flow-shop problems). For instance, in a chocolate factory, every batch production will follow in order: cleaning, roasting, shell removing, grinding and conching. Finally, *implied constraints* model decision-maker preferences or human resources constraints. For example, the decision maker may limit the number of re-starts to limit wear-off, if a machine is hard to access or for human power reasons.

Most of the above constraints are modeled with binary variables. Thus, even though we focus here on a specific problem, the developed approach can be transposed to a large variety of problems. In this paper, we consider a problem with bounded production and set-up costs. In addition, we consider shared resource constraints such that some products cannot be produced simultaneously. Factory energy needs, proportional to production, are met with electricity from a main grid or produced onsite by a micro-grid consisting of solar panels coupled with an ESS see Figure 3.1b.

Electricity from the main grid can be purchased through two different contracts, usually cumulated: Intra-day contract where prices are fixed annually, the factory pays the energy extracted from the main grid at a given time t; In-advance contract where the factory buys energy blocks in advance (*e.g.,* a day ahead of production) at a preferential rate. Decisions are made adjusting energy purchases based on intra-day rates in real-time.

3.1.2 Literature review

We consider a problem coupling production planning and energy supply management. Taken separately, each problem has been widely studied, but considering them simultaneously is less common, especially when taking into account uncertainty, leading to large multistage stochastic optimization problem. In this section, we review the state-of-the-art of energy-aware production planning under uncertainties.

A typical angle for energy-aware production systems is to minimize energy waste, see the reviews [Bän+21], [BG16] and references therein. This part of the literature looks for production plans, or scheduling, that are more energy efficient, adapting tools from well-studied problems like single or parallel machine scheduling, job-shop, flow-shop or lot-sizing [2]. However, few papers discuss the economic impact of integrating renewable energy sources on site: indeed, the industrial energy supply is traditionally guaranteed by an external grid. In their survey [Bän+21], Bänsch et al. count 8 articles (out of 192) that consider an onsite energy generation and an ESS. The literature lacks research on industrial problems with distributed generation systems, though, they are widely studied on their own. We refer to the review [Alo+22] where Alonso-Travesset et al. focuses on recent studies on models under uncertainties in distributed generation systems. They highlight the necessity of properly taking into account uncertainties in those problems, in particular regarding renewable energy generation. In the problem considered here, the main source of uncertainty comes from renewable energies. There are two main ways of handling uncertainty: stochastic optimization and robust optimization.

3.1.2.1 Stochastic optimization for operational management

In the first paradigm, we model uncertain variables as random variables with known distribution, usually represented by a scenario tree. Further, as uncertainties are revealed step by step, stochastic problems are often multistage problems that are known to be challenging, while there exist various methods to tackle 2−stage problems *e.g.,* based on Bender's decomposition (see [BL97]). As a result, multistage problems are classically relaxed into 2−stage problems: all decision variables, except the first stage variable, are assumed to be taken with the full knowledge of the uncertainty. This is the strategy adopted by Golari, Fan, and Jin in [GFJ16] to optimize the production planning of interconnected factories each connected to a micro-grid. Biel et al. take this approach as well in [Bie+18] to solve a flow-shop problem under uncertainties regarding wind energy generation. In another article ([WMG20]), Wang, Mason, and Gangammanavar studies a similar problem with multi-objectives (total completion time and energy costs), where selling an energy excess to the main grid is allowed. They propose an ε-constraint algorithm integrated with the L-shaped method ([Bir85a]), which is a Benders decomposition adapted to 2−stage stochastic programs. To avoid 2−stage approximations, one can turn to dynamic programming reformulations of the multi-stage stochastic problem. However, vanilla dynamic programming for multistage problems is limited by what is known as the *curse of dimensionality*. In 1991, Pereira and Pinto proposed an efficient algorithm to solve those problems: the Stochastic Dual Dynamic Programming (SDDP) algorithm [PP91]. Since then, SDDP has been widely applied to energy management problems and variants have been derived (see [FR21] for a recent survey). We recall the algorithm and present related literature in Section 3.3.2.

[2] The job-shop problem, see *e.g.,* [Man60], looks for an optimal scheduling plan for n jobs, consisting of operations with precedence constraints, on m machines. The flow-shop problem is a variant of the job-shop problem with a strict order of all operations on all jobs. Finally, a lot-sizing problem optimizes the production quantities of each item at each time step.

3.1.2.2 Robust optimization for operational management

In the second paradigm, robust optimization, we consider the worst case in possible uncertainty realizations. This is the choice made by Ruiz Duarte, Fan, and Jin in [RFJ20], where they evaluate the renewable energy integration with an ESS in a factory while optimizing the production planning. This is modeled by a 2−stage problem: in the first stage, a production plan is defined whereas in the second stage, the decisions regarding the energy management system are made to minimize its energy costs under the worst-case energy generation scenario. The robust uncertainty set is determined by statistical tools. Bridging both worlds, Shahandeh, Motamed Nasab, and Li propose in [SML19] to divide random variables into two categories: static and dynamic variables. The idea is to apply robust optimization on one variable category and then stochastic optimization on the other, considering a scenario tree. This results in two hybrid algorithms, mixing robust and stochastic optimization to solve a multistage problem with different uncertainty types.

3.1.2.3 Price and demand uncertainties

Furthermore, in these industrial problems, the solution is not only affected by renewable energies' variability: costs and demands are other known uncertainty sources. If some articles consider time-of-use (TOU) electricity rates ([Bie+18], [MP13], [Li+17] and [WMG20]), which are fixed prices in contract depending on consumption's times, others consider variable prices. In that respect, Bohlayer et al. ([Boh+20]) and Ierapetritou et al. ([Ier+02]) both study mixed-integer multistage stochastic problems under energy prices uncertainty. See also Fazli Khalaf and Wang ([FW18]) who solve a 2−stage stochastic scheduling problem considering both electricity prices and energy generation as random variables. Finally, in lot-sizing problems, the product demand is often random: Higle and Kempf consider a multistage stochastic program in [HK10] to solve a production planning problem under demand uncertainty, trying to avoid cumulating stocks.

3.1.3 Strategic decisions

We have covered stochastic considerations for operational or tactical production planning problems. We now discuss strategic decisions like investing in renewable energies and ESS, with questions of size, technologies and number of ESS and energy generation units. To adapt their energy mix, factories need to design what distributed generation system is suited for their production. In [FMH21], Fattahi, Mosadegh, and Hasani focus on the planning in mining supply chains with renewable energy investment where at each stage, warehouse or generation systems can be installed. Though economic rentability is crucial, the growing interest in microgrids is driven by environmental concerns. Thus, instead of minimizing energy waste, a more direct approach consists of integrating environmental objectives into costs. For example Li et al., in [Li+17], assess wind and solar generation deployment costs in order to achieve net-zero carbon. On another note, microgrids bring flexibility and energy independence. In [Pha+19], Pham et al. extend Golari, Fan, and Jin's work by considering both stochastic demand and microgrid sizing. Their goal is to determine if it is economically viable to provide the system with only renewable energies.

Investing in microgrids doesn't require only sizing but also investigating the different existing technologies and their characteristics. In [Tsi+21], Tsianikas et al. study the capacity extension problem as well as the different storage technologies. An interesting take on the subject is given in [HBF15]: when most micro-grid investment models consider the ESS sizing at the beginning, Hajipour, Bozorg, and Fotuhi-Firuzabad proposes to extend the storage capacity and invest in renewable generation units at different times, leading to a multistage stochastic problem. This model allows life-cycle constraints or decreasing technology efficiency to have an impact on

results.

3.1.4 Contributions

Our contribution in this paper lies in four aspects. First, we propose an optimization model for a coupled management problem with both production and energy supply planning. We take into account the multistage structure of the problem, the uncertainties due to onsite renewable energy generation and binary variables modeling physical production constraints. In particular, we model shared resource constraints: a choice has to be made between different products at each time. Therefore, it is crucial, when reducing the problem to stage t with dynamic programming, to have visibility on the consequences of choosing a product at t. Second, we consider both on-demand supply with TOU pricing and in-advance energy purchasing. The latest brings complexity to the multistage problem with first-stage variables impacting the whole horizon costs. Third, we discuss multiple solution strategies based on well-known and new methodologies: a deterministic approach known as Model Predictive Control (MPC); Stochastic Dynamic Programming (SDP); and an approach solving linear multistage stochastic problems, SDDP. Finally, as there does not exist an efficient algorithm to solve large mixed-integer multistage stochastic problems, we propose heuristic methods relying on the approximated cost-to-go function given by SDDP. We highlight the theoretical and practical limits of these solution strategies on numerical examples.

The remainder of the paper is laid out as follows. Section 3.2 introduces the problem formulation. We present in Section 3.3 dynamic programming methods to solve multistage mixed-integer stochastic problems. Those methods being unsatisfactory for the problem at hand, we then proceed to detail different heuristics in Section 3.4. Finally, Section 3.5 presents numerical results.

3.1.5 Notations

To facilitate understanding, we go through some notation used in this paper. We denote $[a : b] := \{a, \dots, b\}$ the set of integers between a and b, and $[T] := [1 : T]$ the set of non-null integers smaller than T. Accordingly, $X_{[n]}$ denote the collection $X_{[n]} := \{X_i\}_{i\in[n]}$. Generally speaking, we denote the state variables x, the control variables u and the noise ξ. All random variables are in bold characters, further if $\boldsymbol{\xi}$ is a random variable then ξ denotes a realization of this variable. Finally, $\sigma(\xi_{[t]})$ represents the $\sigma-$algebra generated by $\{\xi_\tau\}_{\tau\in[t]}$.

3.2 A multistage stochastic formulation for joint production and energy planning

In this section, we present the mathematical formulation of our problem, presented in Section 3.1.1. We first focus on the operational problem: daily operations the factory has to make. Note that, though we consider a specific production problem constructed from a practical industrial application, the proposed numerical approaches detailed in Sections 3.3 and 3.4 can be adapted to other production problems.

We consider a factory owning solar panels and a battery. Thus, the energy supply is a mix of solar energy available $\boldsymbol{q_t^{\mathrm{PV}}}$ (modeled as random variables), of charge $\boldsymbol{\phi_t^+}$ and discharge $\boldsymbol{\phi_t^-}$ from the battery, and of energy bought from the main grid $\boldsymbol{q_t^{\mathrm{grid}}}$. Energy can be either bought in advance (*e.g.,* on a day-ahead market) or in real-time through industrial contracts with fixed prices. We decompose the energy bought from the grid $\boldsymbol{q_t^{\mathrm{grid}}}$ into energy bought in advance v_t^{DA}, considered for now as a given parameter, plus energy bought during the day $\boldsymbol{v_t^{\mathrm{ID}}}$. With these

elements, we need to ensure that the energy supply exceeds the energy demand $\boldsymbol{q_t^{\text{load}}}$, leading to the following control constraints.

$$\boldsymbol{q_t^{\text{PV}}} + \boldsymbol{\phi_t^-} - \boldsymbol{\phi_t^+} + \boldsymbol{q_t^{\text{grid}}} \geq \boldsymbol{q_t^{\text{load}}} \qquad \forall t \in [T], \tag{3.1a}$$

$$0 \leq \boldsymbol{\phi_t^+} \leq \phi_{max}^+ \qquad \forall t \in [T], \tag{3.1b}$$

$$0 \leq \boldsymbol{\phi_t^-} \leq \phi_{max}^- \qquad \forall t \in [T], \tag{3.1c}$$

$$\boldsymbol{q_t^{\text{grid}}} = v_t^{\text{DA}} + \boldsymbol{v_t^{\text{ID}}} \qquad \forall t \in [T], \tag{3.1d}$$

$$v_t^{\text{DA}}, \boldsymbol{v_t^{\text{ID}}} \geq 0 \qquad \forall t \in [T]. \tag{3.1e}$$

The energy demand $\boldsymbol{q_t^{\text{load}}}$ is shaped by the factory's production, derived from the quantities $(\boldsymbol{u_t^{ij}})_{t,i,j}$ of product j produced on machine i at time t. We also introduce binary variables, $(\boldsymbol{b_t^{ij}})_{t,i,j}$, assigning product j to machine i at time t, leading to the following set of constraint.

$$\sum_j \boldsymbol{b_t^{ij}} \leq 1 \qquad \forall i, t, \tag{3.1f}$$

$$\max_i \boldsymbol{b_t^{ij}} + \max_i \boldsymbol{b_t^{ij'}} \leq 1 \qquad \forall t, \forall (j, j') \in \mathcal{I}, \tag{3.1g}$$

$$u_t^{imin} \boldsymbol{b_t^{ij}} \leq \boldsymbol{u_t^{ij}} \leq u_t^{imax} \boldsymbol{b_t^{ij}} \qquad \forall i, j, t, \tag{3.1h}$$

$$\boldsymbol{q_t^{\text{load}}} = g(\boldsymbol{u_t^{ij}}, \boldsymbol{b_t^{ij}}) \qquad \forall t, \tag{3.1i}$$

$$\boldsymbol{b_t^{ij}} \in \{0, 1\} \qquad \forall i, j, t. \tag{3.1j}$$

At each time t, one machine can be assigned to one product at most (3.1f). $\mathcal{I}$ is the set of incompatible products, meaning a couple of products (j, j') belongs to $\mathcal{I}$ if they share resources. Therefore, they cannot be produced simultaneously (3.1g). Furthermore, production is bounded by machine capacities (3.1h). Finally, the function g gives the energy load induced by energy production (3.1i), we assume g linear.

Hence, the state of the system is described by the products and battery stocks. The stocks of products are modeled with state variables $(\boldsymbol{s_t^j})_{t,j}$. The demand at time t is modeled as a deterministic vector $(d_t^j)_{j \in [J]}$. Initial stocks are empty. Then the stock variables follow dynamic equations and bounding constraints given by

$$\boldsymbol{s_t^j} = \boldsymbol{s_{t-1}^j} - d_t^j + \sum_i \boldsymbol{u_t^{ij}} \qquad \forall t, j, \tag{3.2a}$$

$$\boldsymbol{s_t^j} \geq 0 \qquad \forall t, j, \tag{3.2b}$$

$$s_0^j = 0. \tag{3.2c}$$

Indeed, for each time t and product j, the factory has to satisfy a demand d_t^j, which is ensured by the positivity of stocks requirement (see Equation (3.2b)). Further, the quantity of energy stored in the battery, $(SOC_t)_t$, is also modeled as a state variable:

$$\boldsymbol{SOC_t} = \boldsymbol{SOC_{t-1}} - \frac{1}{\rho}\boldsymbol{\phi_t^-} + \rho\boldsymbol{\phi_t^+} \qquad \forall t, \tag{3.2d}$$

$$SOC_{min} \leq \boldsymbol{SOC_t} \leq SOC_{max} \qquad \forall t. \tag{3.2e}$$

For notational conciseness, we reduce control variables to vector $\boldsymbol{y_t} := (\boldsymbol{b_t}, \boldsymbol{u_t}, \boldsymbol{q_t^{\text{grid}}}, \boldsymbol{\phi_t^+}, \boldsymbol{\phi_t^-})$ and state variables to vector $\boldsymbol{x_t} := (\boldsymbol{s_t}, \boldsymbol{SOC_t})$. Finally, the objective is to minimize total energy purchases *i.e.,* intra-day energy purchases v^{ID}.

$$V(x_0; v^{\text{DA}}) := \min_{\boldsymbol{y_{[T]}}, \boldsymbol{x_{[T]}}} \quad \mathbb{E}\big[\sum_{t=1}^{T} p_t^{\text{ID}} \boldsymbol{v_t^{\text{ID}}}\big] \tag{3.3a}$$

$$s.t. \quad \text{Equations (3.1) and (3.2)}, \tag{3.3b}$$

$$\sigma(\boldsymbol{y_t}) \subset \sigma(\boldsymbol{q_{[t]}^{\text{PV}}}) \qquad \forall t \in [T]. \tag{3.3c}$$

The last constraint Equation (3.3c), commonly known as non-anticipativity constraint, represents the information available when taking decision $\boldsymbol{y_t}$ at t. In particular, in this framework, we observe the random variable q_t^{PV} realization, before making decisions $\boldsymbol{y_t}$, with no knowledge of future random realizations from $t+1$ to T.

We now consider the strategic problem of choosing the best v^{DA} that minimizes day-ahead costs plus operational costs $V(x_0, v^{\text{DA}})$. This can be done by introducing an initial time step $t = 0$ where such strategic variables are decided. This amounts to solving problem 3.4.

$$V(x_0) := \min_{v_t^{\text{DA}} \geq 0} \quad \sum_{t=1}^{T} p_t^{\text{DA}} v_t^{\text{DA}} + V(x_0; v^{\text{DA}}) \tag{3.4}$$

In the next section, we present solution methods for this problem based on Dynamic Programming.

3.3 Dynamic Programming approaches

Assuming that the noises are finitely supported, a multistage stochastic problem like Problem 3.3 can always be cast as a large-scale deterministic problem (see *e.g.,* [BL97]). However, the size of these deterministic equivalents is linear in the number of scenarios, which is often exponential in the horizon. A solution consists in compressing the information required to make a decision. To this end, we make a crucial stagewise independence assumption and turn to Dynamic Programming (DP) tools, presented here.

3.3.1 Stochastic Dynamic Programming

We consider a *controlled dynamic system*, that is a sequence of random vector $\boldsymbol{x}_{[T]}$ that follows a dynamic, affected by a sequence of noises $\boldsymbol{\xi}_{[T]}$. Those random vectors describe the state of the system across time, here product stocks $\boldsymbol{s_t}$ and battery energy level $\boldsymbol{SOC_t}$. Each noise ξ_t takes value in a finite set Ξ_t, and we denote $\Omega := \prod_{t\in[T]} \Xi_t$. We assume that these noises represent all the uncertainty in the problem at hand (here solar energy), with known probability distribution,

resulting in a probability space $(\Omega, \mathcal{A}, \mathbb{P})$. We call *scenario* a sequence $\xi_{[T]}$ of realization of the noise at each time step.

We consider the Problem 3.3, parametrized by v^{DA} and restrained to sub-horizon $[t:T]$ from initial state x_{t-1}, and we denote its expected optimal value $V_t(x_{t-1}; v^{\mathrm{DA}})$. Then, with the stage-wise independence assumption, the DP principle ensures that the value functions follow the following recursive equations:

$$\hat{V}_t(x, \xi; v^{\mathrm{DA}}) = \min_{\substack{y_t \in \mathcal{Y}_t(x,\xi) \\ x_t \in \mathcal{X}_t(x, y_t, \xi)}} \underbrace{p_t^{\mathrm{ID}} \boldsymbol{v}_{\boldsymbol{t}}^{\mathrm{ID}}}_{\text{instantaneous cost}} + \underbrace{V_{t+1}(x_t; v^{\mathrm{DA}})}_{\text{cost-to-go}} \tag{3.5a}$$

$$V_t(x; v^{\mathrm{DA}}) = \mathbb{E}\left[\hat{V}_t(x, \boldsymbol{q}_{\boldsymbol{t}}^{\mathrm{PV}}; v^{\mathrm{DA}})\right], \tag{3.5b}$$

$$V_{T+1}(x; v^{\mathrm{DA}}) = 0. \tag{3.5c}$$

For notational conciseness, we denote $\mathcal{Y}_t(x, \xi, v^{\mathrm{DA}})$ the feasible control set representing constraints Equation (3.1) depending on current state x, noise realization ξ, and strategic variables v^{DA}. Similarly, we denote $\mathcal{X}_t(x, u, \xi)$ the state set representing dynamics Equation (3.2) depending on previous state x, control u and noise ξ.

However, for any $x \in \mathcal{X}_{t-1}$, computing $V_t(x; v^{\mathrm{DA}})$ requires full knowledge of V_{t+1}. With continuous state, it is usually impossible. Therefore, to accommodate for inexact value functions, we introduce the bellman operators which generalize Equations (3.5) so that the dynamic equations hold for any given function R approximating the cost-to-go V_{t+1}. The backward operator $\mathcal{B}_t$, defined in Equation (3.6a),

$$\text{Backward operators} \quad \begin{cases} \hat{\mathcal{B}}_t(R) : x, \xi \mapsto \min\limits_{\substack{y_t \in \mathcal{Y}_t(x,\xi) \\ x_t \in \mathcal{X}_t(x, y_t, \xi)}} p_t^{\mathrm{ID}} \boldsymbol{v}_{\boldsymbol{t}}^{\mathrm{ID}} + R(x_t), \\ \mathcal{B}_t(R) : x \mapsto \mathbb{E}\left[\hat{\mathcal{B}}_t(R)(x, \boldsymbol{q}_{\boldsymbol{t}}^{\mathrm{PV}})\right], \end{cases} \tag{3.6a}$$

returns an approximation, at a given state x, of the cost-to-go V_t starting from time t, given an approximation R of the cost-to-go starting from time $t+1$. Thus, given a discretization of each state space X_t, and an interpolation method we can, recursively, compute an approximation of every cost-to-go function see Algorithm 5.

We then define in Equation (3.6b) the forward operator, which returns the optimal next state x_t, given a starting state x, a noise ξ and an approximation R of the cost-to-go from $t+1$. Note that, in practice, computing $\hat{\mathcal{B}}_t(R)(x, \xi)$ or $\hat{\mathcal{F}}_t(R)(x, \xi)$ consists in solving the same deterministic problem. Nevertheless, if the backward operator is well-defined, the forward operator requires a choice if the optimal solution is not unique. To be completely rigorous, we should say that a forward operator defines a selection of the optimal solution set.

$$\hat{\mathcal{F}}_t(R) : x, \xi \mapsto y_t^\star, x_t^\star \in \operatorname*{arg\,min}_{\substack{y_t \in \mathcal{Y}_t(x,\xi) \\ x_t \in \mathcal{X}_t(x, y_t, \xi)}} p_t^{\mathrm{ID}} \boldsymbol{v}_{\boldsymbol{t}}^{\mathrm{ID}} + R(x_t) \tag{3.6b}$$

DP is a powerful tool as the multistage problem considers $(|\Xi|^T)$ scenarios and turns the exponential complexity in the horizon T into a linear one. However, it is limited by what is

Algorithm 5: Stochastic Dynamic Programming

Input : x_0, discretization grids $\mathcal{X}_t^D$, interpolation operator.
Output : approximated value function $\tilde{V}_t$
$\tilde{V}_{T+1} = 0$.
for $t : T \to 1$ **do**
  for $x_{t-1}^D \in \mathcal{X}_{t-1}^D$ **do**
    // We discretize $\mathcal{X}_t$
    for $\xi_t \in \Xi_t$ **do**
      Solve the one-stage deterministic optimization problem:
        $\tilde{V}_t(x_{t-1}^{\mathrm{D}}, \xi_t; v^{\mathrm{DA}}) = \hat{\mathcal{B}}_t(\tilde{V}_{t+1})(x_{t-1}^{\mathrm{D}}, \xi_t; v^{\mathrm{DA}})$.
    $\tilde{V}_t(x_{t-1}^{\mathrm{D}}; v^{\mathrm{DA}}) = \sum_{\xi_t \in \Xi_t} \pi_{\xi_t} \tilde{V}_t(x_{t-1}^{\mathrm{D}}, \xi_t; v^{\mathrm{DA}})$; // expected value
  Define $\tilde{V}_t$ for any $x \in \mathcal{X}_{t-1}$ by interpolation on $\left\{(x_{t-1}^D, \tilde{V}_t(x_{t-1}^D; v^{\mathrm{DA}}))\right\}_{x_{t-1}^D \in \mathcal{X}_{t-1}^{\mathrm{D}}}$.

known as the *curse of dimensionality*. Indeed, we have to solve, for each time step, $|\mathcal{X}_t|.|\Xi|$ problem. A discretization of $\mathcal{X}_t$ usually requires a number of points exponential in the dimension of $\mathcal{X}_t$. Thus, in practice, DP cannot be used for states with more than 5 dimensions.

Remark 5. *In order to represent the strategic problem as a stochastic dynamic system, we need consider an extended state (x_t, v^{DA}), where v^{DA} is decided at $t = 0$ and then carried on, as part of the state, from stage to stage by the dynamics of the system. This extension increases the dimension of the system from $J+1$ to $J+1+T$, making algorithm 5 computationally intractable as typical T is at least* 24.

When $J \leq 3$ algorithm 5 can be reasonably used to address Problem 3.3. However, computation time is still high as we solve $\mathcal{O}(T.|\mathcal{X}_t^D|.|\Xi|)$ MILP. Thus, we present another algorithm, exploiting sampling methods, in the next section.

3.3.2 Stochastic Dual Dynamic Programming (SDDP)

To counteract the DP computational issues, a class of *Trajectory Following Dynamic Programming* (TFDP) algorithms (see [FL23] for recent overviews) has been developed. The crux of these algorithms is to iterate between forward phases that compute state trajectories, and backward phases that improve cost-to-go estimations. More specifically, in the forward phase of a TFDP algorithm, a state trajectory is computed using the current cost-to-go estimations. Then in a backward phase, the cost-to-go estimations are refined around the state trajectory computed in the forward phase. These approximations are given as the maximum of elementary functions called *cuts*.

For linear multistage stochastic problems with stagewise independence, the Stochastic Dual Dynamic Programming (SDDP) algorithm [PP91] has proven to be an efficient tool, widely used in the energy community in particular for long-term hydro-management. It is the most well-known and studied example of TFDP algorithm, relying on Benders' cut obtained through linear programming duality, assuming the problem is convex and continuous. In line with SDDP, the Stochastic Dual Dynamic integer Programming (SDDiP), [ZAS19], assumes that all state variables are binary, that there exists some continuous recourse ensuring relatively complete recourse assumption, and derives specific linear cuts. As one can always represent bounded integer variables, and approximate continuous variables, through binaries, the algorithm is theoretically

applicable for a large number of settings, including ours, but is limited in practice as each step requires solving a MILP, and as the convergence is generally slow.

Another TFDP algorithm, the Mixed Integer Dynamic Approximation Scheme (MIDAS) (see [PWB20]) assumes the monotonicity of the cost-to-go functions and uses piecewise constant cuts to approximate them. Finally, the Stochastic Lipschitz Dynamic Programming (SLDP) (see [ACF22]), simply assumes Lipschitz regularity of the cost-to-go functions and uses reverse norm cuts. SDDiP, MIDAS and SLDP might be applicable to the industrial microgrid setting but are generally slow to converge without additional, problem-specific, cuts. However, the subject is an active field of research: variants and enhancements of those algorithms are frequently published (*e.g.,* [FR22; QGK23]). Unfortunately, to the best of our knowledge, there is no off-the-shelf implementation of efficient TFDP algorithms for mixed-integer stochastic programs.

Therefore, we consider the continuous relaxation of Problem 3.3, and adapt the tools in Section 3.3.1 for the continuous relaxation, using exponent r to indicate the problem at hand is relaxed. It is the same problem as Problem (3.4) but we assume all binary variables are in $[0,1]$ instead of $\{0,1\}$, represented by $\boldsymbol{y_t^r} \in \mathcal{Y}_t^r(\boldsymbol{x_{t-1}}, \boldsymbol{q_t^{\text{PV}}}; v^{\text{DA}})$.

Algorithm 6: Stochastic Dual Dynamic Programming

```
// Initialization
k = 0, V_t^{r,0} = LB, v^DA.
for k : 0, ... do
    Simulate a scenario {ξ_t^k}_{t∈[T]}.
    // Forward phase
    x_0^k = x_0.
    for t : 1 → T do
        y_t^k, x_t^k = F̂_t^r(V_t^{r,k})(x_{t-1}^k, ξ_t^k; v^DA).
    // Backward phase
    V_{T+1}^{r,k} = 0
    for t : T → 1 do
        // Cut computation
        for ξ realization of ξ_t do
            Solve B̂_t^r(V_{t+1}^{r,k})(x_{t-1}^k, ξ; v^DA) and obtain coefficients α̂_t^k(ξ) and β̂_t^k(ξ) such that:
                α̂_t^k(ξ)^T x + β̂_t^k(ξ) ≤ B̂_t^r(V_{t+1}^{r,k})(x, ξ; v^DA)   ∀x.
        Define α_t^k = E[α̂_t^k(ξ_t)] and β_t^k = E[β̂_t^k(ξ_t)].
        Define V_t^{r,k} : x ↦ max_{κ≤k}(α_t^{κT} x + β_t^κ).
```

Leveraging the convexity of the relaxed Problem 3.3, the SDDP Algorithm 6, approximates each V_{t+1}^r as a maximum of affine functions. More precisely, at iteration k, we first compute a trial trajectory $(x_t^k)_{t\in[T]}$. Then, in the backward phase, we can compute $\mathcal{B}_t^r(V_{t+1}^{r,k})$ by solving $|\Xi_t|$ linear problems. Linear programming duality yields a sub-gradient of $\mathcal{B}_t^r(V_{t+1}^{r,k+1})$ at x_{t-1}^{k+1}, which in turn defines an affine function which underestimates $\mathcal{B}_t^r(V_{t+1}^{r,k+1}) \leq \mathcal{B}_t^r(V_{t+1}^r) = V_t^r$. In particular, at iteration k, the approximate cost-to-go functions $V_t^{r,k}$ are given as a maximum of

affine cuts, *i.e.*, $V_t^{r,k}(x) = \max_{\kappa \leq k}\{\alpha_t^\kappa + \beta_t^\kappa x\}$.

Recall that, given any approximated cost-to-go function, the forward Bellman operator (see Section 3.3.1), produces a state-based feedback, satisfying in particular the binary constraints. Thus, it seems natural to use the functions $V_t^{r,K}$ as approximated cost-to-go, leading to a strategy through the forward operators $\hat{\mathcal{F}}_t(V_t^{r,K})$. The main limit of this approach is that we are quite greedy in the way we repair the binary constraints. Indeed, $V_t^{r,K}$ does not account for binary constraints, and the forward operator only considers their impact on one time-step. In particular in the problem at hand (with shared resource constraints), SDDP approximated cost-to-go functions do not capture the necessity to make a choice between two products in the future. Therefore, a decision at t leading to infeasibility in the future, can have a finite SDDP estimated cost-to-go, and be selected by the forward operator. We illustrate the limit of this approach in the following toy example.

Example 1 (Limit of continuous relaxation.)**.** *Consider a production unit that produces two products $j = A, B$, over $T = 2$ time steps and one machine. The shared resource constraint, modeled through binary variables b_t^j, implies that we must decide which product to produce at $t = 1$, and which at $t = 2$. We look for the production plan minimizing costs while satisfying a demand $D = 1$ in both products at the end of the horizon. The problem is formalized as follows.*

$$\begin{aligned}
\min \quad & 3u_1^A + 2u_1^B + (u_2^A + u_2^B) && && && \text{(3.7a)}\\
s.t \quad & u_1^j + u_2^j \geq D && && j = A, B, && \text{(3.7b)}\\
& 0 \leq u_t^j \leq 2b_t^j && j = A, B && t = 1, 2, && \text{(3.7c)}\\
& b_t^A + b_t^B \leq 1 && && t = 1, 2, && \text{(3.7d)}\\
& b_t^j \in \{0, 1\},\ u_t^j \geq 0 && j = A, B && t = 1, 2. && \text{(3.7e)}
\end{aligned}$$

For the true problem, it is optimal to produce B in the first period and A in the second period, resulting in an optimal cost of 3. However, in the continuous relaxation of Problem (3.7), $b_t^j \in [0, 1]$, and producing both products at the same time is allowed. For instance, producing both products at time $t = 2$ (with $b_2^A = b_2^B = 0.5$) is admissible for the relaxed problem, yielding an optimal cost of 2.

Let V_2^r be the relaxed cost-to-go function given by:

$$\begin{aligned}
V_2^r(u_1^A, u_1^B) = \min_{u_2^A, u_2^B, b_2^A, b_2^B} \quad & u_2^A + u_2^B && && && \text{(3.8a)}\\
s.t \quad & u_1^j + u_2^j \geq D && && j = A, B, && \text{(3.8b)}\\
& 0 \leq u_2^j \leq 2b_2^j && && j = A, B, && \text{(3.8c)}\\
& b_2^A + b_2^B \leq 1, && && && \text{(3.8d)}\\
& b_2^j \geq 0,\ u_2^j \geq 0 && && j = A, B. && \text{(3.8e)}
\end{aligned}$$

Now, using the cost-to-go approximation V_2^r to determine optimal decisions of the mixed-integer problem at $t = 1$, we solve:

$$\min_{u_1^A, u_1^B, b_1^A, b_1^B} \quad 3u_1^A + 2u_1^B + V_2^r(u_1^A, u_1^B) \tag{3.9a}$$

$$s.t \quad b_1^A + b_1^B \leq 1, \tag{3.9b}$$

$$0 \leq u_1^j \leq 2b_1^j \qquad j = A, B, \tag{3.9c}$$

$$b_1^j \in \{0, 1\} \qquad j = A, B. \tag{3.9d}$$

Note that, when solving Problem (3.9), we make decisions at $t = 1$ considering the cost impact at $t = 2$, but not knowing what decisions are attached to this cost. In DP, infeasibility is supposed to be propagated through costs: in this example, with the real cost-to-go function, $V_2(0, 0) = +\infty$ and the solution $u_1^A = u_1^B = 0$ would never be chosen. However, if we use the relaxed cost-to-go function, the infeasible solution $u_1^A = u_1^B = 0$ has a cost $0 + V_2^r(0, 0) = 2$ and is chosen rather than the optimal solution $u_1^A = 1; u_1^B = 0$, whose cost is $2 + V_2^r(0, 1) = 3$.

We move on to present heuristics in Section 3.4, and address this particular limit in Section 3.4.4 through a look-ahead heuristic that considers more than one time-step.

3.4 Heuristics for multistage problems

So far, we have presented a stochastic algorithm with unreasonable computational time and a stochastic algorithm solving a continuous relaxation of our problem. Those are exact methods, but will not allow us to solve the problem in a satisfactory manner. Could we come up with heuristics taking into account uncertainties, using SDDP, and solving mixed-integer problems such as ours? In this section, we present different heuristics, either relaxing information constraints *i.e.,* deterministic, or relaxing integrality to a point.

3.4.1 An Expected Value (EV) corrected heuristic

One of the challenges is to take into account random variables. A common simplification consists in reducing the problem to its deterministic version, by replacing the random variable with our current best estimation. We are then in the anticipative framework which consists in assuming we can look into the future and know the noises realization *i.e.,* relaxing constraint 3.3c. More precisely, solving the anticipative problem, given a scenario $q_{[T]}^{\text{PV}}$, returns a solution perfectly adapted to this scenario, of optimal value $V^{ant}(x_0, q_{[T]}^{\text{PV}})$. Then, the Expected Value (EV) solution amounts to solving the anticipative problem, given the expected scenario $\overline{q}_{[T]}^{PV}$.

However, we are not in a complete recourse setting, meaning that the deterministic production and energy plan computed is not necessarily admissible. Therefore, a first heuristic consists in computing the deterministic solution fixing part of control variables, and then, adjusting the rest of the variables to actual random variable realization. In our particular microgrid problem, we fix production variables and then adjust energy flows to actual solar energy produced. We opt for a simple strategy described in Figure 3.2.

This strategy has no flexibility, which is needed in a system subjected to uncertainties. It serves as a benchmark for stochastic solutions.

3.4.2 Model Predictive Control

To add flexibility to the previous approach, we present the Model Predictive Control (MPC) approach, as a first adaptive approach. To use MPC we need some *forecast* methodology, that

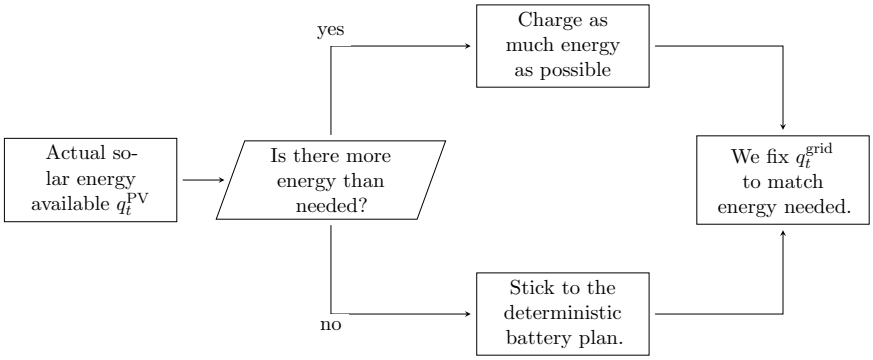


Figure 3.2: Corrected EV heuristic algorithmic scheme

takes available information to predict the values of the random variables $\{\boldsymbol{q_t^{PV}}\}_{t\in[T]}$. The algorithm then consists in solving successive deterministic sub-problems (see Algorithm 13). Step after step, it applies the decision of the first control obtained, reveals the realization of the next random variable, and recomputes all other decisions, updating forecasted values if possible.

Algorithm 7: Model Predictive Control

Input : x_0, initial forecast $\{q^{\text{PV}}_{\tau,0}\}_{\tau\in[T]}$.

$$v^{\text{DA}} = \quad \arg\min \sum_{t=1}^{T} p_t^{\text{DA}} v_t^{\text{DA}} + V^{ant}(x_0, q^{\text{PV}}_{[T],0}; v^{\text{DA}})$$

for $t : 1 \to T$ **do**

Update forecasted values $\{q^{\text{PV}}_{\tau,0}\}_{\tau\in[T]}$.

$$\begin{aligned} y_t^*, \dots, y_T^* = \quad & \arg\min \quad \sum_{\tau=t}^{T} p_t^{\text{ID}} v_t^{\text{ID}} \\ & \text{s.t.} \quad y_\tau \in \mathcal{Y}_\tau(x_{\tau-1}, q^{\text{PV}}_{\tau,0}; v^{\text{DA}}) \qquad \forall \tau \in [t:T], \\ & \qquad\; x_\tau \in \mathcal{X}_\tau(x_{\tau-1}, y_\tau, q^{\text{PV}}_{\tau,0}) \qquad \forall \tau \in [t:T]. \end{aligned}$$

$x_t \in \mathcal{X}(x_{t-1}, y_t^*, q^{\text{PV}}_{t,0})$

As long as we can get a solution to the MILPs in a reasonable time, MPC is an easy option to implement. However, this method yields no performance guarantee, and does not really take randomness into account, as the solution is computed for a single possible realization, but simply recomputes the solution as more information becomes available. Consequently, the quality of the solution provided by MPC depends mainly on the quality of the forecasted values, the flexibility of the problem and the sensitivity of the problem to uncertainty.

3.4.3 2-stage stochastic programming

The strategic design problem balances the design cost $\sum_t p_t^{\text{DA}} v_t^{\text{DA}}$ and the operational cost $V(x_0; v^{\text{DA}})$. The 2-stage stochastic programming consists in relaxing the non-anticipativity constraint for all operational decisions. Hence, the design problem becomes a two-stage stochastic

program, where the first stage decision is the strategic decision v^{DA} and the recourse are the operational decisions.

$$\min_{v^{\mathrm{DA}} \in \mathbb{R}_+^T} \quad \sum_t p_t^{\mathrm{DA}} v_t^{\mathrm{DA}} + \mathbb{E}\left[\hat{V}^{ant}(x_0, \boldsymbol{q}_{[\boldsymbol{T}]}^{\mathrm{PV}}; v^{\mathrm{DA}})\right]. \tag{3.10}$$

However, computing the exact value of $\mathbb{E}\left[\hat{V}^{ant}(x_0, \boldsymbol{q}_{[\boldsymbol{T}]}^{\mathrm{PV}}; v^{\mathrm{DA}})\right]$ would require to solve a deterministic operational problem for each possible scenario $q_{[T]}^{\mathrm{PV}} \in \Omega$. There is usually far too many scenarios to consider, thus, we resort to Sample Average Approximation, which is the 2-stage extension of Monte Carlo methods. We draw S_{MC} scenarios, and obtain the following 2-stage formulation:

$$V^{2S_{MC}}(x_0) := \min_{v^{\mathrm{DA}} \in \mathbb{R}_+^T} \min_{(x_t^s, y_t^s)} \sum_t p_t^{\mathrm{DA}} v_t^{\mathrm{DA}} + \sum_{s=1}^{S_{MC}} \frac{1}{S_{MC}} \left(\sum_{t=1}^{T} p_t^{\mathrm{ID}} v_{t,s}^{\mathrm{ID}} \right) \tag{3.11a}$$

$$\text{s.t.} \quad x_t^s \in \mathcal{X}_t(x_{t-1}^s, y_t^s, q_{t,s}^{\mathrm{PV}}) \qquad \forall t \in [T], \forall s \in [S_{MC}] \tag{3.11b}$$

$$y_t^s \in \mathcal{Y}_t(x_{t-1}^s, q_{t,s}^{\mathrm{PV}}, v^{\mathrm{DA}}) \qquad \forall t \in [T], \forall s \in [S_{MC}]. \tag{3.11c}$$

All the approaches presented in this section up to this point relax non-anticipativity constraints but keep binary constraints by solving MILPs. In Section 3.3.2, we saw that SDDP solves Problem (3.3) with non-anticipativity constraints but relaxing binary constraints. We now look for a trade-off between information relaxation and integrality relaxation.

3.4.4 Look-ahead heuristic

Were the forward operator (see Equation (3.6b)) to have more visibility on the future variable possibilities (or impossibilities), we have the intuition that the algorithm would perform better. Indeed, as it is defined, the operator takes the best decision possible at t by optimizing a one-stage problem minimizing the current cost at t plus an approximate cost-to-go function from $t+1$. Details of the problem complexity are thus only represented over one stage, and the impact of decision at time t on the next stage should all be taken into account by the approximate cost-to-go function.

To have a better representation of the problem, we can consider τ-stage problems with a final cost-to-go function $\tilde{V}_{t+\tau}$ instead of one-stage problems (with final cost-to-go function $\tilde{V}_{t+1}$). More precisely we define a τ-look-ahead Bellman operator $\mathcal{B}_t^\tau$ as:

$$\hat{\mathcal{B}}_t^\tau(R) : x, \xi \mapsto \min_{y_t, x_t} \quad p_t^{\mathrm{ID}} v_t^{\mathrm{ID}} + \min_{(\boldsymbol{x}_{t'}, \boldsymbol{y}_{t'})} \quad \mathbb{E}\left[\sum_{t'=t+1}^{t+\tau} p_{t'}^{\mathrm{ID}} \boldsymbol{v}_{\boldsymbol{t'}}^{\mathrm{ID}} + R(\boldsymbol{x}_{t+\tau}) \right] \tag{3.12a}$$

$$\text{s.t.} \quad x_t \in \mathcal{X}_t(x, y_t, \xi), \tag{3.12b}$$

$$y_t \in \mathcal{Y}_t(x, \xi; v^{\mathrm{DA}}) \tag{3.12c}$$

$$\boldsymbol{x_{t'}} \in \mathcal{X}_{t'}(\boldsymbol{x_{t'-1}}, \boldsymbol{y_{t'}}, \boldsymbol{q_{t'}^{\mathrm{PV}}}) \ \forall t' \in [t+1 : t+\tau] \tag{3.12d}$$

$$\boldsymbol{y_{t'}} \in \mathcal{Y}_{t'}(\boldsymbol{x_{t'-1}}, \boldsymbol{q_{t'}^{\mathrm{PV}}}; v^{\mathrm{DA}}) \ \forall t' \in [t+1 : t+\tau] \tag{3.12e}$$

$$\sigma(\boldsymbol{u_{t'}}) \subset \sigma(\boldsymbol{q_{[t+1:t']}^{\mathrm{PV}}}) \qquad \forall t' \in [t+1 : t+\tau] \tag{3.12f}$$

$$\mathcal{B}_t^\tau(R) : \quad x \mapsto \mathbb{E}\left[\hat{\mathcal{B}}_t^\tau(R)(x, \boldsymbol{q_t^{\mathrm{PV}}})\right]. \tag{3.12g}$$

In this setting, the first-stage decisions are optimized knowing the impact they have on the next $\tau - 1$ stages, thanks to Equations (3.12b) to (3.12f), and a cost-to-go function R from $t + \tau + 1$. However, the $\tau-$stage decisions are taken without any visibility on the future except a given cost-to-go function. For this reason, when solving each $\tau-$stage problem $\mathcal{B}_t^\tau(R_{t+\tau+1})(x_{t-1})$, we only store the first-stage variables y_t and then move along to the next sub-problem $\mathcal{B}_{t+1}^\tau(R_{t+\tau+2})(x_t)$.

In a sense, we allow the operators to look ahead of time to choose their decision at t, and call this method the look-ahead heuristic. We associate to the backward operator $\hat{\mathcal{B}}_t^\tau$ a forward operator $\hat{\mathcal{F}}_t^\tau(R) : \mathcal{X}_{t-1} \times \Xi_t \to \mathcal{X}_t \times \mathcal{Y}_t$ which returns $x_t^\star, y_t^\star$ depending on current state x and noise realization ξ.

For clarity, we explicitly give the 2-look-ahead Bellman operator:

$$\begin{aligned}
\hat{\mathcal{B}}_t^2(R)(x,\xi) = \min_{\substack{x_t,(x_{t+1}^s)_{s\in|\Xi_{t+1}|}\\ y_t,(y_{t+1}^s)_{s\in|\Xi_{t+1}|}}} & p_t^{\mathrm{ID}} v_t^{\mathrm{ID}} + \sum_s \mathbb{P}(\boldsymbol{q_{t+1}^{\mathrm{PV}}} = q_{t+1,s}^{\mathrm{PV}}) \Big[p_{t+1}^{\mathrm{ID}} v_{t+1,s}^{\mathrm{ID}} + \mathcal{R}(x_{t+1}^s) \Big] \\
\text{s.t.} \quad & x_t \in \mathcal{X}_t(x, y_t, \xi), \\
& y_t \in \mathcal{Y}_t(x, \xi; v^{\mathrm{DA}}), \\
& x_{t+1}^s \in \mathcal{X}_{t+1}(x_t, y_{t+1}^s, q_{t+1,s}^{\mathrm{PV}}) && \forall s \in |\Xi_{t+1}| \\
& y_{t+1}^s \in \mathcal{Y}_{t+1}(x_t^s, q_{t+1,s}^{\mathrm{PV}}; v^{\mathrm{DA}}) && \forall s \in |\Xi_{t+1}| \\
\mathcal{B}_t^2(R)(x) = \quad & \mathbb{E}\left[\hat{\mathcal{B}}_t^2(R)(x, \boldsymbol{q_t^{\mathrm{PV}}})\right]. && (3.13a)
\end{aligned}$$

Note that this 2-look-ahead Bellman operator considers the exact cost at t and $t+1$, and uses R as an estimation of expected cost-to-go from $t+2$ to T. In particular, due to the new information, we must consider as many decisions y_{t+1}^s as there are realizations for the random variable $\boldsymbol{q_{t+1}^{\mathrm{PV}}}$.

Combining these new operators with the approximated cost-to-go functions computed by SDDP (see Section 3.3.2), we get a heuristic where the non-anticipativity constraints hold at any time, and the integrality constraints are kept on τ time steps. Unfortunately, increasing the look-ahead horizon *i.e.,* τ, greatly increases the complexity of the sub-problems we solve. For instance, with $|\Omega_t| = 10$, the backward operator $\hat{\mathcal{B}}_t^\tau$ at t solves a problem with $10^{\tau-1}$ times more variables than $\hat{\mathcal{B}}_t$.

3.5 Numerical results

We now present a study case from our industrial partner on which we evaluate the numerical methods presented above. In Section 3.5.1 we detail the study case, intraday results, given in Section 3.5.2, show that the MPC method is most adapted to our study, it is then used for the day-ahead problem in Section 3.5.3 where SDDP shows its advantages.

3.5.1 Study Case

The problem described in Section 3.5.1 is motivated by a cement factory in South Korea. We solve the problem for hourly planning on one day, with $T = 24$ time steps. In the Republic of Korea, electricity rates are fixed for the industry and depend on different time slots and seasons. We took the rates given by the Korea Electricity Power Corporation website [Newa] and thus obtained $\{p_t^{\mathrm{ID}}\}_{t\in[T]}$. We consider that buying energy in advance is cheaper and fix the day-ahead rates to 90% of the real-time rates.

Then we collect solar irradiance data on [Newb]. From this data, we use a forecast algorithm to predict a daily solar energy generation for a park of capacity $C^{PV} \in \{2, 4, 8, 12\}$ in MWc. The model is trained on the last 72 hours data to produce generation scenarios over the next 24

hours. From this model we estimate, at each time step t, 9 quantiles. We finally assume that the noise is stagewise independent, leading to 9^T scenarios.

The factory owns $I = 3$ mills and produces $J = 3$ different types of cement. Production bounds are given by the factory. An analysis of the factory's data leads us to model a mill's energy consumption, on the range $[u_t^{imin}, u_t^{imax}]$, as an affine function of its cement production (3.1i).

We study the impact of three different battery sizes, proportional to the installed renewable capacity: SOC_{max} is equal to the maximum quantity of energy the solar panels can produce in 0.5, 3 or 6 hours. For example, on a solar park of 4 MWc, we consider three battery capacities: 2, 12 or 24 MWh. We also fix ϕ_{max}^+ and ϕ_{max}^- to a quarter of the battery's capacity per time-step and the efficiency factor ρ to 0.9.

3.5.2 Intraday results

In this section, we present and analyze the results obtained when solving problem (3.3) on instances in which energy can only be bought in real-time, which is equivalent to fixing $v_t^{\mathrm{DA}} = 0$ for all t. Further, we only consider a demand at the end of the day: $d_t^j > 0$ only for $t = T$.

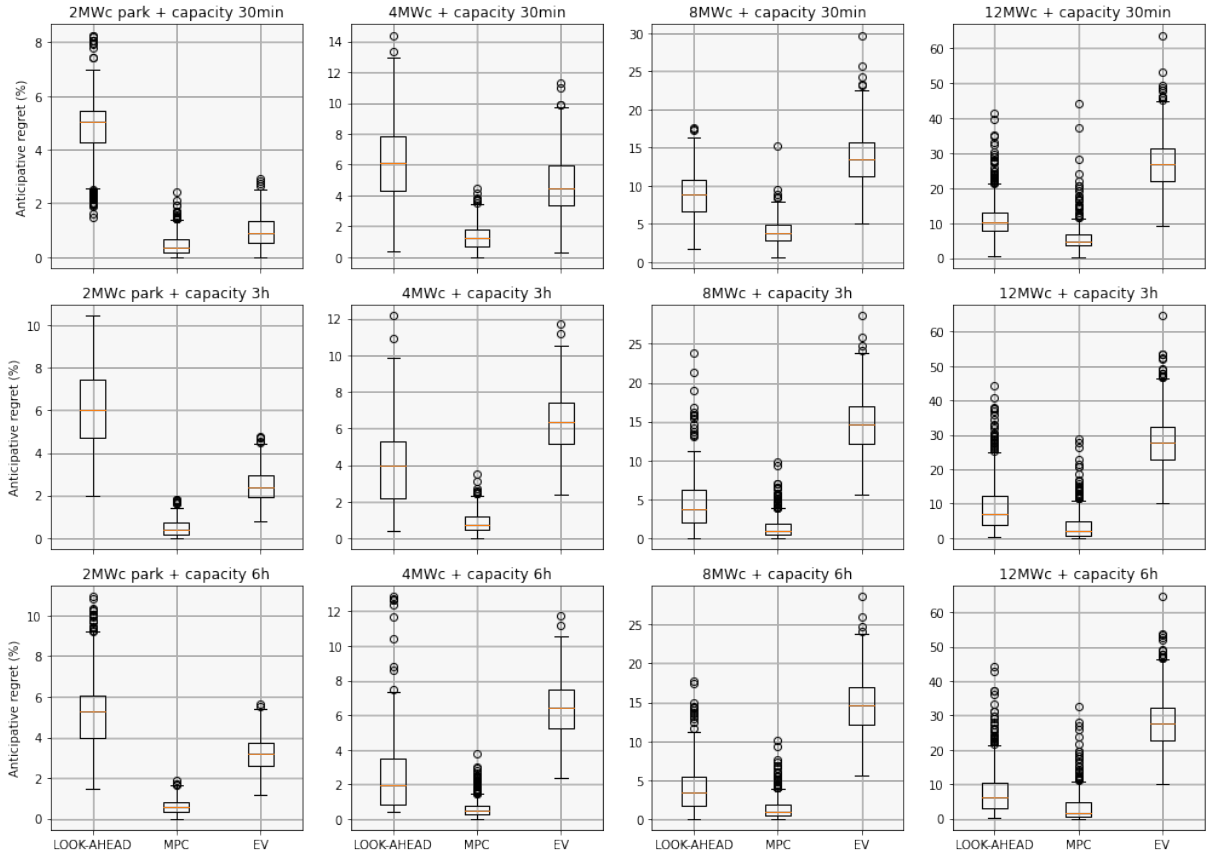


Figure 3.3: Anticipative regret (AR) in percentage for different solar park capacity and ESS capacity: increasing solar energy (and thus variability) from left to right, and increasing battery storage capacity (proportional to solar energy available) from top to bottom.

On a given day, for various renewable size ($C^{PV} \in \{2, 4, 8, 12\}$) and battery sizing ($SOC_{max}$ represents 0.5, 3 or 6 hours of maximum renewable production), we test the different strategies, evaluating them over 500 common scenarios drawn from our statistical model. More precisely,

we compare:

1. the elementary strategy, described in Section 3.4.1, which solves the EV problem abd then adapt energy variables following a deterministic procedure as noises are revealed;

2. the MPC strategy, see Section 3.4.2, which consists in solving deterministic sub-problems at each stage, with updated information, to adjust the solution trajectory accordingly;

3. and the Look-Ahead (LA), with $\tau = 2$, explained in Section 3.4.4, strategy which computes a solution with dynamic programming using an under-approximation of future costs given by SDDP.

Each strategy yields a noise-based policy which, depending on a scenario, computes a trajectory of the system. To evaluate a strategy's performance over a given scenario, we define the *anticipative regret* of admissible noise-based policy π, on a scenario $\xi_{[T]}$, as the relative gap between its cost and the anticipative lower bound:

$$AR^{\pi}(\xi_{[T]}) \;=\; \frac{\hat{V}^{\pi}(x_0, \xi_{[T]}; v^{\mathrm{DA}}) - V^{ant}(x_0, \xi_{[T]}; v^{\mathrm{DA}})}{|V^{ant}(x_0, \xi_{[T]}; v^{\mathrm{DA}})|}. \tag{3.14}$$

In Figure 3.3 we report the anticipative regret of each strategy. The results clearly show MPC's superiority in these instances. On the one side, the EV heuristic yields unsatisfactory results in comparison to MPC: its expected anticipative regret is always higher, and its expected cost as well. Further, except on the first column, which corresponds to instances with few uncertainties (*i.e.,* a solar park of 2MW), and the first instance of the second column (a more uncertain instance but with a small battery), the EV heuristic performs worse than the look-ahead heuristic. As uncertainties grow (from left to right), the costs of the EV heuristic are farther and farther away from the anticipative lower bound, showing that a purely deterministic procedure is not relevant to our problem.

On the other side, the look-ahead heuristic, properly taking uncertainties into account with a stochastic procedure, but relaxing some integrality constraints, does not perform as well as MPC. Indeed, the latter, adjusting the solution trajectory to uncertainties, yields solutions close to their anticipative lower bound: even for the most volatile instances (*i.e.,* the ones with a solar park of 12MWc, all on Figure 3.3's fourth column), the anticipative regret is lower than 5% and in most cases insignificant. These performances can be explained by the problem structure: the uncertainty source does not impact significantly future costs, in the case of solar energy variations at t, MPC foresees the cost impact and adapts accordingly. Furthermore, for industrial problems with renewable generation, we confirm the necessity of installing an ESS to make the system flexible. In Figure 3.4, we plot the optimal expected cost of the various methods on instances with growing ESS capacity. Clearly, the expected optimal expected cost decreases as the ESS capacity increases, although the marginal impact of the ESS capacity is decreasing.

Whereas MPC results are better, we call attention to its limits: on Table 3.1 we can see that MPC takes longer in computation time than the look-ahead heuristic, even more so on instances with the most variability. In these instances, it remains reasonable (a few seconds per problem at the most for an hour step time problem), but with larger instances, and more constraints, it could be unsuitable. Note that SDDP converges after only a 100 iterations, taking approximately $250s$ per instance.

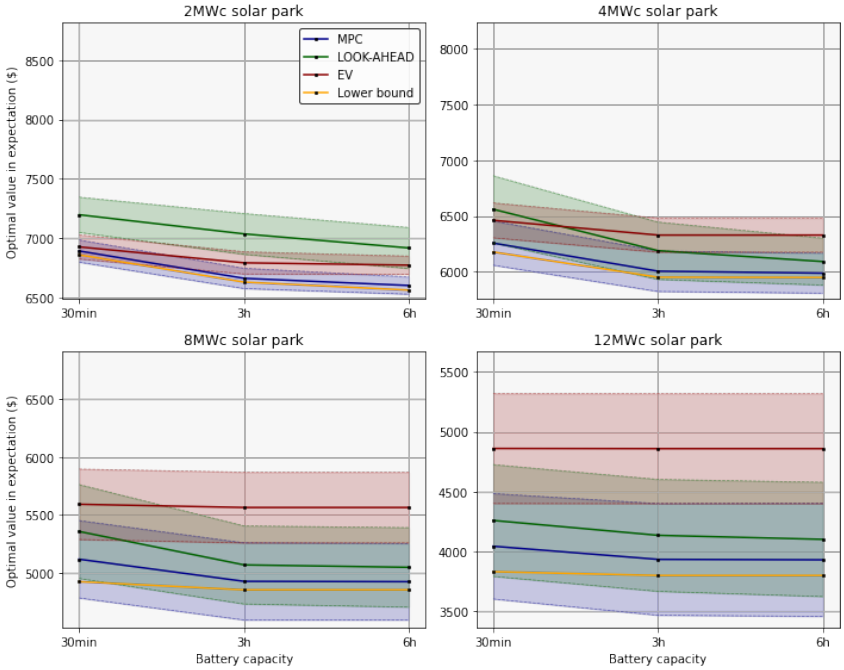


Figure 3.4: Expected value of strategies with 95% confidence interval.

SOC_{max} / C^{PV} (MWc)	0.5h MPC	0.5h L-A	0.5h SDDP	3h MPC	3h L-A	3h SDDP	6h MPC	6h L-A	6h SDDP
2	21	6.5	277	12	7.6	268	25	20	262
4	26	8.0	213	4.4	2.9	225	38	18	238
8	254	11	249	136	26	234	193	24	260
12	248	10	266	125	22	250	135	23	261

Table 3.1: Expected computation time (in seconds) for different solar park capacity and ESS capacity.

3.5.3 Day-ahead results

We now consider the full Problem 3.4 with strategic and operational decisions. In particular, we consider an initial time step ($t = 0$), where the industrial buys in advance energy quantities for the whole horizon. To our knowledge, this type of contract does not exist yet in South Korea, but it could be interesting for the regulator to encourage certain consumption schemes. It can also model the access to energy markets for large consumers or consumers aggregated through virtual power plants. We fix the in-advance prices at 90% of intra-day prices.

The problem can be decomposed into two parts: first a strategical problem with variable v^{DA}, then an operational sub-problem, parametrized by v^{DA}. Our intuition is that a deterministic method might not be flexible enough because first-stage decisions impact the whole horizon. Note that the parametrized problem 3.3 corresponds to the intraday problem we solve in Section 3.5.2. We saw that the most efficient method to solve problem 3.3 is MPC. In this section, we determine through different methods the best strategical decision v^{DA} and then run MPC on the parametrized operational problem.

We assume that the demand is only positive at the end of the day $d_T^j > 0$ and we test various renewable sizes ($C^{PV} \in \{2, 4, 8, 12\}$). In Section 3.5.2, we tested different battery size, and

results showed that extending the battery capacity, to a certain point, improves costs and the system flexibility. Consequently, we now fix the battery capacity to 3 hours of maximum renewable production.

To optimize v^{DA}, we test 3 methods evaluated over 1000 common scenarios:

1. the Expected Value strategy which solves a deterministic version of Problem 3.4 replacing random variables by their expected value;
2. the 2−stage strategy, detailed in Section 3.4.3, which takes the decision v^{DA} minimizing the expected cost over $S_{MC} = 10$ scenarios $(\xi^s_{[T]})_{s\in[S_{MC}]}$. As S_{MC} is small, compared to the noise space, for computational reasons, we consider the median scenario with probability $\frac{1}{2}$;
3. the SDDP strategy in Section 3.3.2 solves the continuous relaxation of Problem 3.4, and yields a solution taking into consideration the uncertainties on the whole horizon, but relaxing integrality.

C^{PV} (MWc)	OPT			AR (in %)		
	EV	2stage	SDDP	EV	2stage	SDDP
2	6067	**6023**	6038	1.6	**0.9**	1.1
4	5471	5483	**5451**	2.1	2.3	**1.7**
8	4552	4553	**4481**	4.2	4.2	**2.5**
12	3714	3691	**3641**	8.7	7.9	**6.7**

Table 3.2: Expected Cost (Opt) and Anticipative Regret (AR) of the solution obtained when finding v^{DA} with the different methods (EV, 2−stage, SDDP); parametrizing the operational problem with this v^{DA}; then solving the parametrized operational problem with MPC.

C^{PV}	EV			2stage			SDDP		
	$I(v^{\mathrm{DA}}_{EV})$	$V(x_0; v^{\mathrm{DA}}_{EV})$	Opt	$I(v^{\mathrm{DA}}_{2S})$	$V(x_0; v^{\mathrm{DA}}_{2S})$	Opt	$I(v^{\mathrm{DA}}_{r})$	$V(x_0; v^{\mathrm{DA}}_{r})$	Opt
2	6002	65	6067	5830	193	**6023**	5659	379	6038
4	5369	102	5471	5123	360	5483	5102	349	**5451**
8	4357	195	4552	4073	480	4553	4043	438	**4481**
12	3394	320	3714	2965	726	3691	3094	548	**3642**

Table 3.3: We obtain $v^{\mathrm{DA}}_{EV}, v^{\mathrm{DA}}_{2S}, v^{\mathrm{DA}}_{r}$ by solving the problem respectively with the EV strategy, 2−stage programming and SDDP; then we parametrize and solve the operational problem with MPC for each v^{DA}.

From Table 3.2, reporting simulated cost and anticipative regret of the various heuristics, we observe that, except for the instance with less uncertainties (first line), the day-ahead energy purchases determined with SDDP yield a lower expected cost as well as a lower anticipative regret than those determined with 2−stage programming or the EV strategy. As uncertainties grow (from top to bottom on the table), the anticipative regret increases and the gap between the AR of EV and the one of SDDP gets wider. Indeed, in the instance with a solar park of 4MWc, the anticipative regret is 0.4% lower for SDDP whereas it is 2% lower for the instance with more uncertainties (solar park of 12MWc).

On Table 3.3 we separate design costs $I(v^{\mathrm{DA}})$ from operational costs $V(x_0; v^{\mathrm{DA}})$ for all instances solved. Whereas the EV strategy essentially pays energy in advance, the two-stage and SDDP strategies have lower design costs and buy more energy in real-time. This can be explained because a stochastic approach is looking for a trade-off between initial and recourse decisions. Assume that we have more energy than predicted, this extra energy comes for free and we better

not have bought too much energy in advance, forcing us to throw this extra energy away (we can't charge the battery more than what is allowed). On the contrary, if we have less energy than predicted, we must either adapt the production plan (which might be possible) or buy energy in real-time which is not that much more expensive than if we bought it in advance (110% of day-ahead prices). Thus, we understand that in this problem, it is more efficient to underestimate the quantity of energy to buy from the main grid, as we have more to gain if the solar realization exceeds its prediction than we have to lose in the opposite case.

We give some additional insights on the various strategies in Figures 3.5 to 3.8. In Figure 3.5, we illustrate the day-ahead purchases over time. As expected, the day-ahead purchases are concentrated in the night and early morning, when energy from the grid is cheaper and no solar energy is available. Note that contrary to other approaches where day-ahead purchases are first-stage decisions, the anticipative day-ahead purchases are scenario dependent. Thus, we plot their expectation, which leads to a smoother function. Indeed, the minimum production constraint induces a discontinuity in the energy-load, and thus the day-ahead purchases. This is also reflected in Figure 3.6, where we plot the expected number of machines (out of 3). This is caused by the day-ahead purchases which shape the production: if energy has been bought for 2 machines, turning a third one on would be costly unless the available solar energy can cover it.

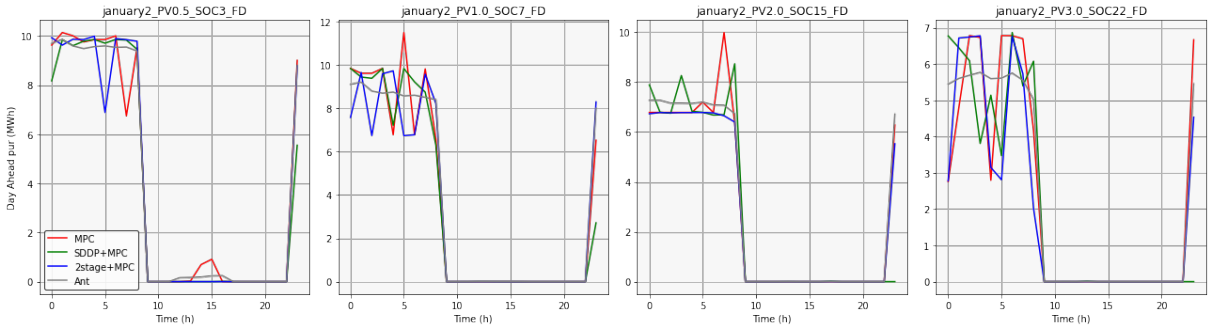


Figure 3.5: Day-ahead purchases y (in MWh) over time with the different methods (EV, 2−stage, SDDP). Averaged anticipative's day ahead purchased are also given.

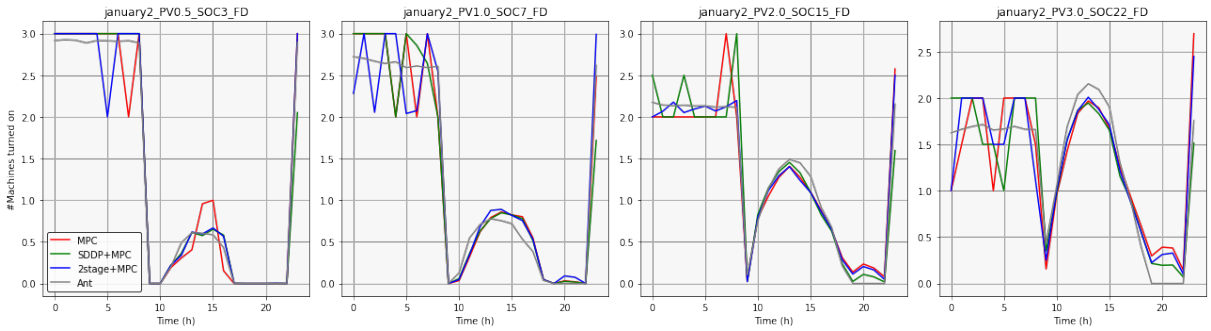


Figure 3.6: Expected number of machines turned on (out of 3) over time with the different methods (EV, 2−stage, SDDP) and with the anticipative solution.

Moreover, on Figure 3.7, we plot the expected battery storage (thick lines) over time for each method and the standard deviation in dashed lines. Notably, the anticipative solution doesn't use the battery as much as the other methods: indeed, as it is aware of the exact amount of solar energy that will be available, it can adapt the production of early stages precisely and does not need the flexibility to compensate for uncertainties. We can observe that the EV strategy

always makes more use of the battery than the stochastic strategies (2−stage and SDDP). This confirms that a deterministic approach needs more flexibility to recover a good solution than a stochastic one.

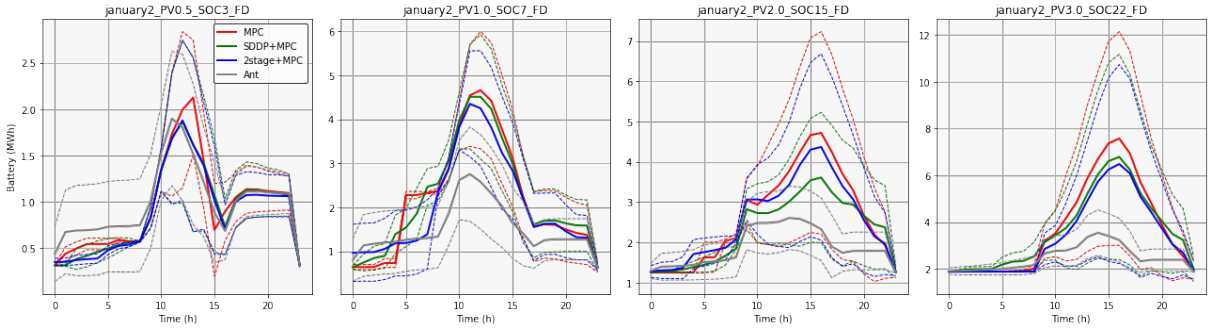


Figure 3.7: Expected trajectory of the battery (in MWh) over time with the different methods (EV, 2−stage, SDDP) and with the anticipative solution.

Finally, we can see in Figure 3.8 the cumulated stocks produced over time for each method. We observe that as uncertainties grow (from left to right), the stochastic solution (green line) gets closer to the anticipative production (grey line).

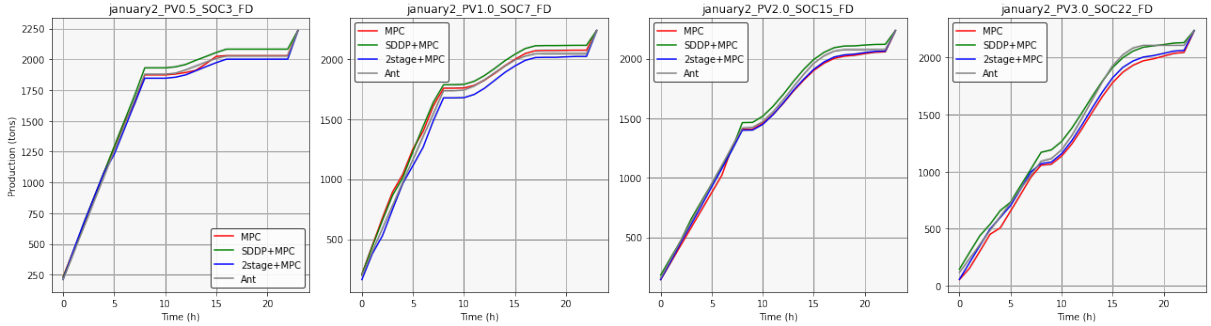


Figure 3.8: Evolution of the cumulated product stocks over time with the different methods (EV, 2−stage, SDDP) and with the anticipative solution.

3.5.3.1 Results on stagewise dependent scenarios

In our use case, the only uncertainty lies in the day-ahead forecast error of solar production. This forecast error was based on an advanced statistical model that we consider as ground truth. As the aim of this work is to compare various methodological approaches we considered a simpler model with stagewise independent residual errors (see Section 3.3).

As this assumption is not strictly satisfied by the advanced statistical model, we simulated the strategies on scenarios obtained from the advanced statistical model (see Figure 3.9). We find that, for these stagewise dependent scenarios, MPC is still the best method for the intraday problem, and the SDDP approach gives significantly better results than the EV strategy for the day-ahead problem (see Figure 3.9). We also observe that the expected optimal value of the SDDP approach is close to the expected anticipative optimal value. Thus, even if it would be possible to use a more advanced statistical model to train the SDDP approach (*e.g.,* autoregressive or Markov Chain models), we do not believe that it would lead to significant improvements in our use case.

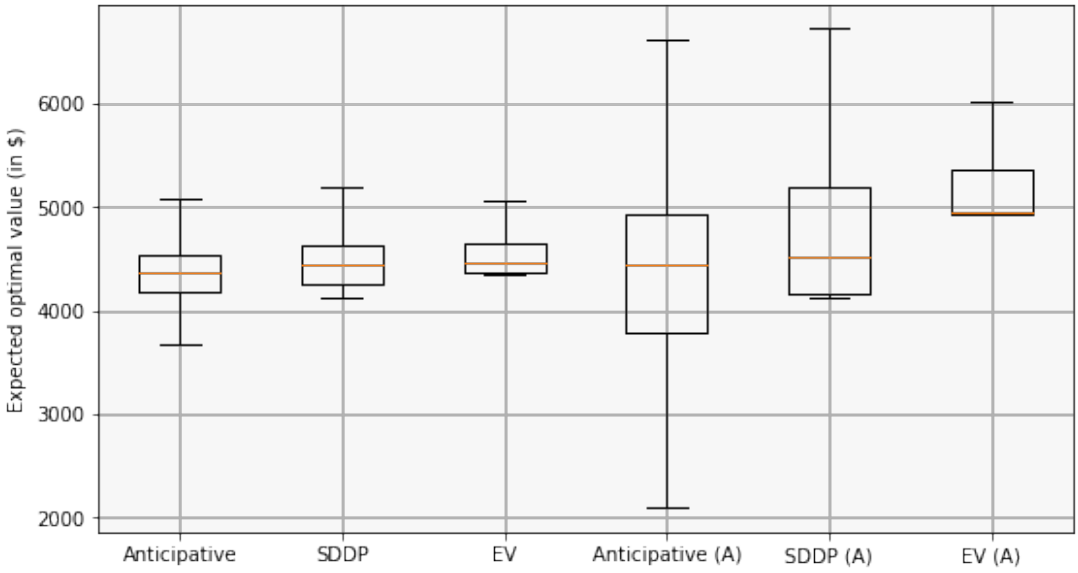


Figure 3.9: Expected Cost (Opt) of the solutions obtained when solving the operational problem parametrized by day-ahead purchases v^{DA} of the Anticipative, EV and SDDP strategies, first on a thousand scenarios with independence assumption; then on a thousand scenarios from the advanced statistical model (A).

3.6 Conclusion

In this paper, we considered a common problem in the industry: jointly optimizing the production planning and the energy supply of a factory, considering intra-day and day-ahead decisions. For ecological reasons we included renewable energies, leading to a stochastic optimization problem. We proposed various solution methodologies and compared them on a realistic industrial case study.

The main difficulty of the production plan comes from binary variables that cannot be relaxed. They are crucial to modeling the hard constraints of the problem. Integrating energy operations into the problem obliges us to handle uncertainties. In all the proposed algorithms to solve the problem, a trade-off must be found between relaxing integrality and relaxing information. For instance, MPC is fully deterministic whereas SDDP solves the continuous relaxation of the stochastic problem. In the tests we have conducted, we found that the right balance depends on the problem.

Indeed, for the intraday problem, where strategic decisions are given, we saw that using the Model Predictive Control algorithm, which consists in replacing the stochastic variables with deterministic ones, and reevaluating decisions at each stage, get the best results. However, for the day-ahead problem, we saw that solving the relaxed version of the problem with For instance, MPC is fully deterministic whereas SDDP yields better strategic decisions as it is more aggressive in its day-ahead buying decisions. Indeed, having too much energy is less costly than having too little. Other methods were considered but did not yield interesting results in our case. In particular, two-stage approaches were not providing better solutions than deterministic MPC; and discretized dynamic programming was too slow. Finally, the look-ahead approach computes a feasible and reasonable solution from For instance, MPC is fully deterministic whereas SDDP cuts. Although it does not beat MPC in this specific setting, the method shows promise as a viable compromise between relaxing integrality and achieving fast computational time.

To conclude, remember that we did not discuss the investment problem. Indeed, the return on investment is difficult to compute, as it highly depends on decarbonization subsidies or tax incentives as well as on the evolution of the energy markets. Recent events in Ukraine have shown that energy prices are volatile and unpredictable, especially in the long term. However, with conservative estimates, we obtain a return on investment of around 10% for solar parks. In our setting, investment in storage is not profitable. Nevertheless, if we allow buying and selling energy at the given prices, which is not completely realistic, the battery would be quite profitable. Without reselling energy, the profitability of storage also depends on the demand load: a high load requiring the machines to be on during peak prices would make the battery profitable. These investment aspects would require further investigation.

Part II

Gardening tools for solving MSbLPs with SDDP

Chapter 4

A branch and bound framework for MSbLPs

Contents

In this chapter, we introduce a Branch-and-Bound (BB) framework that relies on the underlying structure of scenario trees to solve Multistage Stochastic mixed-binary Linear Program (MSbLP) leveraging ideas from classical BB methodology, Dynamic Programming (DP) for multistage problem and especially the Stochastic Dual Dynamic Programming (SDDP) algorithm for MSLP. The work presented in this section has been done in collaboration with Bernardo Freitas Paulo da Costa[1].

First, relying on graph theory [KV12], we define formally scenario trees and different subtrees extracted from a scenario tree in Section 4.1. Then, we introduce in Section 4.2 a class of functions, *assignation functions*, which transform the feasible set of binary variables in an MSbLP. Those functions are fitted to BB methodologies [Mor+16]. Finally, in Section 4.3, we present some specific assignation functions where SDDP can be combined with BB, leading to the algorithm developed in Chapter 5 to solve MSbLP.

[1] Fundaçao Getulio Vargas

4.1 Structuring scenario trees

In Chapter 2, we presented informally the scenario tree representing the uncertainties of an MSbLP. In this section, we present a formal definition of scenario trees that allow us to explore their structure. Through the definition of different subtrees of a scenario tree, we elaborate a flexible framework to solve MSbLP, which relies on variants of the typical bellman operators we presented in Chapter 2.

4.1.1 Elements of graph theory

In this preliminary section, we go over some notations and conventions from graph theory (see [KV12]) that we use later to define formally a scenario tree $\mathcal{T}$.

A *weighted directed graph* $G := (\mathcal{N}, \mathcal{A}, w)$ is described by a set of *nodes* $\mathcal{N}$, a set of *arcs* $\mathcal{A}$, which are ordered pair of nodes, and a function $w : \mathcal{A} \to \mathbb{R}$ assigning a *weight* to each arc in the graph. If $(\nu, \mu) \in \mathcal{A}$, we say that (ν, μ) *leaves* ν and *enters* μ: it is an *outgoing arc* of ν and an *incoming arc* of μ. A *path* in G is a sequence of k nodes $(\nu_1, \dots, \nu_k)$ such that $k > 1$ and for all $i \in [k-1]$, we can order the pair $\{\nu_i, \nu_{i+1}\}$ so that it is in $\mathcal{A}$. We say that the path $(\nu_1, \dots, \nu_k)$ connects ν_1 and ν_k. In particular, a *cycle* is a path connecting ν and itself, and a graph containing no cycle is called *acyclic*. Finally, we say that G is *connected* if, between each pair of nodes, there exists a path.

A *weighted directed tree* $\mathcal{T} := (\mathcal{N}, \mathcal{A}, w)$ is a connected and acyclic weighted directed graph such that all nodes have at most one incoming arc. By definition, a directed tree $\mathcal{T}$ has $|\mathcal{N}| - 1$ arcs, and we call *root* the only node $r \in \mathcal{N}$ which has no incoming arc. We say that $\mathcal{T}$ is $r-$rooted. Then, all other nodes $\mu \in \mathcal{N} \setminus \{r\}$ have exactly one incoming arc (ν, μ) and we denote $a(\mu) = \nu$ the *parent* of ν. By convention, $a(r) = r$. Similarly, we call *children* of a node ν the nodes connected to ν by outgoing arcs *i.e.,* $\mathcal{C}(\nu) := \{\, \mu \in \mathcal{N} \mid (\nu, \mu) \in \mathcal{A} \,\}$. More generally, the children of a node set $S \subset \mathcal{N}$ are $\mathcal{C}(S) := \{\, \mu \in \mathcal{N} \setminus S \mid \exists\, \nu \in S, \quad (\nu, \mu) \in \mathcal{A} \,\}$. The *depth* of ν is defined as the length of the unique path connecting r and ν, and denoted t_ν. We regroup all nodes of same depth t into *layers* $\{\mathcal{N}_t\}_{t \geq 0}$ and $\mathcal{N} = \cup_{t \geq 0} \mathcal{N}_t$. By convention, $\mathcal{N}_0 = \{r\}$. Then, note that for a node $\nu \in \mathcal{N}_t$, $a(\nu) \in \mathcal{N}_{t-1}$ and $\mathcal{C}(\nu) \subset \mathcal{N}_{t+1}$. We refer to the nodes with no children as *leaves* of the tree, and we denote them $\mathcal{L}(\mathcal{T}) = \{\nu \in \mathcal{N} \,|\, \mathcal{C}(\nu) = \emptyset\} \subset \mathcal{N}$. Finally, we say that a weighted directed tree $\mathcal{T} := (\mathcal{N}, \mathcal{A}, w)$ is a *scenario tree* (see an example in Figure 4.1) if the weight w takes value in $[0, 1]$, and for each node $\nu \in \mathcal{N} \setminus \mathcal{L}(\mathcal{T})$, we have $\sum_{\mu \in \mathcal{C}(\nu)} w(\nu, \mu) = 1$.

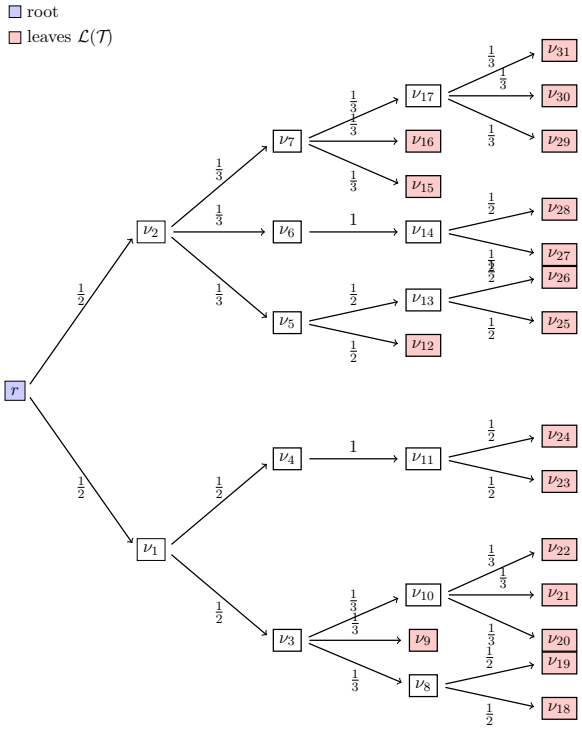


Figure 4.1: We give on this figure an example of a scenario tree $\mathcal{T}$ where $\mathcal{N} = \{\nu_i\}_{i\in[31]}$. We highlight in color the root and leaves of the tree. We read that $a(\nu_3) = \nu_1$ and $\mathcal{C}(\nu_3) = \{\nu_8, \nu_9, \nu_{10}\}$.

In this chapter, we exploit the structure of a scenario tree $\mathcal{T} = (\mathcal{N}, \mathcal{A}, w)$ to decompose efficiently a multistage stochastic program. Thus, we define here different class of subtrees that can be extracted from the initial scenario tree $\mathcal{T}$. To begin with, we call *subtree* $\mathcal{T}' = \mathcal{T}[\mathcal{N}']$ of $\mathcal{T}$, the weighted directed tree induced by $\mathcal{N}' \subset \mathcal{N}$ *i.e.*, $\mathcal{T}' = (\mathcal{N}', \mathcal{A}', w')$ with $\mathcal{A}' = \{(\nu, \mu) \in \mathcal{A} \,|\, (\nu, \mu) \in \mathcal{N}'^2\}$ and $w' = w_{|\mathcal{A}'}$. Note that we cannot construct a subtree $\mathcal{T}' = \mathcal{T}[\mathcal{N}']$ for all node set $\mathcal{N}'$. For instance, we cannot construct a subtree $\mathcal{T}[\mathcal{N}']$, of the scenario tree in Figure 4.1, with $\mathcal{N}' = \{r, \nu_9, \nu_2, \nu_{13}\}$, as the resulting graph is not connected. Further, a subtree $\mathcal{T}'$ is not necessarily a scenario tree, depending on w'. In future sections, we always work on subtrees in comparison with the initial scenario tree $\mathcal{T}$. Thus, $a(\nu)$ and $\mathcal{C}(\nu)$ always indicate the parent and children of ν in $\mathcal{T}$, and t_ν the depth of ν in $\mathcal{T}$. Then the children of $\nu \in \mathcal{N}'$ in the subtree are in the set $\mathcal{C}(\nu) \cap \mathcal{N}'$. We can observe two examples of subtrees in Figure 4.2.

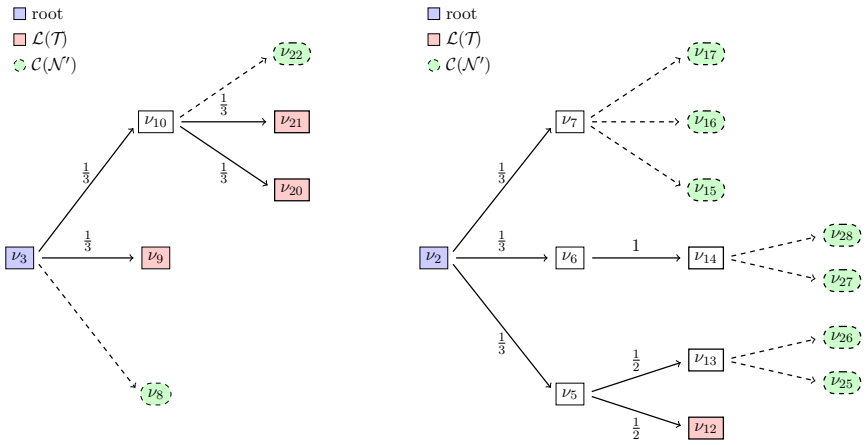


Figure 4.2: Two examples of subtrees of $\mathcal{T}$ represented in Figure 4.1: on the left a ν_3-rooted subtree and on the right a ν_2-rooted subtree. On this figure, we highlight, for a subtree, its root, its children $\mathcal{C}(\mathcal{N}')$, and the leaves of $\mathcal{T}$ that are in the subtree. Whereas the subtree on the right preserves the structure of a scenario tree, the one on the left does not: $w(\nu_3, \nu_{10}) + w(\nu_3, \nu_9) \neq 1$.

We consider two categories of subtrees with convenient structures: the $\nu-$*extracted subtree*, that is the subtree stemming from a node ν up to the leaves; and subtrees constructed by pruning branches of $\mathcal{T}$ as *pruned subtrees*. More precisely, we define the $\nu-$*extracted subtree* $\mathcal{T}_{|\nu} := \mathcal{T}\big[\mathcal{N}_{|\nu}\big]$ as the $\nu-$rooted subtree such that if a node μ is in $\mathcal{N}_{|\nu}$, then its children are too *i.e.*, $\mathcal{C}(\mu) \subset \mathcal{N}_{|\nu}$. In the specific case of $\nu-$extracted subtrees, the structure of scenario tree is preserved (see an example in Figure 4.3).

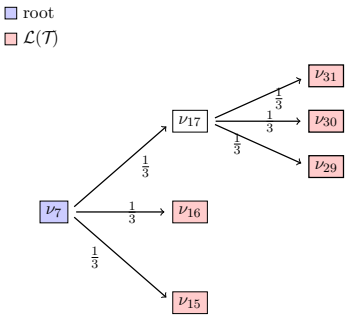


Figure 4.3: The ν_7-extracted subtree from $\mathcal{T}$.

Further, by construction, $\mathcal{C}(\mathcal{N}_{|\nu}) = \emptyset$. A *pruned subtree* $\mathcal{T}_p := \mathcal{T}\big[\mathcal{N}_p\big]$ is defined as a $r-$rooted subtree, such that if a node ν is in $\mathcal{N}_p$, then its parent $a(\nu)$ is too. In particular, a pruned tree is a scenario tree only if either all or none children of a node are in the subtree. For example, the pruned tree represented in Figure 4.4 is not a scenario tree. A useful example of pruned trees is the $t-$*shortsighted subtree* $\mathcal{T}_{1:t} = \mathcal{T}\big[\mathcal{N}_{1:t}\big]$, where we cut all the layers $\mathcal{N}_\tau^\Omega$ representing stages $\tau > t$, *i.e.*, $\mathcal{N}_{1:t} = \cup_{\tau=1}^{t} \mathcal{N}_\tau^\Omega$ and $\mathcal{C}(\mathcal{N}_{1:t}) = \mathcal{N}_{t+1}^\Omega$.

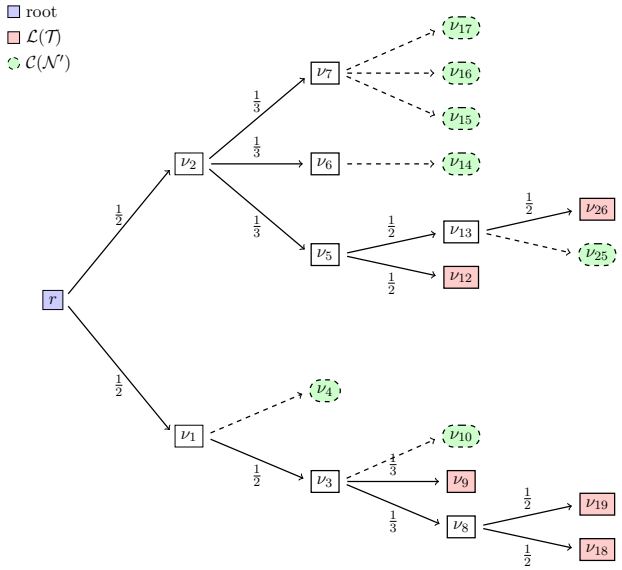


Figure 4.4: Example of a pruned tree of $\mathcal{T}$ which does not preserve the structure of a scenario tree.

Having introduced properly scenario trees, we can now associate a multistage program to a specific scenario tree.

4.1.2 Building a scenario tree

In multistage stochastic programming, at each stage, we optimize decisions that determine the evolution of a dynamic system. This system is affected by a sequence of noises $\boldsymbol{\xi}_{[T]}$, where T is the total number of stages. Each noise $\boldsymbol{\xi_t}$ takes value in a finite set Ξ_t and we denote $\Omega := \prod_{t\in[T]} \Xi_t$. We assume that these noises represent all the uncertainty in the problem at hand, with known probability distribution, resulting in a probability space $(\Omega, \mathcal{A}, \mathbb{P})$. Then, we define a scenario $\omega \in \Omega$ as a sequence of noise realization $\{\xi_t\}_{t\in[T]}$. In general, the scenario tree is informally defined as the collection of all scenarios. Though we find it in the literature in slightly varying forms, the formalism we develop here can be found in [SDR14].

Algorithm 8: Constructing recursively a scenario tree $\mathcal{T}$.

$\mathcal{N}^\Omega = \mathcal{N}_0^\Omega = \{r\},\ \mathcal{A}^\Omega = \emptyset,\ w^\Omega : \mathcal{A}^\Omega \to \mathbb{R},\ \mathcal{T}^\Omega := (\mathcal{N}^\Omega, \mathcal{A}^\Omega, w^\Omega)$
for $t = 1 \dots T$ **do**
  `// We construct a new layer` $\mathcal{N}_t^\Omega$.
  for $\nu \in \mathcal{N}_{t-1}^\Omega$ **do**
    $\mathcal{N}_t^\Omega = \emptyset$
    for $\xi_t \in \Xi_t$ **do**
      `// We construct a child of` ν.
      $\mu = (\nu, \xi_t)$
      `// We update the scenario tree.`
      $\mathcal{N}_t^\Omega = \mathcal{N}_t^\Omega \cup \{\mu\}$
      $\mathcal{A}^\Omega = \mathcal{A}^\Omega \cup \{(\nu, \mu)\}$
      $w^\Omega(\nu, \mu) = \mathbb{P}(\boldsymbol{\xi}_t = \xi_t | \, \nu\,)$
    $\mathcal{N} = \mathcal{N} \cup \mathcal{N}_t$

We construct the *scenario tree* $\mathcal{T}^\Omega = (\mathcal{N}^\Omega, \mathcal{A}^\Omega, w^\Omega)$, as defined in Section 4.1.1, that reflects the multistage structure of the problem. At the beginning of the problem, we have no information on the noises realization yet. This is represented by the root node r, and we define $\mathcal{N}_0^\Omega = \{r\}$. Then, in stage $t = 1$, we uncover the noise realization of $\boldsymbol{\xi}_1$. For each possible realization $\xi \in \Xi_1$, we construct a node $\nu = (r, \xi)$ with parent $a(\nu) = r$, containing the information now available. All those nodes form the layer $\mathcal{N}_1^\Omega = \big\{(r, \xi),\ \xi \in \Xi_1\big\}$ representing stage 1. Further, we add each arc (r, ν) for $\nu \in \mathcal{N}_1^\Omega$ to $\mathcal{A}^\Omega$, and the weight on the arc is given by $w^\Omega(r, \nu) = \mathbb{P}(\boldsymbol{\xi_1} = \xi)$. Moving to the next stage, $t = 2$, in addition of knowing what happened in stage 1, we discover the noise realization of $\boldsymbol{\xi}_2$. For a noise realization $\xi_2 \in \Xi_2$, knowing ξ_1 happened before, we construct a node of the form $\nu = (\mu, \xi_2)$ with parent $a(\nu) = \mu = (r, \xi_1) \in \mathcal{N}_1^\Omega$, and add (μ, ν) to $\mathcal{A}^\Omega$ with weight $w^\Omega(\mu, \nu) = \mathbb{P}(\boldsymbol{\xi}_2 = \xi_2 | \boldsymbol{\xi}_1 = \xi_1)$. Hence, we define the layer representing stage 2 as $\mathcal{N}_2^\Omega = \big\{\ (\nu, \xi),\ \in \mathcal{N}_1 \times \Xi_2\big\}$ *i.e.,* all possible knowledge situation at $t = 2$. Recursively, we construct for all stage t the layer $\mathcal{N}_t^\Omega$,

$$\mathcal{N}_t^\Omega = \Big\{(\nu, \xi) \mid \nu \in \mathcal{N}_{t-1},\ \xi \in \Xi_t\Big\}$$

as described in Algorithm 8.

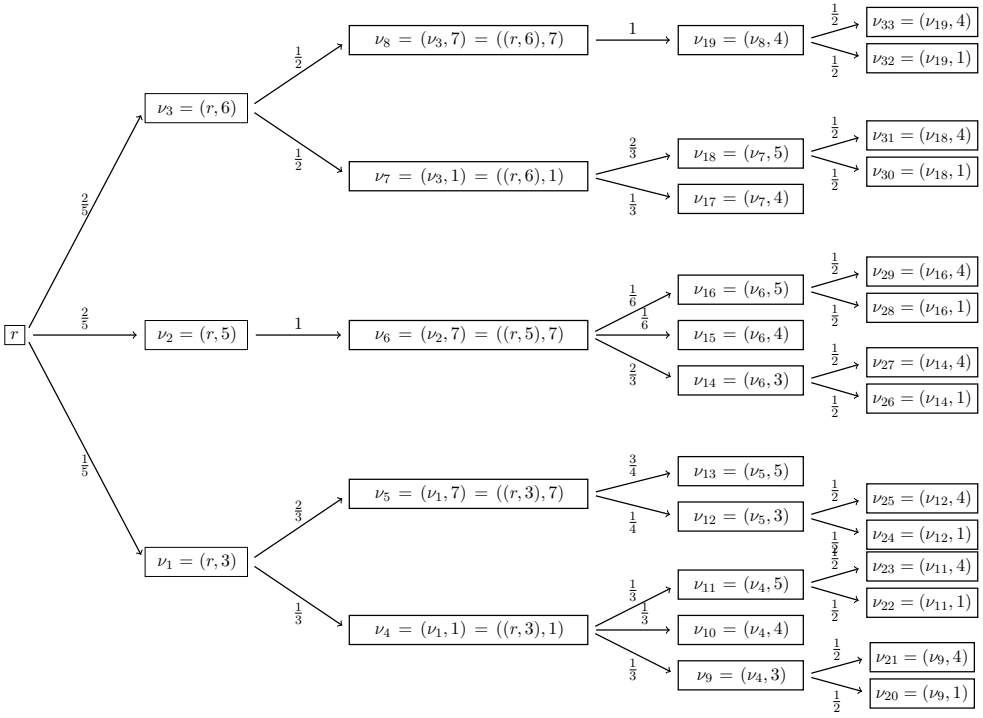


Figure 4.5: Example of a scenario tree $\mathcal{T}_\Omega$

It follows that a node $\nu \in \mathcal{N}_t^\Omega$ in the scenario tree $\mathcal{T}^\Omega$ contains all the information available up to stage t, aligning with the *non-anticipativity constraints* (2.8e), which states that in ν we know the past and present but not the future. Then, $\nu \in \mathcal{N}_t^\Omega$ reads its ancestors' information from the root r:

$$\nu = (\underbrace{a(\nu)}_{\in \mathcal{N}_{t-1}^\Omega}, \xi_\nu) = \Big(\big(\underbrace{a(a(\nu))}_{\in \mathcal{N}_{t-2}^\Omega}, \xi_{a(\nu)}\big), \xi_\nu\Big),$$

where $\xi_\nu \in \Xi_t$ and $\xi_{a(\nu)} \in \Xi_{t-1}$. If we denote π_ν is the probability of being in node ν, we have:

$$\pi_\nu = \mathbb{P}(\boldsymbol{\xi}_t = \xi_t, a(\nu)) = \mathbb{P}(\boldsymbol{\xi}_t = \xi_t | \, a(\nu)\,)\, \pi_{a(\nu)} = w^\Omega(a(\nu), \nu)\, \pi_{a(\nu)}.$$

By convention, $\pi_r = 1$. Thus, each layer $\mathcal{N}_t^\Omega$ does represent clearly a stage of the problem, as it contains the nodes representing all possible scenarios $\{\xi_\tau^k\}_{\tau\in[t]} \in \prod_{\tau=1}^t \Xi_\tau$ up to stage t.

We give an example of a small scenario tree in Figure 4.5, with nodes containing the information accumulated over time. In this example, $T = 4$ and $\Xi_1 = \{3, 5, 6\}$, $\Xi_2 = \{1, 7\}$, $\Xi_3 = \{3, 4, 5\}$, $\Xi_4 = \{1, 4\}$. The probability law $\mathbb{P}$ can be retrieved from the weights $w(\mu, \nu)$ on the figure.

4.1.3 Dynamic programming on a scenario tree

The extensive formulation (2.9) of an MSbLP, presented in Chapter 2, relies on the structure of the scenario tree $\mathcal{T}^\Omega$ we build with Algorithm 8. In practice, the size of $\mathcal{T}^\Omega$ is very large, as it is exponential in the number of stages, and the resulting MILP is intractable. Thus, by considering a subproblem derived from a subtree $\mathcal{T} \subset \mathcal{T}^\Omega$ of the initial scenario tree, we can obtain solvable problems from which we hope to reconstruct a solution of Problem (2.8). We formulate the subproblems $(P_{x_0}^{\mathcal{T}})$ of Problem (2.8) constructed from any ν_0-rooted subtree $\mathcal{T} = \mathcal{T}^\Omega[\mathcal{N}]$:

$$
\begin{aligned}
(P_{x_0}^{\mathcal{T}}) \qquad \min_{x,y,b} \quad & \frac{1}{\pi_{\nu_0}} \sum_{\nu\in\mathcal{N}\setminus\{r\}} \pi_\nu L_\nu(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) && && \text{(4.1a)}\\
& x_\nu = F_\nu(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \subset \mathfrak{X}_\nu && \forall \nu \in \mathcal{N}\setminus\{r\} && \text{(4.1b)}\\
& y_\nu \in \mathfrak{Y}_\nu(x_{a(\nu)}, \xi_\nu) && \forall \nu \in \mathcal{N}\setminus\{r\} && \text{(4.1c)}\\
& b_\nu \in \mathfrak{B}_\nu(x_{a(\nu)}, \xi_\nu) \cap \{0,1\}^{n_b} && \forall \nu \in \mathcal{N}\setminus\{r\} && \text{(4.1d)}\\
& x_{a(\nu_0)} = x_0 && && \text{(4.1e)}
\end{aligned}
$$

where ξ_ν is the information unveiled in node ν. If $\mathcal{T}$ is $r-$rooted, the constraints in Problem (4.1) hold for all nodes in the subtree except r. This is because we consider a setting where there are no decisions at the root, before the first stage[2]. In Equation (4.1a), we recover the probability $\pi_{\nu|\mathcal{T}}$ of being in a node ν in the subtree $\mathcal{T}$ by dividing the probabilities by the probability of being in node ν_0. More specifically, if $(\nu_0, \nu_1, \dots, \nu_k, \nu)$ is the unique path from ν_0 to ν:

$$
\pi_{\nu|\mathcal{T}} = w(\nu_k, \nu) \prod_{t=0}^{k-1} w(\nu_t, \nu_{t+1}) = \frac{\pi_\nu}{\pi_{\nu_0}}.
$$

Though this framework is abstract, we do fall back on Problem (2.8) by considering $(P_{x_{init}}^{\mathcal{T}^\Omega})$. On the one hand, for pruned subtrees $\mathcal{T}_p$, solving $(P_{x_0}^{\mathcal{T}_p})$ amounts to assuming some decisions in the future are unnecessary to construct an optimal policy. On the other hand, for $\nu-$rooted subtrees $\mathcal{T}$, with ν of depth $t_\nu > 0$, $(P_{x_0}^{\mathcal{T}})$ can be interpreted as assuming a certain scenario up to t_ν has happened, resulting in state x_0. In particular, with $\nu-$extracted subtrees $\mathcal{T}_{|\nu}$, we do not cut any potential future scenarios $\{\xi_t\}_{t\in[t_\nu:T]}$. In a multistage setting, we assume at a given node ν, the whole information needed to make optimal decisions is contained in the *state* of the dynamic system. Thus, we introduce the *cost-to-go* functions, $\{\hat{V}_\nu\}_{\nu\in\mathcal{N}}$, as the optimal values of problem $(P_x^{\mathcal{T}_{|\nu}})$ from a node ν depending on the current state of the system x.

[2] However, we could adapt our formulation, to a problem with decisions at the root, by adding an initial stage $t = 0$ with a static noise $\boldsymbol{\xi_0} = \xi_0$

In the same way that we used Bellman's DP principle to obtain recursions (2.11), we have:

$$\hat{V}_\nu(x) = \min_{z,y,b} \quad L_\nu(x,y,b,\xi_\nu) + \sum_{\mu\in\mathcal{C}(\nu)} w(\nu,\mu)\hat{V}_\mu(z) \tag{4.2a}$$

$$z = F_\nu(x,y,b,\xi_\nu) \subset \mathfrak{X}_\nu \tag{4.2b}$$

$$y \in \mathfrak{Y}_\nu(x,\xi_\nu) \tag{4.2c}$$

$$b \in \mathfrak{B}_\nu(x,\xi_\nu) \cap \{0,1\}^{n_b}. \tag{4.2d}$$

$$V_r(x_0) = \sum_{\mu\in\mathcal{C}(r)} w(r,\mu)\hat{V}_\mu(x_0) \tag{4.2e}$$

In comparison with Chapter 2, we make no assumption on independence of variables $\{\boldsymbol{\xi_t}\}_{t\in[T]}$. Thus, the cost-to-go value $\hat{V}_\nu(x)$ for $\nu \in \mathcal{N}_t^\Omega$ is the optimal cost from state x knowing all the information contained in ν, in particular, knowing the noise realization ξ_ν at t. Despite each problem being, individually, small enough to be solved efficiently, the number of cost-to-go functions to compute is as large as the scenario tree $\mathcal{T}^\Omega$, making the dynamic formulation intractable. Even with strong assumptions like stagewise independence (see Chapter 2), dynamic programming is impractical due to the *curse of dimensionality.*

A natural simplification of a multistage program, with a very large number of stages T, is to reduce the problem to a shorter horizon $\tau << T$. This makes sense in a problem where decisions made at t impact less and less the costs in future stages. This is what we propose in Chapter 3, where we implicitly solve our problem reduced to a $2-$shortsighted subtree. We retrieve a forward policy with bellman operators that are associated to $(P_x^{\mathcal{T}_{1:2}})$. In this section, we leverage the definitions introduced in Section 4.1.1 to generalize bellman operators Chapter 3 to any ν_0-rooted subtree $\mathcal{T} = (\mathcal{N},\mathcal{A},w) \subset \mathcal{T}^\Omega$. Assuming we have approximated cost-to-go functions $\{\hat{\mathcal{R}}_\nu\}_{\nu\in\mathcal{N}^\Omega}$, we define the $\mathcal{T}-$backward operator $\hat{\mathcal{B}}_\mathcal{T}(\{\hat{\mathcal{R}}_\mu\}_{\mu\in\mathcal{C}(\mathcal{N})})$:

$$\hat{\mathcal{B}}_\mathcal{T}(\{\hat{\mathcal{R}}_\mu\}_{\mu\in\mathcal{C}(\mathcal{N})})(x_0) = \operatorname*{Min}_{x,y,b} \frac{1}{\pi_{\nu_0}}\left[\sum_{\nu\in\mathcal{N}\setminus\{r\}} \pi_\nu L_\nu(x_{a(\nu)},y_\nu,b_\nu,\xi_\nu) + \sum_{\mu\in\mathcal{C}(\mathcal{N})} \pi_\mu \hat{\mathcal{R}}_\mu(x_{a(\mu)})\right]$$

$$x_\nu = F_\nu(x_{a(\nu)},y_\nu,b_\nu,\xi_\nu) \subset \mathfrak{X}_\nu \qquad \forall \nu \in \mathcal{N}\setminus\{r\} \tag{4.3a}$$

$$y_\nu \in \mathfrak{Y}_\nu(x_{a(\nu)},\xi_\nu) \qquad \forall \nu \in \mathcal{N}\setminus\{r\} \tag{4.3b}$$

$$b_\nu \in \mathfrak{B}_\nu(x_{a(\nu)},\xi_\nu) \cap \{0,1\}^{n_b} \qquad \forall \nu \in \mathcal{N}\setminus\{r\} \tag{4.3c}$$

$$x_{a(\nu_0)} = x_0 \tag{4.3d}$$

In particular, with a ν_0-rooted subtree $\mathcal{T} = \mathcal{T}^\Omega[\{\nu_0\}]$, containing only node ν_0, $\hat{\mathcal{B}}_\mathcal{T}(\{\hat{V}_\mu\}_{\mu\in\mathcal{C}(\mathcal{N})})(x)$ is equivalent to the problem formulated in (4.2). More generally, for any ν_0-rooted subtree $\mathcal{T} = \mathcal{T}^\Omega[\mathcal{N}]$, if we know the true cost-to-go functions $\{\hat{V}_\mu\}_{\mu\in\mathcal{C}(\mathcal{N})}$, then we have $\hat{V}_\nu = \hat{\mathcal{B}}_\mathcal{T}(\{\hat{V}_\mu\}_{\mu\in\mathcal{C}(\mathcal{N})})$. To simulate a policy on any scenario, we derive the $\mathcal{T}-$forward operator

associated to $\hat{\mathcal{B}}_{\mathcal{T}}(\{\hat{\mathcal{R}}_\mu\}_{\mu\in\mathcal{C}(\mathcal{N})})$:

$$\hat{\mathcal{F}}_{\mathcal{T}}(\{\hat{\mathcal{R}}_\mu\}_{\mu\in\mathcal{C}(\mathcal{N})})(x_0) \in \arg\min_{x,y,b} \frac{1}{\pi_{\nu_0}}\Big[\sum_{\nu\in\mathcal{N}\setminus\{r\}} \pi_\nu L_\nu(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) + \sum_{\mu\in\mathcal{C}(\mathcal{N})} \pi_\mu \hat{\mathcal{R}}_\mu(x_{a(\mu)})\Big]$$

$$x_\nu = F_\nu(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \subset \mathfrak{X}_\nu \qquad \forall \nu \in \mathcal{N}\setminus\{r\} \tag{4.4a}$$

$$y_\nu \in \mathfrak{Y}_\nu(x_{a(\nu)}, \xi_\nu) \qquad \forall \nu \in \mathcal{N}\setminus\{r\} \tag{4.4b}$$

$$b_\nu \in \mathfrak{B}_\nu(x_{a(\nu)}, \xi_\nu) \cap \{0,1\}^{n_b} \qquad \forall \nu \in \mathcal{N}\setminus\{r\} \tag{4.4c}$$

$$x_{a(\nu_0)} = x_0 \tag{4.4d}$$

With these operators, we have a way to compare the policies induced by different approximations of the cost-to-go functions. There are two main challenges in this approach: first, deciding which subtree $\mathcal{T}$ leads to good policies with the $\mathcal{T}$-operators; second, which approximations $\{\hat{\mathcal{R}}_\mu\}$ to use. Then, assuming $\{\hat{\mathcal{R}}_\mu\}$ are piecewise linear, we obtain an MILP to solve, where all the information and integrality constraints are considered on a given subtree $\mathcal{T}$. Our goal is to find the right trade-off to compute a *good solution*, meaning that when we simulate the policy of the approximated cost-to-go functions with $\mathcal{T}-$forward operators (4.4), the solution obtained has a reasonable cost.

This framework gives the flexibility of choosing the size of the subproblems we solve. On the one side, with the smallest subproblems formulated with subtrees containing only node $\mathcal{T}_\nu = \mathcal{T}^\Omega[\{\nu\}]$, and with stagewise independence assumptions, we fall back on classic dynamic programming methods presented in Chapter 2 *i.e.,* $\hat{\mathcal{B}}_{\mathcal{T}_\nu}(\mathcal{R})(x) = \hat{\mathcal{B}}_{t_\nu}(\mathcal{R})(x, \xi_\nu)$. Then, the method requires high-quality approximations $\{\hat{\mathcal{R}}_\mu\}$. On the other side, the largest subtree we can choose is the initial scenario tree, and $\hat{\mathcal{B}}_{\mathcal{T}^\Omega}(0)$ is equivalent to the extensive formulation of Problem (2.8). There, no approximated functions are needed as $\mathcal{C}(\mathcal{N}^\Omega) = \emptyset$. This is also the case with large subtrees such as $\nu-$extracted subtrees $\mathcal{T}_{|\nu}$, where $\mathcal{C}(\mathcal{N}^\Omega_{|\nu}) = \emptyset$. In between, with a pruned subtree $\mathcal{T}_p$ of medium size, we might have an MILP of reasonable size. Assuming the decisions made at stage t do not affect so much the decisions in the far future ($\tau \gg t$), and with a rolling horizon heuristic, we hope to find good policies in a reasonable time by solving the problem on a pruned tree.

4.1.4 Stagewise independence

We presented a generic formulation for MSbLPs for any probability space Ω that can be represented with a scenario tree. In Chapter 2, we presented the specific case where the random variables $\{\boldsymbol{\xi}_t\}_{t\in[T]}$ are stagewise independent, *i.e.,* for any scenario $\{\xi_\tau\}_{\tau\in[t]}$ up to stage t, the probability of having noise realization $\xi_t \in \Xi_t$ is the same. In this case, many cost-to-go functions $\{\hat{V}_\nu\}_{\nu\in\mathcal{N}}$ are redundant. Indeed, for two nodes of the same layer $\mathcal{N}^\Omega_t$ in the scenario tree, the subproblems to solve are the same as long as the information unveiled in stage t is the same:

$$\hat{V}_\nu(x) = \hat{V}_{\nu'}(x) = \hat{V}_t(x,\xi) \qquad \forall x, \forall \nu, \nu' \in \mathcal{N}^\Omega_t \text{ with } \xi_\nu = \xi_{\nu'} = \xi \in \Xi_t. \tag{4.5a}$$

Leveraging the bellman recursive equations (4.2), we can reduce the number of cost-to-go functions to one per stage, leading to the cost-to-go functions presented in Chapter 2:

$$\sum_{\mu\in\mathcal{C}(\nu)} w(\nu,\mu)\hat{V}_\mu(x) = \sum_{\mu\in\mathcal{C}(\nu)} \mathbb{P}(\boldsymbol{\xi}_{t+1} = \xi_\mu)\hat{V}_{t+1}(x,\xi_\mu) \tag{4.5b}$$

$$= \mathbb{E}[\hat{V}_{t+1}(x, \boldsymbol{\xi}_{t+1})] \tag{4.5c}$$

$$= V_{t+1}(x) \qquad \forall \nu \in \mathcal{N}^\Omega_t, \forall y. \tag{4.5d}$$

Making the hypothesis of stagewise independence thus reduces significantly the computational burden of solving Problem (2.9) with dynamic programming principles. In this section, we generalize the idea of stagewise independence to $t-$independence: in a $t-$independent multistage stochastic program, no assumptions are made on the random variables up to stage t, and from stage $t+1$ to T, the random variables are stagewise independent.

To that end, we want to compare subtrees from a given layer of the tree, with no knowledge of the past. Those subtrees should look like $\nu-$extracted subtrees $\mathcal{T}_{|\nu}$, except those contain the information of the path leading to ν in $\mathcal{T}^{\Omega}$. Thus, we define the $\nu-$*uprooted subtree* $\mathcal{T}_{\backslash\nu}$, as a scenario tree with the structure of $\mathcal{T}_{|\nu}$ from which we have removed the information in $a(\nu)$. The root r of $\mathcal{T}_{\backslash\nu}$ contains the noise realization revealed in node ν, and form the first layer of the subtree $\{r\} = (r, \xi_\nu)$ (recall that by convention $a(r) = r$). Then, the second layer consists of nodes $\{\mu = (r, \xi_\mu),\ \mu \in \mathcal{C}(\nu)\}$. Recursively, we construct $\mathcal{T}_{\backslash\nu}$ with an analog algorithm of Algorithm 8.

We can observe the difference between $\mathcal{T}_{|\nu}$ and $\mathcal{T}_{\backslash\nu}$ in Figures 4.6 and 4.7. The structure of ν-uprooted subtrees allow us to define $t-$independence in Definition 1.

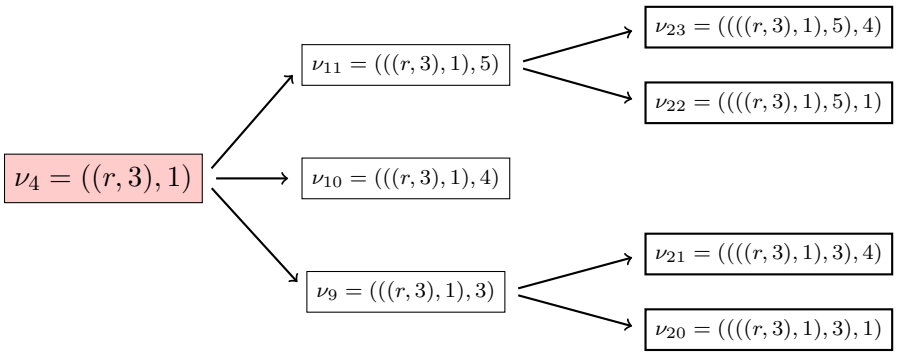


Figure 4.6: Example of an extracted subtree $\mathcal{T}_{|\nu_4}$ from scenario tree in Figure 4.5.

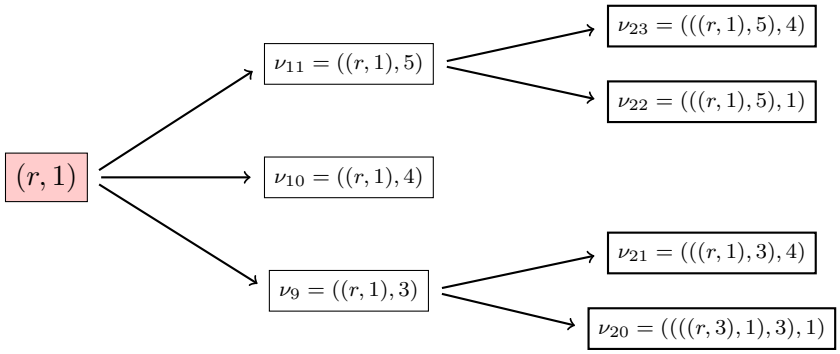


Figure 4.7: Example of an uprooted-subtree: $\mathcal{T}_{\backslash\nu_4}$ of the scenario tree in Figure 4.5.

Definition 1. *We say that the scenario tree $\mathcal{T}$ is $t-$independent if for all $\tau \geq t$ and for all nodes $\nu, \nu' \in \mathcal{N}_\tau^\Omega \cap \mathcal{N}$, we have $\mathcal{T}_{\backslash\nu} = \mathcal{T}_{\backslash\nu'}$. In particular, a $1-$independent scenario tree models a problem with stagewise independence assumptions.*

The notion of $t-$independence, with $t > 1$, is a trade-off between simplifying too much the problem by assuming noises are stagewise independent and reducing the computational burden to solve the problem.

Proposition 1. *If the scenario tree $\mathcal{T}$ is $t-$independent, then there is a unique cost-to-go function*

representing stage $t+1$:

$$\sum_{\mu \in \mathcal{C}(\nu)} w(\nu,\mu)\hat{V}_\mu(x) = \sum_{\mu \in \mathcal{C}(\nu)} \mathbb{P}(\boldsymbol{\xi}_{t+1} = \xi_\mu)\hat{V}_{t+1}(x,\xi_\mu) = V_{t+1}(x), \qquad \forall \nu \in \mathcal{N}_t^\Omega.$$

Thus, t-independence allows the use of an efficient DP approach, that is still subject to the curse of (state) dimensionality. However, we have seen that SDDP can push this limit if the state is continuous. For example, under stagewise independence assumptions, we can solve the continuous relaxation of Problem (2.8) with SDDP and retrieve an approximation of each value function $\hat{V}_t^{SDDP}$. Then, by Proposition 1, the problem is $\tau+1$-independent and there is a unique cost-to-go function representing $\tau+1$, thus the $\mathcal{T}_{1:\tau}-$backward operator $\mathcal{B}_{\mathcal{T}_{1:\tau}}(\{\hat{V}_{\tau+1}^{SDDP}\})$ is the same operator as the one introduced in Chapter 3.

With branch and bound methodology in mind, we present, in the next section, tools for relaxing integrality constraints. From now on, we assume the MSbLP at hand is stagewise independent.

4.2 Assignation functions

In MSbLP, with stagewise independence – which is assumed from now on – the main difficulty comes from the binary variables. Indeed, if we suppress integrality constraints, we obtain a MSLP which can be efficiently solved with SDDP. There are two main ways to suppress integrality constraints: either we relax them and make binary variables continuous, or we fix the variables to a fixed value (0 or 1). Thus, we would like to reduce the problem to multiple continuous subproblems by resorting to BB methods. To this end, this section introduces specific functions that reduce, enforce, or extend the feasibility set of binary variables.

4.2.1 Definition and parameterized Dynamic Programming

We present a generic framework that allows for both relaxing integrality constraints and reducing them by fixing the value of binary variables. This allows us to explore different options for best modifying the problem's feasibility set depending on its structure. Further, the framework encompasses all approximations we can make of an MSbLP by relaxing or fixing integrality constraints.

Definition 2. *An assignation function $\mathfrak{b} : \mathcal{T} \to \{\{0\},\{1\},\{0,1\},[0,1]\}^{n_b}$ is a function that assigns to each binary variables, in each node of the tree, a new feasible space. For each binary variable, there are three possibilities:*

1. *the feasibility space is reduced to a single value (0 or 1): the variable becomes a constant;*
2. *the feasibility space remains the same: the variable stays binary;*
3. *the feasibility space is relaxed: the variable is now continuous.*

Then, we define the parameterized scenario tree $\mathcal{T}^{\Omega,\mathfrak{b}}$ as a scenario tree where the definition of a node is extended with the feasibility set assigned by $\mathfrak{b}$:

$$\nu = (a(\nu), \mathfrak{b}(\nu), \xi_\nu).$$

For example, we consider the scenario tree $\mathcal{T}^\Omega$ in Figure 4.8, which has 4 stages and $|\Xi_1| = 3, |\Xi_2| = |\Xi_3| = |\Xi_4| = 2$. If the number of binary variables per node $n_b = 2$, we represent a parameterized tree $\mathcal{T}^{\Omega,\mathfrak{b}}$ in Figure 4.9. To alleviate the figure, we only write the assignation value in each node.

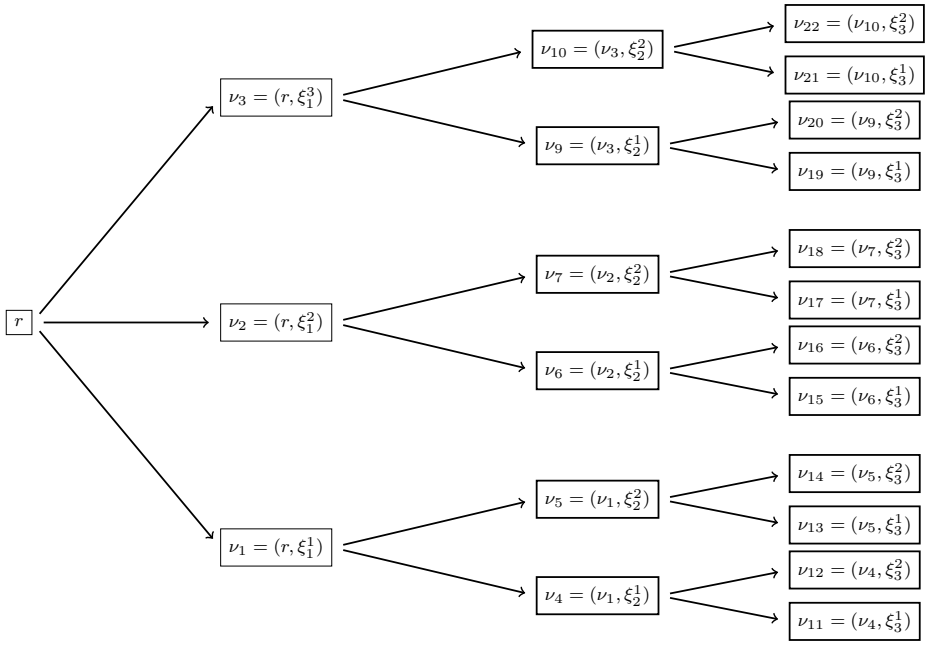


Figure 4.8: Example of a scenario tree $\mathcal{T}^{\Omega}$.

We observe that the assignation operation on the scenario tree in Figure 4.9 does not preserve stagewise independence. Indeed, the binary assignation in node ν_9 is different then the one in ν_7, and thus the cost-to-go functions from node ν_3 and node ν_2 might differ.

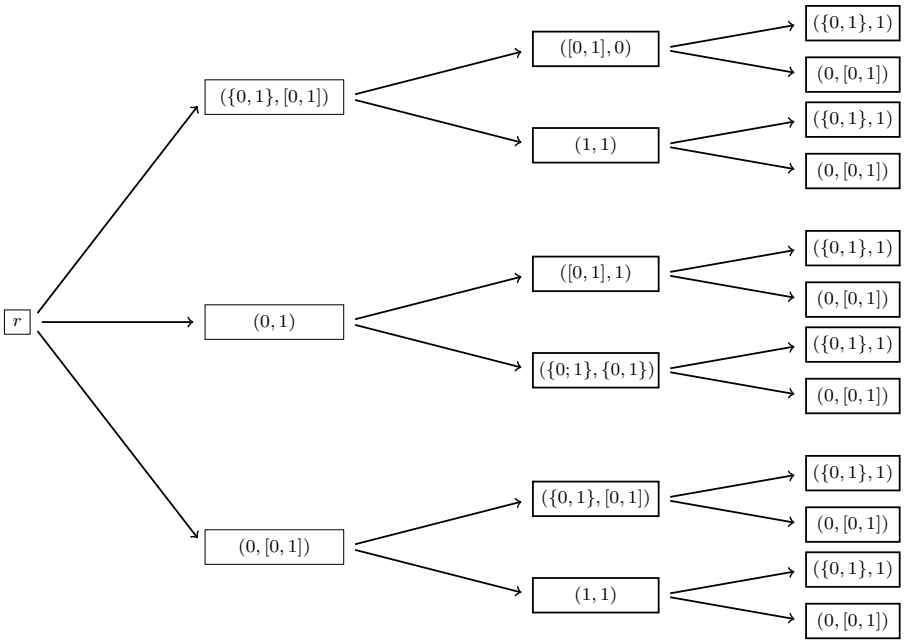


Figure 4.9: Example of a parameterized scenario subtree.

Finally, we adapt the formulation (4.1) of an MSbLP on a subtree to a parameterized subtree

$\mathcal{T}^{\mathfrak{b}}$:

$$(P_{x_0}^{\mathcal{T}^{\mathfrak{b}}}) \quad \min_{x,y,b} \ \frac{1}{\pi_{\nu_0}} \sum_{\nu \in \mathcal{N}\setminus\{r\}} \pi_\nu L_{t_\nu}(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \tag{4.6a}$$

$$x_\nu = F_{t_\nu}(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \qquad \forall \nu \in \mathcal{N}\setminus\{r\} \tag{4.6b}$$

$$y_\nu \in \mathcal{U}_{t_\nu}(x_{a(\nu)}, \xi_\nu) \qquad \forall \nu \in \mathcal{N}\setminus\{r\} \tag{4.6c}$$

$$b_\nu \in \mathfrak{b}(\nu) \qquad \forall \nu \in \mathcal{N}\setminus\{r\} \tag{4.6d}$$

$$x_{a(\nu_0)} = x_0. \tag{4.6e}$$

The only difference between Problem (4.6) and Problem (4.1), for a given subtree $\mathcal{T}$, lies in the constraints (4.6d), where we modify the feasible set of binary variables. Further, under stagewise independence assumptions, the dynamic and instantaneous costs only depend on the stage t_ν, thus we simplify the objective(4.6a) and dynamic constraints(4.6b). However, as illustrated in Figure 4.9, the problem parameterized by $\mathfrak{b}$ is not necessarily stagewise independent. Thus, we introduce the parameterized cost-to-go functions $\{\hat{V}_\nu^{\mathfrak{b}}\}_{\nu\in\mathcal{N}}$ which represent the optimal cost of the parameterized problem from a node ν. Then, we can adapt the $\mathcal{T}-$backward operators (5.4) to the parameterized $\mathcal{T}^{\mathfrak{b}}-$backward operators $\hat{\mathcal{B}}_{\mathcal{T}^{\mathfrak{b}}}$ by changing the feasible set of binary variables:

$$\hat{\mathcal{B}}_{\mathcal{T}^{\mathfrak{b}}}(\{\hat{\mathcal{R}}_\mu\}_{\mu\in\mathcal{C}(\mathcal{N})})(x_0) = \operatorname*{Min}_{x,y,b} \frac{1}{\pi_{\nu_0}} \left[\sum_{\nu \in \mathcal{N}\setminus\{r\}} \pi_\nu L_{t_\nu}(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) + \sum_{\mu\in\mathcal{C}(\mathcal{N})} \pi_\mu \hat{\mathcal{R}}_\mu(x_{a(\mu)}) \right]$$

$$x_\nu = F_{t_\nu}(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \subset \mathfrak{X}_{t_\nu} \qquad \forall \nu \in \mathcal{N}\setminus\{r\} \tag{4.7a}$$

$$y_\nu \in \mathfrak{Y}_{t_\nu}(x_{a(\nu)}, \xi_\nu) \qquad \forall \nu \in \mathcal{N}\setminus\{r\} \tag{4.7b}$$

$$b \in \mathfrak{b}(\nu) \qquad \forall \nu \in \mathcal{N}\setminus\{r\} \tag{4.7c}$$

$$x_{a(\nu_0)} = x_0. \tag{4.7d}$$

Finally, by adapting similarly the constraints, we derive the associated parameterized $\mathcal{T}^{\mathfrak{b}}-$forward operators $\hat{\mathcal{F}}_{\mathcal{T}^{\mathfrak{b}}}$.

Having introduced assignation functions and parameterized scenario trees, we now present some properties of some particular assignation functions. Remember that our primary goal is to leverage SDDP cuts that we can quickly obtain and that approximate well the continuous relaxation of an MSbLP. Note that these assignation functions allow us to generalize a framework containing many approximations of Problem (2.8).

4.2.2 Some specific assignation functions

In all the assignation functions that can approximate Problem (2.8), we first study those that give us lower and upper bounds on the optimal value of the problem.

A natural approximation of MSbLP is obtained by relaxing completely integrality constraints. The resulting problem is a multistage stochastic linear program MSLP which can be solved efficiently with SDDP. This specific case is covered by a unique assignation function that we call the *relaxed-assignation* $\mathfrak{b}^r$, which relaxes all integrality constraints of the problem:

$$\mathfrak{b}^r : \mathcal{T} \to [0,1]^{n_b}. \tag{4.8a}$$

Then, we can run SDDP to solve the problem modeled by the parameterized scenario tree $\mathcal{T}^{\Omega,\mathfrak{b}^r}$. We obtain cuts converging to the true cost-to-go $\hat{V}_\nu^{\mathfrak{b}^r}$ at each node $\nu \in \mathcal{T}^{\Omega,\mathfrak{b}^r}$, and in particular

to the optimal value of the parameterized problem $\hat{V}_r^{\mathfrak{b}^r}(x_{init})$, which is a valid lower-bound on the optimal value $\hat{V}_r(x_{init})$ of Problem (2.8).

We can generalize a category of assignation functions yielding lower-bounds on the true value of the problem. A *lower-assignation* $\mathfrak{b}^{lb}$ is an assignation function that partially relaxes the integrality constraints of the problem and enforces the rest of them:

$$\mathfrak{b}^{lb} : \mathcal{T} \to \{\{0,1\},[0,1]\}^{n_b}. \tag{4.8b}$$

For any assignation function $\mathfrak{b}^{lb}$, the problem modeled on $\mathcal{T}^{\mathfrak{b}^{lb}}$ is a relaxation of Problem (2.8) and we have:

$$\hat{V}_r^{\mathfrak{b}^r}(x_{init}) \leq \hat{V}_r^{\mathfrak{b}^{lb}}(x_{init}) \leq \hat{V}_r(x_{init}).$$

However, solving $(P_{x_0}^{\mathcal{T}^{\mathfrak{b}^{lb}}})$ is not straightforward and can be as hard as solving Problem (2.8). We present in Chapter 5 some particular cases of lower-assignations where we can compute $\hat{V}_r^{\mathfrak{b}^{lb}}(x_{init})$.

On the other hand, we now present assignation functions that give upper-bounds on the problem. We call *fixed-assignation* $\mathfrak{b}^f$ an assignation function where all binary variables are fixed to either 0 or 1:

$$\mathfrak{b}^f : \mathcal{T} \to \{\{0\},\{1\}\}^{n_b}. \tag{4.8c}$$

For any fixed-assignation $\mathfrak{b}^f$, either the problem modeled with $\mathcal{T}^{\Omega,\mathfrak{b}^f}$ has a solution which is feasible for Problem (2.8) and $\hat{V}_r(x_{init}) \leq \hat{V}_r^{\mathfrak{b}^f}(x_{init})$; or the problem is infeasible, $\hat{V}_r^{\mathfrak{b}^f}(x_{init}) = \inf$, and the inequality holds. In particular, for any fixed-assignation $\mathfrak{b}^{f_0} \in \{\{0\},\{1\}\}^{|\mathcal{N}^\Omega| \times n_b}$,

$$\hat{V}_r(x_{init}) = \min_{\mathfrak{b}^f \in \{\{0\},\{1\}\}^{|\mathcal{N}^\Omega| \times n_b}} \hat{V}_r^{\mathfrak{b}^f}(x_{init}) \leq \hat{V}_r^{\mathfrak{b}^{f_0}}(x_{init}).$$

Computing $\hat{V}_r^{\mathfrak{b}^f}(x_{init})$ is not easy: indeed, even if the problem is continuous, to solve it with SDDP, $\mathcal{T}^{\mathfrak{b}^f}$ has to be stagewise independent.

We can enlarge the class of assignation functions yielding upper-bounds on the true value of the problem to *upper-assignation* $\mathfrak{b}^{ub}$ *i.e.,* assignation function that either fixes binary variables to 0 or 1 or keeps integrality constraints

$$\mathfrak{b}^{ub} : \mathcal{T} \to \{\{0\},\{1\},\{0,1\}\}^{n_b}. \tag{4.8d}$$

In particular, a fixed-assignation function is an upper-assignation function. The problems parameterized with upper-assignation functions are also hard to solve. Finally, for any upper-assignation function $\mathfrak{b}^{ub}$:

$$\hat{V}_r(x_{init}) \leq \hat{V}_r^{\mathfrak{b}^{ub}}(x_{init}).$$

As the number of assignation functions is very large, we naturally turn to BB methods to decide which feasibility set to assign to each node.

4.2.3 A Branch-and-Bound methodology

The Branch-and-Bound (BB) algorithmic framework encapsulates a class of algorithms that essentially follow the same procedure. Consider the problem (P) $\text{Min}_{x \in X} f(x)$, of minimizing a function f over a finite set X. As X is finite, the simplest approach would be to enumerate all elements of X and evaluate f on each. The concept of BB is to consider all elements without

having to evaluate f on each by the use of bounding tools. More precisely, we structure the exploration of X by using a specific tree that we call BB-tree to avoid confusion with the scenario tree. In the BB-tree, a BB-vertex corresponds to a subset $X' \subset X$, with the corresponding subproblem (SP) : $\text{Min}_{x \in X'} f(x)$. From a BB-vertex we generate children by partitioning the feasible set X': this is referred to as *branching*. For each BB-vertex we assume that we can compute a lower bound. If this lower bound is higher than the current best upper bound (the value of the best solution $x \in X$ evaluated so far), we know that there is no use in refining further this node: this is *pruning*.

We distinguish three main ingredients that are to be specified in a BB algorithm: first, the *search* strategy, *i.e.,* in which order to explore the BB-tree; second, the *branching* strategy, *i.e.,* the way we partition the feasible set of a given vertex to produce children; third, the *pruning* rules, *i.e.,* rules to decide when to stop exploring areas of the tree. Depending on these three elements, the resulting algorithm, generalized in Algorithm 9, can be more or less efficient.

Algorithm 9: Branch-and-Bound procedure.

Input : initial feasible set X, vertices to explore $V = \{X\}$, current solution $\overline{x} \in X$;
while $V \neq \emptyset$ **do**
  Select $v \in V$ corresponding to a subproblem ; `// search strategy`
  if *we can find* $x' \in v$ *such that* $f(x') < f(\overline{x})$ **then** `// heuristic`
    Set $\overline{x} = x'$;
  Compute lower-bound $\underline{f}(v)$; `// bounding`
  if $\underline{f}(v) \leq f(\overline{x})$ **then** `// v cannot be pruned`
    Partition V into $v_1, \ldots, v_k$; ; `// branching strategy`
    Insert $v_1, \ldots, v_k$ in V;
  Remove v from V;
Return $\overline{x}$;

BB methods are particularly suitable for problems with binary variables. We consider a binary problem $(P) : \text{Min}_{x \in X \subset \{0,1\}^n} f(x)$. A BB-vertex v in the BB-tree corresponds to a subproblem $(SP) : \text{Min}_{x \in v \subset X} f(x)$ where some binary variables have been fixed *i.e.,* $v \subseteq \{0,1\}^{n-n_0-n_1} \times \{0\}^{n_0} \times \{1\}^{n_1}$. Then, a natural branching strategy from BB-vertex v is to choose x_k unfixed in v and to fix either $x_k = 0$ or $x_k = 1$. Further, a lower-bound can be computed by solving the continuous relaxation of (SP). We precise the BB procedure for the binary case in Algorithm 10.

Due to the structure of the assignation functions $\mathfrak{b}$ introduced in this section, we want to derive an algorithm inspired from BB to find the optimal one $\mathfrak{b}^\star$. Whereas in BB, we look for the optimal solution of a problem, we look for the assignation $\mathfrak{b}$ that gives the best solvable approximation of Problem (2.8). Thus, the goal is slightly different, but we can adapt it by choosing suitable pruning and bounding strategies. Starting from the *initial assignation* of Problem (2.8) *i.e.,* $\mathfrak{b}^{int}(\mathcal{T}^\Omega) = \{0,1\}^{|\mathcal{N}| \times n_b}$, we progressively change the assignation in a BB pattern. A BB-vertex v corresponds to an assignation function $\mathfrak{b}^v$, where the assignation of nodes $\mathcal{N}_f \subset \mathcal{N}^\Omega$ has been modified. From a vertex v, we branch by: first, choosing a node $\nu \in \mathcal{N}^\Omega \setminus \mathcal{N}_f$, among those not considered yet; then, generating a child u of v such that $\mathfrak{b}^u(\nu) \in \big\{\{0\}, \{1\}, [0,1]\big\}^{n_b}$ and $\mathfrak{b}^u(\mu) = \mathfrak{b}^v(\mu)$ for $\mu \neq \nu$.

We illustrate in Figure 4.10 this BB methodology to decide on an assignation function applied to the scenario tree in Figure 4.8. To simplify the illustration, we only consider 1 binary variable per node. We observe an example of branching from node v where we have already fixed the

Algorithm 10: B&B procedure for binary problems.

Input : initial feasible set $X \subset \{0,1\}^n$, vertices to explore $V = \{X\}$, current solution $\overline{x} \in X$;
while $V \neq \emptyset$ **do**
  Select a vertex $v \in V$; `// search strategy`
  if *we can find* $x' \in v$ *such that* $f(x') < f(\overline{x})$ **then** `// heuristic`
    Set $\overline{x} = x'$;
  Get lower bound $\underline{f}(v)$ by solving the continuous relaxation of (SP) ; `// bounding`
  if $\underline{f}(v) \leq f(\bar{x})$ **then** `// v cannot be pruned`
    Choose x_k unfixed in v; `// binary branching strategy`
    Insert $v \cap \{x_k = 0\}$ and $v \cap \{x_k = 1\}$ in V;
  Remove v from V;
Return $\overline{x}$;

assignation of nodes $\mathcal{N}_f = \{\nu_1, \nu_4, \nu_9, \nu_{16}, \nu_{22}\}$ in Figure 4.10. From node v, with a given search strategy, we choose a node on which to branch: here ν_6. We generate three children corresponding to potential assignation of ν_6: a fixed value 0, 1, or relaxation $[0,1]$. The case where ν_6 stays binary *i.e.,* $\{0,1\}$, is considered by not branching on node ν_6. It is worth pointing out that a natural search order on the nodes is one that, starting from the root r, corresponds to a pruned subtree.

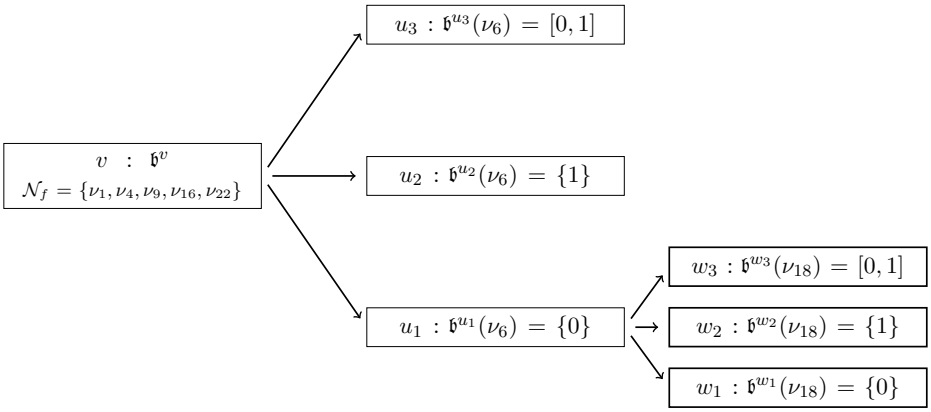


Figure 4.10: A B&B method to construct an assignation function

One of the main challenges we have here is to generate the bounds or find solution with any assignation function (see Section 4.2.2). However, we know that in some cases, for example with the continuous assignation $\mathfrak{b}^c$, we can solve the parameterized problem with SDDP. In the next section, we give conditions under which we can solve $(P^{\mathcal{T}^{\mathfrak{b}}}_{x_{init}})$ with SDDP.

4.3 Merging with SDDP

In order to reduce the computational burden of solving Problem (2.8), we look for ways to use the cuts computed with SDDP. Depending on the assignation function $\mathfrak{b}$, we might be able to approximate portions of the problem with SDDP, and alleviate the complexity of the branch and bound previously presented. In this section, we first discuss the conditions under which we can solve a $\mathfrak{b}-$parameterized problem with SDDP. We then present a particular class of assignation

functions that lead to an efficient branch-and-bound algorithm leveraging SDDP cuts.

4.3.1 Condition for an assignation function to be compatible with SDDP

SDDP is an algorithm that can efficiently solve Multistage Stochastic Linear Program (MSLP) with continuous variables and under stagewise independence assumptions. Thus, to solve the parameterized problem given by $\mathcal{T}^{\mathfrak{b}}$, the resulting subproblem has to be *continuous* and *stagewise independent*. Hence, we look upon interesting properties on assignation functions.

On the one side, we introduce a *continuous-assignation* $\mathfrak{b}^c$ as an assignation function where all binary variables are either fixed or relaxed:

$$\mathfrak{b}^c : \mathcal{T} \to \{\{0\}, \{1\}, [0,1]\}^{n_b}.$$

Then, for any subtree $\mathcal{T}$, the $\mathfrak{b}^c$-parameterized problem $(P_{x_0}^{\mathcal{T}^{\mathfrak{b}}})$ is continuous.

On the other side, we define $t-$symmetry for assignation functions in Definition 3, which is an analog of $t-$independence (see Definition 1).

Definition 3. *We say that the assignation function $\mathfrak{b}$ is $t-$symmetric on subtree $\mathcal{T}$ if for all $\tau \geq t$ and for all nodes ν, ν' in $\mathcal{T}$ of depth τ (*i.e., *$\nu, \nu' \in \mathcal{N}_\tau^\Omega \cap \mathcal{N}$, for all $\xi_{\tau+1} \in \Xi_{\tau+1}$), we have $\mathfrak{b}((\nu, \xi_{\tau+1})) = \mathfrak{b}((\nu', \xi_{\tau+1}))$. In other words, for nodes of a same layer and with same current noise realization, the assignation given by $\mathfrak{b}$ is the same.*

Further, if $\mathcal{T}^{\mathfrak{b}}$ is $\nu-$rooted with $t_\nu = t$ and $\mathfrak{b}$ is $t-$symmetric, then $\mathcal{T}^{\mathfrak{b}}$ is stagewise independent. For instance, in Figure 4.9, the assignation function is $2-$independent, but not $1-$independent. Thus, the parameterized problem corresponding to the figure cannot be solved with SDDP.

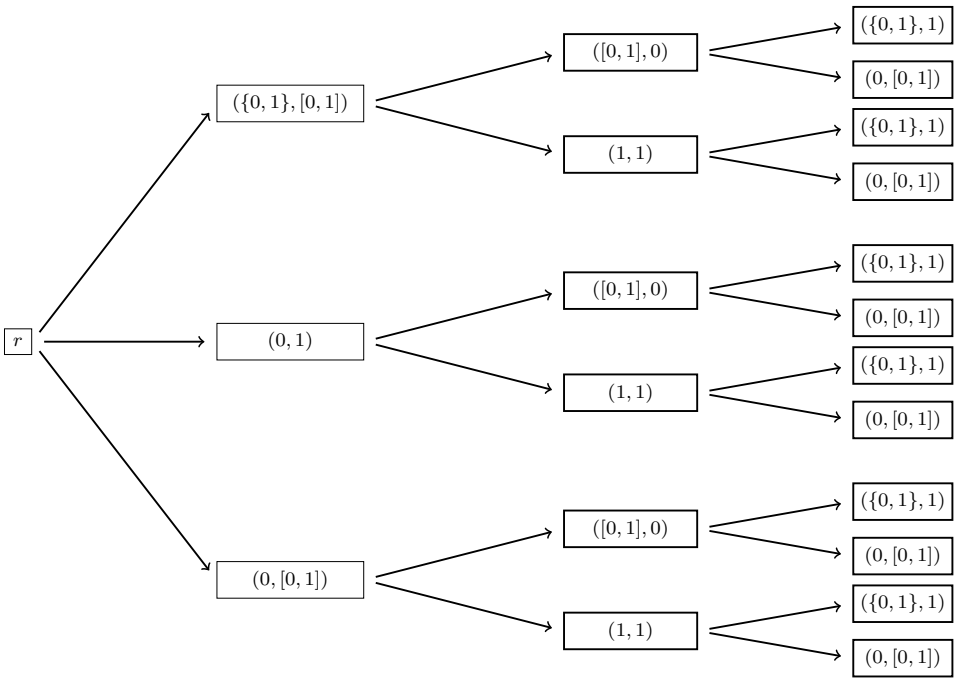


Figure 4.11: Example of a parameterized subtree which preserves stagewise independence.

In opposition, we give an example on Figure 4.11 of an assignation function that is $1-$independent on the same subtree. However, this assignation is not continuous, thus we cannot solve the problem with SDDP.

We give sufficient conditions in Proposition 2 for a parameterized problem to be solved with SDDP.

Proposition 2. *Let $\mathcal{T} = \mathcal{T}^{\Omega}[\mathcal{N}]$ be a $\nu-$rooted subtree, and $\mathfrak{b}$ an assignation function. If $\mathfrak{b}$ is continuous and $t_\nu-$symmetric, then we can solve $(P_{x_0}^{\mathcal{T}^{\mathfrak{b}}})$ with SDDP.*

Having established a class of assignation functions that are compatible with SDDP, we present a particular case, with a convenient structure for branch-and-bound, in the next section.

4.3.2 Branch and Bound for lower assignation function

As we cannot solve exactly Problem (2.8), our goal is to find a good approximation. To that end, in Section 4.2, we introduced assignation functions $\mathfrak{b}$ that parameterize Problem (2.8). We proposed in Section 4.2.3 a branch-and-bound methodology to choose $\mathfrak{b}$, though most of the time the parameterized problems are still too hard to solve.

In the specific case of finding the best lower-assignation functions $\mathfrak{b}^{lb}$(4.8b), for each binary variable b_k in node ν, we have to decide if we relax the integrality on b_k or not. In other words, we try to assess how important each binary variable in the problem is. Recall that $\hat{V}_r^{\mathfrak{b}^{lb}}(x_{init})$ is a lower-bound on Problem (2.8). Then, we can say a lower-assignation $\mathfrak{b}_1^{lb}$ is better than $\mathfrak{b}_2^{lb}$ if $\hat{V}_r^{\mathfrak{b}_1^{lb}}(x_{init}) > \hat{V}_r^{\mathfrak{b}_2^{lb}}(x_{init})$. The problem of finding the best lower-assignation $\mathfrak{b}^{lb}$ can be formulated as

$$\underset{z}{\text{Max}} \quad \hat{V}_r^{\mathfrak{b}^{lb}}(x_{init}) \tag{4.9a}$$

$$z_\nu^k \in \{0,1\} \qquad \forall \nu \in \mathcal{N}^{\Omega}, \forall k \in [n_b] \tag{4.9b}$$

$$\mathfrak{b}^{lb}(\nu)_k = \begin{cases} \{0,1\} \text{ if } z_\nu^k = 1 \\ [0,1] \text{ otherwise.} \end{cases} \qquad \forall k \in [n_b] \tag{4.9c}$$

The optimal value of Problem (4.9) is obviously to keep all integrality constraints *i.e.*, $z = 1$. However, with specific stopping rule for the BB Algorithm 10, we hope to find a good assignation yielding a solvable approximation of Problem(2.8). Next, we introduce assignation functions that are lower-assignation suited to Algorithm 10 as mentioned here, but that also follow conditions in Section 4.3.1 allowing for combining BB with SDDP.

4.3.3 Assignation induced by a pruned subtree

If we branch in a particular order on the nodes, we obtain problems that we can simplify by approximating whole sections of the scenario tree with SDDP cuts. More specifically, if we consider, for a node ν, an assignation function $\mathfrak{b}$ that is continuous and t_ν symmetric on $\mathcal{T}_{|\nu}$, we can solve $(P_{x_0}^{\mathcal{T}_{|\nu}})$ with SDDP by Proposition 2. In other words, if all descendants of ν are continuously relaxed, they model a multistage stochastic continuous sub-problem of Problem (2.8), solvable with SDDP. Thus, the whole portion of the scenario tree corresponding to $\mathcal{T}_{|\nu}$ is simplified and can easily be approximated.

Relying on the structure of a pruned subtree $\mathcal{T}_p$, we obtain a natural separation of the nodes in $\mathcal{T}^{\Omega}$ such that all subtrees that have been pruned can be solved with SDDP, and the rest as a classic MILP. We define a *partially relaxed-assignation* $\mathfrak{b}^{\mathcal{T}_p} : \mathcal{T} \to \{[0,1], \{0,1\}\}^{n_b}$ as an assignation function such that

$$\mathfrak{b}^{\mathcal{T}_p}(\nu) = \begin{cases} \{0,1\} \text{ if } \nu \in \mathcal{T}_p, \\ [0,1] \text{ otherwise.} \end{cases}$$

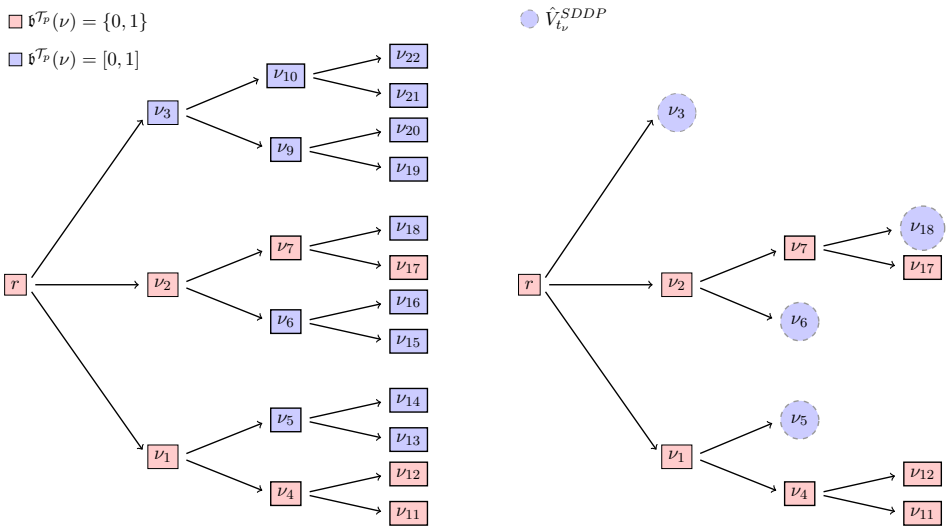


Figure 4.12: Example of a scenario tree parameterized with a partially relaxed-assignation function $\mathfrak{b}^{\mathcal{T}_p}$, based on pruned subtree $\mathcal{T}_p$. On the right, we illustrate the problem's reduction in size with nodes μ in circles when we replace the model with SDDP cuts that approximate the remaining problem on $\mathcal{T}_{|\mu}$.

Let $\mathcal{T}_p = \mathcal{T}^{\Omega}[\mathcal{N}_p]$ be a pruned subtree and $\mathfrak{b}^{\mathcal{T}_p}$ a partially relaxed-assignation, we can develop a method to solve $(P_{x_{init}}^{\mathcal{T}^{\Omega},\mathfrak{b}^{\mathcal{T}_p}})$. First, for a node $\nu \in \mathcal{C}(\mathcal{N}_p)$, we can approximate $\hat{V}_{\nu}^{\mathfrak{b}^r}(x) = \hat{V}_{t_\nu}^{\mathfrak{b}^r}(x, \xi_\nu)$ (recall that we are under stagewise independence assumptions) with cuts generated by SDDP. Then, the $\mathcal{T}_p-$backward operator $\hat{\mathcal{B}}_{\mathcal{T}_p}(\{V_\mu^{SDDP}\}_{\mu \in \mathcal{C}(\mathcal{N})})(x_{init})$ is a classic MILP that we can solve if $\mathcal{T}_p$ is of reasonable size. Indeed, instead of having variables and constraints for all nodes $\nu \in \mathcal{N}^{\Omega} \setminus \mathcal{N}_p$, we model the cost-to-go from $\nu \in \mathcal{C}(\mathcal{N}_p)$ with SDDP cuts, and thus, the model's size is considerably reduced, see an example in Figure 4.12.

Conclusion

In this chapter, we discussed a generic framework to apply BB methods to solving MSbLP, relying on SDDP to (approximately) solve the linear relaxation at each step of the BB tree– which would would otherwise be intractable. Solving the partial relaxation obtained by mixing both technologies implies non-trivial constraints stemming from the scenario tree structure on the searching and branching rules. While the use of assignation functions might appear quite abstract, we develop in the following chapter a reasonably intuitive case that offers a full-fledged methodology to solve MSbLP.

Chapter 5

A growing tree algorithm for solving MSbLPs

Contents

In the previous chapter, we elaborated a complex and abstract framework to explore various ways to approximate Multistage Stochastic mixed-binary Linear Program (MSbLP). This framework enabled us to understand better how we could merge branch-and-bound methodologies with the continuous relaxation approximations given by SDDP. To that end, we have introduced a specific way to approximate the problem, with partially relaxed-assignations $\mathfrak{b}^{\mathcal{T}_p}$, where $\mathcal{T}_p$ is a pruned subtree of the initial scenario tree. In this chapter, we propose a branch-and-bound algorithm to solve an MSbLP derived from those specific assignation functions. As a result, we obtain a class of approximated problems that can be efficiently solved, and from which we reconstruct a

solution of the MSbLP. The work presented in this section has been done in collaboration with Bernardo Freitas Paulo da Costa[1] and Merve Bodur[2].

5.1 Literature Review

Multistage Stochastic mixed-binary Linear Program (MSbLP) can model a broad range of complex problems, such as hydro power scheduling ([PP85],[Hje+19]) and unit commitment ([Bar+06],[TKW00]), which are both large-scale and subject to uncertainties. Despite extensive study, these problems remain challenging due to the need to manage integer variables while addressing decision-making under uncertainty. Currently, most approaches involve approximating one or both of these aspects. For a general introduction to stochastic programming, from a mathematical programming point of view, we refer to Birge and Louveaux's book [BL97].

Decomposition algorithms have been developed for 2−stage stochastic programs and later extended to the multistage case. For example, Progressive-Hedging (PH) [RW91] is an augmented Lagrangian decomposition where the non-anticipativity constraints are relaxed. The Lagrangian coefficients penalizing a non-anticipativity violation are updated by solving a deterministic program per scenario. The method extends straightforwardly to multistage programs but is limited by the number of scenarios, hence the number of stages. The algorithm is based on convex optimization tools, but the non-convex case, in particular with binary variables, has been studied in [HLW01] where PH is used as a meta-heuristic.

Alternatively, Benders decomposition [Ben62] has been adapted to stochastic programs under the name L-shaped method [SW69]. It decomposes the problem into a *master* problem– with first-stage decisions – and *slave* problems– with second-stage decisions, parameterized by the first-stage decisions. If the slave problems are convex, they can be approximated with cuts using dual tools. Then, the master problem is iteratively updated with new cuts approximating each slave problem. These methods can directly be extended to the multistage case [Bir85b], as long as the number of scenarios remains reasonable, which means having a small horizon.

Benders decomposition allows integer variables in the master problem. Thus, in the case of a multistage program with integer variables in each stage, a possibility is to relocate all integer variables of an MSiLP in the first stage [CBS23]. Then, the slave problems are continuous and can be exactly approximated (actually, Castro, Bodur, and Song use SDDP, described next, in their paper to solve them). Additionally, for tractability, we can reduce the number of integer variables by adding information constraints. Further, two-stage linear decision rules can be considered to reduce the number of integer variables to optimize. Indeed, an approach to make MSiLP easier to solve is to restrict the policies with linear decision rules that lead to integral decisions [DBL24].

With some additional assumptions about the noise structure, we can leverage Dynamic Programming (DP) based methods to tackle large horizon instances of MSLP. For example, if we assume noises are stagewise independent, the cost-to-go functions only depend on time and state, which alleviate the computational burden of SDP. Further, in the case of Markov chains, we can handle the problem similarly by expanding the state with information on the previous noise realization [LS19]. Then, the naive approach consists in discretizing the state and estimating the cost-to-go functions in each discretization point – see Section 2.3.2. However, the approach remains limited by the *curse-of-dimensionality* when the state dimension increases (say not larger than 4 or 5).

[1] Fundação Getulio Vargas
[2] University of Edinburgh

Instead of discretizing *a priori* the state space and estimating the cost-to-go from local interpolation, we can use structural information of the cost-to-go function to get global estimation and automatically adapt the discretization point. This leads to a broad class of algorithms, captured under the framework of Trajectory Following Dynamic Programming (TFDP) in [FL23]. Those algorithms iteratively refine estimates of the cost-to-go functions. More precisely, they iterate over a *forward pass*, where they select a trajectory of state points; and a *backward pass* where they compute cuts to improve the cost-to-go estimations at those selected points. More precisely, at iteration k, a system trajectory $x^k_{[T]}$ is computed with forward operators (2.13) using the current cost-to-go approximations $V^{k-1}_{[T]}$. The cost-to-go approximations $V^{k-1}_{[T]}$ are then refined with new cuts derived from incoming state x^k_t. The primary difference between these algorithms is the type of cuts they generate.

One of the most popular and efficient algorithms in TFDPs is Stochastic Dual Dynamic Programming (SDDP) [PP91], which is a randomized version of nested-Benders decomposition introduced in 1991 to manage Brazil's large hydrothermal system, and continues to be widely used today. SDDP is designed to solve Multistage Stochastic Linear Program (MSLP), characterized by their convex and continuous nature. This structure allows for the computation of Benders cuts; in the backward pass, we solve the dual problem of backward operators (2.12) to generate a lower approximation of cost-to-go functions. The algorithm's convergence has been established in [PG08] and [GLP15] for the convex case. More generally, we find a review of different complexity results on TFDPs in [FL23]. Additionally, SDDP has proven effective in various applications and numerous enhancements have been proposed over the years, see [Ser23; FR23] for a comprehensive review.

For non-convex problems, such as those involving binary variables, we find several TFDP algorithms that use specific cuts in the literature. A few notable examples include: [Fer+13], which leverages the Lagrangian relaxation of the stage problem to generate strengthened Benders' cuts using Lagrange multipliers; Mixed Integer Dynamic Approximation Scheme (MIDAS) [PWB20], which employs step functions, assuming the cost-to-go functions are monotonic with respect to the state variables; Stochastic Lipschitz Dynamic Programming (SLDP) [ACF22], which uses concave L_1 cuts for Lipschitz cost-to-go functions; and [RvdL24], which proposes scaled-cuts.

Another approach is to construct a heuristic leveraging the approximations that can easily be obtained with SDDP. For example, Thevenin, Adulyasak, and Cordeau apply SDDP to solve a multi-echelon lot-sizing problem [TAC22], which is naturally formulated as an MSiLP. They propose two strategies: one is to compute the integer variables using Progressive-Hedging (PH); and the other involves iterating between solving the first-stage problem to optimize the integer variables and solving the parameterized problem with SDDP, refining the cost-to-go approximations injected in the first-stage problem. Another example is the approach proposed in [FLG24], where we solve mixed-integer programs in the forward pass while computing Benders-like cuts of the continuous relaxation of stage problems in the backward pass. However, this heuristic can lead to infeasible policies, as demonstrated by a small example in Chapter 3. Further, there are no convergence guarantees to the actual value of the problem with those heuristics.

To tackle specifically binary variables, Stochastic Dual Dynamic integer Programming (SDDiP), introduced in [ZAS19], uses Lagrangian and strengthened Benders' cuts that fit binary variables. Although this paper provides a proof of convergence, the computation of these cuts is complex and significantly increases the computational load. Additionally, these cuts are designed explicitly for binary state variables. Consequently, adapting the algorithm for broader problem classes, such as MSbLP, requires substantial discretization, further impeding convergence. In [QGK21],the

authors suggest enhancements to SDDiP by considering a partial decomposition of the problem into subproblems formulated over a set of nodes– or subtree– of the scenario tree. To approximate those subproblems, they use similar cuts as in SDDiP as well as specific lot-sizing cuts relevant to their application.

5.2 A generic algorithm with convergence results

In this section, we recall the mathematical formulation of an MSbLP as a large MILP. Then, leveraging the results found in Section 4.3, we propose efficiently solvable approximations of an MSbLP. Finally, we obtain a branch-and-bound algorithm to solve an MSbLP with stagewise independent uncertainty.

5.2.1 Branch-and-Bound for solving MSbLP

We consider an MSbLP, where all of the uncertainty can be represented with a scenario tree $\mathcal{T}^\Omega = (\mathcal{N}^\Omega, \mathcal{A}^\Omega, w^\Omega)$, see Section 4.1.2 for more details. We have seen that an MSbLP can be reformulated as a very large MILP. Indeed, leveraging the structure of the $\mathcal{T}^\Omega$, the extensive formulation of the problem reads

$$
\begin{aligned}
(P^{\mathcal{T}^\Omega}_{x_{init}}) \qquad \min_{x,y,b} \quad & \sum_{\nu \in \mathcal{N}^\Omega \setminus \{r\}} \pi_\nu L_{t_\nu}(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) && && \text{(5.1a)}\\
& x_\nu = F_{t_\nu}(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \subset \mathfrak{X}_{t_\nu} && \forall \nu \in \mathcal{N}^\Omega \setminus \{r\} && \text{(5.1b)}\\
& y_\nu \in \mathfrak{Y}_{t_\nu}(x_{a(\nu)}, \xi_\nu) && \forall \nu \in \mathcal{N}^\Omega \setminus \{r\} && \text{(5.1c)}\\
& b_\nu \in \mathfrak{B}_{t_\nu}(x_{a(\nu)}, \xi_\nu) \cap \{0,1\}^{n_b} && \forall \nu \in \mathcal{N}^\Omega \setminus \{r\} && \text{(5.1d)}\\
& x_r = x_{init}, && && \text{(5.1e)}
\end{aligned}
$$

where for each node ν, ξ_ν is the information unveiled in node ν. Then, we have state variables x_ν ruled by dynamics (5.1b) and initialized (5.1e); and control variables (y_ν, b_ν) that are either continuous (5.1c) or binary (5.1d), with n_b the number of binary variables per node. Finally, as we assume that the problem is stagewise independent, the instantaneous cost L_{t_ν}, the dynamics F_{t_ν} and feasible sets $\mathfrak{X}_{t_\nu}, \mathfrak{Y}_{t_\nu}, \mathfrak{B}_{t_\nu}$, at ν depend only of its depth t_ν *i.e.,* the stage in which ξ_ν is occurring.

A classical method for solving MILPs is the so-called Branch-and-Bound approach, which consists of iteratively fixing binary variables and solving some continuous relaxation of the remains (see Algorithm 10). However, in our setting, the resulting MILP (5.1) is so large that even the LP relaxation cannot be solved efficiently in practice, unless we are under stagewise independence assumptions. In such a case, we can obtain a solution efficiently with SDDP. We have seen in the previous chapter that SDDP can tackle some variations of this problem. To solve Problem (5.1) through a Branch-and-Bound approach, we have to be careful to branch on the binary variables in a way such that the subproblem remains tractable by SDDP. In other words, this means that, at every stage of the branch and bound process, we choose a continuous and symmetric assignation function (see Section 4.3.1). We next show how to do that in practice.

5.2.2 Partially relaxing integrality

The algorithm we present in Section 5.2.4 relies on first solving an approximation of $(P^{\mathcal{T}^\Omega}_{x_{init}})$, and then iteratively strengthening this approximation until we get a satisfactory solution. More specifically, we consider a *partial continuous relaxation* of $(P^{\mathcal{T}^\Omega}_{x_{init}})$, meaning we relax some of the

binary constraints. To leverage SDDP cuts, this partial continuous relaxation has to be structured by a pruned subtree of $\mathcal{T}^\Omega$, defined in Section 4.1.1. Let $\mathcal{T}_p = \mathcal{T}^\Omega[\mathcal{N}_p]$ be a pruned subtree, composed of subset of nodes $\mathcal{N}_p$, we call $(PR^{\mathcal{T}_p}_{x_{init}})$ the partially relaxed problem formulated as

$$
\begin{aligned}
(PR^{\mathcal{T}_p}_{x_{init}}) \quad \min_{x,y,b} \quad & \sum_{\nu \in \mathcal{N}^\Omega \setminus \{r\}} \pi_\nu L_{t_\nu}(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) && && \text{(5.2a)}\\
& x_\nu = F_{t_\nu}(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \subset \mathfrak{X}_{t_\nu} && \forall \nu \in \mathcal{N}^\Omega \setminus \{r\} && \text{(5.2b)}\\
& y_\nu \in \mathfrak{Y}_{t_\nu}(x_{a(\nu)}, \xi_\nu) && \forall \nu \in \mathcal{N}^\Omega \setminus \{r\} && \text{(5.2c)}\\
& b_\nu \in \mathfrak{B}_{t_\nu}(x_{a(\nu)}, \xi_\nu) && \forall \nu \in \mathcal{N}^\Omega \setminus \{r\} && \text{(5.2d)}\\
& b_\nu \in \{0,1\}^{n_b} && \forall \nu \in \mathcal{N}_p \setminus \{r\} && \text{(5.2e)}\\
& x_r = x_{init}. && && \text{(5.2f)}
\end{aligned}
$$

Note that in the difference between Problem (5.1) and (5.2) is that integrality constraints (5.2e) are enforced only for nodes in $\mathcal{N}_p$. Note that this specific relaxation corresponds to the partially relaxed-assignation $\mathfrak{b}^{\mathcal{T}_p}$. Further, in $(PR^{\mathcal{T}_p}_{x_{init}})$, the subproblem from $\mu \in \mathcal{C}(\mathcal{N}_p)$, a child of the pruned subtree, is a continuous relaxation of $(P^{\mathcal{T}_{|\mu}}_{x_{a(\mu)}})$. The optimal cost of this relaxed subproblem is denoted $\hat{V}^r_{t_\mu}$ and reads

$$
\begin{aligned}
\hat{V}^r_{t_\mu}(x, \xi_\mu) = \min_{x,y,b} \quad & \Big[L_{t_\mu}(x, u, b, \xi_\mu) + V^r_{t_\mu+1}(z) \Big] && && \text{(5.3a)}\\
& z = F_{t_\mu}(x, u, b, \xi_\mu) \subset \mathfrak{X}_{t_\mu} && && \text{(5.3b)}\\
& u \in \mathfrak{Y}_{t_\mu}(x, \xi_\mu) && && \text{(5.3c)}\\
& b \in \mathfrak{B}_{t_\mu}(x, \xi_\mu) && && \text{(5.3d)}\\
V^r_{t_\mu}(x) = \ & \mathbb{E}\Big[\hat{V}^r_{t_\mu}(x, \boldsymbol{\xi_{t_\mu}}) \Big]. && && \text{(5.3e)}
\end{aligned}
$$

More generally, for any approximations $\{\hat{\mathcal{R}}_t\}_{t\in[T]}$ of cost-to-go functions (5.3), we formulate the corresponding approximation of $(PR^{\mathcal{T}_p}_{x_{init}})$ with the $\mathcal{T}_p-$backward operator $\hat{\mathcal{B}}_{\mathcal{T}_p}(\{\hat{\mathcal{R}}_t\}_{t\in[T]})$:

$$
\begin{aligned}
\hat{\mathcal{B}}_{\mathcal{T}_p}(\{\hat{\mathcal{R}}_t\}_{t\in[T]})(x_{init}) = \operatorname{Min}_{x,y,b} \quad & \sum_{\nu \in \mathcal{N}_p \setminus \{r\}} \pi_\nu L_{t_\nu}(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) + \sum_{\mu \in \mathcal{C}(\mathcal{N}_p)} \pi_\mu \hat{\mathcal{R}}_{t_\mu}(x_{a(\mu)}, \xi_\mu) && && \\
& x_\nu = F_{t_\nu}(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \subset \mathfrak{X}_{t_\nu} && \forall \nu \in \mathcal{N}_p \setminus \{r\} && \text{(5.4a)}\\
& y_\nu \in \mathfrak{Y}_{t_\nu}(x_{a(\nu)}, \xi_\nu) && \forall \nu \in \mathcal{N}_p \setminus \{r\} && \text{(5.4b)}\\
& b_\nu \in \mathfrak{B}_{t_\nu}(x_{a(\nu)}, \xi_\nu) \cap \{0,1\}^{n_b} && \forall \nu \in \mathcal{N}_p \setminus \{r\} && \text{(5.4c)}\\
& x_r = x_{init}. && && \text{(5.4d)}
\end{aligned}
$$

Those operators were previously introduced in Section 4.1.3, in the more general setting without stagewise independence. In particular, if we know the true relaxed cost-to-go functions $\hat{V}^r_t$, the optimal value of $(PR^{\mathcal{T}_p}_{x_{init}})$ is given by $\hat{\mathcal{B}}_{\mathcal{T}_p}(\{\hat{V}^r_t\}_{t\in[T]})(x_{init})$.

We use this formalism to define a converging algorithm in the next section.

5.2.3 An exact algorithm to solve $(PR^{\mathcal{T}_p}_{x_{init}})$

We now give an exact algorithm to solve the partially relaxed problem $(PR^{\mathcal{T}_p}_{x_{init}})$ defined in (5.2), relying on SDDP cuts that approximate the relaxed cost-to-go functions $\hat{V}^r_t$.

Assuming we have initial cost-to-go approximations $\{\hat{V}_t^{r,0}\}_{t\in[T]}$, that are linear cuts, $\hat{\mathcal{B}}_{\mathcal{T}_p}(\{\hat{V}_t^{r,0}\}_{t\in[T]})(x_{init})$ is an MILP that can be solved with any fit for mixed-integer programming solver. We obtain a solution $\{x_\nu^k\}_{\nu\in\mathcal{N}_p}$ and the cost-to-go, from a child μ of $\mathcal{T}_p$, is estimated through $\hat{V}_{t_\mu}^{r,0}(x_{a(\mu)}^k)$ – and therefore is an underestimate of the true cost-to-go function. Observe that evaluating the gap (even in the continuous convex setting) requires either a Monte-Carlo simulation, or some other upper bound techniques. For the algorithm to converge, we must refine the approximation $\hat{V}_t^{r,0}$. To that end, we solve the subproblem(5.3), a multistage stochastic linear continuous problem, at $x_{a(\mu)}^k$ with SDDP which converges to the true value of $\hat{V}_{t_\mu}^r(x_{a(\mu)}^k)$. Then, we update the approximation $\hat{V}_{t_\mu}^{r,0}$ with the new cuts computed by SDDP. If we iterate, we obtain Algorithm 11 which converges to the optimal value of $(PR_{x_{init}}^{\mathcal{T}_p})$. The proof of convergence can be derived straightforwardly using arguments similar to those in SDDP's convergence proof.

Algorithm 11: An exact algorithm to solve $(PR_{x_{init}}^{\mathcal{T}_p})$.

Input: inital approximations $\{\hat{V}_t^{r,0}\}_{t\in[T]}$;
while *stopping criterion is not met* **do** `// Iteration k`
  Solve $\hat{\mathcal{B}}_{\mathcal{T}_p}(\{\hat{V}_t^{r,k-1}\}_{t\in[T]})(x_{init})$; `// we solve an MILP`
  We set $\overline{f}$ its value, and $\{x_\nu^k\}_{\nu\in\mathcal{N}_p}$ its optimal solution;
  Run SDDP with forward trajectories starting from x_ν^k for all $\nu \in \mathcal{L}(\mathcal{T}_p)$ `// refine` $\{\hat{V}_t^{r,k-1}\}_{t\in[T]}$
Return $\overline{f}$;

At each iteration of Algorithm 11, we run SDDP at each leaf of the subtree *i.e.*, $|\mathcal{L}(\mathcal{T}_p)|$ times. This is computationally heavy, especially since cuts computed are shared between leaves ($\hat{V}_t^r$ are stage dependent only). There are other ways of using SDDP to update $\{\hat{V}_t^{r,k-1}\}_{t\in[T]}$. One possibility is to choose, at each iteration, a leaf $\nu \in \mathcal{L}(\mathcal{T}_p)$ and run SDDP from this leaf until convergence at x_ν^k before moving to the next leaf. Another consists in having, for each forward pass of SDDP, $|\mathcal{L}(\mathcal{T}_p)|$ trajectories, each starting from one x_ν^k. Of course, we can look for a middle ground by randomly selecting a subset of leaves for each forward pass while still evaluating the gap for all leaves.

5.2.4 An algorithm to solve MSbLPs

In the previous section, we proposed an exact and converging algorithm to solve $(P_{x_{init}}^{\mathcal{T}_p})$ for any pruned subtree $\mathcal{T}_p$. The optimal value of $(P_{x_{init}}^{\mathcal{T}_p})$ is a lower bound on the optimal value of Problem (5.1). We can also derive from $\mathcal{T}_p$ a policy, using a rolling horizon procedure detailed in Section 5.5.3, for Problem (5.1) with $\hat{\mathcal{F}}_{\mathcal{T}_p}(\{\hat{V}_t^r\}_{t\in[T]})$, where $\hat{V}_t^r$ are given by SDDP. Then, by simulating this policy on a sample of scenarios, we obtain a statistical upper bound on Problem (5.1). Thus, each pruned tree $\mathcal{T}_p$ gives us a bounded interval in which lies $v(P_{x_{init}}^{\mathcal{T}^\Omega})$.

Starting from an empty subtree $\mathcal{T}^\Omega[\{r\}]$, we propose to iteratively grow a pruned subtree $\mathcal{T}_p$, until the policy derived from $\mathcal{T}_p$ gives satisfactory solutions. This *growing strategy* (of the pruned subtree) corresponds to a branch-and-bound branching strategy to find a lower-assignation $\mathfrak{b}^{lb}$, mentioned in Section 4.3.2. We discuss different growing strategies in the next section. Combined with Algorithm 11, we obtain Algorithm 12 to solve an MSbLP.

Algorithm 12: A growing MILP tree algorithm for MSbLP.

Input : initial tree $\mathcal{T}_p = \mathcal{T}^{\Omega}[\{r\}]$,
Compute lower bounds $\underline{V}^r_{[T]}$ via SDDP;
while *stopping criterion is not met* **do**
    Select nodes in $\mathcal{C}(\mathcal{N}_p)$ to grow $\mathcal{T}_p$;
    Solve $(P^{\mathcal{T}_p, \underline{V}^r_{[T]}}_{x_{init}})$ with Algorithm 11 to update the lower bounds $\underline{V}^r_{[T]}$;
    Estimate an upper-bound *ub* through SAA;

We call *pool of candidates* the set of nodes eligible to grow the subtree $\mathcal{T}_p = \mathcal{T}^{\Omega}[\mathcal{N}_p]$ into another subtree $\mathcal{T}_{p'} = \mathcal{T}^{\Omega}[\mathcal{N}_{p'}]$, with $\mathcal{N}_p \subset \mathcal{N}_{p'}$. In Algorithm 12, we propose a method growing a subtree node by node *i.e.,* where the pool of candidates is the children set $\mathcal{C}(\mathcal{N}_p)$ of the current subtree $\mathcal{T}_p$. Indeed, any other node could not grow $\mathcal{T}$ into another subtree alone. More generally, we can grow a subtree with *branches*, where a branch is a path $\{\nu_k\}_{k\in[\tau]}$ in $\mathcal{T}^{\Omega}$ such that $\nu_1 \in \mathcal{C}(\mathcal{N})$ and the path is not in $\mathcal{T}$ *i.e.,* $\nu_k \notin \mathcal{N}$, for all k. Then, we can consider any set $\mathcal{N}_c \subseteq \mathcal{N}^{\Omega} \setminus \mathcal{N}_p$ to be the pool of candidates, if we grow the subtree $\mathcal{T}_p$ with the unique branch connecting $\mathcal{T}_p$ to node $\nu \in \mathcal{N}_c$. In other words, if we want to grow the subtree $\mathcal{T}_p$ with any node $\mu \in \mathcal{N}^{\Omega} \setminus \mathcal{N}_p$, we have to add all descendants of μ as well.

Remark 6 (Additional SDDP run)**.** *Algorithm 12 start by solving the relaxed problem with SDDP, ending with lower approximations* $\underline{V}^r_{[T]}$*. These approximations are good on the support of the policy they induce. However, at each iteration, we add new binary constraints to the model and the obtained solution of* $(P^{\mathcal{T}_p, \underline{V}^r_{[T]}}_{x_{init}})$ *progressively changes. In particular, the state at the leaf of the current tree might go into part of the space where SDDP has not converged, for example, because it was not part of the support of the optimal solution of the convex relaxation. As in Algorithm 11, we may run multiple times SDDP to make sure the approximations* $\underline{V}[T]^r(x)$ *converge, which increase the computation time, and still provide only a lower bound of the (non-relaxed) value function. Consequently, we may want to use only the initial SDDP cuts and check convergence – and add new cuts – at the leaf states only sparingly.*

5.3 Growing (integer) subtree strategies

We presented an exact branch-and-bound algorithm to solve MSbLP in the previous section. In BB, one challenge is choosing a branching strategy (see Section 4.2.3). We elaborate here on different growing strategies to grow the subtree $\mathcal{T}_p$, the goal being to limit the number of iterations of Algorithm 12. Note that we are specifying here the branch-and-bound proposed in Section 4.3.2 to find a lower-assignation $\mathfrak{b}^{lb}$. We give an example of a scenario tree in Figure 5.1. In the next section, we illustrate how the different growing strategies work on this example.

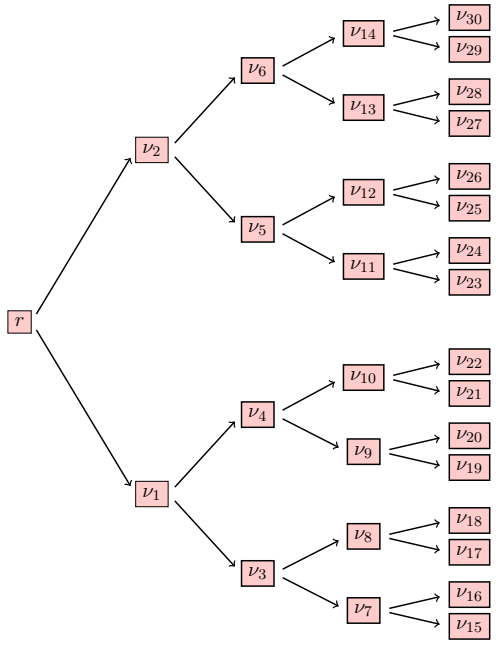


Figure 5.1: A scenario tree $\mathcal{T}^{\Omega}$ with $T = 4$ and for each stage t, $|\Xi_t| = 2$.

Recall that Algorithm 12 solves increasingly large problems ($PR^{\mathcal{T}}_{x_{init}}$) that are structured by pruned subtrees. Starting from a given pruned subtree $\mathcal{T}_p = \mathcal{T}^{\Omega}[\mathcal{N}_p]$, we *grow* $\mathcal{T}_p$ by adding nodes to $\mathcal{T}_p$ while conserving a pruned subtree structure. More precisely, we can grow $\mathcal{T}_p$ with one of its children $\mu \in \mathcal{C}(\mathcal{N}_p)$ into subtree $\mathcal{T}_{p'} = \mathcal{T}^{\Omega}[\mathcal{N}_p \cup \{\mu\}]$.

The structure of a subtree depends significantly on the growing strategy we choose. We introduce two key features to characterize a subtree $\mathcal{T} = \mathcal{T}^{\Omega}[\mathcal{N}]$. First, we define the *depth* of $\mathcal{T}$, denoted $t_{\mathcal{T}}$, as the maximum depth among its leaves *i.e.,*

$$t_{\mathcal{T}} = \max_{\nu \in \mathcal{L}(\mathcal{T})} t_{\nu}.$$

Second, for each subtree $\mathcal{T}$, we define the *density* of a layer $\mathcal{N}_t$, denoted $\rho(\mathcal{N}_t)$, as the proportion of that layer contained in $\mathcal{T}$ *i.e.,*

$$\rho(\mathcal{N}_t) = \frac{\mathcal{N}_t}{\mathcal{N}_t^{\Omega}}.$$

Since $\mathcal{N}_t = \mathcal{N} \cap \mathcal{N}_t^{\Omega}$, $\rho(\mathcal{N}_t)$ ranges from 0, when the layer is empty, to 1, when the layer is complete. By definition, if $\rho(\mathcal{N}_t) = 1$ then $\rho(\mathcal{N}_\tau) = 1$ for all $\tau < t$. Similarly, if $\rho(\mathcal{N}_t) = 0$, then $\rho(\mathcal{N}_\tau) = 0$ for all $\tau > t$. If current decisions have little impact on distant future costs, we might prefer a policy constructed with subtrees that are dense at earlier stages and sparse in future stages.

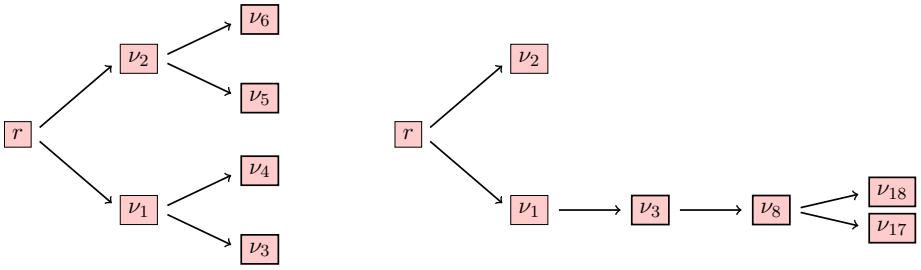


Figure 5.2: Example of two subtrees of the scenario tree in Figure 5.1 of same size.

In Figure 5.2, we illustrate two subtrees of the scenario tree shown in Figure 5.1, each containing six nodes. The subtree on the left is described as *bushy*, meaning it is dense *i.e.,* $\rho(\mathcal{N}_t) \approx 1$ for some t, but shallow *i.e.,* $t_{\mathcal{T}} \ll T$. In contrast, the subtree on the right is *elongated*, characterized by its depth, $t_{\mathcal{T}} \gg 1$, though its layers are *sparse* $\rho(\mathcal{N}_t) \ll 1$ for some t. Unfortunately, we cannot establish dominance of one type of subtree over the other. The performance of *bushy* versus *elongated* subtrees depends on the specific problem structure, including its constraints, size, and shape of the initial scenario tree. Either type may prove more effective depending on the problem instance.

We now present different growing strategies for Algorithm 12.

5.3.1 Shortsighted strategy

Before presenting growing strategies, where we iteratively grow a subtree node by node, we discuss the special case of growing the subtree layer by layer, which amounts to considering only $t-$shortsighted subtrees.

Indeed, in a multistage setting, the most intuitive way to simplify the problem is to consider we have complete information until a given time t, and relax the far future. In other words, we consider all decisions on a smaller horizon $[1:t]$ with $t < T$, and we relax integrality constraints for $\tau > t$. This corresponds to the partially relaxed problem $(P^{\mathcal{T}_{1:t}}_{x_{init}})$. Then, we can grow the subtree $\mathcal{T}$ layer by layer, and we obtain a particular case of Algorithm 12. We refer to this strategy as the *shortsighted strategy* and we illustrate its first iterations in Figure 5.3.

With the shortsighted strategy, there is no choice on how to grow a $t-$shortsighted subtree $\mathcal{T}_{1:t}$ but to add all nodes in $\mathcal{C}(\mathcal{N}_{1:t})$. Hence, we have no control over the size of the expansion of a subtree, and quickly, we fall back on the curse of dimensionality. For example, with $Q = 10$ realization per stage, growing a $t-$shortsighted subtree into a $(t+1)-$shortsighted subtree multiplies the number of variables by 10. Let us assume we cannot solve $(P^{\mathcal{T}_{1:t+1}}_{x_{init}})$ in reasonable time and we are not satisfied with the solution obtained by solving $(P^{\mathcal{T}_{1:t}}_{x_{init}})$. This leads us to construct an intermediate subtree on which the resulting problem is solvable. For example, this subtree could contain all nodes in $\mathcal{N}_{1:t}$ and some nodes in $\mathcal{N}^{\Omega}_{t+1}$.

Further, the shortsighted strategy also leads to bushy subtrees, which, in some problems, can prove less efficient than elongated subtrees. With this in mind, we propose different growing strategies that rely on other pruned subtrees, where we have more flexibility in how to grow the subtree.

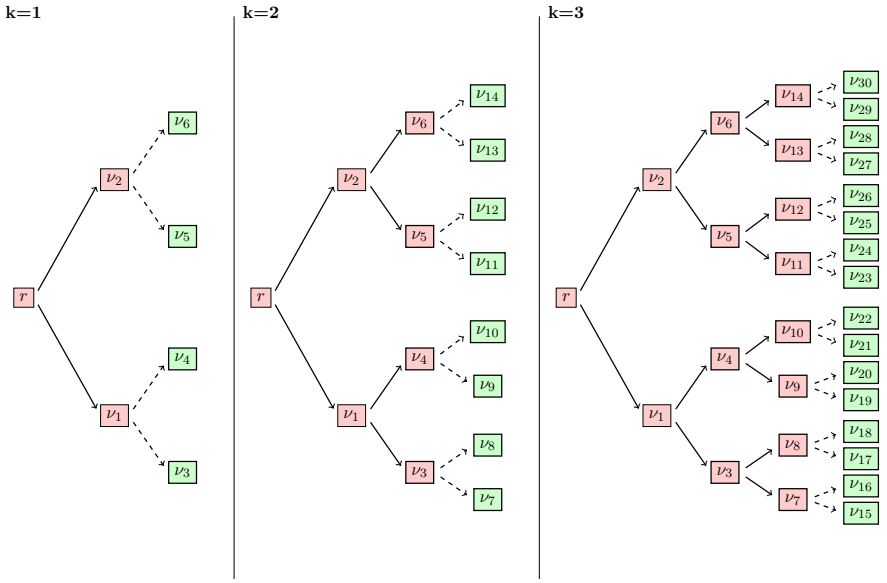


Figure 5.3: Illustration of the shortsighted strategy on the scenario tree in Figure 5.1: in red we represent nodes in the subtree, in green the children of the subtree, which are also candidates to grow the subtree. At iteration 1, we have the subtree $\mathcal{T}_{1:1}$ on the left; at iteration 2, we have the subtree $\mathcal{T}_{1:2}$ on the middle; at iteration 3, we have the subtree $\mathcal{T}_{1:3}$ on the right; and so on.

5.3.2 Random growing strategy

A simple way to choose objects among a set is to select them randomly. Though it could be efficient in some setting, randomly selecting nodes does not give us any guarantee on the quality of the solution we might get. It also prevents interpreting the results unless we do a sensitivity analysis by replicating the random growth.

Starting from a given subtree $\mathcal{T} = \mathcal{T}^{\Omega}[\mathcal{N}]$, we propose a *random growing strategy*, based on two key elements. First, we select a pool of candidate nodes. Second, we define the probability distribution from which nodes are randomly drawn to grow $\mathcal{T}$. Different distributions or candidate pools may be considered depending on the desired structure of the subtree.

The candidate pool is naturally $\mathcal{C}(\mathcal{N}_p)$ in a strategy where the subtree is grown node by node. Starting from the empty subtree $\mathcal{T}_{p_0} = \mathcal{T}^{\Omega}\big[\{r\}\big]$, we randomly select a node ν among $\mathcal{C}(\mathcal{N}_{p_0})$ with uniform distribution. In Figure 5.4, we observe six iterations of this *random growing strategy*. This strategy tends to produce elongated subtrees, as the early layers contain fewer nodes and thus have a lower probability of being selected.

Alternatively, we can define the candidate pool as a specific layer $\mathcal{N}_t^{\Omega} \cap \mathcal{C}(\mathcal{N}_p)$ and draw nodes uniformly from this layer. This strategy offers more control over the density of the subtree: for bushy subtrees, we can select a dense, partially filled layer, while for more balanced subtrees, a sparse layer can be chosen.

Another strategy involves growing the subtree with any node $\nu \in \mathcal{N}^{\Omega} \setminus \mathcal{N}_p$. As detailed in Section 5.2.4, this implies growing the subtree with the branch connecting it to the chosen node. One option is to use the leaves $\mathcal{L}(\mathcal{T}^{\Omega}) \setminus \mathcal{N}_p$ as candidates, meaning we incorporate an entire scenario's information into the partial relaxation at once. In this case, we randomly select a leaf from $\mathcal{L}(\mathcal{T}^{\Omega}) \setminus \mathcal{N}_p$ with a uniform distribution.

The outcome remains uncertain although we can influence the random strategy by choosing

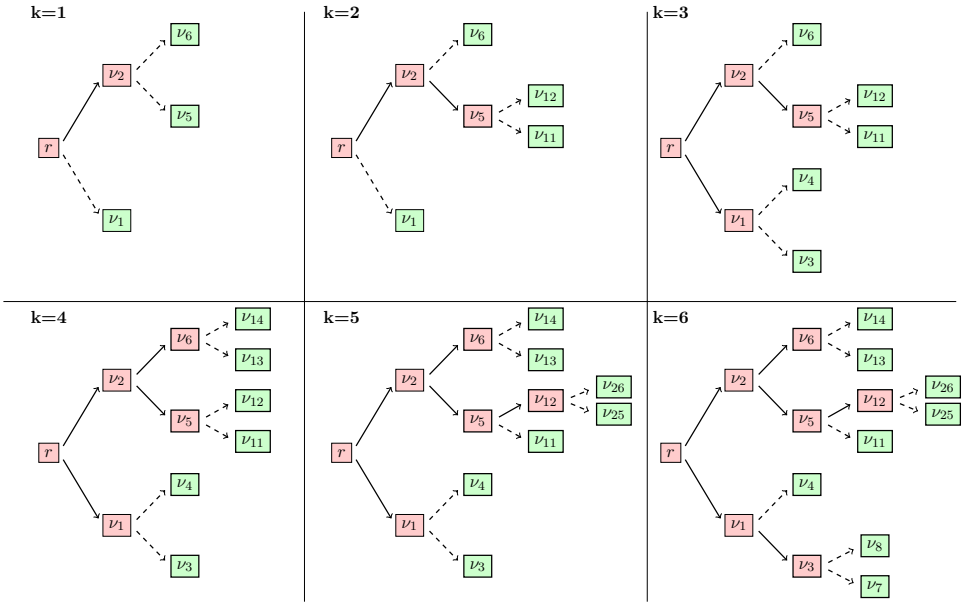


Figure 5.4: Illustration of the random growing strategy on the scenario tree in Figure 5.1: in red we represent nodes in the subtree, in green the candidate pool, here the children of the subtree $\mathcal{C}(\mathcal{N}_p)$. We represent six iterations of the random strategy.

candidates and the probability distribution. The following sections discuss growing strategies designed to maximize solution quality. Rather than using fixed (like the shortsighted) or random strategies, we evaluate a *score* to each candidate, representing the importance of enforcing integrality constraints at this node. We then grow the subtree with the highest-scoring candidate.

5.3.3 Gap with Integrality

We aim to develop a growing strategy that constructs a subtree providing the best possible approximation. To grow a given subtree $\mathcal{T}_p$, we here consider $\mathcal{C}(\mathcal{N}_p)$ as the candidate pool. Our challenge here is to determine which node $\mu \in \mathcal{C}(\mathcal{N}_p)$ should have integrality constraints *i.e.,* is the most impactful on the solution of our problem.

To that end, we introduce the *integrality gap* function $\text{gap}_{\text{int}} : [0,1] \to [0,1]$, represented in Figure 5.5, which represents the gap between a variable and $\{0,1\}$:

$$\text{gap}_{\text{int}}(b_k^r) = 1 - (1 - 2b_k^r)^2. \tag{5.5a}$$

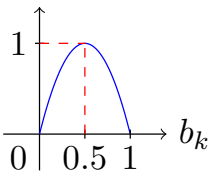


Figure 5.5: Representation of gap_{int}.

As the binary variables b in Problem (5.1) are in $\{0,1\}^{n_b}$, we extend gap_{int} to n_b variables by

summing their integrality gap:

$$\begin{aligned} \text{gap}_{\text{int}}(b^r): \quad [0,1]^{n_b} &\to [0, n_b] && (5.5b)\\ b^r &\mapsto \sum_{k=1}^{n_b} \text{gap}_{\text{int}}(b^r_k). && (5.5c) \end{aligned}$$

The *integrality-gap branching strategy* consists in adding to the tree the node of $\mathcal{C}(\mathcal{T}_p)$ whose solution b^r has the largest integrality gap. More precisely, at an iteration of Algorithm 12, we solve $\hat{\mathcal{B}}_{\mathcal{T}_p}(\{\underline{V}_t\}_{t\in[T]})(x_{init})$, the problem on the current subtree $\mathcal{T}_p$, where the relaxed portions of the scenario tree are approximated with SDDP cuts $\underline{V}^r_{[T]}$. Because $\hat{\mathcal{B}}_{\mathcal{T}_p}(\{\underline{V}_t\}_{t\in[T]})(x_{init})$ is formulated on a pruned subtree, we only have a local solution for nodes in $\mathcal{T}_p$. However, for candidate nodes $\mu \in \mathcal{C}(\mathcal{N}_p)$, which are not in $\mathcal{N}_p$, we can compute a relaxed solution $(x^r_\mu, u^r_\mu, b^r_\mu)$ by solving:

$$\begin{aligned} (\hat{x}^r_\mu, \hat{u}^r_\mu, \hat{b}^r_\mu) = \arg\min_{x,y,b} \quad & \Big[L_{t_\mu}(\hat{x}_{a(\mu)}, u, b, \xi_\mu) + \underline{V}^r_{t_\mu+1}(x) \Big] && (5.6a)\\ x = F_{t_\mu}(\hat{x}_{a(\mu)}, u, b, \xi_\mu) &\subset \mathfrak{X}_{t_\mu} && (5.6b)\\ u \in \mathfrak{Y}_{t_\mu}(\hat{x}_{a(\mu)}, \xi_\mu) & && (5.6c)\\ b \in \mathfrak{B}_{t_\mu}(\hat{x}_{a(\mu)}, \xi_\mu), & && (5.6d) \end{aligned}$$

where $\underline{V}^r_{t_\mu+1}$ is our current cost-to-go approximation at $t_\mu + 1$ and $\hat{x}_{a(\mu)}$ is the local solution of $\mu's$ parent. Then, We can assess the integrality gap of candidate node μ by computing $\text{gap}_{\text{int}}(b^r_\mu)$. Finally, we choose the node that maximizes the integrality gap $\mu^\star := \arg\max_{\mu\in\mathcal{C}(\mathcal{N}_p)} \{\text{gap}_{\text{int}}(b^r_\mu)\}$.

Remark 7. *Note that from one iteration to another, the solution we obtain when solving* $(P^{\mathcal{T}_p}_{x_{init}})$ *can evolve. More precisely, if at iteration* k *we decide to grow the subtree* $\mathcal{T}_p$ *with node* μ*, we begin iteration* $k+1$ *with the subtree* $\mathcal{T}_{p'} = \mathcal{T}^\Omega[\mathcal{N}_p \cup \{\mu\}]$*. The problem we solve* $(P^{\mathcal{T}_{p'}}_{x_{init}})$ *is very close to* $(P^{\mathcal{T}_p}_{x_{init}})$*, but for* $\nu \in \mathcal{N}_{p'}$*, the solution* $\hat{x}_\nu$ *can be differ. In Figure 5.6, we illustrate how growing the subtree with one node can infer on the relaxed variables computed for candidate nodes.*

If we want to consider all nodes not in $\mathcal{N}_p$ as candidates, it is slightly more complicated to compute $\hat{b}^r_\mu$ for a candidate μ. Indeed, if μ is connected to $\mathcal{T}_p$ by the branch $(\nu_k)_{k\in[\tau]}$, we first have to compute $\hat{x}^r_{\nu_1}$ by solving Problem(5.6) for ν_1 with incoming state $\hat{x}_{a(\nu_1)}$. Then, we compute recursively all $\hat{x}^r_{\nu_k}$ by solving Problem(5.6) for ν_k with incoming state $\hat{x}^r_{\nu_{k-1}}$. In the end, we obtain $(\hat{x}^r_\mu, \hat{u}^r_\mu, \hat{b}^r_\mu)$ and can assess the integrality gap $\text{gap}_{\text{int}}(\hat{b}^r_\mu)$ of node μ.

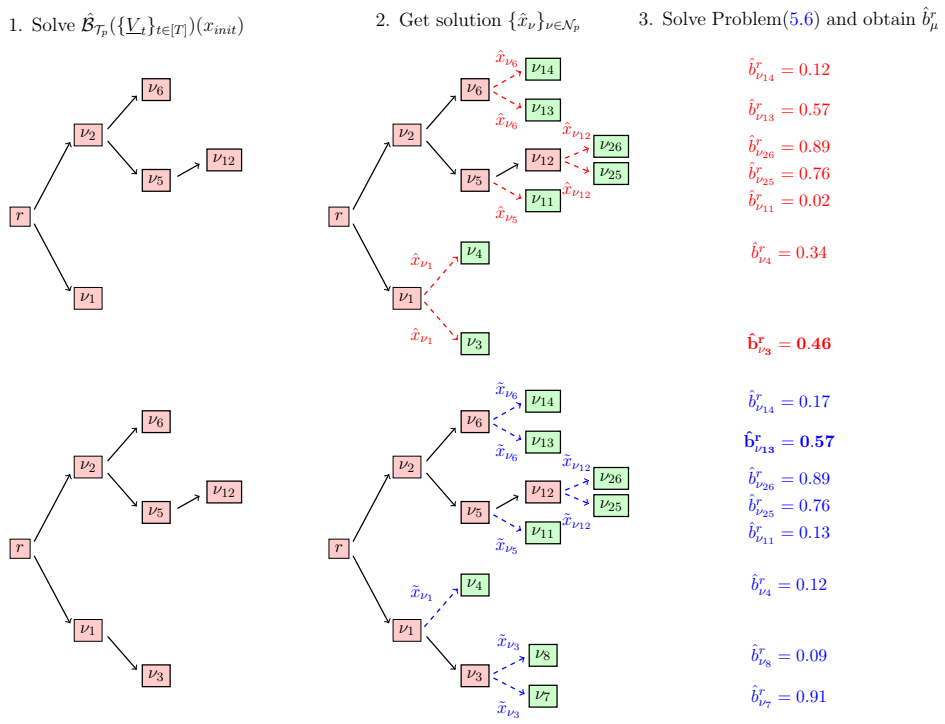


Figure 5.6: Illustration of the methodology to compute the integrality gap of candidate nodes for the integrality gap growing strategy. In red, we represent the current subtree $\mathcal{T}_p$ and the candidate pool in green. Here, we first solve the MILP obtained by partially relaxing the problem on $\mathcal{T}_p$. From the solution, we can compute the relaxed solution of candidate nodes by solving Problem(5.6). Once we have $\hat{b}^r_\mu$ for a candidate μ, we can compute its integrality gap $\text{gap}_{\text{int}}(\hat{b}^r_\mu)$. We choose the node maximizing gap_{int}: here ν_3. The lower part of the figure indicates the next iteration.

5.3.4 Strong Branching

Finally, we propose a strategy led by the improvement of the lower bound obtained by solving $(PR^{\mathcal{T}_p}_{x_{init}})$. Similarly, as in Section 4.3.2, we want to choose to keep integrality in order to improve the most the lower-bound obtained with our lower assignation. Here, starting from a subtree $\mathcal{T}_p$, for each candidate $\mu \in \mathcal{C}(\mathcal{N}_p)$, we consider the candidate subtree $\mathcal{T}'_p = \mathcal{T}^\Omega\big[\mathcal{N}_p \cup \{\mu\}\big]$. We solve each problem $\hat{\mathcal{B}}_{\mathcal{T}'_p}(\{\underline{V}_t\}_{t\in[T]})(x_{init})$ and choose the candidate that improves the most the lower-bound. This strategy corresponds to the commonly used strong branching method in branch-and-bound literature. We thus refer to it as the *strong-branching growing strategy*. For example, in Figure 5.7, we observe an iteration of the algorithm, starting from subtree $\mathcal{T}_p = \mathcal{T}^\Omega\big[\{r, \nu_1, \nu_2, \nu_5\}\big]$. For each candidate $\mu \in \mathcal{C}(\mathcal{N}_p) = \{\nu_3, \nu_4, \nu_6, \nu_{11}, \nu_{12}\}$, we solve $\hat{\mathcal{B}}_{\mathcal{T}^\Omega\left[\mathcal{N}_p\cup\{\mu\}\right]}(\{\underline{V}_t\}_{t\in[T]})(x_{init})$ and obtain a lower-bound lb. We then choose to add the node that improves the most the lower-bound when growing the subtree, in this example ν_4.

Note that this method is extremely heavy in terms of computation. Indeed, at each iteration, we have to solve $|\mathcal{C}(\mathcal{N}_p)|$ MILPs. Further, the size of the children $|\mathcal{C}(\mathcal{N}_p)|$ increases, unless we add a leaf of the initial scenario tree, and the MILPs we have to solve are increasing large, and harder to solve. To find a balance between robustness of the strategy and computational time, we suggest reducing the candidate pool. The $C-$*strong branching strategy* is a strong branching strategy conduced on a batch of candidates, that we select randomly, with uniform distribution,

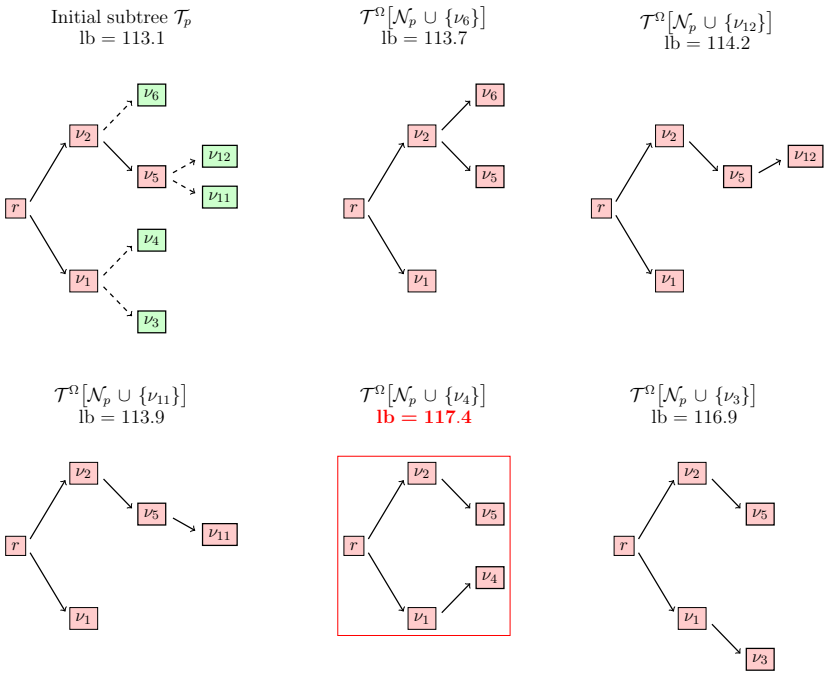


Figure 5.7: Illustration of an iteration of the strong branching growing strategy. In red, we represent nodes in the current subtree $\mathcal{T}_p$ and in green the candidate pool. Then, each subtree corresponds to the candidate subtree $\mathcal{T}'_p = \mathcal{T}^\Omega\big[\mathcal{N}_p \cup \{\mu\}\big]$ for a candidate node μ, and we can read the lower-bound obtained by solving the partially relaxation on $\mathcal{T}'_p$.

in $\mathcal{C}(\mathcal{N}_p)$. Depending on the size of the batch, this strategy reduces significantly the number of computations. For instance, if we take the example in Figure 5.7, with a 2−strong branching strategy we only solve 2 MILP instead of 5.

5.4 Information gained from single scenario model

We have presented a new approach to solve MSbLP relying on partial relaxations of binary variables. In contrast, we confront those approaches with widely used deterministic approaches which relax information– but not integer– constraints. In this section, we first introduce the anticipative framework which relaxes entirely non-anticipativity constraints and assumes the future is known. Though it is unrealistic, we derive from this framework an indicator to measure the impact of uncertainty on a problem. Then, we present the Model Predictive Control (MPC) methodology which is an adaptive deterministic algorithm employed in practice to solve problems under uncertainties.

5.4.1 The Perfect Information lower bound

A natural lower bound for stochastic problems comes from relaxing the non-anticipativity constraint (2.8e). We are then in the anticipative, or Perfect Information (PI), framework which consists in assuming that we can look into the future and know the noises realization (*e.g.,* what energy costs at any given time). More precisely, the anticipative policy returns, for each scenario $\xi^{\mathrm{PI}}_{[T]}$ a solution that perfectly fits this scenario *i.e.,* that is an optimal solution to the deterministic problem:

$$\hat{V}^{\mathrm{PI}}(x_{init}, \xi^{\mathrm{PI}}_{[T]}) := \min_{x,y,b} \quad \sum_{t=1}^{T} L_t(x_{t-1}, y_t, b_t, \xi^{\mathrm{PI}}_t) \tag{5.7a}$$

$$x_t = F_t(x_{t-1}, y_t, b_t, \xi^{\mathrm{PI}}_t) \subset \mathfrak{X}_t \qquad \forall t \tag{5.7b}$$

$$y_t \in \mathfrak{Y}_t(x_{t-1}, \xi^{\mathrm{PI}}_t) \qquad \forall t \tag{5.7c}$$

$$b_t \in \mathfrak{B}_t(x_{t-1}, \xi^{\mathrm{PI}}_t) \cap \{0,1\}^{n_b} \qquad \forall t \tag{5.7d}$$

$$x_0 = x_{init}. \tag{5.7e}$$

Note that, obviously, this policy is usually not admissible for Problem (2.9) as it requires unavailable information. However, by definition, $\hat{V}^{\mathrm{PI}}(x_{init}, \xi^{\mathrm{PI}}_{[T]})$ is a lower bound of the cost incurred by any admissible policy on this scenario. Then, the PI lower bound is defined as the average over all scenarios of the value yielded by the anticipative policy:

$$V^{\mathrm{PI}}(x_{init}) := \mathbb{E}\big[\hat{V}^{\mathrm{PI}}(x_{init}, \boldsymbol{\xi}_{[\boldsymbol{T}]})\big]. \tag{5.8}$$

Finally, the Expected Value of Perfect Information (EVPI) [RS61] is defined as:

$$EVPI = V_r(x_{init}) - V^{\mathrm{PI}}(x_{init}). \tag{5.9}$$

The EVPI characterizes the impact of uncertainty on a stochastic problem.

5.4.2 Model Predictive Control

In the industry, the Model Predictive Control (MPC) algorithm is widely used to solve operational problems with an adaptive approach. Although this algorithm (see Algorithm 13) is deterministic—meaning all uncertain parameters are fixed— it adapts by solving a new optimization problem at each stage to adjust for real-time uncertainties, fixing only the decisions for the current stage t. Concretely, at each stage t, Algorithm 13 solves a deterministic approximation of the problem on horizon $[t : T]$, but only enforces the decisions related to stage t.

Though MPC does not guarantee optimality, it has proven effective in practice. For instance, in Chapter 3, we show that for the operational problem MPC yields better policies than stochastic approaches. However, MPC can construct infeasible policies, as illustrated in Example 2.

Algorithm 13: Model Predictive Control

Input : x_{init}, initial forecast $\xi^f_{[T]}$;
for $t : 1 \to T$ **do**
  Observe random realization ξ_t at t;
  Update forecast $\xi^f_{[t:T]}$;

$$
\begin{aligned}
(u^\star_{[t:T]}, b^\star_{[t:T]}) \in \underset{x_{[t:T]}, y_{[t:T]}, b_{[t:T]}}{\arg\min} \quad & \sum_{\tau=t}^{T} L_\tau(x_{\tau-1}, y_\tau, b_\tau, \xi^f_\tau) \\
& x_\tau = F_\tau(x_{\tau-1}, y_\tau, b_\tau, \xi^f_\tau) \subset \mathfrak{X}_\tau && \forall \tau \\
& y_\tau \in \mathfrak{Y}_\tau(x_{\tau-1}, \xi^f_\tau) && \forall \tau \\
& b_\tau \in \mathfrak{B}_\tau(x_{\tau-1}, \xi^f_\tau) \cap \{0,1\}^{n_b} && \forall \tau \\
& x_{\tau-1} = x_{t-1}.
\end{aligned}
$$

$x_t := F_t(x_{t-1}, u^\star_t, b^\star_t, \xi^f_t)$;

Example 2 (An example where MPC fail)**.** *Consider a production unit that produces two products $j = A, B$ over $T = 2$ time steps and one machine. The shared resource constraint, modeled through binary variables b^j_t, implies that we must decide which product to produce at $t = 1$, and which at $t = 2$. The demand in product B is a random variable $\boldsymbol{d^B}$ uniformly distributed in $\{0, 2\}$. The problem reads*

$$
\begin{aligned}
\min \quad & 2u^A_1 + u^B_1 + u^A_2 + u^B_2 && && (5.10a)\\
s.t. \quad & u^A_1 + u^A_2 \geq 1 && && (5.10b)\\
& u^B_1 + u^B_2 \geq \boldsymbol{d^B} && && (5.10c)\\
& 0 \leq u^j_t \leq 2b^j_t && j = A, B, t = 1, 2 && (5.10d)\\
& b^A_t + b^B_t \leq 1 && t = 1, 2 && (5.10e)\\
& b^j_t \in \{0, 1\}, u^j_t \geq 0 && j = A, B, t = 1, 2. && (5.10f)
\end{aligned}
$$

If we solve this problem with MPC, then at $t = 1$, using the expected value of $\boldsymbol{d^B}$ as an oracle, we solve the following problem:

$$
\begin{aligned}
\min \quad & 2u^A_1 + u^B_1 + u^A_2 + u^B_2 && && (5.11a)\\
s.t. \quad & u^A_1 + u^A_2 \geq 1 && && (5.11b)\\
& u^B_1 + u^B_2 \geq \mathbb{E}[\boldsymbol{d^B}] = 1 && && (5.11c)\\
& 0 \leq u^j_t \leq 2b^j_t && j = A, B, t = 1, 2 && (5.11d)\\
& b^A_t + b^B_t \leq 1 && t = 1, 2 && (5.11e)\\
& b^j_t \in \{0, 1\}, u^j_t \geq 0 && j = A, B, t = 1, 2. && (5.11f)
\end{aligned}
$$

The optimal solution is to produce B first and then A, as A is cheaper in the second stage i.e., *$u^B_1 = 1, u^A_1 = 0$. Then, at $t = 2$, if the uncertainty that realizes is $\boldsymbol{d^B} = 2$, we have to solve*

the following problem:

$$
\begin{aligned}
\min \quad & u_2^A + u_2^B && && (5.12a)\\
s.t. \quad & u_2^A \geq 1 && && (5.12b)\\
& 1 + u_2^B \geq 2 && && (5.12c)\\
& 0 \leq u_2^j \leq 2b_2^j && j = A, B && (5.12d)\\
& b_2^A + b_2^B \leq 1 && && (5.12e)\\
& b_2^j \in \{0,1\}, u_2^j \geq 0 && j = A, B. && (5.12f)
\end{aligned}
$$

This problem is infeasible and the value obtained with MPC is thus $+\infty$.

5.5 Numerical results

In this section, we test Algorithm 12 on small instances to compare the different branching strategies introduced in Section 5.3. Bear in mind that the example we chose is similar to the one in Chapter 3, where we showed that a deterministic method worked better than the look-ahead methodology, corresponding to the shortsighted branching strategy. However, this example allows us, in small instances, to understand how the different growing strategies perform. We first present the model of the problem, see Chapter 2 for a more detailed explanation of modeling choices.

5.5.1 An industrial problem with exclusion constraints

We consider a facility with I machines that produce up to J types of products that can be stored. Moreover, the facility owns an Energy Storage System (ESS). Our goal is to provide the facility with a joint production and energy supply planning, on a discrete horizon $t \in [T]$. One of the particularities of the problem is that some products cannot be produced simultaneously, which translates as *exclusion constraints*. We consider energy prices $\{\boldsymbol{p}_t^{\boldsymbol{grid}}\}_{t\in[T]}$ as random variables and choose a multistage stochastic formulation, leading to Problem (5.13), with notations regrouped in Table 5.1.

$$\min_{\boldsymbol{y_{[T]}},\boldsymbol{x_{[T]}},\boldsymbol{b_{[T]}}} \quad \mathbb{E}\left[\sum_{t=1}^{T} \boldsymbol{p_t^{grid}} \boldsymbol{q_t^{\text{grid}}}\right] \tag{5.13a}$$

$$s.t. \quad \boldsymbol{s_t^j} = \boldsymbol{s_{t-1}^j} - d_t^j + \sum_i \boldsymbol{u_t^{i,j}} \qquad \forall t, j \tag{5.13b}$$

$$u_{min}^{i,j} \boldsymbol{b_t^{ij}} \leq \boldsymbol{u_t^{i,j}} \leq u_{max}^{i,j} \boldsymbol{b_t^{ij}} \qquad \forall i, j, t \tag{5.13c}$$

$$\sum_j \boldsymbol{b_t^{ij}} \leq 1 \qquad \forall i, t \tag{5.13d}$$

$$\max_{i\in[I]} \boldsymbol{b_t^{ij}} + \max_{i\in[I]} \boldsymbol{b_t^{ij'}} \leq 1 \qquad \forall (j, j') \in \mathcal{J} \tag{5.13e}$$

$$\boldsymbol{\phi_t^-} - \boldsymbol{\phi_t^+} + \boldsymbol{q_t^{\text{grid}}} \geq \sum_{i,j} \alpha^{i,j} \boldsymbol{u_t^{i,j}} + \beta^{i,j} \boldsymbol{b_t^{ij}} \qquad \forall t \tag{5.13f}$$

$$\boldsymbol{SOC_t} = \boldsymbol{SOC_{t-1}} - \frac{1}{\rho}\boldsymbol{\phi_t^-} + \rho \boldsymbol{\phi_t^+} \qquad \forall t \tag{5.13g}$$

$$\boldsymbol{\phi_t^+}, \boldsymbol{\phi_t^-} \leq \overline{\Phi} \qquad \forall t \tag{5.13h}$$

$$SOC_{min} \leq \boldsymbol{SOC_t} \leq SOC_{max} \qquad \forall t \tag{5.13i}$$

$$\boldsymbol{s_T^j} \geq s_{\text{final}}^j \qquad \forall t, j \tag{5.13j}$$

$$\boldsymbol{b_t^{ij}} \in \{0, 1\} \qquad \forall i, j, t \tag{5.13k}$$

$$\sigma(\boldsymbol{y_t}) \subset \sigma(\boldsymbol{q_{[t]}^{\text{grid}}}) \qquad \forall t. \tag{5.13l}$$

To simplify notations, we regroup state variables at t in the vector $\boldsymbol{x_t} := \left(\boldsymbol{s_t^{[J]}}, \boldsymbol{SOC_t}\right)$ and continuous control variables in $\boldsymbol{y_t} := \left(\boldsymbol{\phi_t^-}, \boldsymbol{\phi_t^+}, \boldsymbol{q_t^{\text{grid}}}, \boldsymbol{u_t^{[I],[J]}}\right)$. The objective (5.13a) is to minimize energy costs, which are reduced to the cost of the energy purchased from the main grid $\boldsymbol{q_t^{\text{grid}}}$. We must satisfy the demand for each product, and to this end, we can decide how much quantity to produce $\boldsymbol{u_t^{i,j}}$ and then store the product. Typical stock and ESS dynamics are modeled in (5.13b) and (5.13g). The production is bounded (5.13c), and each machine can only produce one product at a time (5.13d). We denote $\mathcal{J} \subset J^2$ the set of incompatible products such that if $(j, j') \in \mathcal{J}$, they cannot be produced simultaneously (5.13e). We ensure there is always enough energy for the production plan (5.13f), where the energy consumption is linear in the production. Finally, the charge and discharge variables are bounded (5.13h), so is the battery (5.13i), and there is a target end stock s_{final}^j for each product t (5.13j). Finally, to retrieve *relatively complete recourse* hypothesis, that are required for SDDP, we allow and penalize unsatisfied demand d in the objective.

In order to test Algorithm 12, we randomly generate instances of Problem (5.13) with varying sizes (more details on the instance generation are given in Chapter A). Our goal here is mainly to try to understand how the different growing strategies perform. Thus, we look for instances of the problem where we can solve any partial relaxation of the problem to optimality in a reasonable time, and in particular, the initial Problem (5.1). This has guided our choices for the instance sizes (given by T and Q) we study in the next sections. Observe that the computational time does not only depend on the scenario tree size, but also on the constraints, which can significantly make the problem harder.

Table 5.1: Notations for Problem (5.13)

Parameters		Variables	
s^j_{final}	final targeted stock	s^j_t	quantity stored
$[SOC_{min}, SOC_{max}]$	ESS bounds	SOC_t	energy stored
$[u^{i,j}_{min}, u^{i,j}_{max}]$	production bounds	$u^{i,j}_t$	quantity produced
$\mathcal{J}$	incompatibility set	b^{ij}_t	production binary decision
ρ	charging efficiency	ϕ^+_t	energy charged
$\overline{\Phi}$	(dis)charge bounds	ϕ^-_t	energy discharged
$\alpha^{i,j}, \beta^{i,j}$	energy consumption parameters	q^{grid}_t	energy bought
d^j_t	demand of j at t	$\boldsymbol{p}^{\boldsymbol{grid}}_t$	random energy prices
Indices			
t	stages	x_t	state variables
i	machines	y_t	control variables
j	products	b_t	binary variables

5.5.2 Lower bounds for $(P^{\mathcal{T}_p}_{x_{init}})$

In this section, we evaluate the growing strategies introduced in Section 5.3, namely the shortsighted strategy, the random strategy, the Integrality Gap (IG) strategy and the Strong-Branching (SB) strategy. Except for the shortsighted strategy, the growing strategies have different variants (see Section 5.3). For simplicity, at iteration i, the pool of candidates for the random, IG and SB strategies, given a current subtree $\mathcal{T}_i := \mathcal{T}^{\Omega}[\mathcal{N}_i]$, are its children $\mathcal{C}(\mathcal{N}_i)$.

For any subtree $\mathcal{T}_p$, solving $(PR^{\mathcal{T}_p}_{x_{init}})$ with Algorithm 11 gives us a lower bound on Problem (5.1). Thus, we propose to assess a subtree with its corresponding lower bound *i.e.,* the optimal value of $(PR^{\mathcal{T}_p}_{x_{init}})$. In addition, we compute the Perfect Information (PI) lower bound, presented in Section 5.4.1, and the continuous relaxation lower bound with SDDP. Finally, we assess how far the lower bounds are from the optimal value of the problem, $v(P^{\mathcal{T}^{\Omega}}_{x_{init}})$. Indeed, for the chosen instances, the scenario trees are sufficiently small to solve the extensive formulation of Problem (5.1).

Three instances of Problem (5.13) with distinct scenario tree structures are selected to test the methodology. We refer to a particular instance as $\text{I}_{T,Q}$ where T is its number of stages and Q its number of noise realization per stage. First, we consider $\text{I}_{3,9}$, where the initial scenario tree looks like a *bush*, meaning it is short but each layer is well furnished ($T \ll Q$), with a total of $Q^T = 729$ scenarios. In contrast, the scenario tree representing $\text{I}_{8,2}$ (with $Q^T = 256$ scenarios) is more of a twig, high but with sparse layers ($Q \ll T$). Finally, $\text{I}_{5,4}$ (with 1048 scenarios) corresponds to a balanced scenario tree, where $T \approx Q$.

For each instance, the $T-1$ shortsighted subtrees are constructed. To enable comparison and limit computations, random subtrees of the same size as the shortsighted ones are generated. For a given size N, we compute a 95% confidence interval for the lower bound of the random strategy by generating 100 random subtrees of size N. Finally, as the IG and SB strategies are iterative, subtrees of increasing size are solved until solution times become prohibitive. Due to the high computational load, we set a time limit for solving the partially relaxed subproblems: 1000 seconds for shortsighted subtrees and 180 seconds for other strategies, given the greater

number of computations involved.

We observe the lower bounds obtained with different growing strategies depending on the subtrees' size for $I_{3,9}$, $I_{8,2}$ and $I_{5,4}$ on Figures 5.8 to 5.10. Note that when the partially relaxed problem could not be solved to optimality, the optimality gap obtained is represented as an interval. Note that for the random strategy, all subproblems of all instances are solved to optimality and thus the optimality gap interval is not to be confused with the one representing statistical fluctuation. For example in $I_{8,2}$, we observe a gap for the $\{5, 6, 7\}-$shortsighted lower bounds and for the SB strategy from $N = 35$.

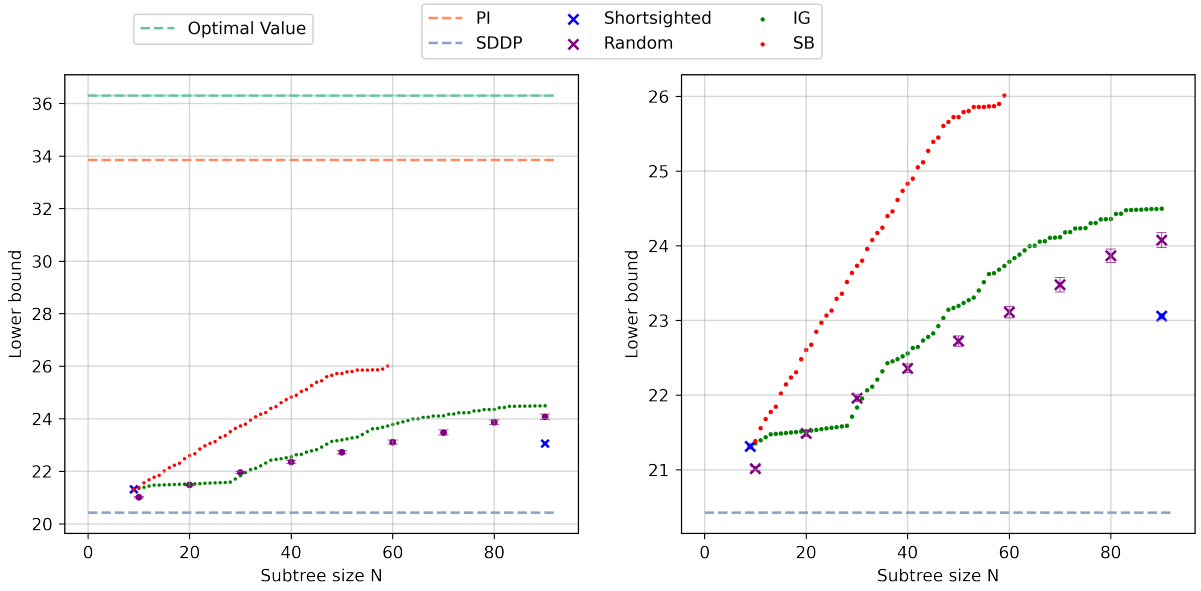


Figure 5.8: $v(PR^{\mathcal{T}}_{x_{init}})$ depending on $\mathcal{T}$ of size N for $I_{3,9}$, zoomed on the right for clarity.

Regarding the improvement of the lower bound, no strategy consistently outperforms the others on these small instances: whereas the SB strategy is preferable for $I_{3,9}$, the random and IG strategies perform just as well for $I_{5,4}$, and for $I_{8,2}$, the shortsighted strategies yield the best lower bounds by far. This is likely due to the structure of the problems: depending on how uncertainties affect future costs or the significance of binary constraints, one strategy may be more advantageous than others. Thus, at this point, we cannot provide specific guidelines on which growing strategy to select.

Finally, we observe in each example that even the best lower bounds computed are far from the optimal value (we synthesize the results in Tables B.1 to B.3). In $I_{3,9}$, the best lower bound reaches a value of 26, leaving a 28% gap from the optimal value, which is approximately 36.3. Here, considering subtrees containing at most 12.3% of the initial scenario tree is not enough to obtain a good lower bound on the value of the problem. Similarly, in $I_{8,2}$ we have at best a 15% optimality gap and a 22% one for $I_{5,4}$. Thus, even though this methodology yields improved lower bounds compared to SDDP, it does not give a satisfactory approximated optimal value of Problem (2.9).

In the next section, we assess the different growing strategies through another metric: we look at the policies that can be derived from subtrees grown in a certain way to solve Problem (2.9).

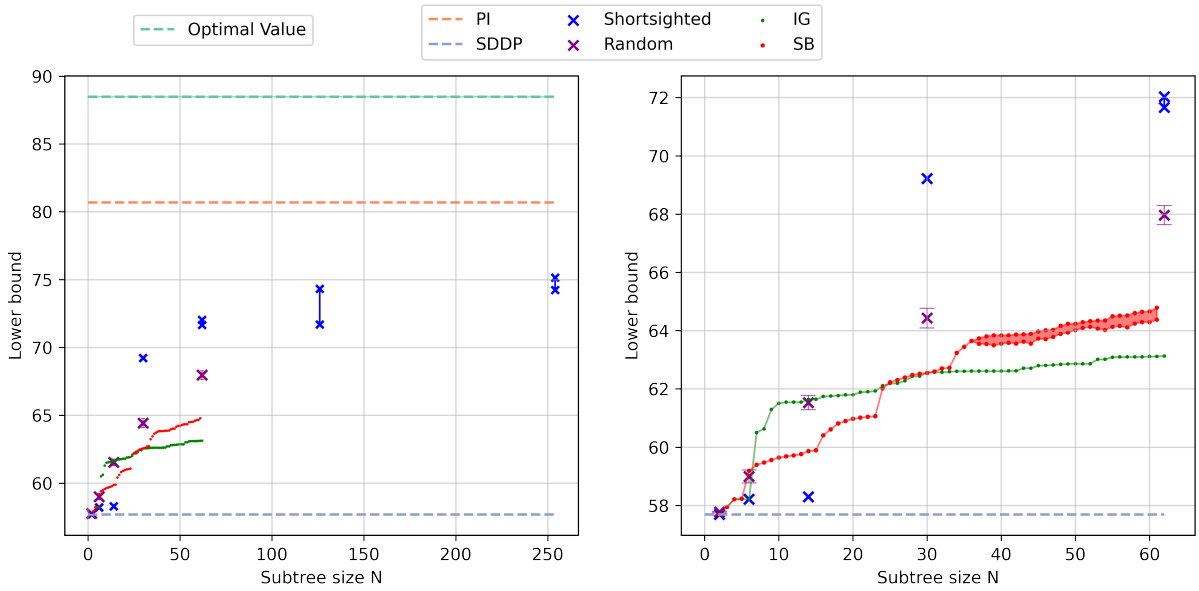


Figure 5.9: $v(PR^{\mathcal{T}}_{x_{init}})$ depending on $\mathcal{T}$ of size N for $I_{8,2}$.

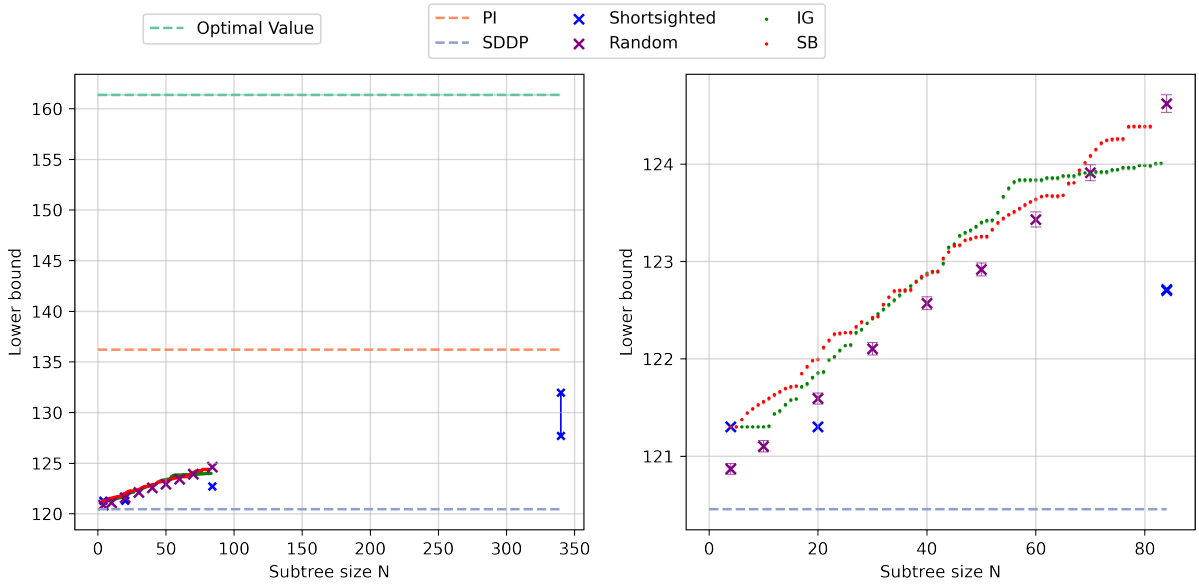


Figure 5.10: $v(PR^{\mathcal{T}}_{x_{init}})$ depending on $\mathcal{T}$ of size N for $I_{5,4}$.

5.5.3 Simulating policies

In order to compute a policy for stage t, we introduce the *forward bellman operators* $\hat{\mathcal{F}}_{\mathcal{T}_{|\nu_0}}$ which depend on a ν_0-extracted subtree with $t_{\nu_0} = t$ and $\nu_0 \neq r$. These operators rely on the cost-to-go

approximations retrieved from SDDP cuts $V^r_{[T]}$:

$$\hat{\mathcal{F}}_{\mathcal{T}_{|\nu_0}}(V^r_{[T]})(x) = \arg\min_{x,y,b} \sum_{\nu \in \mathcal{N}_{|\nu_0}} \pi_\nu L_{t_\nu}(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) + \sum_{\mu \in \mathcal{C}(\mathcal{N}_{|\nu_0})} \pi_\mu V^r_{t_\mu}(x_{a(\mu)}, \xi_\mu)$$

$$x_\nu = F_{t_\nu}(x_{a(\nu)}, y_\nu, b_\nu, \xi_\nu) \subset \mathfrak{X}_{t_\nu} \qquad \forall \nu \in \mathcal{N}_{|\nu_0} \qquad (5.14a)$$

$$y_\nu \in \mathfrak{Y}_{t_\nu}(x_{a(\nu)}, \xi_\nu) \qquad \forall \nu \in \mathcal{N}_{|\nu_0} \qquad (5.14b)$$

$$b_\nu \in \mathfrak{B}_{t_\nu}(x_{a(\nu)}, \xi_\nu) \cap \{0,1\}^{n_b} \qquad \forall \nu \in \mathcal{N}_{|\nu_0} \qquad (5.14c)$$

$$x_{a(\nu_0)} = x. \qquad (5.14d)$$

With those operators, we can simulate for any scenario a policy depending on a growing strategy with the rolling horizon procedure. Starting from state x, we observe the uncertainty at stage t corresponding to node ν_0. Given the chosen growing strategy, we construct a ν_0-rooted subtree $\mathcal{T}_{|\nu_0}$ with its related forward operator $\hat{\mathcal{F}}_{\mathcal{T}_{|\nu_0}}$. We then compute a solution $(\hat{x}_\nu, \hat{y}_\nu, \hat{b}_\nu) \in \hat{\mathcal{F}}_{\mathcal{T}_{|\nu_0}}(V^r_{[T]})(x)$, where $V^r_{[T]}$ are the cuts computed by SDDP. Finally, we enforce only the decisions $(\hat{x}_{\nu_0}, \hat{y}_{\nu_0}, \hat{b}_{\nu_0})$ computed for node ν_0, resulting in outgoing state x'. We iterate until we reach the final stage.

We can compare the strategies obtained following this procedure with the one computed by MPC, presented in Section 5.4.2. Bear in mind that in the previous section, we tested the performance of those strategies on small instances of the problem. We did not specifically select instances where MPC performs poorly; in fact, we observe that MPC produces almost optimal policies in these cases. In the next section, we will present a slight adaptation of the application, for which deterministic methods, such as MPC, perform poorly.

Due to the numerous computations, we select a number of strategies to test and simulate their induced policies over all scenarios. This is possible with the limited number of scenarios ($256, 729$ and 1024) for the selected instances. The distribution over all scenarios obtained with different policies for $\mathrm{I}_{3,9}, \mathrm{I}_{2,8}$ and $\mathrm{I}_{5,4}$ are represented in Figures 5.11 to 5.13. The expected value and standard deviation obtained with different policies are gathered in Tables B.4 to B.6 in Chapter B. Before analyzing results, we have to point out that in the case of $\mathrm{I}_{3,9}$, since the problem has only 3 stages and there are no decisions at the root r, the policy induced by the $2-$shortsighted strategy is actually the optimal policy of the problem.

As in the previous section, we cannot identify a growing strategy that consistently outperforms the others. If for $\mathrm{I}_{3,9}$ and $\mathrm{I}_{8,2}$, the shortsighted strategy yields lower average costs, the IG strategy performs better in $\mathrm{I}_{5,4}$. This suggests that the growing strategy to adopt varies based on the problem's specific characteristics and size. The distribution of results also shows variability, indicating no clear dominance of one approach.

However, for small-scale problems like these, the shortsighted strategy is viable as it remains computationally feasible. With more time steps or uncertainty realizations, this strategy may no longer be practical. On the other hand, the MPC approach proves to be highly effective, achieving competitive results with relatively low computation time. This efficiency is likely due to the high linearity of costs: solving the problem on average is quite effective.

Overall, when the size of the subtrees generated is adequately large, the shortsighted, random, and IG strategies yield satisfactory policies. In particular for $\mathrm{I}_{5,4}$, the distribution of costs with subtrees of size 20, compared to size 4 significantly, is way more compact, meaning the risk of having costly simulations diminishes. This is promising for larger problems where MPC may

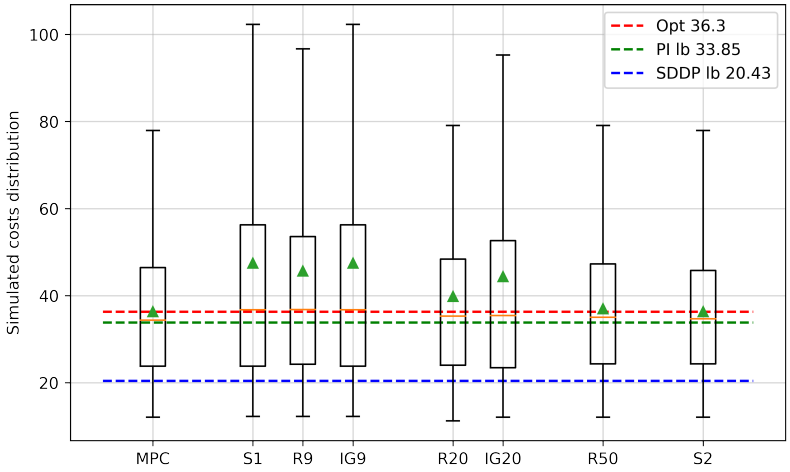


Figure 5.11: Distribution of costs obtained by simulating different policies to solve $I_{3,9}$, over all 729 scenarios.

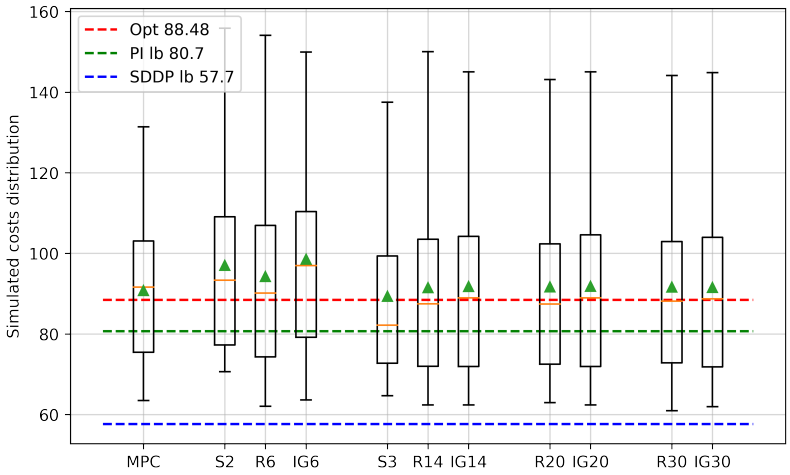


Figure 5.12: Distribution of costs obtained by simulating different policies to solve $I_{8,2}$, over all 256 scenarios.

not perform well, suggesting that these methods could effectively address MSbLP. In the next section, we introduce specific constraints (*e.g.,* bounding grid purchase variables) where our approach leads to better performance than MPC, even in smaller cases.

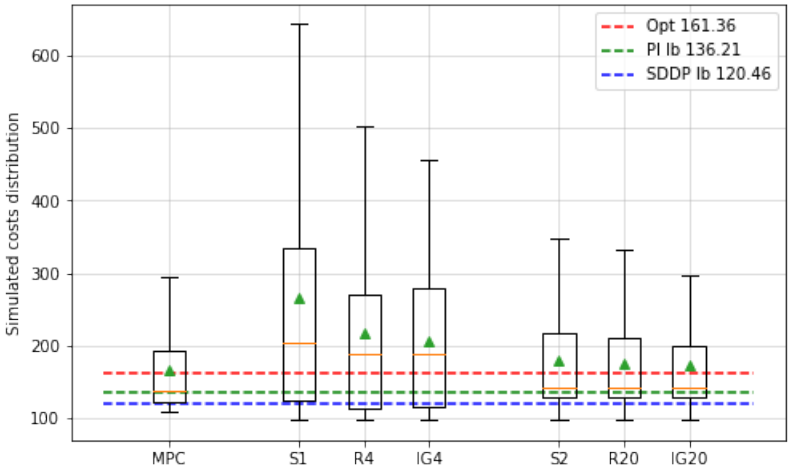


Figure 5.13: Distribution of costs obtained by simulating different policies to solve $I_{5,4}$, over all 1048 scenarios.

5.5.4 Bounding the energy purchases

We concluded in the previous section that the problem presented in Section 5.5.1 can efficiently be solved with a deterministic approach, MPC. The methodology we developed is meant for solving problems where uncertainties must be taken into account. To complement the performance analysis made in Sections 5.5.2 and 5.5.3, we consider in this section a slightly different application.

In this section, by slightly modifying the problem, we find an application where having a deterministic approach can be highly penalized. Specifically, at each stage t, we bound the instantaneous costs, which is in this problem the energy purchased on the main grid. In other words, we give a budget for grid purchases at each stage. This is modeled with the following constraint that is added to the model (5.13):

$$\boldsymbol{p}_t^{\boldsymbol{grid}} \boldsymbol{q}_{\boldsymbol{t}}^{\text{grid}} \leq B \qquad \forall t. \tag{5.15}$$

Since the energy prices are random variables, a lack of anticipation, for instance in deterministic approaches, can affect feasibility. However, we recover relatively complete recourse hypothesis, as in Section 5.5.1, by penalizing the violation of this new constraint (5.15).

Finally, we evaluate the different strategies through their induced policies, as in the previous section, for two instances: $I_{8,2}$ and $I_{15,4}$. $I_{8,2}$ allows us to compute the exact policies (as we can simulate over all scenarios) and to estimate the gap with the optimal value of the problem. In contrast, $I_{15,4}$ is a large problem on which we are unable to solve the extensive formulation with an MILP solver. Neither can we compute the exact policies as there are 4^{15} scenarios: we thus evaluate policies on 1000 randomly selected scenarios.

We can compare the policies obtained for $I_{8,2}$ in Figure 5.14. First, the policy given by MPC is far from the optimal value, with a 61% optimality gap, which is reduced to 10% with the S4 policy. Actually, the policies induced by our approach, on subtrees containing more than 14 nodes, yield a better expected cost than the MPC policy. However, the cost distribution of the MPC policy is more compact than the other policies, which indicates that outliers impact significantly its expected cost.

Regarding the computational time of the different methods (see Table 5.2), for $I_{8,2}$, MPC is the fastest method, which is not surprising as the problem is small. For the shortsighted and random

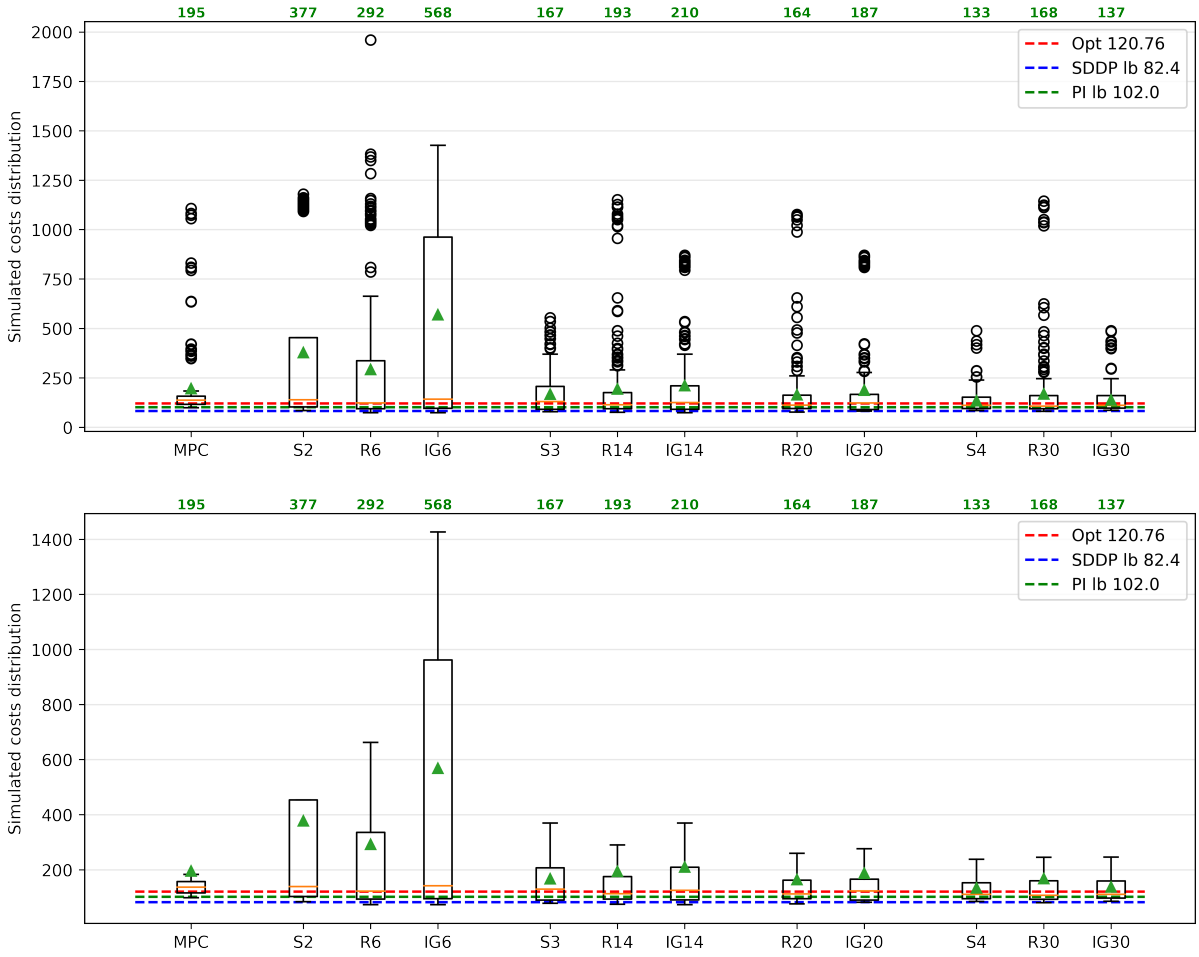


Figure 5.14: Distribution of costs obtained by simulating different policies to solve $I_{8,2}$, over all 256 scenarios. The expected value of each policy is on top of the figure in green.

strategies, computational time are reasonable until a size of 30. Indeed, the computational time increases with the size of the subtrees generated to compute a policy. In particular, the IG policy is heavy to compute: for each time step $t \in [8]$, constructing a subtree requires solving $|\mathcal{T}|$ optimization problems. Thus, even with small subtrees, it takes longer than all other methods.

	Shortsighted			Random			IG			MPC		
$\|\mathcal{T}\|$	min	mean	max	min	mean	max	min	mean	max	min	mean	max
6	0.8	1.0	3.0	0.7	1.1	2.3	4.1	6.2	12	0.8	0.2	0.1
14	2.6	4.8	8.5	2.4	3.9	7.2	24	30	44			
20	-	-	-	4.4	9.6	22	45	53	64			
30	9.8	26	74	8.9	22	51	130	178	300			

Table 5.2: Minimum, average and maximum computational time (in seconds) to simulate a trajectory with different approaches for $I_{8,2}$. To compute a policy with our approach, we first need to run SDDP on the continuous relaxation of the problem, which here takes about 2 seconds.

In Figure 5.15, we present results for a larger instance, $I_{15,4}$. While the optimal value cannot be computed due to the extensive problem size, SDDP provides a valid lower bound. The cost distributions from our approach, regardless of the growing strategy used, are more compact than those obtained with MPC: this is clear when reading Table 5.5, Additionally, whereas MPC has an average cost of 1197, our approach performs better with any growing strategy, though only slightly for S1 and IG4 (with respectively an average cost of 1130 and 1106). Among these, the random policy outperforms all others, even with small subtrees.

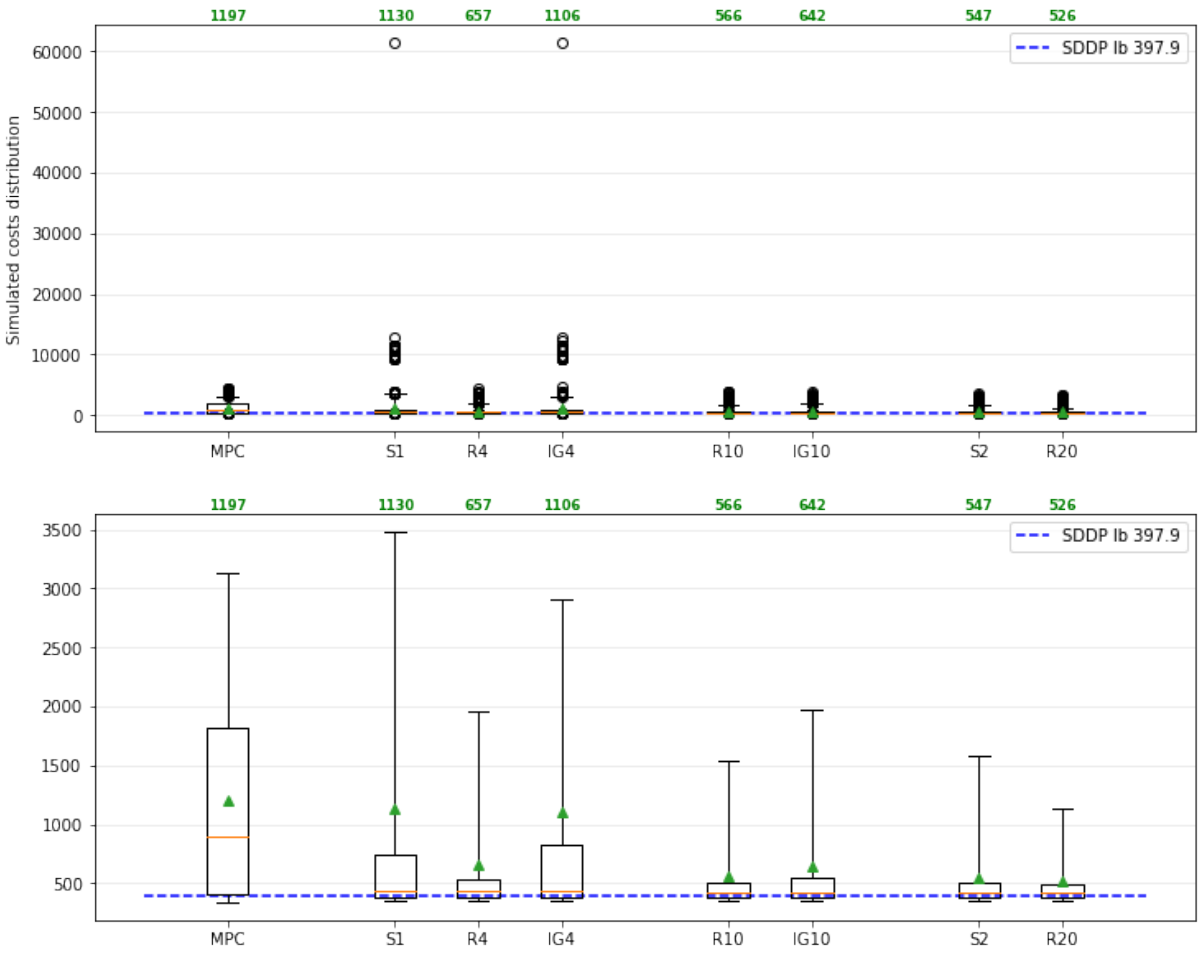


Figure 5.15: Distribution of costs obtained by simulating different policies to solve $I_{15,4}$, over 1000 scenarios. The expected value of each policy is on top of the figure in green.

Increasing subtree sizes can significantly improve policy performances. For example, increasing the size of the subtrees with the random strategy allows to decrease the average cost from 657 with R4 down to 526 with R20. This average cost is again improved with subtrees of size 20 but this comes at the price of higher computational time. For example, with subtrees of size 20 nodes, the average computation time reaches 81 seconds (see Table 5.3). Notably, the IG strategy is the most computationally expensive (we did not compute IG20 for this reason) and, despite this, performs worse than the random policies.

	Shortsighted			Random			IG			MPC		
$\|\mathcal{T}\|$	min	mean	max	min	mean	max	min	mean	max	min	mean	max
4	1.8	2.9	7.9	2.0	3.6	12	8.1	12	61	1.5	4.9	27
10	-	-	-	6.7	13	38	43	88	312			
20	20	126	504	20	81	496	x	x	x			

Table 5.3: Minimum, average and maximum computational time (in seconds) to simulate a trajectory with different approaches for $I_{15,4}$. To compute a policy with our approach, we first need to run SDDP on the continuous relaxation of the problem, which here takes about 22 seconds.

We observe in Figure 5.15 that some outliers are particularly costly and have a huge impact on the expected cost of a policy. This is the case for S1 and IG4 where we can see a single outliers at a cost higher than 60000. Those very costly simulations are due to infeasible solutions. Indeed, for relatively complete recourse assumptions, we relax hard constraints by penalizing the violation in the objective. In Table 5.4, we find details on the solutions computed with all policies regarding the unsatisfied demand (violation of constraint (5.1b)) and the energy budget violation (5.15). Then, we can see how MPC violates the energy budget constraints way more than our approach (due to a lack of anticipativity on the uncertain prices), while our approaches on subtrees that are too small struggle more to satisfy the demand at all time. Finally, in Table 5.5, we find a table with the percentiles of the distributions represented in Figure 5.15. In particular, by comparing the mean and the 50% percentile of S1 and IG4, we can explain how their average cost is so high compared to their distributions in Figure 5.15.

	Shortsighted		Random		IG		MPC	
$\|\mathcal{T}\|$	(5.1b)	(5.15)	(5.1b)	(5.15)	(5.1b)	(5.15)	(5.1b)	(5.15)
4	0.50	0.20	0.06	0.16	0.47	0.20	0.0	0.77
10	-	-	0.03	0.10	0.07	0.14		
20	0.02	0.09	0.01	0.09	x	x		

Table 5.4: Average penalization for demand satisfaction (5.1b) and energy purchase bound (5.15) equations in model $I_{15,4}$.

		4			10		20	
	MPC	S	Random	IG	Random	IG	S	Random
mean	1198	1131	658	1107	567	643	547	526
min	277	326	322	327	321	329	324	323
1%	315	343	339	340	339	341	342	337
10%	363	362	361	360	359	360	361	360
25%	404	382	383	382	381	379	380	378
50%	902	432	433	437	420	422	422	417
75%	1820	748	541	831	507	549	504	489
90%	2513	1959	1440	1959	772	1570	721	637
99%	3929	11165	3078	10540	2994	2677	2490	2400
max	4432	61346	4318	61346	3762	3746	3505	3352

Table 5.5: Percentiles for $I_{15,4}$.

Conclusion

In conclusion of these preliminary results, our approach shows promise. Indeed, in cases such as in Section 5.5.4 where MPC performs poorly, and SDDP cannot compute a feasible solution, our approach can compute feasible policies in a reasonable time. Further, this framework is very flexible since we could use any approximations of the cost-to-go functions to compute a policy. Future works will focus on finding a growing strategy that outperforms the random one.

Appendix A

Instance generation

More precisely, for a given size of the instance, *i.e.,* T, Q, I, J, we fix some parameters and draw others randomly in specific uniform distributions. The values chosen for uniform distributions are inspired from the application presented in Chapter 3, but adapted to a given instance size. Then, some parameters are the same for all instances, for example the battery parameters:

$$SOC_{min} = 1, SOC_{max} = 12, \rho = 0.9, \overline{\Phi} = 3.$$

Then, the production bounds $u^{i,j,min}$ and $u^{i,j,max}$ are randomly drawn from respective uniform distribution $U(30, 55)$ and $U(60, 90)$. For all instances, i and j, we fix $\alpha^{i,j} = 0.001$ and we draw $\beta^{i,j}$ from uniform distribution $U(0.001, 0.06)$. In the next subsections, we specify for each instance the actual values on which we tested the different strategies, as well as the demand and prices.

A.1 Parameters for $\mathrm{I}_{3,9}$

For this instance, we fix the number of machines $I = 3$ and of products $J = 4$. As there are few time steps, there is no stage demand *i.e.,* $d_t^j = 0$ for $t \in [3]$ and $j \in [4]$, but a final stock target $s_{\text{final}} := (59, 58, 41, 63)$. We set:

$$\beta = \begin{pmatrix} 1.3 & 2.3 & 0.3 & 2.3 \\ 0.3 & 1.3 & 0.3 & 0.3 \\ 0.3 & 0.3 & 2.3 & 1.3 \end{pmatrix}, u_{min} = \begin{pmatrix} 45 & 30 & 45 & 47 \\ 33 & 50 & 46 & 34 \\ 45 & 45 & 54 & 49 \end{pmatrix}, u_{max} = \begin{pmatrix} 67 & 68 & 74 & 76 \\ 69 & 63 & 79 & 62 \\ 70 & 63 & 61 & 69 \end{pmatrix}.$$

The products that are incompatible are the couples in set $\mathcal{J} = \{(1, 4), (2, 3), (3, 4)\}$. Finally, the discrete values taken by $\boldsymbol{p}_t^{\boldsymbol{grid}}$ at each stage are represented with the matrix:

$$\begin{pmatrix} 39 & 36 & 41 & 11 & 22 & 30 & 8 & 9 & 12 \\ 20 & 40 & 33 & 5 & 31 & 20 & 41 & 22 & 45 \\ 47 & 9 & 5 & 17 & 14 & 47 & 27 & 10 & 14 \end{pmatrix}.$$

A.2 Parameters for $I_{8,2}$

A.2.1 Application described in Section 5.5.1

For this instance, we fix the number of machines $I = 3$ and of products $J = 4$. We set:

$$\beta = \begin{pmatrix} 0.3 & 2.3 & 1.3 & 0.3 \\ 0.3 & 1.3 & 2.3 & 2.3 \\ 0.3 & 0.3 & 2.3 & 1.3 \end{pmatrix}, u_{min} = \begin{pmatrix} 33 & 35 & 40 & 44 \\ 47 & 48 & 48 & 44 \\ 48 & 35 & 41 & 35 \end{pmatrix}, u_{max} = \begin{pmatrix} 61 & 81 & 78 & 74 \\ 73 & 82 & 86 & 83 \\ 72 & 83 & 69 & 81 \end{pmatrix}$$

The demand and random prices over time are given by:

$$d = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 11 & 34 & 25 & 24 \\ 28 & 10 & 2 & 20 \\ 2 & 15 & 37 & 24 \\ 2 & 2 & 5 & 12 \\ 18 & 10 & 0 & 5 \end{pmatrix}, \qquad \boldsymbol{p^{grid}} := \begin{pmatrix} 29 & 29 \\ 43 & 18 \\ 30 & 35 \\ 11 & 26 \\ 9 & 22 \\ 10 & 46 \\ 36 & 45 \\ 38 & 20 \end{pmatrix}.$$

Finally, the products that are incompatible are the couples in set $\mathcal{J} = \{(1,2),(1,3),(1,4),(3,4)\}$, and the final stock target is $s_{\text{final}} := (58, 65, 37, 34)$.

A.2.2 Adaptation when bounding energy purchases

In Section 5.5.4, we add a constraint (5.15) which represents a budget on the energy purchases. We then test our method on the same instance presented just previously, with slight modifications: the demand and the final stock target are multiplied by 1.5 and we change the scenarios representing $\boldsymbol{p^{grid}}$:

$$d = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 16.5 & 51.0 & 37.5 & 36 \\ 42 & 15.0 & 3 & 30 \\ 3 & 22.5 & 55.5 & 36 \\ 3 & 3 & 7.5 & 18 \\ 27 & 15 & 0 & 7.5 \end{pmatrix}, \qquad \boldsymbol{p^{grid}} := \begin{pmatrix} 20 & 30 \\ 20 & 30 \\ 20 & 30 \\ 20 & 30 \\ 5 & 40 \\ 5 & 40 \\ 5 & 40 \\ 5 & 40 \end{pmatrix}.$$

The final stock is thus $s_{\text{final}} := (87, 97.5, 55.5, 51)$ and we set $B = 60$.

A.3 Parameters for $I_{5,4}$

For this instance, we fix the number of machines $I = 3$ and of products $J = 3$. We set:

$$\beta = \begin{pmatrix} 2.3 & 0.3 & 2.3 \\ 2.3 & 2.3 & 0.3 \\ 2.3 & 2.3 & 1.3 \end{pmatrix}, u_{min} = \begin{pmatrix} 43 & 34 & 34 \\ 42 & 48 & 55 \\ 52 & 32 & 45 \end{pmatrix}, u_{max} = \begin{pmatrix} 65 & 89 & 89 \\ 88 & 81 & 82 \\ 76 & 90 & 60 \end{pmatrix}$$

The demand and random prices over time are given by:

$$d = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 37 & 12 & 5 \\ 27 & 2 & 26 \end{pmatrix}, \qquad \boldsymbol{p^{grid}} := \begin{pmatrix} 12 & 41 & 13 & 37 \\ 37 & 40 & 27 & 38 \\ 18 & 43 & 28 & 19 \\ 9 & 49 & 37 & 9 \\ 47 & 42 & 50 & 20 \end{pmatrix}.$$

Finally, the products that are incompatible are the couples in set $\mathcal{J} = \{(1,3),(2,3)\}$, and the final stock target is $s_{\text{final}} := (136, 102, 55)$.

A.4 Parameters for $\mathrm{I}_{15,4}$

For this instance, we fix the number of machines $I = 3$ and of products $J = 4$. We set:

$$\beta = \begin{pmatrix} 0.3 & 1.3 & 0.3 & 1.3 \\ 1.3 & 0.3 & 2.3 & 1.3 \\ 0.3 & 2.3 & 0.3 & 1.3 \end{pmatrix}, u_{min} = \begin{pmatrix} 49 & 51 & 37 & 47 \\ 40 & 35 & 45 & 54 \\ 40 & 47 & 39 & 54 \end{pmatrix}, u_{max} = \begin{pmatrix} 66 & 85 & 60 & 79 \\ 68 & 62 & 64 & 66 \\ 86 & 60 & 70 & 66 \end{pmatrix}$$

The demand and random prices over time are given by:

$$d = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 28 & 21 & 27 & 32 \\ 27 & 3 & 33 & 2 \\ 5 & 40 & 24 & 24 \\ 33 & 7 & 5 & 39 \\ 6 & 28 & 5 & 4 \\ 12 & 33 & 23 & 35 \\ 14 & 3 & 16 & 15 \\ 9 & 14 & 0 & 32 \\ 7 & 17 & 35 & 30 \\ 17 & 10 & 40 & 40 \\ 31 & 26 & 15 & 32 \\ 15 & 29 & 13 & 8 \\ 29 & 26 & 26 & 35 \end{pmatrix}, \qquad \boldsymbol{p^{grid}} := \begin{pmatrix} 20 & 24 & 26 & 30 \\ 20 & 24 & 26 & 30 \\ 20 & 24 & 26 & 30 \\ 20 & 24 & 26 & 30 \\ 20 & 24 & 26 & 30 \\ 20 & 24 & 26 & 30 \\ 20 & 24 & 26 & 30 \\ 20 & 24 & 26 & 30 \\ 20 & 24 & 26 & 30 \\ 4 & 7 & 40 & 45 \\ 4 & 7 & 40 & 45 \\ 4 & 7 & 40 & 45 \\ 4 & 7 & 40 & 45 \\ 4 & 7 & 40 & 45 \\ 4 & 7 & 40 & 45 \end{pmatrix}.$$

Finally, the products that are incompatible are the couples in set $\mathcal{J} = \{(1,3),(2,3)\}$, the final stock target is $s_{\text{final}} := (415, 333, 259, 314)$, and we set $B = 60$.

Appendix B

Additional content on numerical experiments

In this appendix, we provide tables summarizing the results from Sections 5.5.2 to 5.5.4. In the tables, a '-' indicates subtree sizes that cannot be constructed with the shortsighted strategy, while an 'x' signifies results that could have been computed but were not.

B.1 Values and gap of the lower bounds computed

	Shortsighted		Random		IG		SB	
$\|\mathcal{T}\|$	lb	gap	lb	gap	lb	gap	lb	gap
9	21.3	41	21.0 ± 0.03	42	21.3	41	21.3	41
20	-	-	21.5 ± 0.04	41	21.5	41	22.6	38
30	-	-	22.0 ± 0.04	39	22.0	39	23.7	35
40	-	-	22.4 ± 0.06	38	22.6	38	24.8	32
50	-	-	22.7 ± 0.07	37	23.2	36	25.7	29
60	-	-	23.1 ± 0.08	36	23.8	34	26	**28**
70	-	-	23.5 ± 0.10	35	24.1	34	x	x
80	-	-	23.9 ± 0.09	34	24.4	33	x	x
90	23.0	37	24.1 ± 0.10	0	24.5	34	x	x

Table B.1: Lower bounds and optimality gap (in %) obtained with different strategies for $I_{3,9}$, where the optimal value is opt= 36.3.

	Shortsighted		Random		IG		SB	
$\|\mathcal{T}\|$	value	gap	value	gap	value	gap	value	gap
2	57.7	35	57.7± 0.03	35	x	x	x	x
6	58.2	34	59.0 ± 0.2	33	58.2	34	57.7	35
14	58.3	34	61.5 ± 0.2	30	61.6	30	59.1	33
30	69.2	22	64.4 ± 0.3	27	62.6	30	59.8	32
62	72	19	68.0 ± 0.3	23	63.1	29	62.5	29
126	[71.7, 74.3]	≈ 19	x	x	x	x	x	x
254	75.1	**15**	x	x	x	x	x	x

Table B.2: Lower bounds and optimality gap (in %) obtained with different strategies for $I_{8,2}$, where the optimal value is opt= 88.5. The interval obtained for the shortsighted strategy with 126 nodes is due to the partially relaxed problem not being solved to optimality: this is the gap obtained between the best lower bound and the best solution found.

	Shortsighted		Random		IG		SB	
$\|\mathcal{T}\|$	value	gap	value	gap	value	gap	value	gap
4	121.3	25	120.8 ± 0.05	25	x	x	x	x
10	-	-	121.1 ± 0.06	25	121.3	25	121.3	25
20	121.3	25	121.6 ± 0.06	25	121.3	25	121.6	25
30	-	-	122.1 ± 0.06	24	121.8	24	122.0	24
40	-	-	122.6 ± 0.07	24	122.4	24	122.4	24
50	-	-	122.9 ± 0.07	24	122.9	24	122.9	24
60	-	-	123.4 ± 0.08	23	123.4	23	123.3	23
70	-	-	123.9 ± 0.08	23	123.8	23	123.6	23
84	122.7	24	124.6 ± 0.09	23	123.9	23	124.1	23
340	131.9	**18**	x	x	x	x	x	x

Table B.3: Lower bounds and optimality gap (in %) obtained with different strategies for $I_{5,4}$, where the optimal value is opt= 161.3.

B.2 Simulated policies' average value and standard deviation

	Shortsighted		Random		Integrality Gap		MPC	
$\|\mathcal{T}\|$	mean	std	mean	std	mean	std	mean	std
9	47	33	45	31	47	34		
20	-	-	40	22	44	30		
50	-	-	37	**15**	x	x		
90	**36**	**15**	x	x	x	x		
							36	15

Table B.4: L and best standard deviation of the simulated costs of different growing strategies over the whole scenario set for $I_{3,9}$.

	Shortsighted		Random		Integrality Gap		MPC	
$\|\mathcal{T}\|$	mean	std	mean	std	mean	std	mean	std
6	97	22	94	24	98	25		
14	**89**	**21**	91	23	91	23		
20	-	-	92	23	91	23		
30	-	-	92	23	91	23		
							91	17

Table B.5: Expected value and standard deviation of the simulated costs of different growing strategies over the whole scenario set for the instance with $T = 8$ and $Q = 2$.

	Shortsighted		Random		Integrality Gap		MPC	
$\|\mathcal{T}\|$	mean	std	mean	std	mean	std	mean	std
4	267	181	217	123	205	92		
20	180	76	174	70	**173**	**68**		
							167	70

Table B.6: Expected value and standard deviation of the simulated costs of different growing strategies over the whole scenario set for the instance with $T = 5$ and $Q = 4$.

Part III

Modeling fairness in decision problems

Chapter 6

Fairness: from concept to mathematical models

Contents

In this chapter, we give a general overview of how fairness is defined and modeled across the scientific literature, while making the link to our energy application. We first cover some definitions of fairness, before diving into the existing mathematical treatment of the subject. This work and the following chapter have been done while visiting Pierre Pinson[1] in the second year of the PhD.

First, we present two examples to illustrate the concepts introduced in this section.

Example 3 (Multiportfolio management)**.** *An advisor is in charge of N portfolios with individual interests across various assets. The aggregation of portfolios can be modeled with the following optimization model:*

$$\underset{\{x_i\}_{i\in[N]}}{\text{Max}} \quad \sum_{i\in[N]} r_i(x_i) - c\Big(\sum_{i\in[N]} x_i\Big) \tag{6.1a}$$

$$x_i \in \mathcal{X}_i \qquad \forall i \in [N], \tag{6.1b}$$

where x_i are the trades of i, constrained to be in set $\mathcal{X}_i$, r_i is the revenue function, and c the trading cost function.

Example 4 (Shared Energy storage system (ESS) management)**.** *A manager is in charge of managing an ESS, in which M buildings have invested collaboratively. This example is inspired*

[1] Imperial College of London

by the problem presented in [Jor+24]. We model the aggregation of buildings with:

$$\underset{\{x_t^j\}_{j\in[M],t\in[T]}}{\text{Min}} \quad \sum_{t\in[T]} \sum_{j\in[M]} c_t^j q_t^j \tag{6.2a}$$

$$p_t^j + \phi_t^j + q_t^j \geq d_t^j \qquad \forall t \in [T], \forall j \in [M] \tag{6.2b}$$

$$x_t^j := (p_t^j, \phi_t^j, q_t^j) \in \mathcal{X}_t^j \qquad \forall t \in [T] \forall j \in [M] \tag{6.2c}$$

$$SoC_t = SoC_{t-1} + \sum_{j\in[M]} \phi_t^j \qquad \forall t \in [T], \tag{6.2d}$$

where a building j is modeled by d_t^j, its energy demand at time t; p_t^j, its the energy production at t; q_t^j, the quantity of energy bought from the grid at price c_t^j; and ϕ_t^j, if positive (resp. negative), is the quantity of energy charged to (resp. discharged from) the ESS at t. All variables at t are constrained by set $\mathcal{X}_t^j$. Finally, Soc_t is the State of Charge in the shared battery at t, modeled with dynamic equations (6.2d).

In both examples, to make all participants benefit from the aggregation, the aggregator must ensure fair treatment. In Example 3, the advisor must guarantee equitable distribution of market costs among portfolios. In Example 4, deciding how energy from the shared battery should be allocated is not straightforward as it affects directly the cost of each prosumer. Therefore, the aggregator's choices regarding prosumer access to energy storage has to be fair: should access be proportional to each building's investments, based on energy needs, or should alternative criteria be considered?

6.1 Defining, modeling and accommodating fairness

In the Oxford Dictionary, fairness is defined as *the quality of treating people equally or in a way that is reasonable*. The definition is simple but subjective. Is treating people equally, regardless of any token of individuality, considered fair in society? Furthermore, what does it mean to be reasonable? Whatever take we have on fairness is necessarily subjective and context-dependent. We present here some notions of the philosophical approach to fairness (see [Kon03] for a deeper analysis). We do not pretend to give a thorough description of the philosophical literature, but merely outline some concepts relevant to the following work.

When speaking of fairness, [Kon03] distinguish *fair processes* from *fair outcomes*. By opposition to fairness, we call *unfair* a process or an outcome that is not fair. In the first paradigm, fairness is evaluated not through the outcomes, but through the treatment of each individual in the group that results in said outcomes. This concept is relevant in Machine Learning, for applications like granting or denying loans, bail or parole decisions etc. In such problems, the inherent biases in the data used for algorithm training can lead to unfair predictive outcomes, in the sense that individuals of a given social class are favored compared to others. Hence, it becomes imperative to integrate fairness into the learning process and think of ways to assess fairness across different data populations. We refer to [Jab+17; CH20; RTK22] for more details on the way fairness can be addressed in machine learning.

On the other hand, the *fair outcomes* paradigm takes into consideration the individual outcomes and makes sure that everyone gets their fair share. This approach is favored in game theory where each individual (or player) is modeled with a utility function whose actual value depends on the actions of all players. For a given set of actions, we obtain a utility vector, denoted $u := (u_1, \ldots, u_n)$, representing every agent's utility u_i. A utility vector is then said to be *fair* if it satisfies a set of properties that might vary from one specific fairness definition to another.

Among them, individual rationality is key as it ensures every individual is better off in the aggregation, and therefore accepts to be a part of it. In Example 3, the utility is the benefit of each agent, whereas in Example 4 the utility is the energy costs of each building. In the remainder of this paper, we discuss fair outcomes approaches.

Intuitively, fairness can be confused with *Egalitarianism* where a utility vector is said to be fair if all coordinates are equal, meaning that everyone gets the same share. For instance, the *Gini coefficient* [Gin21] is a commonly used indicator to measure equality—mistaken for fairness—which evaluates how far a given distribution is from the equal distribution. Although it makes sense in some applications, it is impractical most of the time since people have unequal access to resources and different needs: indeed, in Example 4, if we consider equality through the quantity of energy given from the ESS, we would add constraints ensuring everyone gets the exact same amount of energy:

$$\sum_{t=1}^{T} \phi_t^j = \sum_{t=1}^{T} \phi_t^{j'} \qquad \forall j \neq j'.$$

We can see the limits of such modeling as it would provide too much energy to buildings with smaller energy consumption. Moreover, it is found to be unpopular in surveys ([Kon03]), as people feel they get less than they should. Thus, equal resource (and opportunity) distribution does not always address social inequality.

To counteract these side effects, the *Need Principle* aims at satisfying basic needs equally first and then focusing on efficiency. This is a trade-off between need and other distributive goals. In Example 4, each building could decompose its energy demand d_t^j into the minimum energy needed n_t^j plus energy asked for comfort s_t^j. Then, additional constraints (6.3a) can be added to the aggregation model to ensure that everyone gets free energy to satisfy its needs:

$$\begin{aligned} p_t^j + \phi_t^j + q_t^j &\geq n_t^j + s_t^j && \forall t \in [T], \forall j \in [M], \\ p_t^j + \phi_t^j &\geq n_t^j && \forall t \in [T], \forall j \in [M]. \end{aligned}$$

Then, depending on the energy available in the ESS and the energy produced by each building, the manager dispatches the energy to minimize the aggregated costs of energy bought to the grid. The Need Principle finds practical application in Euphemia [EUP16], an algorithm developed to optimize the orders to be executed on the European coupled electricity market. Euphemia pursues a dual objective: first, equitably distributing curtailment among areas where a portion of orders are unaccepted; and second, maximizing social welfare by optimizing the total market value in the Day-Ahead auction. The emphasis is on maximizing order acceptance for each area, and then the algorithm seeks an efficient solution.

A different approach was introduced by [Raw71]: assuming that a group of individuals has no idea of their rank or situation in society, they will agree on a social contract aiming at maximizing the well-being of the least well-off. If the agents possess distinct characteristics, it might be difficult to compare them and ensure equitable treatment among them. This approach to fairness is often referred to as *minimax fairness*, as this amounts to optimizing for the worst objective

among agents. In Example 3, the minimax aggregator is modeled as:

$$\underset{x,t}{\text{Max}} \quad \min_i \ \{ r_i(x_i) - t_i \} \tag{6.3a}$$

$$\text{s.t.} \quad x_i \in \mathcal{X}_i \qquad \forall i \in [N] \tag{6.3b}$$

$$\sum_{i\in[N]} t_i = c \left(\sum_{i\in[N]} x_i \right), \tag{6.3c}$$

where t_i represents what the aggregation charges portfolio i for trading. Then, (6.3c) ensures the sum of cost charged to portfolios equals to the trading cost of the aggregation. This amounts to having transfer variables in-between agents, which is proposed by [IT14] to solve a multi-portfolio problem with fairness considerations. We avoid transfer variables in this paper, as they may raise privacy and trust concerns in practical application. Instead, we simplify the approach by designating the aggregator as the sole entity with complete information on the problem.

Now that we have presented some of the philosophical concepts that define the foundations of fairness, we discuss in the following mathematical ways to model and assess fairness.

6.2 Mathematical models of fairness

6.2.1 Notions of fair solutions

For more than a century, fairness or inequality has been widely discussed in the literature. The first fairness notion can be traced back to [Par14]: a utility vector is said to be Pareto optimal if there are no other accessible utility vector where an individual is better off without negatively impact another. Pareto optimality does not imply fairness in a solution, but guarantees a stability. This concept is also used when trying to find a balance between multiple objectives: traditionally in portfolio management to find a trade-off between a high expected revenue and low risk. In our context, this can be adapted as we look for a trade-off between the total revenue of the aggregation (efficiency) and the fairness of the solution.

One of the main challenges facing fairness challenges is that of resource allocation among agents. This naturally falls into the scope of game theory. In a founding article [Nas50], John Nash introduced the bargaining problem where two rational agents, allowed to bargain, try to maximize the sum of their utilities. Agents are rational, hence individual rationality is ensured through a *disagreement point*, which is the strategy decided by players if they cannot reach an agreement. As agents can cooperate, they must have an agreement on properties a utility vector, u, should satisfy. Nash proposed four axioms to constitute this agreement: *Pareto optimality*; *Symmetry*, applying the same permutation to two utility vectors does not change their order; *Independence of irrelevant alternatives*, if a utility vector is the optimal utility vector within the feasible set, it remains so if the set is reduced. *Scale invariance*: applying affine transformations to the utility vector does not change the social ranking. Then, he showed that, under a number of assumptions (among them, the set of feasible utility vectors must be convex and compact), there exists a unique utility vector satisfying those axioms. This unique utility vector is regarded in the literature as a viable option when seeking fairness. It has been demonstrated that under convexity of the feasible set, it can be obtained by maximizing the product of utilities ([Nas53; Mut99]), and thus by maximizing a logarithmic sum of utilities:

$$\max_{u\in\mathcal{U}} \quad \sum_{i=1}^{N} \log(u_i - d_i),$$

where $u \in \mathcal{U}$ is a feasible utility vector among N players, and d the disagreement point. This approach is referred to as *proportional fairness*. Some papers criticized the *Independence of irrelevant alternatives* for having undesirable side effects. To overcome those issues, [KS75] proposed to replace it with a *monotonicity* axiom, resulting in another unique utility vector, and a slightly different vision on fairness.

In opposition to bargaining games, cooperative games study games where forming coalitions is allowed. In this theory, it is assumed that players can achieve superior outcomes by cooperating. Players must establish their common interest and then work together to achieve it, which requires information exchanges. In [Sha52], Shapley studied a class of functions that evaluate players participation in a coalition. Considering a set of axioms (symmetry, efficiency and law of aggregation), Shapley showed that there exists a unique value function satisfying those axioms. He derived an explicit formula to compute the value of a player i in a cooperative game with a set N of players:

$$\phi_i(v) = \sum_{S \subset N \setminus \{i\}} \binom{|N|-1}{|S|}^{-1} \big(v(S \cup \{i\}) - v(S)\big),$$

where $v(S)$ gives the total expected sum of payoffs the cooperation S can obtain. The values obtained $\{\phi_i(v)\}_{i \in N}$ are called Shapley values. They are considered as a fair redistribution of gains in the group. However, they are very hard to compute in practice (as the size of the problem grows, those values are not computable).

6.2.2 Evaluating the fairness of outcomes

As we study fairness, we naturally look for ways to measure it. In [Lan+10], the authors proposed a mathematical framework based on five axioms (continuity, homogeneity, saturation, partition and starvation) to define and evaluate fairness measures of utility vectors. They established a class of functions satisfying those axioms, which comprises various known measures on fairness, such as Atkinson's index, $\alpha-$fairness, Jain's index etc. Removing the axiom of homogeneity, this class is extended to measures looking for a trade-off between fairness and efficiency. Although a variety of indices exist, the Gini coefficient, mentioned before, is the most commonly used. For example, in a recent paper [Hey+19] studying the fairness in power system reliability, the authors compared a Gini-based index to a variance-based index (similar to the standard deviation index).

When fairness is considered in the problem (through the objective or constraints), it comes at a price: a fair solution might not be the most efficient one. Indeed, many articles try to find a balance between efficiency (have the best objective possible) and fairness (have a fair solution). In [BFT11], the authors established bounds on the price of fairness for two approaches (proportional fairness and minimax fairness) in resource allocation problems among self-interested players.

In this section, we referred to work that lay the foundations of fairness modeling in mathematics. For a more complete review, we refer to [XH23] where the authors provided guidelines for readers to choose the appropriate definition and modeling of fairness. They went through the list of indicators and criteria that exist to measure and define fairness. However, they assumed that fairness can always be reflected through the social welfare function (which corresponds to utilities in game theory) of agents. This means that the well-being of different agents are comparable through a single value.

Chapter 7

Fairness by design in shared-energy allocation problems

Contents

In this chapter, we derive tools to model fairness in an aggregation problem, leveraging the discussion on the concept of fairness in Chapter 6. The content of this chapter led to preprint [FLP24] and is currently under review in Computational Management Science.

7.1 Introduction

Many domains, such as telecommunication networks, healthcare, disaster management, and energy-sharing systems, require fairness as a key criterion. However, fairness is not easy to define or implement, as it can have different meanings and implications in different contexts.

Nevertheless, mathematical models that address real-world problems should not ignore fairness, even if it adds complexity to the problem. In this paper, we investigate various methods to incorporate fairness in a multi-agent problem. Specifically, we apply fairness to the problem of aggregating prosumers, who are both electricity producers and consumers, in the electricity market.

We focus on electric energy management application, where the aggregation of prosumers is becoming more relevant due to the increasing number of prosumers. Renewable energy generation capacities are becoming more affordable and effective, as renewable energy investments are rising (19% in 2022, according to a report by [Intd] on global trends in renewable energy). This enables smaller prosumers, such as medium-sized industries, to invest in onsite energy generation and storage. However, prosumers are usually too small to access the electricity market directly, so some companies offer to aggregate them in electricity markets. For example, CPower is an American company that aggregates a total of 2.000 MW of power [CPO]. We refer to [CJA17] for an extensive review on aggregators and their role in electricity markets.

Those aggregators can be external entities responsible for every prosumer energy transfers. In this case, there is a necessity to think of how the aggregation affects the participants to ensure a fair allocation of benefits. This is highlighted in a report [EUR15] on designing fair and equitable market rules for demand response aggregation, published by the association representing the common interests of the European electricity industry, Euralectric. Indeed, there is a practical need to guarantee that each prosumer benefits from staying in the aggregation. Further, prosumers need to feel like they are not being disfavored compared to others, leading the aggregator to choose a solution with a fair allocation of benefits.

In the literature, one distinguishes two main approaches in handling fairness: solve the problem efficiently and then reallocate the benefits [YHS21; Wan+19; Yan+23]; or change the objective function in order to get a fair solution [XH23]. In the first approach, we model a multi-agent problem with a utilitarian objective, i.e., we optimize the aggregated objectives of agents. Then, a protocol is implemented to reallocate the benefits among agents. For example, *Shapley values* [Sha52] assess the marginal contribution of each agent in the group and determine their fair share. The second approach prioritizes fair solutions through the modeling by changing the objective function. We refer to [XH23] for a comprehensive overview and guidelines on selecting an appropriate objective function to reflect fairness. The two most studied objective functions are *the minimax objective* [Raw71], which optimizes the least well-off agent's objective, and *the proportional objective* [Nas50], derived from Nash's *bargaining solution*, which optimizes the logarithmic sum of agents' objectives. Note that this approach, through the objective function, means that the well-being of different agents are compared through a single value.

However, these approaches present some limitations. On the one side, the proportional and minimax approaches focus merely on the objective function and not decisions. This can be a problem, as in some applications there can be different characteristics which are valuable. For example in an energy contract, both the flexibility and the volume of energy traded are important features. Thus, it is hard to take into account both of them when the quality of a solution is determined by a single value. On the other hand, post-allocation distributions of benefits are not adapted to problems that are formulated over long periods of time, such as contracts in electricity markets. Indeed, those approaches require to solve the whole problem before allocating costs. Then, it is impractical in most cases to expect each agent to wait until the problem's completion, which could span several months or years, to receive their fair share. Furthermore, given the inherent uncertainties linked to most problems, we also want our approach to hold in a stochastic framework. Then, fairness criteria must be redefined considering utility distributions

and associated risks over time.

In this paper, we introduce various strategies for integrating fairness considerations into optimization problems. Our primary focus is what we refer to as *fairness-by-design*. Instead of relying on *ex post* redistribution, as is usual in game theory [Sha52], we can establish a degree of fairness directly within the model. Our main contribution is to provide a framework and tools to accommodate fairness into mathematical models, in particular in the context of prosumer aggregation. What sets our approach apart is the extension of this framework to dynamic and stochastic settings, allowing for risk-averse and time-consistent guarantees.

More specifically, we present two key elements for achieving fair allocation in an aggregation. First, we model fair cost allocation through an operator ordering the costs of the different prosumers. For the choice of this operator, we present three traditional approaches (*utilitarian, proportional* and *minimax*). Additionally, we propose *acceptability constraints* that ensure each agent's outcome improves in a predefined sense within the aggregation. In their simplest form, these acceptability constraints correspond to individual (or self) rationality in game theory, ensuring each agent benefits from being in the group. We then extend the problem to a dynamic framework where decisions are made sequentially over time. In this context, agent's cost are multidimensional, the acceptability (or individual rationality) constraints thus need to choose a (partial) order. We discuss a few relevant partial order choices. Similarly, in a stochastic framework, agents cost are random variables, and we discuss relevant stochastic orders. Compared to [GKW23], who propose a risk-averse stochastic bargaining game, our approach handles uncertainties through the objective function but also acceptability constraints. This enables us to consider various aspects of the impact of uncertainties on the problem. As a result, our proposed model is well-suited for addressing inherent uncertainties within multistage stochastic programs, enhancing its practical applicability. Finally, we assess these different strategies on a toy model where we aggregate 4 electricity consumers to access the day-ahead market. We discuss each modeling choice consequences.

The remainder of the paper is organized as follows. In Section 7.2, we delve into definitions of fairness and its integration into optimization models. We propose, in Section 7.3, to model prosumers aggregation with acceptability constraints and a fair objective function. We then illustrate the introduced framework on a toy model in Section 7.4. Section 7.5 expands the notion of acceptability into the dynamic framework, while Section 7.6 adapts acceptability and fairness to the stochastic framework.

Notations

To facilitate understanding, we go through some notations used in this paper. We denote $[N] := \{1, \dots, N\}$ the set of non-null integers smaller than N. Accordingly, $X_{[n]}$ denote the collection $\{X_i\}_{i\in[n]}$. Random variables are denoted in bold characters, with their realization is normal font. The $\sigma-$algebra generated by $\{\xi_\tau\}_{\tau\in[t]}$ is denoted $\sigma(\xi_{[t]})$. Finally, in this paper, the term operator always refers to a mathematical operator.

7.2 Fairness in the literature

In the Oxford Dictionary, fairness is defined as *the quality of treating people equally or in a way that is reasonable*. The definition is simple but subjective. Is treating people equally, regardless of any token of individuality, considered fair in society? Furthermore, what does it mean to be reasonable? Whatever take we have on fairness is necessarily subjective and context-dependent (see [Kon03] for a philosophical analysis of fairness). In this section, we give a general overview of

how fairness is defined and modeled across the scientific literature, while linking it to our energy application. Bear in mind that each approach on fairness adopts a specific definition of fairness, which is not consensual.

7.2.1 Modeling and accommodating fairness

One of the main challenges facing fairness is the allocation of resources between agents. This naturally falls into the scope of game theory, where each of the N individuals (or players) is modeled with a utility function whose actual value depends on the actions of all players. For a given set of actions, we obtain a utility vector, denoted $u := (u_1, \dots, u_N)$, representing the utility of every agent u_i. A utility vector is then said to be *fair* if it satisfies a set of properties that might vary from one specific fairness definition to another. Among them, *individual rationality*, which ensures that every individual is better off in the aggregation, and therefore accepts to be a part of it, is often required.

In a seminal contribution [Nas50], John Nash introduced the bargaining problem where two rational agents, allowed to bargain, try to maximize the sum of their utilities. If agents are rational, individual rationality must be ensured. This is modeled using a *disagreement point* which represents the outcome obtained by players if they cannot reach an agreement. For agents to cooperate, they must agree on properties a utility vector, $u_{[N]}$, should satisfy to be admissible. Nash proposed four axioms to constitute this agreement: *Pareto optimality* – we cannot improve the utility of one agent without decreasing another's utility; *Symmetry* – applying the same permutation to two utility vectors does not change their order; *Independence of irrelevant alternatives* – if a utility vector is the optimal utility vector within the feasible set, it remains so if the set is reduced; *Scale invariance* – applying affine transformations to the utility vector does not change the social ranking. Nash showed that, under some assumptions including convexity and compactness of feasible utility vectors, there exists a unique utility vector satisfying those axioms. This unique utility vector is regarded in the literature as a viable option when seeking fairness. It has been demonstrated that under convexity of the feasible set, it can be obtained by maximizing the product of utilities ([Nas53; Mut99]), and thus by maximizing a logarithmic sum of utilities:

$$\max_{u \in \mathcal{U}} \quad \sum_{i=1}^{N} \log(u_i - d_i),$$

where $u \in \mathcal{U}$ is a feasible utility vectors among N players, and d the disagreement point. This approach is referred to as *proportional fairness*. Some papers criticized the *Independence of irrelevant alternatives* for having undesirable side effects. To overcome those issues, [KS75] proposed to replace it with a *monotonicity* axiom, resulting in another unique utility vector and a slightly different vision on fairness.

In contrast to bargaining games, *cooperative games* study games in which coalition formation is allowed: see [OR94] for a complete introduction. In this theory, it is assumed that players can achieve superior outcomes by cooperating. Players must establish their common interest and then work together to achieve it, which requires information exchanges. In *transferable utility games*, payoffs are given to the group which then divides among players through a post-allocation scheme. In [Sha52], Shapley studied a class of functions that evaluate the participation of players in a coalition. Considering a set of axioms (symmetry, efficiency and law of aggregation), Shapley showed that there exists a unique value function satisfying those axioms. He derived an explicit

formula to compute the value of a player i in a cooperative game with a set N of players:

$$\phi_i(v) = \sum_{S \subset N \setminus \{i\}} \binom{|N|-1}{|S|}^{-1} \big(v(S \cup \{i\}) - v(S)\big),$$

where $v(S)$ gives the total expected sum of payoffs the coalition S can obtain. The values obtained $\{\phi_i(v)\}_{i \in N}$ are called Shapley values. They are considered a fair redistribution of gains in the group. However, they are very hard to compute in practice (as the size of the problem grows, those values are not computable).

A different approach was introduced by John Rawl in [Raw71]: assuming that a group of individuals has no idea of their rank or situation in society, they will agree on a social contract aiming at maximizing the well-being of the least well-off. If the agents possess distinct characteristics, it might be difficult to compare them and ensure equitable treatment among them. This approach to fairness is often referred to as *minimax fairness*, as this amounts to optimizing the worst objective among agents.

When fairness is considered in the problem (through the objective or constraints), it comes at a price: a fair solution might not be the most efficient one. Indeed, many articles try to find a balance, or trade-off, between efficiency (have the best objective possible) and fairness (have a fair solution). In [BFT11], the authors established bounds on the price of fairness for two approaches, previously introduced– proportional fairness and minimax fairness– in resource allocation problems among self-interested players.

In this section, we referred to work that lay the foundations of fairness modeling in mathematics. In the following section, we present some applications of aggregations and the way fairness is considered or evaluated.

7.2.2 Applications of fairness in the literature

In this paper, we focus on a *by-design* approach, meaning that fairness is already accommodated in an optimization model. Although fairness is commonly recognized as crucial, in most articles the approach adopted derives from act utilitarianism: one should at every moment promote the greatest aggregate happiness, which consists in maximizing social welfare regardless of individual costs. For example, in [Xia+20], the authors studied an aggregator in charge of multiple agents within a power system. They optimized the total revenue of the aggregation without considering the impact on each agent individually. In [MP19], a prosumers' aggregator can focus on different indicators (import/export costs, exchange with the system operator, peak-shaving services etc.) to optimize its trades with the energy market, and the trades between prosumers. The indicator to focus on must be agreed on by the prosumers. The authors gave a sensitivity analysis of the parameters of the problem to determine what would increase the social acceptability of such an aggregation system. However, the model is utilitarian as it does not consider the allocation of costs among agents.

Other papers have proposed to first optimize the problem and then handle fairness through benefit post-allocation schemes. One way to deal with post-allocation is to model the aggregation as a coalitional game. This is the case of [Fre+15], where the authors studied a risk-averse renewable-energy multi-portfolio problem. In order to get a fair and stable allocation of profits, they chose *the Nucleolus* approach which finds a vector utility that minimizes the incentive to leave the aggregation for the worst coalition. In particular, this solution is in *the core* of the game, meaning every players gains from staying in the grand coalition. Similarly, in [YHS21], the authors studied a group of buildings with solar generation that mutually invest in an ESS.

The approach is to, first, optimize the problem formulated as a two-stage stochastic coalition game. Then, a fair reallocation of costs is determined by computing the nucleolus allocation which minimizes the minimal dissatisfaction of agents.

Some papers propose different methods to elaborate post-allocation schemes. For example in [Yan+23], the authors studied the joint participation of wind farms with a shared energy storage. The solution is found by first solving a two-stage stochastic program, and then reallocating the lease cost among users in a proportional scheme. They chose to make a wind farm pay depending on its increase of revenue after using the energy storage leasing service. In [Wan+19], the authors valued cooperation in their model, which is another way to look at cost redistribution. They considered an aggregator which participates in capacity and energy market for a number of energy users. In their model, the aggregator is not in charge of the users decisions but of the trades with the energy market, therefore he must incentivize users to deviate from their optimal scheduling for minimizing total revenue. They proposed to solve an asymmetric Nash bargaining problem to determine the incentizing costs. In another approach, the authors solved a multi-portfolio problem with fairness considerations in [IT14]. Instead of splitting the market impact costs in a pro-rata fashion, they introduced charging variables, constrained to satisfy some properties, that are optimized in the model. This approach amounts to having transfer variables, which we avoid in this paper, as they may raise privacy and trust concerns in practical application. Instead, we simplify the approach by designating the aggregator as the sole entity with complete information on the problem, which pays directly agents depending on their actions.

Typically, fairness is dealt with through the objective function, or in a post-allocation scheme. However, some researchers proposed constraints to ensure fairness. For example in [AKY22], the authors constrained the allocation feasibility set for a resource allocation problem. They introduced a welfare function dominance constraint: the admissible set of social welfare functions must dominate a referenced one. Then, with a utilitarian objective, a trade-off between fairness and efficiency is obtained. An alternative approach, proposed in [Oh22], is to bound a fairness indicator. The authors studied the energy planning of multiple agents over a virtual energy storage system (VESS), where energy dispatch is managed by an aggregator. They introduced two fairness indicators depending on the energy allocation, and added constraints bounding them in a utilitarian model. Then, they compared the results with a minimax approach, where they optimize the minimal fairness indicator over agents.

In many cases, uncertainties are inherent to the problem. If multiple articles have dealt with uncertainties, they rarely have a stochastic take on fairness. For example, in both [Yan+23] and [YHS21], the authors solved their problem with a two-stage program and then redistributed the costs fairly after uncertainty realization. Thus, there is no stochastic policy for fair redistribution. Other articles accommodated risk-averse profiles to game theory approaches. In [GKW23], the authors studied a risk-averse extension of the Bargaining Problem. They adapted Nash bargaining axioms to constrain the feasible utility vectors depending on the risk profile of players.

7.3 A shared-resource allocation problem in the context of a prosumer aggregator

We present here a general framework where a so-called *aggregator* aggregates independent agents' needs (industrial prosumers, residential units, virtual power plants...) and makes economic transactions for the collective. To make aggregation contracts attractive to agents, we encounter two distinct challenges: first, each agent needs to find the contract *acceptable*, ensuring that each agent derives substantial benefits from the aggregation; second, the decisions made by

the aggregator, leading to benefits or losses for each agent, should be made fairly. Recall that, for practical reasons, we do not allow money transfers between agents. Finally, to align with standard optimization frameworks, we aim to minimize the costs of agents and thus consider them as buyers.

In the following, Section 7.3.1 formalize the setting, Section 7.3.2 explore various objective functions that model fair decisions, and finally Section 7.3.3 introduce acceptability constraints.

7.3.1 Prosumers and market structure

We denote by $x^i \in \mathcal{X}^i$ the set of state and decision variables modeling an agent i. The technical constraintsproper to agent i are represented through feasible set $\mathcal{X}^i$, while external constraints (for instance market exchanges), common to all agents, are represented with feasible set $\mathcal{M}$. Finally, each prosumer wants to minimize a cost function $L^i : \mathcal{X}^i \to \mathbb{R}$, yielding the model (P^i). Note that (P^i) can model problems in various contexts. In Section 7.4, we present the particular application of this framework to prosumers aggregation on energy markets.

We now consider an aggregator in charge of I agents, we denote $x := (x^i)_{i\in[I]}$. The aggregator in problem (A), aggregates agents' decisions into $h(x^1, \dots, x^I)$ to satisfy external constraints $\mathcal{M}$ (see (7.1c)). Further, the physical constraint of each agent must be conserved (see (7.1b)), while the external constraints bind all agents' decisions. Finally, on the one hand, constraint (7.1d) ensures that the cost of an agent i is within an acceptable set $\mathcal{A}^i_\alpha$ they have agreed on prior to optimization. On the other hand, $\mathcal{F}_I$ is the agent operator that computes the objective of the aggregator considering the I objective functions of all agents. Depending of the choices of the acceptability sets $\mathcal{A}^i_\alpha$ and the agent operator $\mathcal{F}_I$, discussed, respectively, in Section 7.3.2 and Section 7.3.3, We have obtained different approaches to the shared resource allocation problem.

$$
\begin{aligned}
(P^i) \qquad & \min_{x^i} \; L^i(x^i) && (A) \qquad \min_{x} \; \mathcal{F}_I((L^i(x^i))_{i\in[I]}) && && (7.1a)\\
& \text{s.t.} \; x^i \in \mathcal{X}^i && \text{s.t.} \; x^i \in \mathcal{X}^i && \forall i \in [I] && (7.1b)\\
& \quad x^i \in \mathcal{M}. && \quad h(x^1, \dots, x^I) \in \mathcal{M} && && (7.1c)\\
& && \quad L^i(x^i) \in \mathcal{A}^i_\alpha && \forall i \in [I]. && (7.1d)
\end{aligned}
$$

We assume that the aggregation can decide that it is optimal for agents to operate independently *i.e.,* if x^i is an optimal solution of (P^i), then $(x^1, \dots, x^I)$ is an admissible solution of (A).

7.3.2 Fair cost aggregation

Assuming that all agents have agreed to participate in the aggregation (we discuss acceptability in Section 7.3.3), we focus on the way the aggregator operates to allocate aggregation benefits among prosumers.

The most natural and efficient method is the so-called *utilitarian approach*:

$$
\mathcal{F}^U_I((L^i(x^i))_{i\in[I]}) = \sum_{i\in[I]} L^i(x^i). \tag{7.2a}
$$

This approach aims to minimize total costs independently from the distribution of costs among prosumers: fairness is set aside. Indeed, in case of heterogeneity of the objective functions, it is possible that one of the objective function L^i dominates the others, *i.e.,*

$$
L^i(x^i) \geq L^k(x^k), \qquad \forall x^i \in \mathcal{X}^i, \quad \forall x^k \in \mathcal{X}^k,
$$

in which case all efforts of the aggregation are focused on minimizing the dominant objective function. A possibility that falls out of the scope of this paper (see Section 7.2.1) is to solve (A) and then reallocate resources with a fair scheme or put money transfers in place. We study alternative agent operators that ensure fair allocation for various fairness definitions.

First, we consider the *proportional approach* based on Nash bargaining solutions (see Section 7.2.1). For this approach, we consider the set of reachable (dis)utilities $\mathcal{L} = \{(L^1(x^1), \dots, L^I(x^I) \mid x^i \in \mathcal{X}^i, \forall i \in [I],\ M^i x^i \in \mathcal{M}\}$, and set the optimal values of (P^i), v^i [1], as the chosen disagreement point. Then, Nash [Nas50] introduces a set of axioms that must respect a fair distribution of (dis)utilities, and show that, if $\mathcal{L}$ is convex and compact, there exists a unique (dis)utility vector satisfying those axioms. Furthermore, it is proven that Nash's distribution is obtained by maximizing the sum of logarithmic utilities. For our problem, it corresponds to using the agent operator :

$$\mathcal{F}_I^P((L^i(x^i))_{i\in[I]}) := -\sum_{i\in[I]} \log(v^i - L^i(x^i)). \tag{7.2b}$$

Note that this approach tends to act in favor of smaller participants. Indeed, increasing a small cost improvement is preferred to increasing an already large cost improvement.

Finally, Rawls' theory of justice leads to the *minimax approach* favoring the least well-off. Here, the operator we obtain is:

$$\mathcal{F}_I^{MM}((L^i(x^i))_{i\in[I]}) := \max_{i\in[I]} L^i(x^i). \tag{7.2c}$$

For similar reasons to the utilitarian approach, this method may not be adequate for heterogeneous agents as it only focuses on minimizing the dominant objective function. To address this issue, we quantify an agent's well-being by looking at the proportional savings he makes in the aggregation. Then, applying Rawls' principle, we minimize the maximum proportional costs over agents, and we obtain the following agent operator:

$$\mathcal{F}_I^{PMM}((L^i(x^i))_{i\in[I]}) := \max_{i\in[I]} \frac{L^i(x^i)}{v^i}, \tag{7.2d}$$

which we refer to as the *Scaled Minimax approach*. Note that in both the scaled minimax approach ($\mathcal{F}_I^{PMM}$) and the proportional approach ($\mathcal{F}_I^P$), there are multiple solutions with different aggregated costs. We assume here we have defined a way to select a solution among them.

7.3.3 Acceptability constraints

Having delineated several methodologies for equitable cost distribution, we must convince agents to be part of the aggregation. We consider that agents are individually rational, that is a contract cannot be deemed acceptable if at least one agent is not better off independently *i.e.,* $v^i \leq L^i(x^i)$, where $L^i(x^i)$ is the cost of i in the aggregation. We can go one step further and require that, to find the contract acceptable, they benefit from it, *i.e.,* $v^i > L^i(x^i)$. We thus define the acceptability set $\mathcal{A}^i_\alpha$ appearing in (7.1d) as follows:

$$\mathcal{A}^i_\alpha := \{\ u^i \mid u^i \leq \alpha\ v^i\ \}, \tag{7.3}$$

[1] We implicitly assume here that either there is a unique solution, or that we have defined a way to select a solution among the set of optimal solutions.

where $\alpha \in (0, 1]$ is given. Then, we say a solution is $\alpha-$acceptable if it is contained in $\mathcal{A}^i_\alpha$.

Acceptability sets are independent from one agent to another. We then define *global acceptability* as the cartesian product of all acceptability sets $\mathcal{A}_\alpha := \mathcal{A}^{i_1}_\alpha \times \cdots \times \mathcal{A}^{i_I}_\alpha$.

Enforcing acceptability constraints to (A) restricts the feasible solution set, potentially leading to higher aggregated cost. We define the *price of acceptability* as

$$PoA := v^\star_{\mathcal{A}_\alpha} - v^\star_\emptyset, \tag{7.4}$$

where $v^\star_{\mathcal{A}_\alpha}$ is the optimal value of (A) with acceptability constraints $\mathcal{A}_\alpha$, and $v^\star$ is the optimal value of (A) without them.

Remark 8. *In the scaled minimax model with agent operator $\mathcal{F}^{PMM}_I$, the optimal solution is $1-$acceptable. Indeed, if the agents don't take advantage of the aggregation, then $L^i(x^i) = v^i$ and we get a feasible solution with respect to $\mathcal{A}_\alpha$ of optimal value 1. Further, if we consider the problem:*

$$\begin{aligned}
\underset{\alpha}{\text{Min}} \quad & \alpha && && \text{(7.5a)}\\
s.t. \quad & x^i \in \mathcal{X}^i && \forall i \in [I] && \text{(7.5b)}\\
& h(x^1, \ldots, x^I) \in \mathcal{M} && && \text{(7.5c)}\\
& L^i(x^i) \in \mathcal{A}^i_\alpha && \forall i \in [I], && \text{(7.5d)}
\end{aligned}$$

it is equivalent to problem (A) with agent operator $\mathcal{F}^{PMM}_I$ with no acceptability constraints:

$$\begin{aligned}
\underset{x}{\text{Min}} \quad & \underset{i\in[I]}{\text{Max}} \quad \frac{L^i(x^i)}{v^i} && && \text{(7.6a)}\\
s.t. \quad & x^i \in \mathcal{X}^i && \forall i \in [I] && \text{(7.6b)}\\
& h(x^1, \ldots, x^I) \in \mathcal{M}. && && \text{(7.6c)}
\end{aligned}$$

Then, when we linearize the maximum in (7.6a), we fall back into Problem (7.5), by definition of $\mathcal{A}^i_\alpha$.

Remark 9. *Note that our problem with the proportional operator $\mathcal{F}^P_I$ necessarily yields a solution $(1-\epsilon)-$acceptable, with $\epsilon > 0$. Indeed, if for agent i, $L^i(x^i) \geq v^i$, then $\log(v^i - L^i(x^i))$ is undefined.*

For simplicity, in the rest of the paper, we assume $\alpha = 1$. We later discuss how to extend the acceptability constraint to a dynamic (see Section 7.5.2) and stochastic framework (see Section 7.6.3). Finally, combining different objective functions with acceptability constraints, we observe on a small illustration their impact on the solution in the following section.

7.4 Application to consumer aggregation on the day-ahead and balancing market

In this section, we adapt and illustrate the framework presented in Section 7.3 to the problem of prosumers aggregation on electricity markets. More specifically, the prosumers have access to: the *day-ahead market*, where every day at 2 pm, prices and electrical energies are set for all across Europe for the twenty-four hours of the next day; and the *balancing market* on which

t	**1**	**2**	**3**	**4**	**5**
p_t^{DA}	2	16	1	10	1
p_t^B	6	25	5	15	5
$\underline{q}_t^{DA}$	11	11	11	11	11

	A_1	A_2	A_3	A_4
$\underline{q}_i$	0	5	0	2
$\overline{q}_i$	5	5	4	3
Q_i	10	25	8	15

Table 7.1: Parameters values

prosumers must buy or sell electricity at real-time prices to ensure power system balance. A minimum trade of 0.1 MWh of energy is required to participate in the day-ahead market.

We propose a toy model to illustrate the implications of each model proposed in Section 7.3. Therefore, we consider a problem with four consumers ($I = 4$) on five stages ($T = 5$). At each stage t, we must decide how much energy $q_{t,i}^{DA}$ (*resp.* $q_{t,i}^{B}$) to purchase from the day-ahead (*resp.* balancing) market for consumer i. Thus in (P^i), we have $x^i := (q_{t,i}^{DA}, q_{t,i}^{B})$. Each consumer has bounds $[\underline{q}_i; \overline{q}_i]$ on its electricity consumption, and a total consumption Q_i to meet at the end of the horizon, amounting to feasible set $\mathcal{X}^i$. Note that the upper bounds on electricity consumption simplify physical constraints that would ensure a finite volume of traded electricity. We introduce binary variables b_t^{DA}, representing the decision to buy a day-ahead, to model the minimum volume requirement for the day-ahead market, which composes the external constraints $\mathcal{M}$. The objective for consumer i is to minimize its electricity costs:

$$L^i(x^i) = \sum_{t=1}^{T} [\, p_t^{DA}\ q_{t,i}^{DA} + p_t^B\ q_{t,i}^B \,], \tag{7.7a}$$

where p_t^{DA} (*resp.* p_t^B) is the price of electricity at t on the day-ahead (*resp.* balancing) market. We obtain the simple prosumer (P^i) and the aggregated model (A):

$$(P^i)\ \min_{x^i} \quad L^i(x^i) \qquad\qquad (A)\ \min_{x} \mathcal{F}_I\Big((L^i(x^i))_{i\in[I]}\Big) \tag{7.7b}$$

$$\text{s.t. } \underline{q}_i \le q_{t,i}^{DA} + q_{t,i}^B \le \overline{q}_i \quad \forall t \qquad \text{s.t. } \underline{q}_i \le q_{t,i}^{DA} + q_{t,i}^B \le \overline{q}_i \quad \forall t, \forall i \tag{7.7c}$$

$$\sum_{t=1}^{T} (q_{t,i}^{DA} + q_{t,i}^B) \ge Q_i \qquad \sum_{t=1}^{T} (q_{t,i}^{DA} + q_{t,i}^B) \ge Q_i \quad \forall i \tag{7.7d}$$

$$\underline{q}_t^{DA} b_t^{DA} \le q_{t,i}^{DA} \le M\, b_t^{DA} \ \forall t \qquad \underline{q}_t^{DA} b_t^{DA} \le \sum_{i\in[I]} q_{t,i}^{DA} \le M b_t^{DA} \quad \forall t \tag{7.7e}$$

$$b_t^{DA} \in \{0,1\} \quad \forall t, \qquad b_t^{DA} \in \{0,1\} \quad \forall t, \tag{7.7f}$$

where $\mathcal{F}$ is the chosen agent operator for the aggregation. We solve this small problem with the utilitarian operator $\mathcal{F}_I^U$, with the scaled minimax operator $\mathcal{F}_I^{PMM}$ and with the proportional operator $\mathcal{F}_I^P$. For all agent operator , we solve the problem with and without acceptability constraints, with $\alpha = 1$. In the proportional and scaled minimax approaches, we do not optimize the aggregated costs. As the optimal solution is not necessarily unique, there can be different optimal solutions with different aggregated costs. Among those, we choose one with minimum aggregated costs.

We refer to the model with agent operator $f \in \{U, SM, P\}$ (for Utilitarian, Scaled Minimax and Proportional) and acceptability constraints set $a \in \{\emptyset, \alpha\}$ (for no acceptability constraints,

or acceptability constraints given by $\mathcal{A}_\alpha$) as m_a^f, and $\mathcal{A} = \emptyset$ corresponds to a model without acceptability constraints. Finally, we compute here Shapley's values (see Chapter C for more details), which are commonly recognized as a fair solution, to compare them to the solutions we obtain with our models.

We show on a small artificial illustration how all these models can lead to different solutions. Each model can be evaluated through two metrics: first, the efficiency of the model *i.e.,* the overall costs of the aggregation; second, the fairness of the model *i.e.,* how distributed are the costs over prosumers. For the prosumers' parameters and market prices we use the data on Table 7.1. We observe the allocation of costs over consumers on Figure 7.1, the resulting percentage of savings made by each consumer on Table 7.2, and the detail of day-ahead and balancing purchases on Table 7.3.

First, it's worth noting that none of the consumers can individually access the day-ahead market as for any prosumer $\overline{q}_i \leq \underline{q}_t^{DA}$ and thus constraint (7.7e) excludes any purchase on the day-ahead market. In the utilitarian model $m_\emptyset^U$,the primary focus is to minimize aggregated costs, making it optimal to always consistently access the day-ahead market as a group. To achieve this, consumer A_1 redistributes its energy load across 4 time steps, incurring a higher individual cost (64% higher) than when acting independently. By adding acceptability constraints to the model (m_α^U), the model loses in efficiency but now satisfies individual rationality: $PoA = 3\%$ of $m_\emptyset^U$'s costs. We observe that the aggregated costs of consumers slightly increases, but now the charge of energy needed to access the day-ahead market is shared between A_1 and A_3, although A_3 does not gain anything from the the aggregation (0% of savings).

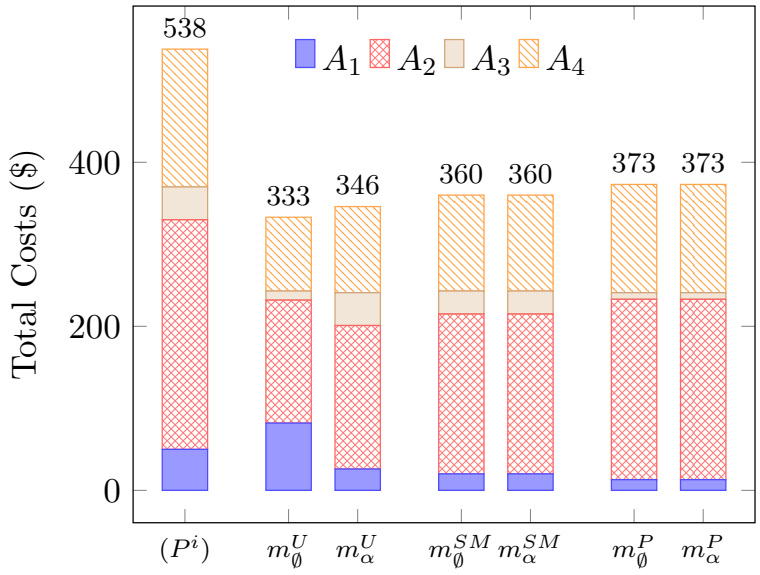


Figure 7.1: We observe the result of the static Problem (7.7f) with parameters given in Table 7.1. The bars correspond to the outcome of different models, the number above being the total cost. The first bar is the non-aggregated model: we solve each (P^i) independently. Then, there are three groups of two bars, each group corresponding to a choice of agent operator $(\mathcal{F}_I^U, \mathcal{F}_,^{PMM} \mathcal{F}_I^P)$. Then, for each objective function, we present the results of the model, first without and then with, acceptability constraints $\mathcal{A}_\alpha$, with $\alpha = 1$. Each bar is decomposed in 4 blocks corresponding to the cost incurred by each consumer i. At the top of each bar, we can read the sum of aggregated costs in the corresponding model.

Table 7.2: Percentage of savings $\frac{v^i - L^i(x^i)}{v^i}$ made by A_i in the model m_a^f depending on agent operator $f \in \{U, SM, P\}$ and acceptability set $a \in \{\emptyset, \alpha\}$ and PoA (in percentage) of the corresponding model.

	Utilitarian $\mathcal{F}_I^U$					Minimax $\mathcal{F}_I^{PMM}$					Proportional $\mathcal{F}_I^P$				
	A1	A2	A3	A4	PoA	A1	A2	A3	A4	PoA	A1	A2	A3	A4	PoA
$\emptyset$	-64	46	72	46	0	60	30	30	30	0	74	21	80	21	0
$\mathcal{A}_\alpha$	48	37	0	37	4	60	30	30	30	0	74	21	80	21	0
Shapley	114	20	111	28	0										

Conversely, the proportional solution (from model $m_\emptyset^P$) adopts a more bargaining-oriented approach, resulting in collaboration only during time slots ($t \in \{1, 3, 5\}$) with lower prices. Indeed, as A_1 and A_3 are not forced to consume energy at all times ($\underline{q}_1 = \underline{q}_3 = 0$), they can shift their consumption to time slots with lower prices. On the contrary, A_2 and A_4 must always consume energy ($\underline{q}_2 = 5, \underline{q}_4 = 2$), and the two of them together cannot access the day-ahead market either. Thus, in $m_\emptyset^P$, the solution is for A_1 and A_3 to consume only in time steps $\{1, 3, 5\}$, which leaves A_2 and A_4 to operate independently at $t = 2, t = 4$, resulting in limited savings (21%) compared to the scaled minimax approach ($m_\emptyset^{SM}$). As noticed in Remark 9, the solution is necessarily 1-acceptable. Therefore, the solution is the same in $m_\emptyset^P$ and m_α^P. Moreover, the proportional solution yields the worst aggregated costs *i.e.,* the less efficient solution.

	A1		A2		A3		A4	
t	DA	B	DA	B	DA	B	DA	B
1	***0***	0	5	0	3	0	3	0
2	3	0	5	0	***0***	0	3	0
3	2	0	5	0	1	0	3	0
4	3	0	5	0	***0***	0	3	0
5	2	0	5	0	4	0	3	0

(a) $m_{\emptyset}^{U}$

A1		A2		A3		A4	
DA	B	DA	B	DA	B	DA	B
0	0	5	0	3	0	3	0
1.01	0	5	0	1.93	0	3	0
3.93	0	5	0	***0***	0	3	0
0	0	0	***5***	***0***	0	0	***3***
5	0	5	0	3	0	3	0

(b) m_{α}^{U}

	A1		A2		A3		A4	
t	DA	B	DA	B	DA	B	DA	B
1	***0***	0	5	0	3	0	3	0
2	***0***	0	0	***5***	***0***	0	0	***3***
3	4.44	0	5	0	1.56	0	3	0
4	1.13	0	5	0	1.87	0	3	0
5	4.44	0	5	0	1.56	0	3	0

(c) $m_{\emptyset}^{SM}$

A1		A2		A3		A4	
DA	B	DA	B	DA	B	DA	B
0	0	5	0	3	0	3	0
0	0	0	***5***	***0***	0	0	***3***
4.44	0	5	0	1.56	0	3	0
1.13	0	5	0	1.87	0	3	0
4.44	0	5	0	1.56	0	3	0

(d) m_{α}^{SM}

	A1		A2		A3		A4	
t	DA	B	DA	B	DA	B	DA	B
1	3	0	5	0	***0***	0	3	0
2	***0***	0	0	***5***	***0***	0	0	***3***
3	3.5	0	5	0	4	0	3	0
4	***0***	0	0	***5***	***0***	0	0	***3***
5	3.5	0	5	0	4	0	3	0

(e) $m_{\emptyset}^{P}$

A1		A2		A3		A4	
DA	B	DA	B	DA	B	DA	B
3	0	5	0	***0***	0	3	0
0	0	0	***5***	***0***	0	0	***3***
3.5	0	5	0	4	0	3	0
0	0	0	***5***	***0***	0	0	***3***
3.5	0	5	0	4	0	3	0

(f) m_{α}^{P}

Table 7.3: Quantity of energy purchased on the day-ahead and balancing markets per stage for all consumers depending on different models: in bold italic, we highlight purchases on the (more expensive) balancing market and stages where no purchases are made on the day-ahead market.

With the scaled minimax approach, the model $m_{\emptyset}^{SM}$ yields a trade-off between efficiency and fairness compared to m_{α}^{U}: we observe that A_1 and A_3 decide to stop consuming at expensive time steps, thus achieving greater savings. The model also encourages more cooperation than the proportional model $m_{\emptyset}^{P}$, as we can observe on Table 7.3. As a result, in this model, all consumers achieve similar proportional savings, amounting to approximately 30% compared to operating independently, at the exception of A_1 that can save up to 60%. This means that any solution where A_1 shifts its consumption to other time slots to help others access the day-ahead market, would increase its costs too much, and A_3 would save less than 30%: this cannot be an optimal solution of $m_{\emptyset}^{SM}$. However, the aggregated cost of the aggregation is higher than with $m_{\emptyset}^{U}$ and m_{α}^{U}. Again, adding acceptability constraints does not change the solution, as the scaled minimax problem is innately 1-acceptable (see Remark 8).

Lastly, we observe on Table 7.2 the allocation of savings with a post-allocation rule based on Shapley's values. This approach leverages the efficiency of $m_{\emptyset}^{U}$ but then re-allocates costs to obtain a fair and acceptable solution. In this application, A_1 and A_3 save respectively 114% and 111% of their costs compared to operating independently, which amounts to them being paid by the aggregation to participate. Even though this allows A_2 and A_4 to gain from the aggregation, this questions the acceptability of this solution as some would earn money while others have residual costs. Furthermore, this method becomes impractical when dealing with large problems involving many prosumers and time steps due to the extensive computational

requirements. Additionally, adapting this method to dynamic and stochastic contexts is unclear, so we do not consider it further.

7.5 Fairness across time

In most use cases, we can assume that the aggregation of agents is thought to stay in place over long periods. One of the challenges of this long-term setting is incentivizing agents not to leave the aggregation, which requires adjusting the acceptability constraints of the static case.

7.5.1 Problem formulation

We consider a problem with T stages corresponding to consecutive times where decisions are made. At each stage $t \in [T]$, agent i makes a decision $x_t^i \in \mathcal{X}_t^i$, incurring a cost $L_t^i(x_t^i)$. Those stage costs are aggregated through a time operator $\mathcal{F}_T^i : \mathbb{R}^T \to \mathbb{R}$. Thus, the agent i's problem reads:

$$
\begin{aligned}
(P_T^i) \;:=\; \min_{x_t^i} \quad & \mathcal{F}_T^i\Big((L_t^i(x_t^i))_{t\in[T]}\Big) && && \text{(7.8a)}\\
\text{s.t} \quad & x_t^i \in \mathcal{X}_t^i && \forall t && \text{(7.8b)}\\
& x_t^i \in \mathcal{M}_t && \forall t. && \text{(7.8c)}
\end{aligned}
$$

A typical example of time-aggregator $\mathcal{F}_T^i$ is the discounted sum of stage costs *i.e.,* dropping the dependence in x_i for clarity's sake:

$$
\mathcal{F}_T^i((L_t)_{t\in[T]}) = \sum_{t\in[T]} r^t L_t,
$$

for $r \in (0, 1]$. Alternatively, $\mathcal{F}_T^i$ can be defined as the maximum of stage costs. This might happen for electricity markets where a prosumer aims at *peak shaving i.e.,* minimizing peak electricity demand. Further, time-aggregation operators may vary among agents, who may express different sensitivity to time *i.e.,* the discounted rate r varies among prosumers.

We now write the aggregation problem within this framework. Note that we can cast the current multistage setting into the setting of Section 7.3, by decomposing each agent into T independent stage-wise sub-agents: then we have $I \times T$ and we can use the methodology of Section 7.3. Thus we need to define an operator $\mathcal{F}_{I\times T}$ that takes $\{L_t^i\}_{t\in[T],i\in[I]}$ as input.

However, in most settings, it is reasonable to assume that an agent is time-homogeneous, meaning that, in some sense, for all $i \in [N]$, the agents $(i, t)_{t\in[T]}$ are the same, and aggregates the stage costs across time. Consequently, the global aggregation operator $\mathcal{F}_{I\times T}$ can be modeled as aggregating, over agents, their aggregated stage-costs, *i.e.,* $\mathcal{F}_{I\times T} = \mathcal{F}_I \odot \mathcal{F}_T$ where the $\odot$ notation stands for

$$
\mathcal{F}_I \odot \mathcal{F}_T\Big((L_i^t)_{i\in[I],t\in[T]}\Big) = \mathcal{F}_I\Big(\mathcal{F}_T^1\Big((L_t^1)_{t\in[T]}\Big), \ldots, \mathcal{F}_T^I\Big((L_t^I)_{t\in[T]}\Big)\Big). \tag{7.9}
$$

Finally, we obtain the following model for the aggregation of agents in a dynamic framework:

$$
\begin{aligned}
(A^T) \ := \quad & \underset{x_t^i}{\text{Min}} && \mathcal{F}_I \odot \mathcal{F}_T\Big((L_t^i)_{i\in[I],t\in[T]}\Big) && && (7.10a)\\
& \text{s.t.} && x_t^i \in \mathcal{X}_t^i && \forall t && (7.10b)\\
& && h(x_t^1,\dots,x_t^I) \in \mathcal{M}_t && \forall t && (7.10c)\\
& && (L_t^i)_{t\in[T]} \in \mathcal{A}^i, && && (7.10d)
\end{aligned}
$$

where we recall that we defined $\mathcal{F}_T$ as the sum, and suggest to choose $\mathcal{F}_I$ from the fairness operators $(\mathcal{F}_I^U, \mathcal{F}_I^P, \mathcal{F}_I^{PMM})$ introduced in Section 7.3.2. Thus, we obtain a fair objective function of the aggregated model (A^T). However, we have yet to adapt the notion of acceptability from Section 7.3.3 to this long-term framework, which is our next topic.

7.5.2 Dynamic acceptability

In long-term problems, for the aggregation to be acceptable, agents should not be tempted to leave the aggregation in between stages. Therefore, we extend our notion of acceptability constraint (7.3) to a dynamic framework. First, denote $v_t^i := L_t^i(x_t^{i,\star})$, the optimal independent cost of an agent i at stage t, where $x^{i,\star}$ is the optimal solution of Problem (7.8).

The acceptability constraint (7.3) consist in requiring, for each agent i, that its vector of costs $(L_t^i)_{t\in[T]}$ is less than $(v_t^i)_{t\in[T]}$. Unfortunately, there is no natural ordering of $\mathbb{R}^T$, and each (partial) order will define a different extension of the acceptability constraint (7.3). We present now some extensions of the acceptability constraint derived from standard partial orders.

Maybe the most intuitive choice is the component-wise order (induced by the positive orthant), *i.e.,* comparing coordinate by coordinate. This results in the *stage-wise acceptability* constraint $\mathcal{A}_s$, which enforces that each agent benefits from the aggregation at each stage:

$$
\mathcal{A}_s^i = \big\{\ (u_t^i)_{t\in[T]} \quad | \quad u_t^i \le v_t^i, \quad \forall t \in [T]\ \big\}. \tag{7.11a}
$$

As this approach might be too conservative for our model, *i.e.,* constrain the aggregation too much to take advantage of it, we consider two other orderings.

First, we can relax the stage-wise acceptability by considering that at each stage t, each agent benefits from the aggregation if we consider its costs aggregated up to time t. This result in *progressive acceptability* constraint $\mathcal{A}_p^i$:

$$
\mathcal{A}_p^i = \big\{\ (u_t^i)_{t\in[T]} \quad | \quad \sum_{\tau=1}^{t} u_\tau^i \le \sum_{\tau=1}^{t} v_\tau^i, \quad \forall t \in [T]\ \big\}. \tag{7.11b}
$$

Second, we ensure that each agent, aggregating its cost over the whole horizon, benefits from the aggregation (which amounts to the set in (7.3)) if we consider only the aggregated costs at the end of the horizon. We thus define the *average acceptability* constraint:

$$
\mathcal{A}_{av}^i = \big\{\ (u_t^i)_{t\in[T]} \quad | \quad \sum_{t=1}^{T} u_t^i \le \sum_{t=1}^{T} v_t^i\ \big\}. \tag{7.11c}
$$

Remark 10. *We have that $\mathcal{A}_s^i \subseteq \mathcal{A}_p^i \subseteq \mathcal{A}_{av}^i$. The acceptability constraint should be chosen as to strike a balance between aggregated cost efficiency (obtained with a less constrained acceptability set), and incentive to stay in the aggregation (obtained with a more constrained acceptability set).*

7.5.3 Numerical illustration

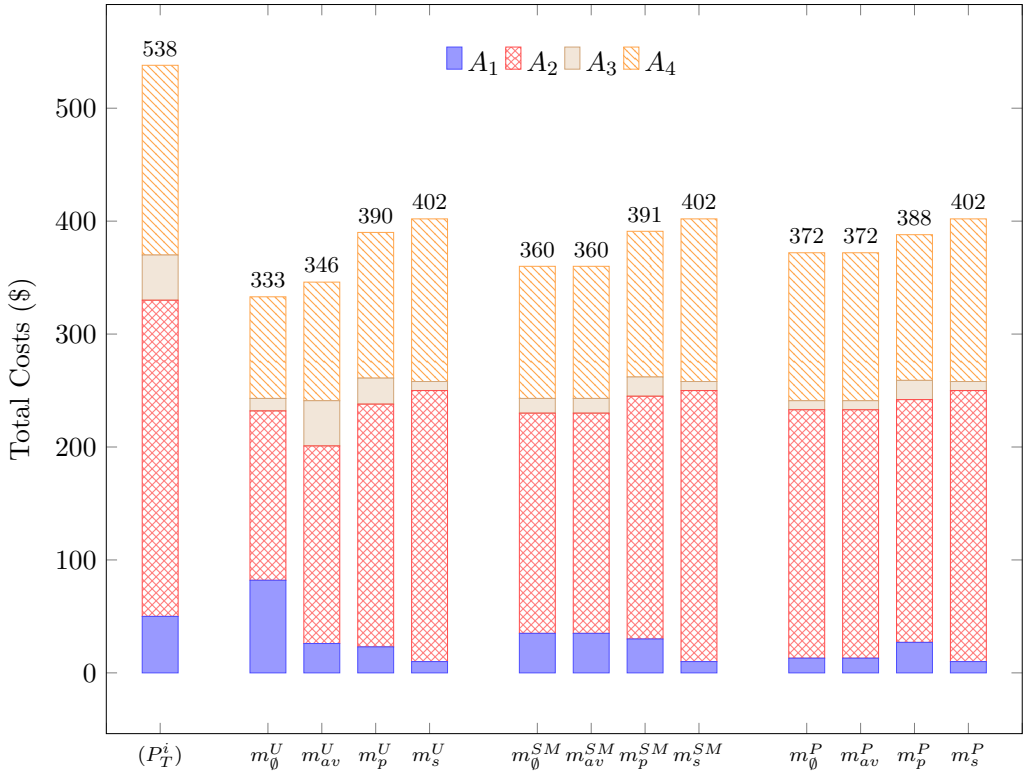


Figure 7.2: The bars correspond to the results of different models we solve. The first one is the independent model: we solve each (P_T^i) independently. Then, there are three groups of four bars, each group corresponds to a choice of agent operator $(\mathcal{F}^U, \mathcal{F}^{PMM}, \mathcal{F}^P)$. Then, given an operator, we have the model first without then with different acceptability constraints $(\emptyset, \mathcal{A}_{av}, \mathcal{A}_p, \mathcal{A}_s)$. Each bar is decomposed in 4 blocks corresponding to the share of each consumer i. At the top of each bar, we can read the sum of aggregated costs in the corresponding model.

We take the same example as in Section 7.4 and try out different combinations of operator $\mathcal{F}_{I\times T}$ $(\mathcal{F}^U, \mathcal{F}^{PMM}, \mathcal{F}^P)$ and acceptability set $\mathcal{A}$ $(\emptyset, \mathcal{A}_{av}, \mathcal{A}_p, \mathcal{A}_s)$. We denote m_a^f the model with agent operator $f \in \{U, SM, P\}$ and acceptability set $a \in \{\emptyset, av, p, s\}$. Figure 7.2 represent the distribution of prosumers' costs for these different cases, while Table 7.4 report their proportional savings. Finally, on Table 7.5, we report the day-ahead and balancing purchases of the different models with progressive and stagewise acceptability, while the results with no acceptability and average acceptability are in Section 7.4 on Table 7.3.

Table 7.4: Percentage of savings $\frac{v^i - L^i(x^i)}{v^i}$ achieved by A_i in the model m_a^f depending on agent operator f and acceptability set a and PoA (in percentage) of the corresponding model.

	Utilitarian $\mathcal{F}_I^U$					Minimax $\mathcal{F}_I^{PMM}$					Proportional $\mathcal{F}_I^P$				
	A1	A2	A3	A4	PoA	A1	A2	A3	A4	PoA	A1	A2	A3	A4	PoA
$\emptyset$	-64	46	73	46	0	30	30	67	30	0	74	21	80	21	0
$\mathcal{A}_{av}$	48	37	0	37	4	30	30	67	30	0	74	21	79	21	0
$\mathcal{A}_p$	55	23	44	23	17	44	23	57	23	8.6	45	23	56	23	4.3
$\mathcal{A}_s$	80	14	80	14	21	80	14	80	14	12	80	14	80	14	8.0

We observe on Figure 7.2 that increasing acceptability constraints (from none to average, progressive, and finally stage-wise) comes at a price but gives stronger guarantees to each prosumer. Indeed, we have seen that $m_\emptyset^U$ is the most efficient model but yields solutions in contradiction with individual rationality. We can correct this defect by enforcing average acceptability, but this is not enough to ensure everyone gains from the aggregation, as A_3 makes 0% of savings. With more constrained acceptability, $\mathcal{A}_p$ and $\mathcal{A}_s$, we enforce individual rationality over time or at all times. This leads to solutions where the savings among prosumers are shared in fairer proportions- at the loss of efficiency.

On the other hand, with an agent operator reflecting fairness (like scaled minimax or proportional), we obtain solutions that already aim at a fairer distribution of savings. Consequently, if we can observe solutions changing with increasing acceptability constraints, those changes are clearer with a utilitarian operator. Indeed, in the utilitarian model, A_2 achieves savings ranging from 14% to 46% of his independent cost. In contrast, under the scaled minimax approach, the savings range from 14% to 30%, and with the proportional approach, the savings fall between 14% and 21%.

Note that even though the acceptability constraints and the agent operator are two distinct tools, they both drive the model to fairer solutions for all agents in the aggregation.

	A1		A2		A3		A4	
t	DA	B	DA	B	DA	B	DA	B
1	***0***	0	0	***5***	***0***	0	0	***3***
2	***0***	0	0	***5***	***0***	0	0	***3***
3	5	0	5	0	4	0	3	0
4	1.4	0	5	0	1.6	0	3	0
5	3.6	0	5	0	2.4	0	3	0

(a) $m^U_{\mathcal{A}_p}$

	A1		A2		A3		A4	
	DA	B	DA	B	DA	B	DA	B
	0	0	0	***5***	***0***	0	0	***3***
	0	0	0	***5***	0	***0***	0	***3***
	5	0	5	0	4	0	3	0
	0	0	0	***5***	0	***0***	0	***3***
	5	0	5	0	4	0	3	0

(b) $m^U_{\mathcal{A}_s}$

	A1		A2		A3		A4	
t	DA	B	DA	B	DA	B	DA	B
1	***0***	0	0	***5***	***0***	0	0	***3***
2	***0***	0	0	***5***	***0***	0	0	***3***
3	5	0	5	0	4	0	3	0
4	2	0	5	0	1	0	3	0
5	3	0	5	0	3	0	3	0

(c) $m^{SM}_{\mathcal{A}_p}$

	A1		A2		A3		A4	
	DA	B	DA	B	DA	B	DA	B
	0	0	0	***5***	***0***	0	0	***3***
	0	0	0	***5***	0	***0***	0	***3***
	5	0	5	0	4	0	3	0
	0	0	0	***5***	0	***0***	0	***3***
	5	0	5	0	4	0	3	0

(d) $m^{SM}_{\mathcal{A}_s}$

	A1		A2		A3		A4	
t	DA	B	DA	B	DA	B	DA	B
1	***0***	0	0	***5***	***0***	0	0	***3***
2	***0***	0	0	***5***	***0***	0	0	***3***
3	3.8	0	5	0	3.5	0	3	0
4	1.9	0	5	0	1.1	0	3	0
5	4.3	0	5	0	3.5	0	3	0

(e) $m^{P}_{\mathcal{A}_p}$

	A1		A2		A3		A4	
	DA	B	DA	B	DA	B	DA	B
	0	0	0	***5***	***0***	0	0	***3***
	0	0	0	***5***	0	***0***	0	***3***
	5	0	5	0	4	0	3	0
	0	0	0	***5***	0	***0***	0	***3***
	5	0	5	0	4	0	3	0

(f) $m^{P}_{\mathcal{A}_s}$

Table 7.5: Quantity of energy purchased on the day-ahead and balancing markets per stage for all consumers depending on different models: in bold italic, we highlight purchases on the (more expensive) balancing market and stages where no purchases are made on the day-ahead market.

7.6 Accommodating fairness to uncertainties with stochastic optimization

Problems with energy generation, especially from renewable sources, and prices on energy markets are inherently uncertain as we have decisions to make over time, and the future is uncertain. Then, in addition to acceptability and fairness, we must tackle the challenge of handling uncertainties (while being fair about how we handle those). We want to address this issue by extending the problem presented in Section 7.3 to a stochastic framework. To that end, we introduce random variable $\boldsymbol{\xi}$, along with probability space $(\Omega, \mathcal{A}, \mathbb{P})$, which gathers all sources of uncertainties in the problem. We assume that Ω is finite, a common simplification in stochastic optimization to make problems more tractable. If Ω is not finite we rely on sample average approximation.

In the same way that we decomposed the problem in Section 7.5 with T time steps, we can decompose the problem here with Ω scenarios. Thus, there are similarities with the previous section. The main difference is that the set of time-step $\{1, \dots, T\}$ has a natural ordering, while the set of scenario Ω does not, which leads to discussing different partial orders on $\mathbb{R}^\Omega$ than on $\mathbb{R}^T$.

7.6.1 Static stochastic problem formulation

The problem at hand is naturally formulated as a multi-stage stochastic problem. For simplicity reasons, we first consider a 2−stage relaxation of the problem: in the first stage, *here-and-now* decisions must be made before knowing the noise's realization; in the second stage, once the noise's realization is revealed, *recourse* actions can be decided.

We first adapt the individual model (P^i) to a stochastic framework:

$$
\begin{aligned}
(P^{i,\rho}) \;:=\; & \min_{\boldsymbol{x}^i(\boldsymbol{\xi})} && \rho\left[L^i(\boldsymbol{x}^i(\boldsymbol{\xi}),\boldsymbol{\xi})\right] && && \text{(7.12a)}\\
& \text{s.t.} && \boldsymbol{x}^i(\boldsymbol{\xi}) \in \mathcal{X}^i && \text{a.s.} && \text{(7.12b)}\\
& && \boldsymbol{x}^i(\boldsymbol{\xi}) \in \mathcal{M} && \text{a.s.,} && \text{(7.12c)}
\end{aligned}
$$

where ρ is a (coherent) risk-measure *i.e.,* a function which gives a deterministic cost equivalent to a random cost, reflecting the risk of a decision for prosumer i, see *e.g.,* [Art+99]. The choice of ρ depends on the attitude of i towards risk. For example, the risk measure associated with a risk-neutral approach is the mathematical expectation $\mathbb{E}_\xi$. Alternatively, a highly risk-averse profile will opt for the worst-case measure $\sup_\xi$. Another widely used risk measure is the Average Value at Risk (a.k.a Conditional Value at Risk, or expected shortfall, see [RU+00]), or a convex combination of expectation and Average Value at Risk.

Now, we adapt the deterministic aggregation model (A). We face the same challenge as in Section 7.5. With multiple scenarios, we can consider that we have $I \times \Omega$ prosumers and we need to choose an operator $\mathcal{F}_{I\times\Omega} : \mathbb{R}^{I\times\Omega} \to \mathbb{R}$, leading to:

$$
\begin{aligned}
(A^\rho) \;:=\; & \operatorname*{Min}_{\boldsymbol{x}} && \mathcal{F}_{I\times\Omega}\Big((L^i(\boldsymbol{x}^i(\boldsymbol{\xi}),\boldsymbol{\xi}))_{i\in[I]}\Big) && && \text{(7.13a)}\\
& \text{s.t.} && \boldsymbol{x}^i(\boldsymbol{\xi}) \in \mathcal{X}^i && \forall i \in [I] \quad \text{a.s.} && \text{(7.13b)}\\
& && h(\boldsymbol{x}^1(\boldsymbol{\xi}),\dots,\boldsymbol{x}^I(\boldsymbol{\xi})) \in \mathcal{M} && \text{a.s.} && \text{(7.13c)}\\
& && L^i(\boldsymbol{x}^i(\boldsymbol{\xi}),\boldsymbol{\xi}) \in \mathcal{A}^i && \forall i \in [I] \quad \text{a.s..} && \text{(7.13d)}
\end{aligned}
$$

We assume the aggregator knows risk measures and prosumers objectives. As in Section 7.5.1, there are multiple possible choices for such operators. We assume that this operator $\mathcal{F}_{I\times\Omega}$ results from the composition of two operators: an uncertainty-operator $\mathcal{F}^i_\Omega$ dealing with the scenarios, which can differ from one prosumer to another; and an agent operator $\mathcal{F}_I$, as defined in Section 7.3.2. However, contrary to Section 7.5, it is not clear if we should aggregate first with respect to uncertainty (meaning that a prosumer manages its own risk) or with respect to prosumers (meaning that the risks are shared). We next discuss reasonable modeling choices of aggregation operators and acceptability constraints.

7.6.2 Stochastic objective

For the sake of conciseness, we are going to consider two possible uncertainty aggregators: a risk-neutral choice, where $\mathcal{F}^i_\Omega$ is the mathematical expectation $\mathbb{E}_\xi$, and a worst-case operator where $\mathcal{F}^i_\Omega$ is the supremum over the possible realization $\sup_\xi$. For the agent operator $\mathcal{F}_I$, which reflects the way to handle fairness, we consider either the utilitarian $\mathcal{F}^U_I$ or the scaled minimax $\mathcal{F}^{PMM}_I$ options (see Section 7.3.3 for definitions).

We suggest four different compositions of $\mathcal{F}^i_\Omega$ and $\mathcal{F}_I$ to construct the aggregation operator $\mathcal{F}_{I\times\Omega}$. Again, for simplicity of notations, we write $\boldsymbol{L^i}$ instead of $L^i(\boldsymbol{x}^i(\boldsymbol{\xi}),\boldsymbol{\xi})$.

First, we introduce the risk-neutral and utilitarian operator $\mathcal{F}^{US}_{I\times\Omega}$, which aims at minimizing the aggregated expected costs of prosumers:

$$\mathcal{F}^{US}_{I\times\Omega}\Big((\boldsymbol{L^i})_{i\in[I]}\Big) = \mathcal{F}^{U}_{I} \odot \mathbb{E}_{\Omega}\Big((\boldsymbol{L^i})_{i\in[I]}\Big) \tag{7.14a}$$

$$= \sum_{i=1}^{I} \sum_{\xi\in\Omega} \pi_\xi \, L^i(x^i(\xi),\xi). \tag{7.14b}$$

Alternatively, considering a robust approach to uncertainties, we have the operator $\mathcal{F}^{UR}_{I\times\Omega}$ which minimizes the worst-case aggregated costs of prosumers:

$$\mathcal{F}^{UR}_{I\times\Omega}\Big((\boldsymbol{L^i})_{i\in[I]}\Big) = \sup_{\xi\in\Omega} \odot \mathcal{F}^{U}_{I}\Big((\boldsymbol{L^i})_{i\in[I]}\Big) \tag{7.15a}$$

$$= \sup_{\xi\in\Omega}\Big\{ \sum_{i=1}^{I} L^i(x^i(\xi),\xi) \Big\}. \tag{7.15b}$$

Remark 11. *We claim that* $\sup_{\xi\in\Omega}\odot\mathcal{F}^{U}_{I}$ *makes more sense than* $\mathcal{F}^{U}_{I}\odot\sup_{\xi\in\Omega}$ *as the later aggregates each prosumer's worst costs. Indeed, if the worst-case costs for different prosumers occur in different scenarios, the aggregated costs calculated might never happen or happen in a scenario* ξ *not in* Ω*.*

On the other hand, we have $\mathbb{E}_{\xi\in\Omega}\odot\mathcal{F}^{U}_{I} = \mathcal{F}^{U}_{I}\odot\mathbb{E}_{\xi\in\Omega}$*, by associativity of sums. Similarly, by associativity of supremum, we have* $\sup_{\xi\in\Omega}\odot\mathcal{F}^{PMM}_{I} = \mathcal{F}^{PMM}_{I}\odot\sup_{\xi\in\Omega}$*.*

As the first two operators do not model fairness considerations into the model, we now look for a fair distribution by using $\mathcal{F}^{PMM}_{I}$ to aggregate prosumers' costs. First, let $\boldsymbol{x}^{i,\star}(\boldsymbol{\xi})$ be the[2] optimal solution of $(P^{i,\rho})$, and denote $v^{i,\rho}_{\xi} := L^i(x^{i,\star}(\xi),\xi)$, the cost incurred by i when operating alone under uncertainty realization ξ. Finally, $\boldsymbol{v}^{i,\rho}$ is the random variable taking values $v^{i,\rho}_{\xi}$ for the respective realization ξ.

Results given in Sections 7.4 and 7.5.3 suggest that the scaled minimax approach suits our problem more than the proportional approach. Thus, in a stochastic framework, we propose the operator $\mathcal{F}^{SMS}_{I\times\Omega}$:

$$\mathcal{F}^{SMS}_{I\times\Omega}\Big((\boldsymbol{L^i})_{i\in[I]}\Big) = \mathcal{F}^{PMM}_{I} \odot \mathbb{E}_{\Omega}\Big((\boldsymbol{L^i})_{i\in[I]}\Big) \tag{7.16a}$$

$$= \max_{i\in[I]}\left\{ \frac{\mathbb{E}\big[\boldsymbol{v}^{i,\mathbb{E}}\big] - \sum_{\xi\in\Omega}\, \pi_\xi \, L^i(x^i(\xi),\xi)}{\mathbb{E}\big[\boldsymbol{v}^{i,\mathbb{E}}\big]} \right\}. \tag{7.16b}$$

Finally, combining the robust and the scaled minimax approaches, we obtain the operator $\mathcal{F}^{SMR}_{I\times\Omega}$, which focuses on the prosumer having the worst worst-case proportional costs:

$$\mathcal{F}^{SMR}_{I\times\Omega}\Big((\boldsymbol{L^i})_{i\in[I]}\Big) = \sup_{\xi\in\Omega} \odot \mathcal{F}^{PMM}_{I}\Big((\boldsymbol{L^i})_{i\in[I]}\Big) \tag{7.17a}$$

$$= \sup_{\xi\in\Omega}\left\{ \max_{i\in[I]} \left\{ \frac{v^{i,\mathbb{E}}_{\xi} - L^i(x^i(\xi),\xi)}{v^{i,\mathbb{E}}_{\xi}} \right\} \right\}. \tag{7.17b}$$

[2] We assume uniqueness of a way of selecting an optimal solution, as in Section 7.3)

Remark 12. *Note that here, depending on the sense of the combination between uncertainty-operator and agent-operator, we could have a model with different risk-measure profiles for the prosumers.*

We now turn to extending the acceptability constraint (7.3) to a stochastic setting.

7.6.3 Stochastic dominance constraints

As in Section 7.5.2, to induce acceptability, we require that, for each prosumer i, its random cost $L^i(\boldsymbol{x}^i(\boldsymbol{\xi}), \boldsymbol{\xi})$ is less than the random cost of the independent model $\boldsymbol{v}^{i,\mathbb{E}}$. Unfortunately, there is no natural ordering of random variable (or equivalently of $\mathbb{R}^\Omega$), and each (partial) order will define a different extension of the acceptability constraint (7.3).

We now present four acceptability constraints, using various ordering on the space of random variable, leveraging the stochastic dominance theory (see [DR03] for an introduction in the context of stochastic optimization). In this section, we give the mathematical expression of acceptability constraints, but mixed integer formulation can be found in Chapter D.

In a very conservative perspective, we consider the almost-sure order, comparing random variables scenario by scenario:

$$\mathcal{A}^{i,\rho}_{a.s} := \left\{ \boldsymbol{u}^{i,\rho} \,|\, u^{i,\rho}_\xi \leq v^{i,\rho}_\xi, \quad \forall \xi \right\}. \tag{7.18a}$$

We can relax the almost-sure ordering by not requiring the benefit of aggregation for all scenarios but distributionally. For example, if we have two scenarios ξ and ζ, with same probability, we consider that it is acceptable to lose on ξ if we do better on ζ, that is such that $u^{i,\rho}_\xi \leq v^{i,\rho}_\xi$ and $u^{i,\rho}_\zeta \geq v^{i,\rho}_\zeta$. To formalize this approach, we turn to *stochastic first-order dominance constraints* (see [DR03]), and leverage 1^{st} order acceptability:

$$\begin{aligned}
\mathcal{A}^{i,\rho}_{(1)} :=& \left\{ \boldsymbol{u}^{i,\rho} \,|\, \boldsymbol{u}^{i,\rho} \preceq_{(1)} \boldsymbol{v}^{i,\rho} \right\} && (7.18b)\\
:=& \left\{ \boldsymbol{u}^{i,\rho} \,|\, \mathbb{P}(\boldsymbol{u}^{i,\rho} > \eta) \leq \mathbb{P}(\boldsymbol{v}^{i,\rho} > \eta), && \forall \eta \in \mathbb{R} \right\}\\
:=& \left\{ \boldsymbol{u}^{i,\rho} \,|\, \mathbb{E}\big[g(\boldsymbol{u}^{i,\rho})\big] \leq \mathbb{E}\big[g(\boldsymbol{v}^{i,\rho})\big] \qquad \forall g : \mathbb{R} \to \mathbb{R}, \text{ non-decreasing} \right\}.
\end{aligned}$$

One downside of this acceptability constraint is that the modeling entails numerous binary variables, posing practical implementation challenges.

We can thus consider a relaxed, less risk-averse version of 1^{st} order acceptability, relying on *stochastic second-order dominance constraints*, also known as increasing convex acceptability, which is equivalent to :

$$\begin{aligned}
\mathcal{A}^{i,\rho}_{(ic)} :=& \left\{ \boldsymbol{u}^{i,\rho} \,|\, \boldsymbol{u}^{i,\rho} \preceq_{(ic)} \boldsymbol{v}^{i,\rho} \right\} && (7.18c)\\
=& \left\{ \boldsymbol{u}^{i,\rho} \,|\, \mathbb{E}\big[(\boldsymbol{u}^{i,\rho} - \eta)^+ \big] \leq \mathbb{E}\big[(\boldsymbol{v}^{i,\rho} - \eta)^+ \big] && \forall \eta \in \mathbb{R} \right\}\\
=& \left\{ \boldsymbol{u}^{i,\rho} \,|\, \mathbb{E}\big[g(\boldsymbol{u}^{i,\rho}) \big] \leq \mathbb{E}\big[g(\boldsymbol{v}^{i,\rho}) \big], \qquad \forall g : \mathbb{R} \to \mathbb{R}, \text{convex, non-decreasing} \right\}.
\end{aligned}$$

Moreover, increasing convex acceptability is also easier to implement than 1^{st} order acceptability (see Chapter D).

Finally, the risk-neutral acceptability constraint compares two random variables through their expectation:

$$\mathcal{A}^{i,\rho}_{\mathbb{E}} := \left\{ \boldsymbol{u}^{i,\rho} \,|\, \mathbb{E}_{\mathbb{P}}\left[\boldsymbol{u}^{i,\rho}\right] \leq \mathbb{E}_{\mathbb{P}}\left[\boldsymbol{v}^{i,\rho}\right] \right\}. \tag{7.18d}$$

We can use another convex risk measure instead of the expectation in (7.18d).

Remark 13. *We have that* $\mathcal{A}^{i,\rho}_{a.s} \subseteq \mathcal{A}^{i,\rho}_{(1)} \subseteq \mathcal{A}^{i,\rho}_{(ic)} \subseteq \mathcal{A}^{i,\rho}_{\mathbb{E}}$. *Therefore, in the same way as in Remark 10, the acceptability constraint yields a balance between risk-neutral* ($\mathcal{A}^{i,\rho}_{\mathbb{E}}$) *and a robust approach on risk* ($\mathcal{A}^{i,\rho}_{a.s}$), *with intermediary visions on risk* ($\mathcal{A}^{i,\rho}_{(ic)}$, $\mathcal{A}^{i,\rho}_{(1)}$)

7.6.4 Numerical illustration

We consider the stochastic version of the example presented in Section 7.4, where balancing prices $\{\boldsymbol{p}^B_t\}_{t\in[T]}$ are random variables with uniform, independent, distribution over $[0.35p^{DA}_t, 5p^{DA}_t]$. The problem can be formulated as a multi-stage program, where day-ahead purchases are decided in the first stage (a day-ahead of following stages), and then each stage corresponds to a time slot where we can buy energy on the balancing market at a price $\boldsymbol{p}^B_t$.

We solve and discuss the sample average approximation of the two-stage approximation of this problem. More precisely, we draw 50 prices scenario, and solve a two-stage program where the first stage decisions are the day-ahead purchases, and the second stage decisions are the balancing purchases from time slot 1 to T. We set $I = 4, T = 10$, and we draw $\Omega = 50$ scenarios of balancing prices. For the prosumers' parameters and market prices, we use the data on Tables 7.1 and 7.6.

Table 7.6: Prices on both markets

t	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
p^{DA}_t	3	3	7	4	2	10	7	4	7.5	8
q^{DA}_t	12	12	12	12	12	12	12	12	12	12

Table 7.7: Percentage of expected savings $\frac{\mathbb{E}\left[\boldsymbol{v}^{i,\text{sup}}\right]-\mathbb{E}\left[L^i(\boldsymbol{x}^i(\boldsymbol{\xi})\right]}{\mathbb{E}\left[\boldsymbol{v}^{i,\text{sup}}\right]}$ made by A_i, expected aggregated costs $\mathbb{E}\left[\mathcal{F}_I(L^1(\boldsymbol{x}^{i,\star}(\boldsymbol{\xi}))_{i\in[I]})\right]$ and PoA in the corresponding model.

	Utilitarian Stochastic $\mathcal{F}^{US}_{I\times\Omega}$						Scaled Minimax Stochastic $\mathcal{F}^{SMS}_{I\times\Omega}$					
	A_1	A_2	A_3	A_4	(A)	PoA	A_1	A_2	A_3	A_4	(A)	PoA
$\emptyset$	2	52	-14	52	684	0	32	36	32	32	770	0
$\mathcal{A}^{\mathbb{E}}_{\mathbb{E}}$	4	51	1	48	684	0	32	36	32	32	770	0
$\mathcal{A}^{\mathbb{E}}_{(ic)}$	22	44	25	36	721	37	32	36	32	32	770	0
$\mathcal{A}^{\mathbb{E}}_{(1)}$	36	24	34	18	882	198	23	21	28	19	918	148
$\mathcal{A}^{\mathbb{E}}_{a.s}$	43	17	41	14	930	246	37	17	33	16	941	171
	Utilitarian Robust $\mathcal{F}^{UR}_{I\times\Omega}$						Scaled Minimax Robust $\mathcal{F}^{SMR}_{I\times\Omega}$					
	A_1	A_2	A_3	A_4	(A)	PoA	A_1	A_2	A_3	A_4	(A)	PoA
$\emptyset$	-19	53	6	49	686	0	22	48	21	38	693	0
$\mathcal{A}^{\mathbb{E}}_{\mathbb{E}}$	0	51	0	48	686	0	22	48	21	38	693	0
$\mathcal{A}^{\mathbb{E}}_{(ic)}$	22	43	25	33	738	52	29	43	29	33	720	27
$\mathcal{A}^{\mathbb{E}}_{(1)}$	25	22	28	15	920	234	28	26	33	18	881	188
$\mathcal{A}^{\mathbb{E}}_{a.s}$	43	17	40	14	930	244	43	17	40	14	930	237

We solve the problem with different combinations of aggregation operators and acceptability sets

and can compare the impact of each combination on the solution. We denote m_a^f the model with aggregation operator $f \in \{US, SMS, UR, SMR\}$ and acceptability set $a \in \{\emptyset, \mathbb{E}, (ic), (1), (a.s)\}$. We read prosumers' expected percentage of savings with risk-neutral and worst-case approaches on Table 7.7. For example, in model $m_{(ic)}^{US}$, we read that A_1 (resp. A_2, A_3, A_4) saves 22% (resp. 44%, 25%, 36%) of its costs. The expected cost of the aggregation is 721\$, thus asking for increasing-convex acceptability costs 37\$. Moreover, we can observe the distribution of prosumers' expected costs with a risk-neutral (resp. worst-case) approach on Figure 7.3 (resp. Figure 7.4).

Our first comment is that the problems previously identified from a utilitarian perspective with no acceptability constraints are still present in a stochastic framework. Indeed, both with the risk-neutral utilitarian $\mathcal{F}_{I\times\Omega}^{US}$ and worst-case utilitarian $\mathcal{F}_{I\times\Omega}^{UR}$ operators, we observe on Table 7.7 that some prosumers can pay more in the aggregation compared to being alone (A_3 pays +14% in the stochastic approach, and A_1 pays +19% in the robust approach). This highlights the necessity for either acceptability constraints or an aggregation operator.

If we choose a fair approach through the objective (operators $\mathcal{F}_{I\times\Omega}^{SMS}$ and $\mathcal{F}_{I\times\Omega}^{SMR}$), we guarantee a higher percentage of savings to all prosumers than in the utilitarian approach. For example, with no acceptability constraints, all prosumers save at least 32% of their costs in a risk-neutral approach and 21% in a robust approach, compared to respectively −14% and −19% with the utilitarian approach. This comes at the price of efficiency, especially in the risk-neutral case, as the expected aggregated costs of the scaled minimax approach is 13% higher than with the utilitarian approach. This remains true as we use smaller acceptability set.

Conversely, when solving this problem with a utilitarian approach (operators $\mathcal{F}_{I\times\Omega}^{US}$ and $\mathcal{F}_{I\times\Omega}^{UR}$), we can increase the guaranteed percentage of savings by constraining more the acceptability. Indeed, with $\mathcal{F}_{I\times\Omega}^{US}$, all prosumers save at least from 1% with expected acceptability to 22% with increasing convex acceptability, and with $\mathcal{F}_{I\times\Omega}^{UR}$, it is from 0% to 22%. However, increasing the acceptability to first-order or almost sure does not improve this guarantee, as the problem is now getting too constrained. For example, with almost-sure acceptability, the choice of the operator on uncertainty is inconsequential: the distribution of costs is the same with both operators $\mathcal{F}_{I\times\Omega}^{US}$ and $\mathcal{F}_{I\times\Omega}^{UR}$. Notably, there exists a substantial gap between increasing-convex acceptability and first-order acceptability. For example, with the scaled minimax stochastic operator $\mathcal{F}_{I\times\Omega}^{SMS}$, the costs increase from 770 with increasing convex acceptability to 918 with first-order acceptability.

Thus, we obtain various solutions with different balances between efficiency and fairness and different risk visions. In this example, in the stochastic case, if we want to give the same guarantees to every prosumer, the natural choice is operator $\mathcal{F}_{I\times\Omega}^{SMS}$. However, this approach costs 13% more than in $m_{\emptyset}^{\mathcal{F}_{I\times\Omega}^{US}}$. If we want to opt for a less costly approach, the operator $\mathcal{F}_{I\times\Omega}^{US}$ combined with increasing convex acceptability might be considered a better trade-off between efficiency and fairness: it ensures at least 22% of savings to each prosumer and induces 5% of efficiency loss. Finally, increasing the variability of the scenarios drawn for these tests does not significantly impact the empirical conclusions we make in this section.

7.7 Conclusion

In this paper, we have provided a framework with tools to accommodate fairness in prosumer aggregation problems. Through the discussion in Section 2, we emphasized the importance of fairness and the need to carefully consider how to model it and be aware of the different approaches available. Since modeling fairness aims to improve the social acceptability of mathematical models, we connected underlying philosophical concepts to the fairness modeling process.

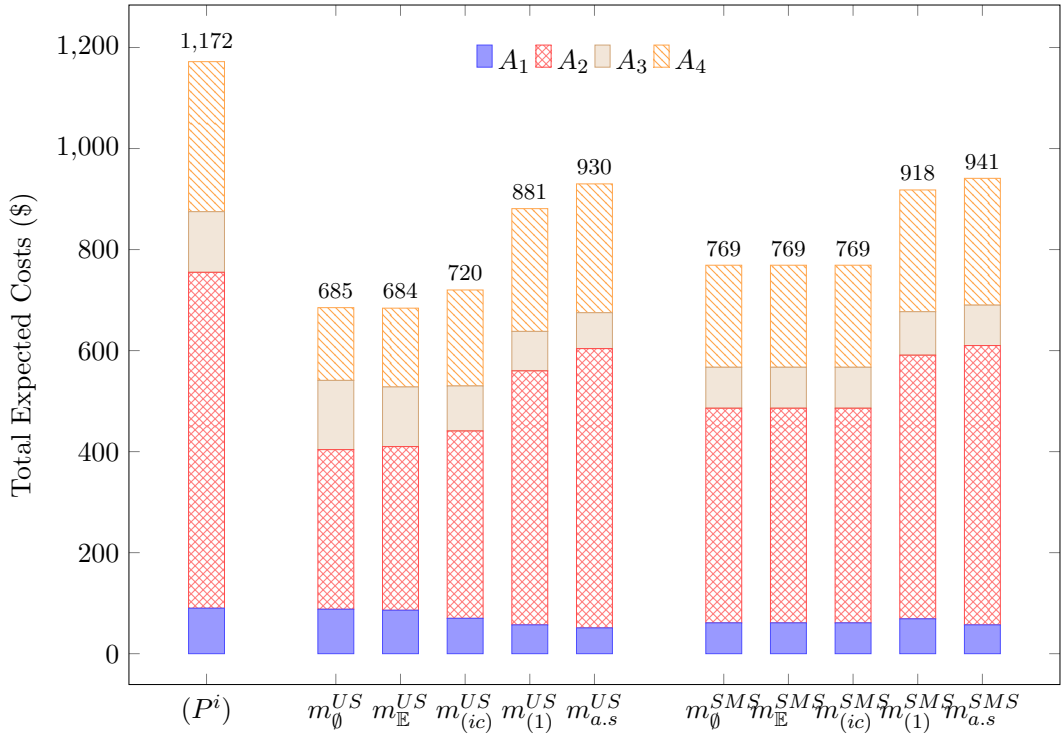


Figure 7.3: The bars correspond to the results of different models we solve with a stochastic approach. The first one is the model without aggregation
: we solve each $(P^{i,\mathbb{E}})$ independently. The second bar corresponds to the problem solved with operator $\mathcal{F}^{US}_{I\times\Omega}$ without acceptability constraints. Then, the four following bars correspond to the same problem with increasingly strong acceptability $(\mathcal{A}^{\mathbb{E}}_{\mathbb{E}}, \mathcal{A}^{\mathbb{E}}_{(ic)}, \mathcal{A}^{\mathbb{E}}_{(1)}, \mathcal{A}^{\mathbb{E}}_{a.s})$. The following bar is for the problem solved with operator $\mathcal{F}^{SMS}_{I\times\Omega}$ without acceptability constraints, followed by four bars with different acceptability sets. Each bar is decomposed in 4 blocks corresponding to the expected share $\mathbb{E}[L^i(\boldsymbol{x}^i(\boldsymbol{\xi}), \boldsymbol{\xi})]$ of each consumer i. At the top of each bar, we can read the sum of expected aggregated costs in the corresponding model.

First, we discussed acceptability constraints to discourage prosumers from leaving the aggregation. We then compared different choices of the objective function (utilitarian, scaled minimax, and proportional). Through a stylized (and more straightforward to interpret) deterministic case study, we showed how different combinations of objectives and constraints influence solutions, emphasizing the importance of fairness and acceptability considerations. We then extended the model to dynamic and stochastic frameworks, aligning it with what we expect practical problems to be. In this context, we adapt acceptability constraints to account for long-term horizons and uncertainties, and we showcase their impact on solutions using similar stylized instances.

In our numerical example, we obtained a spectrum of options from various combinations of acceptability sets and objective functions, ranging from the most efficient models (with the lowest aggregated costs) to the fairest models (where agents' gains are more comparable). Too-restrictive acceptability sets or a bargaining approach (proportional operator $\mathcal{F}^P_I$) can significantly reduce efficiency, while an intermediate approach leverages aggregation benefits without excessively favoring certain prosumers. Thus, we recommend the proportional min-max agent aggregator with progressive acceptability constraint in the dynamic case (resp. increasing convex acceptability in the stochastic case), which balances efficiency and fairness well. Recall that the framework discussed here is not reduced to prosumer aggregation in energy markets only, and can be adapted

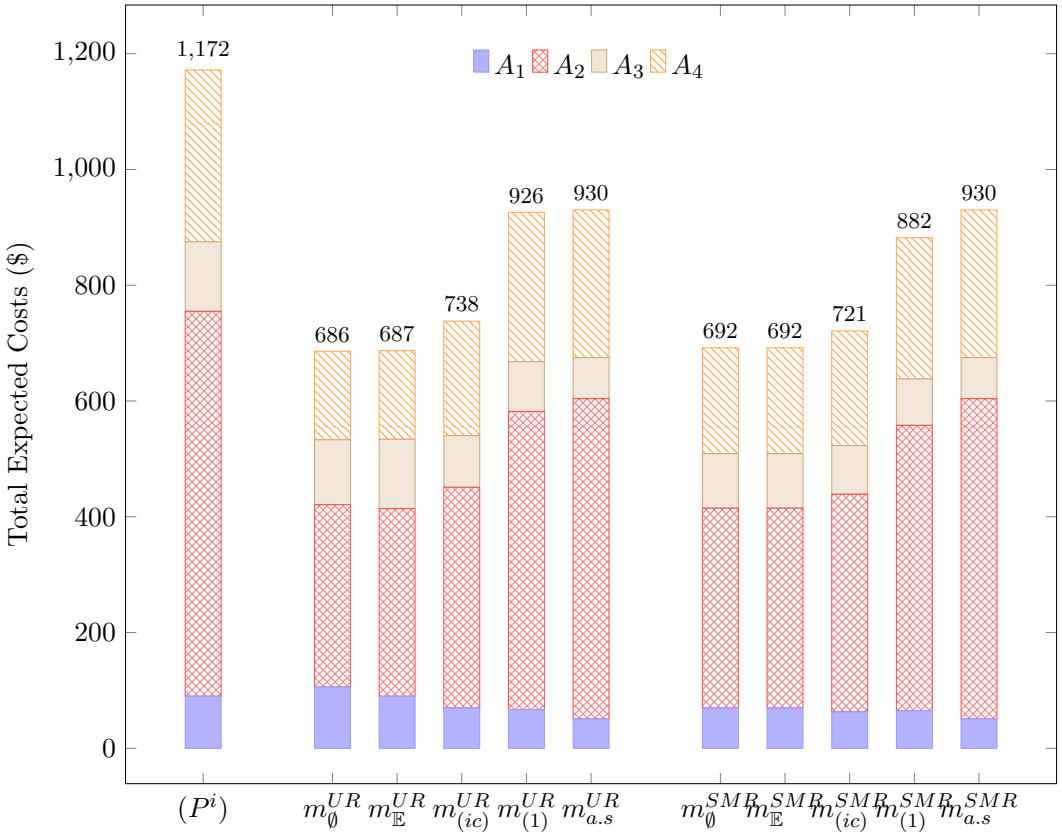


Figure 7.4: This figure can be read like Figure 7.3, except that the two considered operators are $\mathcal{F}_{I\times\Omega}^{UR}$ and $\mathcal{F}_{I\times\Omega}^{SMR}$.

to other aggregation problems in energy system management problems (*e.g.,* virtual power plant, portfolio management in energy markets, ancillary service provision, etc.).

In future work, we plan to discuss the extension of the aggregation problem to a multistage stochastic program, where we would have to combine the partial orders presented in the dynamic framework in Section 7.5 and the stochastic orders of the stochastic framework in Section 7.6. This will require a discussion of possible aggregators $\mathcal{F}_{T\times\Omega\times I}$ over agent, time, and uncertainty simultaneously. Although we can easily assume a factorization of the form $\mathcal{F}_I \odot \mathcal{F}_{T\times\Omega}$, it would not be realistic to describe $\mathcal{F}_{T\times\Omega}$ as the composition of a time aggregator and an uncertainty aggregator. Indeed, such a factorization would not guarantee time-consistency of the problem and might not even preserve non-anticipativity. Further, acceptability constraints must be defined using multivariate stochastic order (see [DR09; AL15; DW16]) whose mathematical programming representations are more involved. Finally, it would be interesting to investigate potential bounds on *price of acceptability*.

Appendix C

Computing Shapley's values

In this appendix, we give more details on how to compute Shapley's values for the application in Section 7.4. First, we introduce the general formulas and definitions required to compute Shapley's values, then we apply them to our example.

C.1 Definitions and Formulas

Shapley's values are commonly seen as a fair distribution of costs (or revenues) when agents cooperate in a cooperative game. The Shapley Value returns each player's fair share of the total gains by averaging their contributions across all possible ways they can join the coalition. Let N be the number of agents in the game. We denote $w : 2^{|N|} \to \mathbb{R}$, the worth function associating to a coalition S, the expected payoff obtained by cooperation.

To compute Shapley's values, first, we compute $\delta_i(S)$, the marginal contribution of agent i to coalition $S \subset N$:

$$\delta_i(S) = w(S \cup \{i\}) - w(S) \tag{C.1a}$$

Then, the Shapley value of agent i, given a characteristic function w, is $\phi_i(w)$:

$$\phi_i(w) = \frac{1}{n} \sum_{S \subset N \setminus \{i\}} \binom{n-1}{|S|}^{-1} \delta_i(S) \tag{C.1b}$$

C.2 Application to our example

We consider an example with 4 agents. For each coalition $S \subset [4]$, we solve the aggregated problem with utilitarian operator $\mathcal{F}_S^U$ and acceptability constraints $\mathcal{A}_1$-as in cooperation games we must satisfy individual rationality properties- and obtain optimal solution c_S Then, we introduce the characteristic function assessing the worth of coalition S as:

$$w(S) = \sum_{i \in S} v^i - c_S, \tag{C.2a}$$

where v^i is the optimal value of problem (P^i). Thus $w(S)$ designates the savings made by coalition S when they cooperate. On Tables C.1 and C.2, we detail the intermediate computations, and in Equations (C.2b) to (C.2e) we compute shapley's values.

	{1}	{2}	{3}	{4}	{1,2}	{1,3}	{1,4}	{2,3}	{2,4}	{3,4}	{1,2,3}	{1,2,4}	{1,3,4}	{2,3,4}	{1,2,3,4}
$\sum_{i\in S} v^i$	50	280	40	168	330	90	218	320	448	208	370	498	258	488	538
c_S	50	280	40	168	330	90	218	320	448	208	244	365	162	392	333
$w(S)$	0	0	0	0	0	0	0	0	0	0	126	133	96	96	205

Table C.1: Costs with and without cooperation and worth w of each coalition S.

	{1}	{2}	{3}	{4}	{1,2}	{1,3}	{1,4}	{2,3}	{2,4}	{3,4}	{1,2,3}	{1,2,4}	{1,3,4}	{2,3,4}
$\delta_1(S)$	-	0	0	0	-	-	-	126	133	96	-	-	-	109
$\delta_2(S)$	0	-	0	0	-	126	133	-	-	96	-	-	109	-
$\delta_3(S)$	0	0	-	0	126	-	96	-	96	-	-	72	-	-
$\delta_4(S)$	0	0	0	-	133	96	-	96	-	-	79	-	-	-

Table C.2: Marginal contributions of agent i to each coalition.

$$\begin{aligned}\phi_1(w) = \frac{1}{4}\Bigg[&\binom{3}{1}^{-1} \big(\delta_1(\{2\}) + \delta_1(\{3\}) + \delta_1(\{4\})\big) \\ &+ \binom{3}{2}^{-1} \big(\delta_1(\{2,3\}) + \delta_1(\{2,4\}) + \delta_1(\{3,4\})\big) \\ &+ \binom{3}{3}^{-1} \delta_1(\{2,3,4\})\Bigg]\end{aligned}$$

$$\phi_1(w) = \frac{1}{4}\left[\frac{1}{3}\times 0 + \frac{1}{3}(126+133+96) + 1\times 109\right] = 56.8 \tag{C.2b}$$

$$\phi_2(w) = \frac{1}{4}\left[\frac{1}{3}\times 0 + \frac{1}{3}(126+133+96) + 1\times 109\right] = 56.8 \tag{C.2c}$$

$$\phi_3(w) = \frac{1}{4}\left[\frac{1}{3}\times 0 + \frac{1}{3}(126+96+96) + 1\times 72\right] = 44.5 \tag{C.2d}$$

$$\phi_4(w) = \frac{1}{4}\left[\frac{1}{3}\times 0 + \frac{1}{3}(133+96+96) + 1\times 79\right] = 46.8 \tag{C.2e}$$

As the values computed here represent the way to distribute savings, we must subtract $\phi_i(w)$ from the cost of agent i to obtain its costs in the aggregation after fair allocation through Shapley's values in Table C.3.

	A_1	A_2	A_3	A_4
$\phi_i(w)$	56.8	56.8	44.5	46.8
v^i	50	280	40	168
$L^i(x^i)$	-6.8	223.2	-4.5	121.2
$100\frac{v^i - L^i(x^i)}{v^i}$	114	20	111	28

Table C.3: Costs and proportional savings of the agents with Shapley's post-allocation scheme.

Appendix D

Modeling of stochastic dominance constraints

We present here practical formulas to implement the stochastic orders dominance constraints introduced in Section 7.6.3. Those constraints establish a dominance between $\boldsymbol{v}^{i,\rho}$, the random variable representing i independent costs, and $\boldsymbol{u}^{i,\rho}$, the random variable representing i costs in the aggregation.

D.1 First-order dominance constraint model

The first-order dominance constraints (7.18b) model is based on [GNS08].

Lemma 1. *In Problem (A^ρ), acceptability constraints $\boldsymbol{u}^{i,\rho} \preceq_{(1)} \boldsymbol{v}^{i,\rho}$ can be modeled with:*

$$b^i_{\xi,\eta} \in \{0,1\} \qquad \forall \eta \in [\Omega],\ \forall \xi \in \Omega \tag{D.1a}$$

$$u^{i,\rho}_\xi - v^{i,\rho}_\eta \leq M\, b^i_{\xi,\eta} \qquad \forall \eta \in [\Omega],\ \forall \xi \in \Omega \tag{D.1b}$$

$$\sum_{\xi=1}^{\Omega} \pi_\xi b^i_{\xi,\eta} \leq a_\eta \qquad \forall \eta \in [\Omega]. \tag{D.1c}$$

We denote $a_\eta := \mathbb{P}(\boldsymbol{v}^{i,\rho} > v^{i,\rho}_\eta)$, which is a parameter for the aggregation problem.

Proof. As Ω is assumed to be finite, $\boldsymbol{v}^{i,\rho}$ follows discrete distribution with realizations $v^{i,\rho}_\eta$ for $\eta \in \Omega$. Then,

$$\begin{aligned} \boldsymbol{u}^{i,\rho} \preceq_{(1)} \boldsymbol{v}^{i,\rho} &\iff \mathbb{P}(\boldsymbol{u}^{i,\rho} > \eta) \leq \mathbb{P}(\boldsymbol{v}^{i,\rho} > \eta) && \forall \eta \in \mathbb{R} \\ &\iff \mathbb{P}(\boldsymbol{u}^{i,\rho} > v^{i,\rho}_\eta) \leq \mathbb{P}(\boldsymbol{v}^{i,\rho} > v^{i,\rho}_\eta) && \forall \eta \in \Omega. \end{aligned}$$

Then, using $\mathbb{P}(\boldsymbol{X} > x) = \mathbb{E}[\mathbb{1}_{\boldsymbol{X}>x}]$, and introducing binary variables $b^i_{\xi,\eta} = \mathbb{1}_{u^{i,\rho}_\xi > v^{i,\rho}_\eta}$, we get:

$$\left(\mathbb{P}(\boldsymbol{u}^{i,\rho} > v^{i,\rho}_\eta) \leq \mathbb{P}(\boldsymbol{v}^{i,\rho} > v^{i,\rho}_\eta) \iff \sum_{\xi=1}^{\Omega} \pi_\xi b^i_{\xi,\eta} \leq a_\eta\right) \qquad \forall \eta \in \Omega.$$

To linearize the definition of $b^i_{\xi,\eta}$, we rely on big-M constraint:

$$\begin{aligned} b^i_{\xi,\eta} &\in \{0,1\} && \forall \eta \in \Omega, \forall \xi \in \Omega \\ u^{i,\rho}_\xi - v^{i,\rho}_\eta &\leq M b^i_{\xi,\eta} && \forall \eta \in \Omega, \forall \xi \in \Omega. \end{aligned}$$

□

D.2 Increasing convex dominance constraint model

The increasing convex dominance constraints (7.18c), is based on [CGS09].

Lemma 2. *In problem* (A^ρ)*, the acceptability constraint* $\boldsymbol{u}^{i,\rho} \preceq_{(ic)} \boldsymbol{v}^{i,\rho}$ *can be modeled with:*

$$s^i_{\xi,\eta} \geq 0 \qquad \forall \eta \in [\Omega],\ \forall \xi \in \Omega \tag{D.2a}$$

$$s^i_{\xi,\eta} \geq u^{i,\rho}_\xi - v^{i,\rho}_\eta \qquad \forall \eta \in [\Omega],\ \forall \xi \in \Omega \tag{D.2b}$$

$$\sum_{\xi=1}^{\Omega} \pi_\xi s^i_{\xi,\eta} \leq a^{ic}_\eta \qquad \forall \eta \in [\Omega]. \tag{D.2c}$$

We denote $a^{ic}_\eta := \mathbb{E}[(\boldsymbol{v}^{i,\rho} - v^{i,\rho}_\eta)^+]$.

Proof. As in Section D.1, we know that $\boldsymbol{v}^{i,\rho}$ follows a discrete distribution with realizations $v^{i,\rho}_\eta$) for $\eta \in \Omega$. Then,

$$\begin{aligned} \boldsymbol{u}^{i,\rho} \preceq_{(ic)} \boldsymbol{v}^{i,\rho} \iff & \quad \mathbb{E}\big[\,(\boldsymbol{u}^{i,\rho} - \eta)^+\,\big] \leq \mathbb{E}\big[\,(\boldsymbol{v}^{i,\rho} - \eta)^+\,\big] && \forall \eta \in \mathbb{R} \\ \iff & \quad \mathbb{E}\big[\,(\boldsymbol{u}^{i,\rho} - v^{i,\rho}_\eta)^+\,\big] \leq \mathbb{E}\big[\,(\boldsymbol{v}^{i,\rho} - v^{i,\rho}_\eta)^+\,\big] && \forall \eta \in \Omega. \end{aligned}$$

We introduce positive variables $s^i_{\xi,\eta} = (u^{i,\rho}_\xi - v^{i,\rho}_\eta)^+$, for $\eta \in \Omega$. Thus, we can model the increasing convex dominance constraints as: follows

$$\left(\mathbb{E}\big[\,(\boldsymbol{u}^{i,\rho} - v^{i,\rho}_\eta)^+\,\big] \leq \mathbb{E}\big[\,(\boldsymbol{v}^{i,\rho} - v^{i,\rho}_\eta)^+\,\big] \iff \sum_{\xi=1}^{\Omega} \pi_\xi s^i_{\xi,\eta} \leq a^{ic}_\eta\right) \qquad \forall \eta \in [\Omega].$$

□

Appendix E

Additional numerical results

E.1 Impact of α

We test the model with different values of α that represent the targeted gap between a prosumer's cost in the aggregation and its individual optimal cost. Figure E.1 presents the proportional savings of prosumers when solving m_α^f for $f \in \{U, SM, P\}$ and $\alpha \in \{0.6, 0.7, 0.8, 0.9, 1.0\}$. Bear in mind that solving $m_\emptyset^{SM}$ is equivalent to maximizing α (see Remark 8). Hence, if $\alpha^\star$ is the optimal solution of Problem (7.5), enforcing acceptability constraints with $\alpha \geq \alpha^\star$ results in the same solution, while solving m_α^{SM} with $\alpha < \alpha^\star$ leads to the disagreement point. In our study case, $\alpha^\star = 0.6964285$.

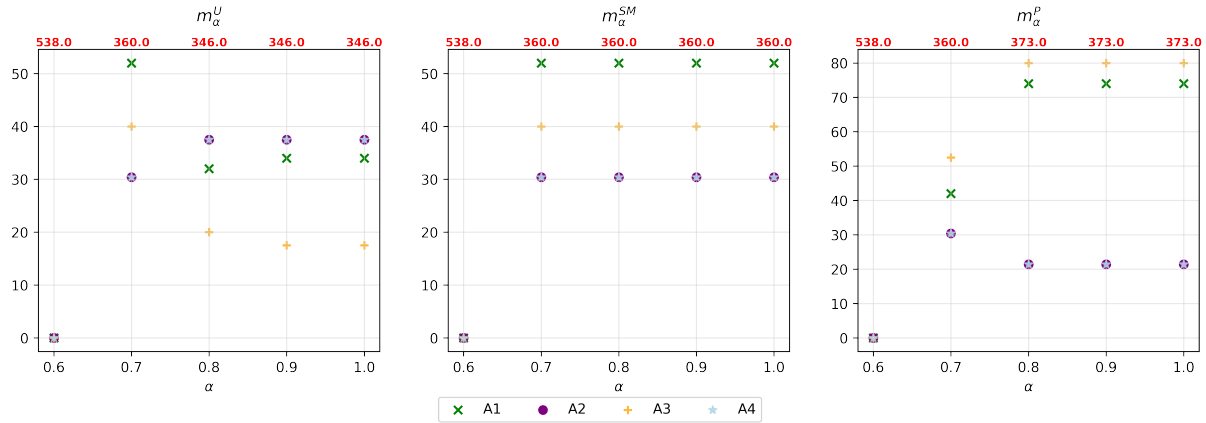


Figure E.1: Proportional savings $\frac{v^i - L^i(x^i)}{v^i}$ of prosumers depending on the chosen fairness operator and the acceptability coefficient α. In red above the figure is the cost of the aggregation in the corresponding model.

In the utilitarian case ($f = U$), increasing α relaxes the model, leading to a reduction in aggregation costs between $m_{0.7}^U$ and $m_{0.8}^U$. In some cases, while the aggregation cost remains unchanged, varying α affects the cost allocation of prosumers. For example, with $\alpha = 0.8$, A_1 and A_3 save respectively 32% and 20%, whereas with $\alpha = 0.9$ they save 34% and 17.5%. Finally, with the proportional operator, we observe a switch in the aggregation cost between $m_{0.7}^P$ and $m_{0.8}^P$ leading to a change in cost allocation. Unlike the utilitarian operator, cost distribution among prosumers remains unchanged for the same aggregation cost. Here, enforcing stronger acceptability results in more evenly distributed proportional savings. Specifically, for $\alpha \geq 0.8$,

savings range from 21.4% to 80%, whereas with $\alpha < 0.8$, they range from 30.5% to 52.5%.

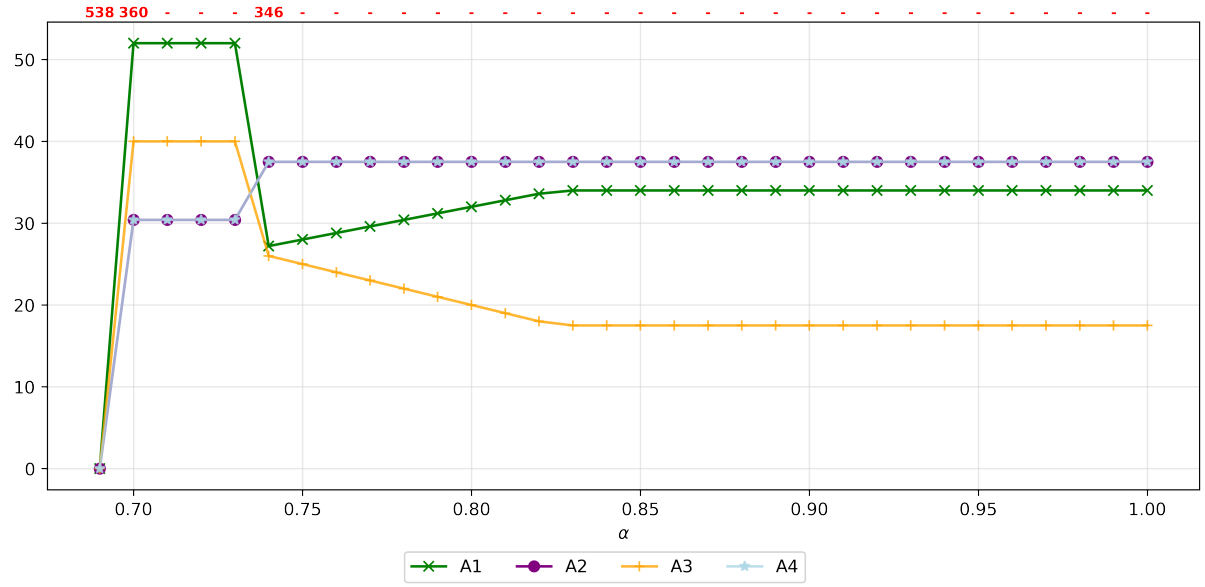


Figure E.2: Proportional savings $\frac{v^i - L^i(x^i)}{v^i}$ of prosumers in the model m_α^U depending on α. Above the figure, the aggregation cost for each model is shown in red, with '$-$' indicating no change.

We test the utilitarian model m_α^U with a finer discretization of α. As shown in Figure E.2, for α between 0.74 and 0.83, the aggregation cost remains unchanged, while cost distribution shifts between A_1 and A_3. In this case, A_1 and A_3 have flexible energy consumption, allowing the aggregation to rely on either for day-ahead market access. A shift in cost distribution between them indicates a change in which consumer is prioritized.

E.2 Increasing the gap between day-ahead and balancing prices

We conduct tests on various instances to verify that our observations are not solely problem-dependent. Specifically, we consider cases with different gaps between day-ahead and balancing prices

As a reference, we use the balancing prices in Table E.1. We then evaluate the models with day-ahead prices set as $p_t^{DA} = \gamma p_t^B$, where $\gamma \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. A lower γ increases the benefit of aggregation. The parameters for the minimum and maximum energy consumption of prosumers are given in Table 7.1. Since we set $T = 10$, the total demand is adjusted to $Q = [20, 50, 16, 27]$.

Table E.1: Balancing prices per stage

t	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
p_t^B	34	35	30	33	24	25	38	35	39	25

To compare the different instances, we consider three indicators. First, the **minimal proportional savings**:

$$\alpha^\star := \min_{i \in [I]} \{ \frac{v^i - L^i(x^i)}{v^i} \},$$

representing the smallest proportional savings received by any agent. Then, the **dispersion of proportional savings**, denoted δ:

$$\delta := \max_{i\in[I]} \Big\{ \frac{v^i - L^i(x^i)}{v^i} \Big\} - \alpha^\star,$$

that measures the difference between the highest and lowest proportional savings. Finally, **global aggregation gains**:

$$v^+ := \sum_{i\in[I]} v^i - v(m^f_\alpha),$$

quantifying the overall benefit of aggregation.

Figure E.3 presents the different indicators for $\gamma \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ across models m^f_1 with $f \in \{U, SM, P\}$ and $\alpha = 1$. Several general observations hold across the five instances.

First, as expected, the minimax operator consistently yields the highest minimal proportional savings $\alpha^\star$, regardless of the acceptability level. It also minimizes the dispersion of proportional savings δ. Second, for a given acceptability set, the utilitarian operator achieves the highest overall gains. As γ decreases, the gap between day-ahead and balancing prices widens, making access to the day-ahead market more advantageous. Consequently, aggregation becomes more beneficial for each agent, leading to higher global gains and $\alpha^\star$. However, as γ increases from 0.5 to 0.8, the dispersion of proportional savings grows under the utilitarian operator, while it decreases under the proportional one.

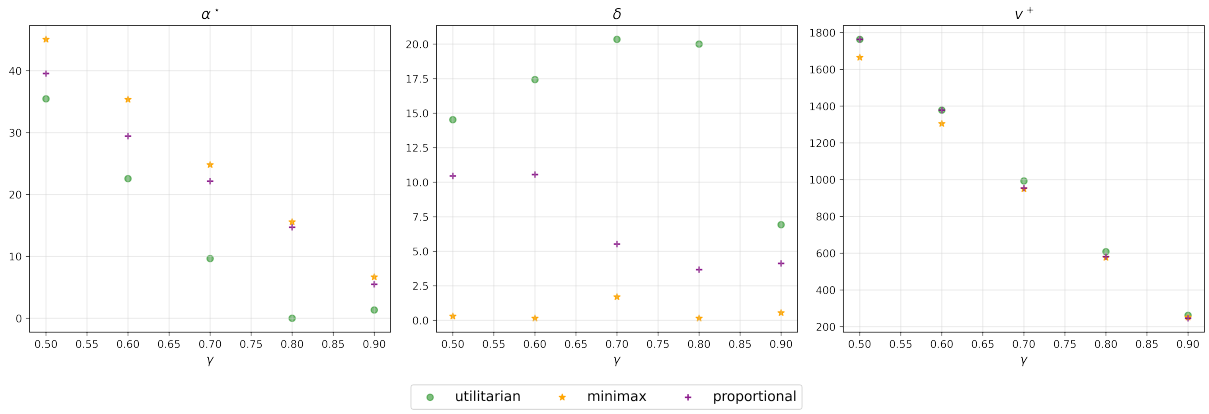


Figure E.3: Minimal proportional saving $\alpha^\star$, dispersion of proportional savings δ and global aggregation gains v^+ obtained by solving m^f_1 for $f \in \{U, SM, P\}$ for instances depending on the gap γ between day-ahead and balancing prices.

E.2.1 Testing the stochastic case on different scenario samples

We present additional tests for different instances of the problem described in Section 7.6.4. We consider instances where balancing prices $\{\boldsymbol{p^B_t}\}$ are random variables with uniform, independent distribution over $[0.3p^{DA}_t, 3p^{DA}_t]$. From this distribution, we randomly generate 20 samples of 50 scenarios. Prosumers' parameters and day-ahead market prices are taken from Tables 7.1 and 7.6.

For each scenario sample, we solve m^f_a with $f \in \{US, SMS\}$ and $a \in \{\emptyset, \mathbb{E}, (ic), (1), (a.s)\}$. As in Chapter B, we report the minimum proportional savings $\alpha^\star$, dispersion of proportional savings δ and global aggregation gains v^+ for each model. For each indicator, we compute the expectation,

standard deviation, minimum, and maximum values across the 20 instances and summarize the results in Tables E.2 and E.3.

	$\alpha^\star$				δ				v^+			
	average	std	min	max	average	std	min	max	average	std	min	max
$m_{\emptyset}^{US}$	-39.8	16	-75	-9.5	69	18.8	30.8	108.5	148.1	12.9	123.7	172.7
$m_{\mathbb{E}}^{US}$	0	0	0	0	27	2.9	21.4	31.4	147.7	13.1	123.7	171.1
$m_{(ic)}^{US}$	0	0	0	0	27	2.9	21.4	31.4	147.7	13.1	123.7	171.1
$m_{(1)}^{US}$	6.8	3.4	0	11.6	10.8	5.3	0	17	80.5	41.5	0	135.7
$m_{(a.s)}^{US}$	0.7	2.2	0	8.3	1.5	4.5	0	15.3	8.4	26.3	0	97.4

Table E.2: Average value, standard deviation, minimum and maximum values of the minimal proportional saving $\alpha^\star$, the dispersion of proportional savings δ and global aggregation gains v^+ obtained by solving the models with the risk-neutral utilitarian operator $\mathcal{F}_{I\times\Omega}^{US}$ over 20 different samplings of scenarios.

	$\alpha^\star$				δ				v^+			
	average	std	min	max	average	std	min	max	average	std	min	max
$m_{\emptyset}^{SMS}$	16.8	1.2	13.8	18.7	0.6	0.9	0	4.2	134.7	13	107.2	159.6
$m_{\mathbb{E}}^{SMS}$	16.8	1.2	13.8	18.7	0.6	0.9	0	4.2	134.7	13	107.2	159.6
$m_{(ic)}^{SMS}$	16.8	1.2	13.8	18.7	0.6	0.9	0	4.2	134.7	13	107.2	159.6
$m_{(1)}^{SMS}$	7.2	4.3	0	13.6	8.2	5.4	0	14	72.6	41.5	0	119.2
$m_{(a.s)}^{SMS}$	0.7	2.2	0	8.3	1.4	4.5	0	15.1	8.4	26.3	0	97.3

Table E.3: Average value, standard deviation, minimum and maximum values of the minimal proportional saving $\alpha^\star$, the dispersion of proportional savings δ and global aggregation gains v^+ obtained by solving the models with the risk-neutral scaled-minimax operator $\mathcal{F}_{I\times\Omega}^{SMS}$ over 20 different samplings of scenarios.

From these tables, we observe that although the values fluctuate between different samples, the comparison between models remains consistent. Notably, under the risk-neutral scaled-minimax operator $\mathcal{F}_{I\times\Omega}^{SMS}$, $\alpha^\star$ and δ almost do not fluctuate. Moreover, when comparing the utilitarian approach ($\mathcal{F}_{I\times\Omega}^{US}$) with the minimax approach ($\mathcal{F}_{I\times\Omega}^{SMS}$), the latter consistently yields the highest $\alpha^\star$ and minimal δ. Finally, increasing the level of acceptability reduces the dispersion of proportional savings but also leads to a decrease in global gains.

Conclusion and perspectives

Starting from a client study case of METRON, we modeled a joint production and energy planning problem as a Multistage Stochastic mixed-binary Linear Program (MSbLP) in Chapter 3. MSbLP are known for their complexity and the main focus of this thesis has been to elaborate efficient algorithms to solve them. We analyzed existing methods like MPC and SDP and introduced a heuristic leveraging cost-to-go approximations computed with SDDP. This served as the foundation for Part II, where we explored partial relaxations of integrality constraints to incorporate SDDP. In Chapter 4, we proposed an abstract framework combining SDDP with Branch-and-Bound (BB) techniques to solve MSbLP. This led to an exact algorithm in Chapter 5 that iteratively solves partial relaxations of an MSbLP that rely on subtrees structure.

Let us now discuss potential future research directions.

Smart and efficient growing strategies Our partial relaxation approach enforces integrality on a subtree while relaxing it elsewhere, approximating these regions with SDDP. We proposed various strategies to grow subtrees and construct policies with reasonable average costs. Numerical results demonstrated the potential of this approach, particularly for problems where deterministic approximations are overly optimistic and full integrality relaxation yields infeasible solutions. However, the random strategy outperformed the more theoretically sophisticated Integrality Gap (IG) strategy in both solution quality and computational time. Future work will focus on developing smarter growing strategies that outperform random approaches while ensuring convergence.

Partially relaxing information constraints In solving an MSbLP, the challenge is to account both for uncertainties and integrality constraints. In Part II, we prioritized uncertainty by relaxing integrality constraints. Alternatively, we propose, in future work, to prioritize integrality by relaxing information constraints instead. We can actually keep the structure of Algorithm 12, where partial relaxation rely on the structure of subtrees. This means enforcing all constraints on a subtree while approximating the remaining horizon as a deterministic problem. This approach results in a partial relaxation of information constraints, and the same iterative process from Algorithm 12 can be applied. Comparing the results of Algorithm 12 with those of this alternative procedure would provide insights into the relative importance of modeling uncertainty and integrality for a given problem.

Fairness in a multistage stochastic framework In Part III, we addressed fairness in aggregation problems by introducing *acceptability constraints*, ensuring all agents benefit from collaboration. These constraints enforce an order between aggregated and independent costs. While the static deterministic case relies on a total order on real numbers, the dynamic and stochastic cases require partial orders in higher dimensions. We proposed several partial orders

for multistage and stochastic settings. An interesting direction for future work is to identify partial orders that simultaneously suit both multistage and stochastic contexts.

Acronyms

PSMLB Problème Stochastic Multiétapes Linéaire en variables continues et Binaires. 3, 23, 24, 26, 29, 30

MPC Model Predictive Control. 3, 29, 30, 49, 56, 63, 64, 66, 68, 69, 72, 73, 110, 111, 112, 118, 119, 120, 122, 123, 124, 177

SDP Stochastic Dynamic Programming. 3, 27, 28, 29, 47, 48, 49, 56, 98, 177

SDDP Stochastic Dual Dynamic Programming. 3, 5, 24, 27, 28, 29, 30, 41, 44, 47, 48, 49, 51, 54, 56, 60, 61, 62, 63, 65, 66, 68, 70, 72, 73, 77, 87, 89, 92, 93, 94, 95, 97, 98, 99, 100, 101, 102, 103, 108, 114, 115, 116, 118, 122, 123, 124, 177

BB Branch-and-Bound. 3, 5, 30, 49, 77, 87, 90, 91, 94, 95, 103, 177

MSbLP Multistage Stochastic mixed-binary Linear Program. 5, 29, 42, 43, 44, 45, 48, 49, 51, 77, 78, 83, 85, 87, 88, 89, 95, 97, 98, 99, 100, 102, 103, 110, 119, 177

PV Solar photovoltaic. 14, 34

CBAM Carbon Border Adjustment Mechanism. 15, 35

RO Recherche Opérationnelle. 16, 17, 20

PLM Programme Linéaire en variables Mixtes. 20, 24, 25, 26

SSE Système de Stockage d'Énergie. 21, 22

PD Programmation Dynamique. 22, 23, 24, 26, 27

OR Operational Research. 36, 37

MILP Mixed-Integer Linear Program. 39, 44, 45, 83, 85, 94, 95, 100, 102, 109, 110, 120

ESS Energy Storage System. 40, 41, 52, 53, 54, 55, 68, 113, 114, 115

DP Dynamic Programming. 41, 43, 44, 45, 46, 47, 58, 59, 60, 63, 77, 84, 87, 98

MSLP Multistage Stochastic Linear Program. 41, 77, 87, 89, 93, 98, 99

MSiLP Multistage Stochastic mixed-Integer Linear Program. 42, 98, 99

IPCC Intergovernmental Panel on Climate Change. 52

IEA International Energy Agency. 52

TFDP Trajectory Following Dynamic Programming. 60, 61, 99

SDDiP Stochastic Dual Dynamic integer Programming. 60, 61, 99, 100

MIDAS Mixed Integer Dynamic Approximation Scheme. 61, 99

SLDP Stochastic Lipschitz Dynamic Programming. 61, 99

EV Expected Value. 63, 68, 70, 71, 72

PH Progressive-Hedging. 98, 99

SAA Sample Average Approximation. 103

PI Perfect Information. 110, 111, 115

EVPI Expected Value of Perfect Information. 111

IG Integrality Gap. 115, 116, 118, 121, 122, 123, 124, 129, 130, 177

SB Strong-Branching. 115, 116, 129, 130

Bibliography

[ACF22] Shabbir Ahmed, Filipe Goulart Cabral, and Bernardo Freitas Paulo da Costa. "Stochastic Lipschitz Dynamic Programming". In: *Mathematical Programming* 191.2 (Feb. 2022).

[AKY22] Nikolaos Argyris, Özlem Karsu, and Mirel Yavuz. "Fair Resource Allocation: Using Welfare-Based Dominance Constraints". In: *European Journal of Operational Research* 297.2 (Mar. 2022), pp. 560–578. (Visited on 12/01/2023).

[AL15] Benjamin Armbruster and James Luedtke. "Models and formulations for multivariate dominance-constrained stochastic programs". In: *IIE Transactions* 47.1 (2015), pp. 1–14.

[Alo+22] Àlex Alonso-Travesset et al. "Optimization Models under Uncertainty in Distributed Generation Systems: A Review". In: *Energies* 15.5 (Mar. 2022).

[Art+99] Philippe Artzner et al. "Coherent measures of risk". In: *Mathematical finance* 9.3 (1999), pp. 203–228.

[Bän+21] Kristian Bänsch et al. "Energy-Aware Decision Support Models in Production Environments: A Systematic Literature Review". In: *Computers & Industrial Engineering* 159 (Sept. 2021).

[Bar+06] R. Barth et al. "A Stochastic Unit-commitment Model for the Evaluation of the Impacts of Integration of Large Amounts of Intermittent Wind Power". In: *2006 International Conference on Probabilistic Methods Applied to Power Systems*. 2006 International Conference on Probabilistic Methods Applied to Power Systems. Stockholm, Sweden: IEEE, June 2006, pp. 1–8. URL: http://ieeexplore.ieee.org/document/4202207/ (visited on 10/21/2024).

[Ben62] J F Benders. "Partitioning Procedures for Solving Mixed-Variables Programming Problems". In: *Numerische Mathematik* (1962).

[BFT11] Dimitris Bertsimas, Vivek F. Farias, and Nikolaos Trichakis. "The Price of Fairness". In: *Operations Research* 59.1 (Feb. 2011), pp. 17–31. (Visited on 12/07/2023).

[BG16] Konstantin Biel and Christoph H. Glock. "Systematic Literature Review of Decision Support Models for Energy-Efficient Production Planning". In: *Computers & Industrial Engineering* 101 (Nov. 2016).

[Bie+18] Konstantin Biel et al. "Flow Shop Scheduling with Grid-Integrated Onsite Wind Power Using Stochastic MILP". In: *International Journal of Production Research* 56.5 (Mar. 2018).

[Bir85a] John R. Birge. "Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs". In: *Operations Research* 33.5 (Oct. 1985).

[Bir85b] John R. Birge. "Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs". In: *Operations Research* 33.5 (Oct. 1985), pp. 989–1007. URL: https://pubsonline.informs.org/doi/10.1287/opre.33.5.989 (visited on 10/21/2024).

[BL97] John Birge and François Louveaux. *Introduction to Stochastic Dynamic Programming*. Jan. 1997.

[Boh+20] Markus Bohlayer et al. "Energy-Intense Production-Inventory Planning with Participation in Sequential Energy Markets". In: *Applied Energy* 258 (Jan. 2020).

[CBS23] Margarita P. Castro, Merve Bodur, and Yongjia Song. *Markov Chain-based Policies for Multi-stage Stochastic Integer Linear Programming with an Application to Disaster Relief Logistics*. May 10, 2023. URL: http://arxiv.org/abs/2207.14779 (visited on 10/21/2024). Pre-published.

[CGS09] Miguel Carrión, Uwe Gotzes, and Rüdiger Schultz. "Risk Aversion for an Electricity Retailer with Second-Order Stochastic Dominance Constraints". In: *Computational Management Science* 6.2 (May 2009), pp. 233–250. (Visited on 06/22/2023).

[CH20] Simon Caton and Christian Haas. "Fairness in machine learning: A survey". In: *ACM Computing Surveys* (2020).

[CJA17] Andreia M. Carreiro, Humberto M. Jorge, and Carlos Henggeler Antunes. "Energy management systems aggregators: A literature survey". In: *Renewable and Sustainable Energy Reviews* 73 (2017), pp. 1160–1172. URL: https://www.sciencedirect.com/science/article/pii/S1364032117301776.

[COP16] COP21. *Paris Agreement*. 2016. URL: https://treaties.un.org/pages/ViewDetails.aspx?src=TREATY&mtdsg_no=XXVII-7-d&chapter=27&clang=_en (visited on 10/07/2024).

[CPO] CPOWER. URL: https://cpowerenergy.com/.

[DBL24] Maryam Daryalal, Merve Bodur, and James Luedtke. "Lagrangian Dual Decision Rules for Multistage Stochastic Mixed-Integer Programming". In: *Operations Research* 72.2 (2024).

[DR03] Darinka Dentcheva and Andrzej Ruszczynski. "Optimization with Stochastic Dominance Constraints". In: *SIAM Journal on Optimization* 14.2 (Jan. 2003), pp. 548–566. (Visited on 06/22/2023).

[DR09] Darinka Dentcheva and Andrzej Ruszczyński. "Optimization with multivariate stochastic dominance constraints". In: *Mathematical Programming* 117.1-2 (2009), pp. 111–127.

[DW16] Darinka Dentcheva and Eli Wolfhagen. "Two-stage optimization problems with multivariate stochastic order constraints". In: *Mathematics of Operations Research* 41.1 (2016), pp. 1–22.

[EUP16] EUPHEMIA. *EUPHEMIA Public Description PCR Market Coupling Algorithm*. 2016. URL: https://www.nemo-committee.eu/assets/files/euphemia-public-description.pdf (visited on 12/05/2023).

[EUR] EURO. *What is Operational Research?* URL: https://www.euro-online.org/web/pages/301/or-and-euro (visited on 10/09/2024).

[EUR15] EURELECTRIC. *Designing fair and equitable market rules for demand response aggregation*. 2015. URL: https://cdn.eurelectric.org/media/1923/0310_missing_links_paper_final_ml-2015-030-0155-01-e-h-D4D245B0.pdf.

[Fer+13] Thomé Fernanda et al. "Non-Convexities Representation on Hydrothermal Operation Planning Using SDDP". In: (2013).

[FL23] Maël Forcier and Vincent Leclère. "Trajectory Following Dynamic Programming Algorithms without Finite Support Assumptions". In: *Journal of Convex Analysis* (2023).

[FLG24] Zoé Fornier, Vincent Lèclere, and Dorian Grosso. "Joint production and energy supply planning of an industrial microgrid". In: *Energy Systems* (2024).

[FLP24] Zoé Fornier, Vincent Leclère, and Pierre Pinson. *Fairness by design in shared-energy allocation problems*. 2024. URL: https://arxiv.org/abs/2402.00471.

[FMH21] Mohammad Fattahi, Hadi Mosadegh, and Aliakbar Hasani. "Sustainable Planning in Mining Supply Chains with Renewable Energy Integration: A Real-Life Case Study". In: *Resources Policy* 74 (Dec. 2021).

[FR21] C Füllner and S Rebennack. "Stochastic Dual Dynamic Programming and Its Variants". In: (2021).

[FR22] Christian Füllner and Steffen Rebennack. "Non-Convex Nested Benders Decomposition". In: *Mathematical Programming* 196.1-2 (2022), pp. 987–1024. (Visited on 08/11/2023).

[FR23] Christian Füllner and STEFFEN Rebennack. "Stochastic Dual Dynamic Programming and Its Variants—A Review". In: *Preprint, available at http://www. optimization-online. org/DB_FILE/2021/01/8217. pdf* (2023).

[Fre+15] Lucas Freire et al. "A Hybrid MILP and Benders Decomposition Approach to Find the Nucleolus Quota Allocation for a Renewable Energy Portfolio". In: *IEEE Transactions on Power Systems* 30.6 (Nov. 2015), pp. 3265–3275. (Visited on 07/31/2023).

[FW18] Alireza Fazli Khalaf and Yong Wang. "Energy-Cost-Aware Flow Shop Scheduling Considering Intermittent Renewables, Energy Storage, and Real-Time Electricity Pricing". In: *International Journal of Energy Research* 42.12 (Oct. 2018).

[Geo+21] Ramy Georgious et al. "Review on Energy Storage Systems in Microgrids". In: *Electronics* 10.17 (Jan. 2021).

[GFJ16] Mehdi Golari, Neng Fan, and Tongdan Jin. "Multistage Stochastic Optimization for Production-Inventory Planning with Intermittent Renewable Energy". In: *Production and Operations Management* 26 (Sept. 2016).

[Gin21] Corrado Gini. "Measurement of Inequality of Incomes". In: *The Economic Journal* 31.121 (Mar. 1921), pp. 124–125.

[GKW23] Walter J. Gutjahr, Raimund M. Kovacevic, and David Wozabal. "Risk-Averse Bargaining in a Stochastic Optimization Context". In: *Manufacturing & Service Operations Management* 25.1 (Jan. 2023), pp. 323–340. (Visited on 07/31/2023).

[GLP15] Pierre Girardeau, Vincent Leclere, and Andrew B Philpott. "On the convergence of decomposition methods for multistage stochastic convex programs". In: *Mathematics of Operations Research* 40.1 (2015), pp. 130–145.

[GNS08] Ralf Gollmer, Frederike Neise, and Rüdiger Schultz. "Stochastic Programs with First-Order Dominance Constraints Induced by Mixed-Integer Linear Recourse". In: *SIAM Journal on Optimization* 19.2 (Jan. 2008), pp. 552–571. (Visited on 06/26/2023).

[HBF15] Ehsan Hajipour, Mokhtar Bozorg, and Mahmud Fotuhi-Firuzabad. "Stochastic Capacity Expansion Planning of Remote Microgrids With Wind Farms and Energy Storage". In: *IEEE Transactions on Sustainable Energy* 6.2 (Apr. 2015).

[Hey+19] Evelyn Heylen et al. "Fairness and Inequality in Power System Reliability: Summarizing Indices". In: *Electric Power Systems Research* 168 (Mar. 2019), pp. 313–323. (Visited on 12/01/2023).

[Hje+19] Martin N. Hjelmeland et al. "Nonconvex Medium-Term Hydropower Scheduling by Stochastic Dual Dynamic Integer Programming". In: *IEEE Transactions on Sustainable Energy* 10.1 (Jan. 2019), pp. 481–490. URL: https://ieeexplore.ieee.org/document/8289405/ (visited on 10/21/2024).

[HK10] Julia L. Higle and Karl G. Kempf. "Production Planning Under Supply and Demand Uncertainty: A Stochastic Programming Approach". In: *Stochastic Programming*. 2010.

[HLW01] Kjetil K. Haugen, Arne Løkketangen, and David L. Woodruff. "Progressive hedging as a meta-heuristic applied to stochastic lot-sizing". In: *European Journal of Operational Research* 132.1 (2001), pp. 116–122. URL: https://www.sciencedirect.com/science/article/pii/S0377221700001168.

[HPG18] Adam Hirsch, Yael Parag, and Josep Guerrero. *Microgrids A Review of Technologies, Key Drivers, and Outstanding Issues*. 2018.

[IEA24] IEA. *World Energy Investment*. 2024. URL: https://www.iea.org/reports/world-energy-investment-2024 (visited on 10/08/2024).

[Ier+02] M. G. Ierapetritou et al. "Cost Minimization in an Energy-Intensive Plant Using Mathematical Programming Approaches". In: *Industrial & Engineering Chemistry Research* 41.21 (Oct. 2002).

[Ins24] Energy Institute. *Statistical Review of World Energy*. 2024. URL: https://www.energyinst.org/statistical-review (visited on 10/08/2024).

[Inta] International Energy Adgency. *Global Energy Review 2021*. https://www.iea.org/reports/global-energy-review-2021. Online; Accessed: 2022-09-13.

[Intb] International Energy Adgency. *The cost of capital in clean energy transitions*. https://www.iea.org/articles/the-cost-of-capital-in-clean-energy-transitions. Online; Accessed: 2023-04-24.

[Intc] International Energy Adgency. *Tracking Industry 2021*. https://www.iea.org/reports/tracking-industry-2021. Online; Accessed: 2022-09-13.

[Intd] International Renewable Energy Adgency. *Renewable power generation costs in 2021*.

[IT14] Dan A. Iancu and Nikolaos Trichakis. "Fairness and Efficiency in Multiportfolio Optimization". In: *Operations Research* 62.6 (Dec. 2014), pp. 1285–1301. (Visited on 10/20/2022).

[Jab+17] Shahin Jabbari et al. "Fairness in reinforcement learning". In: *International conference on machine learning*. PMLR. 2017, pp. 1617–1626.

[Jor+24] Natalia Jorquera-Bravo et al. "Fair Energy Allocation for Collective Self-Consumption". In: *ha preprint ffhal-04409750f* (2024).

[Kon03] James Konow. "Which Is the Fairest One of All? A Positive Analysis of Justice Theories". In: *Journal of Economic Literature* 41.4 (Nov. 2003), pp. 1188–1239. (Visited on 08/04/2023).

[KS04] Daniel Kirschen and Goran Strbac. *Fundamentals of Power System Economics*. 1st ed. Wiley, Mar. 2004. (Visited on 10/09/2024).

[KS75] Ehud Kalai and Meir Smorodinsky. "Other solutions to Nash's bargaining problem". In: *Econometrica: Journal of the Econometric Society* (1975), pp. 513–518.

[KV12] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. 5th. Springer Publishing Company, Incorporated, 2012.

[Lan+10] Tian Lan et al. "An Axiomatic Theory of Fairness in Network Resource Allocation". In: *2010 Proceedings IEEE INFOCOM*. 2010, pp. 1–9.

[Li+17] Binbin Li et al. "Toward Net-Zero Carbon Manufacturing Operations: An Onsite Renewables Solution". In: *Journal of the Operational Research Society* 68.3 (Mar. 2017).

[LS19] Nils Löhndorf and Alexander Shapiro. "Modeling Time-Dependent Randomness in Stochastic Dual Dynamic Programming". In: *European Journal of Operational Research* 273.2 (Mar. 2019), pp. 650–661. URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221718306787 (visited on 10/21/2024).

[Man60] Alan S. Manne. "On the Job-Shop Scheduling Problem." In: *Operations Research* (1960).

[Mor+16] David R. Morrison et al. "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning". In: *Discrete Optimization* 19 (2016), pp. 79–102. URL: https://www.sciencedirect.com/science/article/pii/S1572528616000062.

[MP13] Joon-Yung Moon and Jinwoo Park. "Smart Production Scheduling with Time-Dependent and Machine-Dependent Electricity Cost by Considering Distributed Energy Resources and Energy Storage". In: *International Journal of Production Research* 52 (Dec. 2013).

[MP19] Fabio Moret and Pierre Pinson. "Energy Collectives: A Community and Fairness Based Approach to Future Electricity Markets". In: *IEEE Transactions on Power Systems* 34.5 (Sept. 2019), pp. 3994–4004. (Visited on 12/01/2023).

[Mut99] Abhinay Muthoo. "Bargaining theory with applications". In: *Cambridge University Press* (1999).

[Nas50] John F. Nash. "4. The Bargaining Problem". In: *The Essential John Nash*. Ed. by Sylvia Nasar. Princeton: Princeton University Press, 1950, pp. 37–48. (Visited on 08/04/2023).

[Nas53] John Nash. "Two-Person Cooperative Games". In: *Econometrica* 21.1 (Jan. 1953), p. 128. (Visited on 12/21/2023).

[Newa] New Energy and industrial technology Development Organization. https://home.kepco.co.kr/kepco/EN/main.do. Online; Accessed: 2022-09-13.

[Newb] New Energy and industrial technology Development Organization. https://appww1.infoc.nedo.go.jp/appww/index.html?lang=2. Online; Accessed: 2022-09-13.

[Oh22] Eunsung Oh. "Fair Virtual Energy Storage System Operation for Smart Energy Communities". In: *Sustainability* 14.15 (Aug. 2022), p. 9413. (Visited on 12/01/2023).

[OR94] Martin J. Osborne and Ariel Rubinstein. *A course in game theory*. electronic edition. Cambridge, USA: The MIT Press, 1994.

[Par14] Vilfredo Pareto. "Manual of political economy: a critical and variorum edition". In: *OUP Oxford* (2014).

[PG08] A.B. Philpott and Z. Guan. "On the Convergence of Stochastic Dual Dynamic Programming and Related Methods". In: *Operations Research Letters* 36.4 (July 2008), pp. 450–455. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167637708000308 (visited on 10/21/2024).

[Pha+19] An Pham et al. "A Multi-Site Production and Microgrid Planning Model for Net-Zero Energy Operations". In: *International Journal of Production Economics* 218 (Dec. 2019).

[PP85] M. V. F. Pereira and L. M. V. G. Pinto. "Stochastic Optimization of a Multireservoir Hydroelectric System: A Decomposition Approach". In: *Water Resources Research* 21.6 (June 1985), pp. 779–792. URL: https://agupubs.onlinelibrary.wiley.com/doi/10.1029/WR021i006p00779 (visited on 10/21/2024).

[PP91] M. V. F. Pereira and L. M. V. G. Pinto. "Multi-Stage Stochastic Optimization Applied to Energy Planning". In: *Mathematical Programming* 52.1-3 (May 1991).

[PS16] Yael Parag and Benjamin K. Sovacool. "Electricity Market Design for the Prosumer Era". In: *Nature Energy* 1.4 (Mar. 2016), p. 16032. (Visited on 10/09/2024).

[PWB20] A. B. Philpott, F. Wahid, and J. F. Bonnans. "MIDAS: A Mixed Integer Dynamic Approximation Scheme". In: *Mathematical Programming* 181.1 (May 2020).

[QGK21] Franco Quezada, Céline Gicquel, and Safia Kedad-Sidhoum. "Combining Polyhedral Approaches and Stochastic Dual Dynamic Integer Programming for Solving the Uncapacitated Lot-Sizing Problem under Uncertainty". In: (2021).

[QGK23] Franco Quezada, Céline Gicquel, and Safia Kedad-Sidhoum. "A Stochastic Dual Dynamic Integer Programming Based Approach for Remanufacturing Planning under Uncertainty". In: *International Journal of Production Research* 61.17 (2023), pp. 5992–6012. (Visited on 08/14/2023).

[Raw71] John Rawls. "A Theory of Justice". In: *The Belknap press of Harvard University Press* (1971). Eleventh printing, 1981.

[RFJ20] José Luis Ruiz Duarte, Neng Fan, and Tongdan Jin. "Multi-Process Production Scheduling with Variable Renewable Integration and Demand Response". In: *European Journal of Operational Research* 281.1 (Feb. 2020).

[RM21] Paolo Renna and Sergio Materi. "A Literature Review of Energy Efficiency and Sustainability in Manufacturing Systems". In: *Applied Sciences* 11.16 (Jan. 2021), p. 7366. (Visited on 08/18/2022).

[ROA] ROADEF. *Présentation de la ROADEF*. URL: https://roadef.org/app/pages/roadef-qui-sommes-nous.

[RS61] H. Raiffa and R. Schlaifer. *Applied Statistical Decision Theory*. Studies in Managerial Economics, 1961.

[RTK22] Yves Rychener, Bahar Taskesen, and Daniel Kuhn. "Metrizing Fairness". In: *arXiv preprint arXiv:2205.15049* (2022).

[RU+00] R Tyrrell Rockafellar, Stanislav Uryasev, et al. "Optimization of conditional value-at-risk". In: *Journal of risk* 2 (2000), pp. 21–42.

[RvdL24] Ward Romeijnders and Niels van der Laan. "Benders Decomposition with Scaled Cuts for Multistage Stochastic Mixed-Integer Programs". In: (2024).

[RW91] Roger R.T. Rockafellar and J.-B. West. "Scenarios and Policy Aggregation in Optimization Under Uncertainty". In: *Mathematics of Operations Research* 16.1 (1991).

[SDR14] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory, Second Edition*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611973433.

[Ser23] Bonn Kleiford Seranilla. "On the applications of Stochastic Dyal Dynamic Programming". In: (2023).

[Sha06] Alexander Shapiro. "On Complexity of Multistage Stochastic Programs". In: *Operations Research Letters* 34.1 (Jan. 2006).

[Sha52] Lloyd S. Shapley. "A Value for N-Person Games". In: *RAND Corporation* (1952).

[SML19] Hossein Shahandeh, Farough Motamed Nasab, and Zukui Li. "Multistage Stochastic Capacity Planning of Partially Upgraded Bitumen Production with Hybrid Solution Method". In: *Optimization and Engineering* 20.4 (Dec. 2019).

[Sus17] World Business Council for Sustainable Development. *Microgrids for commercial and industrial companies*. 2017. URL: https://www.wbcsd.org/resources/microgrids-for-commercial-and-industrial-companies/ (visited on 10/08/2024).

[SW69] R. M. Van Slyke and Roger Wets. "L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming". In: *SIAM Journal on Applied Mathematics* 17.4 (1969), pp. 638–663. URL: http://www.jstor.org/stable/2099310.

[TAC22] Simon Thevenin, Yossiri Adulyasak, and Jean-François Cordeau. "LStochastic Dual Dynamic Programming for Multiechelon Lot Sizing with Component Substitution". In: *INFORMS Jounral on Computing* 34.6 (2022).

[TKW00] Samer Takriti, Benedikt Krasenbrink, and Lilian S.-Y. Wu. "Incorporating Fuel Constraints and Electricity Spot Prices into the Stochastic Unit Commitment Problem". In: *Operations Research* 48.2 (Apr. 2000), pp. 268–280. URL: https:

//pubsonline.informs.org/doi/10.1287/opre.48.2.268.12379 (visited on 10/21/2024).

[Tsi+21] Stamatis Tsianikas et al. "A Storage Expansion Planning Framework Using Reinforcement Learning and Simulation-Based Optimization". In: *Applied Energy* 290 (May 2021).

[Wan+19] Jianxiao Wang et al. "Incentivizing Distributed Energy Resource Aggregation in Energy and Capacity Markets: An Energy Sharing Scheme and Mechanism Design". In: *Applied Energy* 252 (Oct. 2019), p. 113471. (Visited on 12/01/2023).

[Wan+20] Richard Wang et al. "Renewable Energy Microgrids: Economic Evaluation and Decision Making for Government Policies to Contribute to Affordable and Clean Energy". In: *Applied Energy* 274 (Sept. 2020), p. 115287. (Visited on 10/08/2024).

[WMG20] Shasha Wang, Scott J. Mason, and Harsha Gangammanavar. "Stochastic Optimization for Flow-Shop Scheduling with on-Site Renewable Energy Generation Using a Case in the United States". In: *Computers & Industrial Engineering* 149 (Nov. 2020).

[XH23] V Xinying Chen and J.N Hooker. "A Guide to Formulating Equity and Fairness in an Optimization Model". In: *Annals of Operations Research* 326 (Nov. 2023), pp. 581–619.

[Xia+20] Xiangsheng Xiao et al. "Large-Scale Aggregation of Prosumers toward Strategic Bidding in Joint Energy and Regulation Markets". In: *Applied Energy* 271 (2020), p. 115159.

[Yan+23] Xiyun Yang et al. "Day-Ahead and Real-Time Market Bidding and Scheduling Strategy for Wind Power Participation Based on Shared Energy Storage". In: *Electric Power Systems Research* 214 (Jan. 2023), p. 108903. (Visited on 12/05/2023).

[YHS21] Yu Yang, Guoqiang Hu, and Costas J Spanos. "Optimal sharing and fair cost allocation of community energy storage". In: *IEEE Transactions on Smart Grid* 12.5 (2021), pp. 4185–4194.

[ZAS19] Jikai Zou, Shabbir Ahmed, and Xu Andy Sun. "Stochastic Dual Dynamic Integer Programming". In: *Mathematical Programming* 175.1-2 (May 2019).