UNIVERSITÉ DE MONTRÉAL

MÉTHODES NUMÉRIQUES APPLIQUÉES À LA PROGRAMMATION DYNAMIQUE STOCHASTIQUE POUR LA GESTION D'UN SYSTÈME HYDROÉLECTRIQUE

KENJY DEMEESTER
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
AOÛT 2017

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

MÉTHODES NUMÉRIQUES APPLIQUÉES À LA PROGRAMMATION DYNAMIQUE STOCHASTIQUE POUR LA GESTION D'UN SYSTÈME HYDROÉLECTRIQUE

présenté par : DEMEESTER Kenjy
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. GAMACHE Michel, Ph. D., président
M. ORBAN Dominique, Doctorat, membre et directeur de recherche
M. CÔTÉ Pascal, Ph. D., membre et codirecteur de recherche
M. BASTIN Fabian, Doctorat, membre externe

DÉDICACE

À ma famille ...

REMERCIEMENTS

Je tiens à remercier en premier lieu mes directeurs pour m'avoir encadré tout au long de cette expérience. Ces quelques mots ne suffiront pas à décrire toute la reconnaissance que j'ai pour vous. Vous m'avez transmis avec passion une partie de votre domaine d'expertise et vos judicieux conseils m'ont permis de me dépasser à chaque étape du projet. Merci d'avoir cru en moi pendant ces deux années.

J'aimerais ensuite remercier Monsieur Michel Gamache d'avoir accepté de présider le jury, ainsi que Monsieur Fabian Bastin d'accepter d'en faire partie.

Je remercie profondément le Conseil de Recherche en Sciences Naturelles et en Génie (CRSNG), le Fonds de Recherche du Québec - Natures et Technologies (FRQNT) et Rio Tinto de m'avoir octroyé un financement pendant cette maîtrise.

Je ne serais jamais arrivé où je suis à présent sans toutes ces rencontres qui m'ont guidé à travers mes années d'étude. Je tiens à saluer mes premiers collègues du groupe de recherche du DRAME de l'École de technologie supérieure. Les gars vous avez été pour moi une véritable source d'inspiration. Vous m'avez initié à cet incroyable domaine qu'est la recherche et je vous serai toujours infiniment reconnaissant. Durant ces deux années de maîtrise au sein du GERAD, j'ai eu la chance de rencontrer d'incroyables personnes. Tous à la fois aussi talentueux que généreux, j'ai pris plaisir à vous côtoyer chaque jour durant nos cours, nos lunchs et aussi nos pauses café ! J'espère que nos chemins se recroiseront bientôt.

Un grand merci pour l'accueil reçu dans la région du Saguenay. Tout d'abord, aux membres du groupe de ressources hydriques de Rio Tinto pour leur amabilité et leur professionnalisme. J'adresse une mention particulière à Richard Arsenault pour ces nombreux conseils lors de la rédaction, mais aussi à Catherine Bergeron pour m'avoir hébergé et intégré à la vie au Saguenay !

Finalement, je tiens à conclure ces remerciements par le soutien de ma famille. Vous avez toujours été une source de motivation pour moi. Je suis désolé de ne pas pouvoir passer autant de temps que je le souhaiterais avec vous. Je vous aime de tout mon coeur.

RÉSUMÉ

La Programmation Dynamique Stochastique (PDS) est une méthode couramment utilisée pour la gestion de petits systèmes hydroélectriques. La PDS décompose le problème principal en une succession de petits problèmes non linéaires à résoudre. Néanmoins, son utilisation peut être problématique dans un contexte opérationnel en raison du temps de calcul important requis pour résoudre tous ces sous-problèmes. L'objectif du projet de recherche consiste donc à améliorer sa vitesse d'exécution en proposant une nouvelle méthode de résolution des sous-problèmes générés par la PDS. Cette approche combine deux méthodes d'optimisation non linéaire, soit une méthode de Programmation Linéaire Séquentielle (PLS) pour la phase de récursion de la PDS et une méthode de points intérieurs pour la simulation.

La méthode proposée sera comparée à trois grandes familles de méthodes d'optimisation non linéaires avec contraintes : les méthodes du lagrangien augmenté, les méthodes séquentielles et les méthodes de points intérieurs. Pour évaluer convenablement l'efficacité de chacune de ces méthodes, les logiciels d'optimisation les plus établis sont utilisés dans cette analyse. Par contre, presque aucune implémentation de la méthode de la PLS n'a été trouvée dans la littérature. Une partie de ce travail sera consacrée à l'implémentation de cet algorithme.

Les résultats numériques sont obtenus par l'application de la PDS sur le système hydro-électrique du Saguenay-Lac-Saint-Jean, opéré par la compagnie Rio Tinto. Étant donné la tendance linéaire des fonctions de production sur une majeure partie de leur domaine, il est possible d'approcher efficacement ces dernières par une approximation linéaire par morceaux. Ainsi, une approche hybride combinant la PLS et le logiciel IPOPT s'avère une solution efficace autant pour le temps de calcul que la qualité de la solution.

ABSTRACT

Stochastic Dynamic Programming (SDP) is a common approach used for hydropower systems management. SDP decomposes the main problem by a sequence of small nonlinear subproblems. Nevertheless, its application can be difficult in an operational context because of the excessive computation time required to solve all its subproblems. The objective of this study consists of increasing SDP speed by introducing a new methodology to solve these subproblems. This approach combines two nonlinear methods, the Sequential Linear Programming (SLP) is used for the computation of policy and an interior points method for the simulation.

The method proposed will be evaluated to the main nonlinear constraints methods: augmented Lagrangian, sequential methods, and interior points methods. To adequately compare their efficiency, state-of-the-art solvers are used in the analysis. Few established implementations of SLP algorithm have been found in the literature. Thus, a part of this work will be an implementation of this method.

Numerical results are obtained by SDP application on the Saguenay-Lac-Saint-Jean hydropower system, operated by the company Rio Tinto. Considering that hydropower functions exhibit a linear behavior over large portions of the domain, it is possible to accurately approximate the function by a piecewise linear approximation. In that case, an hybrid approach combining SLP and the software IPOPT is proved to be efficient to reduce the computation time with no lack of quality solution.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

PDS Programmation Dynamique Stochastique
PLS Programmation Linéaire Séquentielle
PQS Programmation Quadratique Séquentielle
SLSJ Saguenay-Lac-Saint-Jean

CHAPITRE 1 INTRODUCTION

L'utilisation des énergies renouvelables prend une grande importance dans les enjeux de notre société moderne. En effet, de nombreuses technologies émergent dans le but de réduire l'utilisation des énergies fossiles et notre empreinte écologique. L'hydroélectricité possède de multiples avantages en tant que source d'énergie propre. Entre autres, cette technologie offre une certaine souplesse pour répondre aux fluctuations de la demande d'électricité, grâce à l'utilisation de barrages permettant de conserver l'eau. La gestion d'un système de production hydroélectrique consiste à déterminer une certaine quantité d'eau à utiliser pour subvenir en tout temps aux besoins des consommateurs. Cependant, ces décisions peuvent s'avérer être difficiles lorsqu'un système hydroélectrique est composé de plusieurs centrales. Les modèles d'optimisation jouent alors un rôle important en tant qu'outils d'aide à la décision afin d'assister les ingénieurs dans leur planification des opérations quotidiennes.

1.1 Définitions et concepts de base

Un système hydroélectrique est constitué d'une ou plusieurs centrales installées le long des rivières. La production d'hydroélectricité est générée par des groupes de turbo-alternateurs. Ces groupes permettent de convertir l'énergie de l'eau en énergie mécanique, par la rotation de leurs palmes, puis en énergie électrique via l'alternateur. On distingue deux types de complexes hydroélectriques : la centrale à réservoir et la centrale au fil de l'eau.

Une centrale à réservoir est composée de trois éléments : une centrale, un barrage et un évacuateur de crue. L'alimentation de la centrale en eau est régularisée à l'aide d'un barrage situé en amont. Ce barrage permet de conserver un certain volume d'eau pour un usage futur et de créer une hauteur de chute. La hauteur de chute correspond à la différence entre le niveau de l'eau du réservoir en amont et en aval. Cette hauteur exerce une certaine pression sur les palmes des turbines favorisant la génération hydroélectrique. Par conséquent, plus la hauteur d'eau est importante, plus les turbines seront efficaces. Il est donc primordial de conserver une bonne hauteur de chute tout au long de l'année afin de maximiser la production d'énergie. Cette gestion comporte cependant des risques. Si le niveau d'eau dans le réservoir est trop élevé, le risque de déverser de l'eau augmente. Bien évidemment, cette eau n'a aucune valeur énergétique, elle est donc considérée comme perdue.

Le second type de centrale est au fil de l'eau. Ces centrales ont la particularité de disposer d'un réservoir dont la contenance est fixe ou dont la variation est marginale par rapport aux

intervalles dans lesquels les décisions de gestion sont prises. Par conséquent, aucune décision de gestion n'est requise pour ces ouvrages puisque le débit turbiné et déversé ne varie qu'en fonction du débit de la rivière en amont.

La gestion d'un système hydroélectrique peut être difficile pour plusieurs raisons. Premièrement, plusieurs sources d'incertitude sont à prendre en compte. En effet, les apports naturels d'eau dans le système sont considérés incertains. Dans une région spécifique, couramment appelée un bassin versant, la quantité d'eau apportée par la pluie ou la neige ruisselle selon plusieurs processus hydrologiques. De ce fait, les apports en eau sont généralement beaucoup plus importants au printemps, notamment par la fonte de la neige, et plus faibles durant l'été. En considérant les conditions météorologiques et les caractéristiques du bassin, il est possible de simuler les débits des rivières grâce à l'utilisation de modèles hydrologiques. Néanmoins, ces prévisions ne sont jugées fiables que pour une courte durée.

D'autre part, la modélisation d'un problème de gestion de réservoir nécessite de prendre en compte les fonctions de production des centrales. Ces informations sont généralement modélisées par des fonctions non linéaires, ce qui ajoute une difficulté supplémentaire au processus de résolution.

L'un des défis les plus importants des ingénieurs responsables des opérations d'un système hydroélectrique est donc de déterminer la meilleure gestion en dépit de ces incertitudes. L'optimisation stochastique joue alors un rôle très important. On distingue trois types de problèmes d'optimisation dans la gestion hydroélectrique : court terme, moyen terme et long terme.

Le problème à court terme est utilisé pour déterminer la répartition de la génération d'hydro-électricité entre plusieurs groupes de turbo-alternateur. Ces modèles sont notamment utilisés dans le but de minimiser les pertes de production ou de maximiser l'efficacité des groupes turbo-alternateurs. L'horizon de cette planification est généralement court, de l'ordre d'une à plusieurs semaines, et le pas de temps peut généralement varier de l'heure à la journée.

Les modèles de gestion à moyen terme consistent à déterminer un volume d'eau à soutirer de chacun des réservoirs d'un système hydroélectrique sur un horizon annuel. Ces modèles doivent être capables de subvenir à différents besoins tout au long de l'année tout en respectant les règles d'opération du système. Le pas de temps dans ce type de modèle est de l'ordre de quelques jours voire hebdomadaire.

Finalement, les modèles à long terme sont utilisés comme outils de conception afin d'évaluer la rentabilité de nouveaux projets hydroélectriques. Ce genre de modèle n'est pas utilisé en opération. Son horizon est de plusieurs années avec un pas de temps mensuel voire saisonnier.

Les travaux présentés dans ce mémoire sont spécifiques au modèle de gestion à moyen terme.

1.2 Éléments de la problématique

Une quantité importante d'électricité est nécessaire pour le procédé de fabrication d'aluminium. Pour subvenir aux demandes énergétiques des alumineries, la compagnie Rio Tinto, producteur d'aluminium, dispose des droits d'eau lui permettant d'opérer le système hydroélectrique du Saguenay-Lac-Saint-Jean (SLSJ). Ce système est composé de 5 centrales hydroélectriques, soit 3 centrales à réservoir et 2 centrales au fil de l'eau, installées le long des rivières Péribonka et Saguenay. Ce système a une capacité de production annuelle de 3100MW. La figure 1.1 présente un schéma du système du SLSJ.

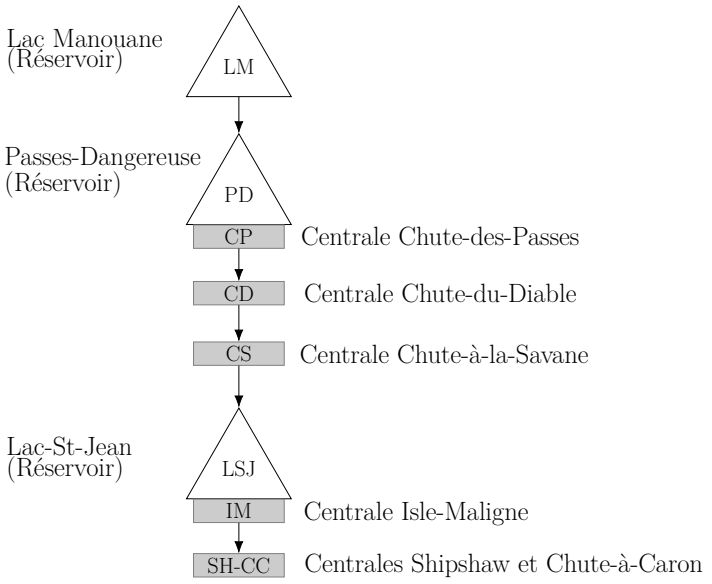

Figure 1.1 Réseau hydroélectrique du Saguenay-Lac-Saint-Jean.

Rio Tinto a récemment développé un outil d'aide à la décision pour la gestion à moyen terme de ce système. Le logiciel est basé sur l'algorithme de la Programmation Dynamique Stochastique (PDS). La PDS est une méthode très populaire pour calculer une politique (ou règle) de gestion. Elle consiste à décomposer le problème principal en une série de petits problèmes non linéaires à résoudre. Cela implique à la fois une discrétisation de l'horizon temporel du problème et du domaine de décision.

Pour chaque période de planification, une politique de gestion détermine le volume d'eau à soutirer à chacune des centrales. Ces décisions dépendent de l'état du système, caractérisé notamment par le volume d'eau présent dans les réservoirs.

Étant donné les incertitudes liées à la gestion, cette politique doit maximiser un certain objectif tout en étant la plus sécuritaire possible. Dans le cas d'étude du SLSJ, l'objectif de la politique de gestion est de satisfaire la demande énergétique des alumineries à moindre coût. Si cette demande n'est pas respectée, le déficit d'électricité peut être alors importé du système d'Hydro-Québec. Dans le cas contraire, les surplus sont vendus grâce à des contrats.

Le principal problème relié à l'utilisation du logiciel de PDS développé par Rio Tinto, est le temps de calcul. En effet, pour un système à trois réservoirs comme celui du SLSJ, le nombre de sous-problèmes à résoudre est significativement grand, à cause de la discrétisation, et ils doivent être résolus de façon efficace.

1.3 Objectifs de recherche

L'objectif principal de ce mémoire est d'étudier différentes stratégies dans le but d'améliorer le temps de calcul du logiciel de la PDS développé par Rio Tinto.

Celui-ci est directement influencé par la rapidité de résolution de chacun des sous-problèmes. Présentement, une méthode de points intérieurs non linéaire est utilisée pour réaliser cette tâche. Toutefois, la méthode n'a jamais été comparée à d'autres approches. Comme il sera montré dans le chapitre 2, de nombreuses autres méthodes peuvent être employées pour la résolution des sous-problèmes. Actuellement, très peu de travaux sur la PDS argumentent le choix de la méthode de résolution des sous-problèmes. L'idée est donc dans un premier temps de répertorier chacune de ces méthodes, puis de déterminer lesquelles sont les plus appropriées pour résoudre les sous-problèmes de PDS dans notre cas du SLSJ.

D'autre part, on propose d'analyser l'utilisation d'une approximation du problème lors de l'évaluation de la politique de gestion. Sachant que l'usage de contraintes non linéaires complique la résolution des sous-problèmes, une approximation linéaire par morceaux des contraintes sera effectuée grâce à l'emploi d'hyperplans tangents.

Pour valider ces stratégies, une comparaison sera effectuée lors d'un processus de simulation de la politique de gestion. En utilisant un ensemble de scénarios d'apport naturel, la simulation évalue le comportement de la politique au cours d'une gestion annuelle du système. La stratégie retenue correspondra donc à celle qui sera parvenue à définir la politique de gestion le plus rapidement, tout en conservant une bonne performance de gestion.

1.4 Plan du mémoire

Ce mémoire est rédigé par article. Le chapitre 2 présente une revue de littérature du développement de la méthode de la PDS appliquée à la gestion des réservoirs ainsi qu'une revue des méthodes d'optimisation non linéaire avec contraintes. Le chapitre 3 introduit la démarche suivie dans l'article présenté au chapitre 4. Le chapitre 5 est consacré à une discussion générale sur le projet. Pour terminer, le chapitre 6 expose les conclusions et recommandations des travaux.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre présente une revue de littérature sur les notions abordées dans ce mémoire. Dans un premier temps, une étude du développement de la méthode de Programmation Dynamique Stochastique (PDS) est présentée dans le contexte de la gestion des réservoirs. Ensuite, une revue des méthodes d'optimisation non linéaire est introduite afin d'accomplir les objectifs de recherche.

2.1 Méthode de programmation dynamique stochastique

La gestion d'un système de réservoirs est associée à un problème d'optimisation séquentiel où l'on cherche à déterminer une quantité d'eau à relâcher pour chacune des périodes d'opération.

Dans le cadre de la gestion à moyen terme, le problème est défini sur un horizon de planification annuel, décomposé en T périodes. Pour la formulation mathématique, on appelle $\mathbf{s}_t := (s_{1,t}, s_{2,t}, \ldots, s_{n,t})$, le vecteur représentant le volume d'eau contenu dans un système à n réservoirs au début de la période t. La quantité d'eau à soutirer de chaque réservoir lors d'une période t est représentée par le vecteur $\mathbf{u}_t := (u_{1,t}, u_{2,t}, \ldots, u_{n,t})$. Finalement, les apports naturels à chaque réservoir lors d'une période t sont définis par le vecteur $\mathbf{q}_t := (q_{1,t}, q_{2,t}, \ldots, q_{n,t})$. Ces apports étant jugés aléatoires, une variable hydrologique $\mathbf{h}_t$ est généralement utilisée comme indicateur des apports à venir. La politique de gestion est alors un vecteur $\Pi := (\pi_1, \pi_2, \ldots, \pi_T)$ où chacune des fonctions π_t retourne la décision $\mathbf{u}_t$ à appliquer au système selon les états du système $\mathbf{s}_t$ et $\mathbf{h}_t$.

L'algorithme de la PDS, proposé par Bellman (1957), est l'une des méthodes de référence pour la gestion de réservoirs (Labadie, 2004). Sa popularité peut être expliquée par sa simplicité d'utilisation ainsi que la prise en charge de l'aspect stochastique et non linéaire du problème.

De façon récursive, la PDS cherche à estimer chaque fonction π_t grâce à l'approximation d'une fonction de coût futur espéré $\tilde{J}_t$, appelée aussi valeur de l'eau et définie par

$$\pi_t(\mathbf{s}_t, \mathbf{h}_t) \in \arg\min_{\mathbf{u}_t} \quad \tilde{J}_t(\mathbf{s}_t, \mathbf{h}_t). \tag{2.1}$$

La fonction de la valeur de l'eau $\tilde{J}_t(\mathbf{s}_t, \mathbf{h}_t)$ est la valeur optimale de :

$$\underset{\mathbf{u}_t}{\text{Minimiser}} \quad \underset{\mathbf{q}_t|\mathbf{h}_t}{\mathbb{E}} \left\{ G_t(\mathbf{s}_t, \mathbf{u}_t) + \tilde{J}_{t+1}(\mathbf{s}_{t+1}, \mathbf{h}_{t+1}) \right\}, \tag{2.2a}$$

$$\text{sujet à} \quad \mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{q}_t - C\mathbf{u}_t, \tag{2.2b}$$

$$s_{\min} \leq \mathbf{s}_{t+1} \leq s_{\max}, \tag{2.2c}$$

$$0 \leq \mathbf{u}_t \leq u_{\max}. \tag{2.2d}$$

où G_t est la fonction de coût à l'instant t et $\mathbb{E}$ est l'opérateur d'espérance. Cette espérance est généralement conditionnée sur la variable hydrologique.

La contrainte (2.2b) correspond au bilan hydrique du système de réservoirs. Afin d'assurer la continuité des états à chacune des périodes, le volume d'eau des réservoirs est mis à jour en fonction du volume d'eau relâché et des apports naturels parvenus durant la période. La matrice C correspond quant à elle a une matrice de connectivité liant les réservoirs entre eux.

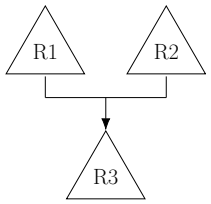

Figure 2.1 Exemple d'un système composé de trois réservoirs

Pour visualiser cette équation, on considère un petit système composé de 3 réservoirs comme l'illustre la Figure 2.1. Dans cette situation, (2.2b) est alors défini par le système suivant

$$\begin{bmatrix} s_{1,t+1} \\ s_{2,t+1} \\ s_{3,t+1} \end{bmatrix} = \begin{bmatrix} s_{1,t} \\ s_{2,t} \\ s_{3,t} \end{bmatrix} + \begin{bmatrix} q_{1,t} \\ q_{2,t} \\ q_{3,t} \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_{1,t} \\ u_{2,t} \\ u_{3,t} \end{bmatrix}.$$

Afin d'estimer la valeur de la fonction $\tilde{J}_t$ sur l'ensemble de son domaine, (2.2) est résolu pour un ensemble fini de points grâce à une discrétisation du domaine des variables $\mathbf{s}_t$ et $\mathbf{h}_t$. La fonction est ensuite modélisée grâce à des techniques d'interpolation. Par conséquent, le nombre de sous-problèmes à résoudre augmente exponentiellement en fonction du nombre de variables d'état. Cette limitation est connue dans la littérature sous le nom de la malédiction de la dimensionalité (Bellman, 1957).

En dépit de cette caractéristique, l'utilisation de la PDS est très répandue dans la littérature. On y distingue deux catégories de travaux : l'application de la PDS à de petits systèmes, i.e., moins de 4 réservoirs (Kim et al., 2007; Côté et al., 2011; Anvari et al., 2014; Etkin et al., 2015) et à des grands systèmes (Goor et al., 2010; Rougé et Tilmant, 2016; Raso et al., 2017). Dans le cas de notre étude, nous nous concentrons sur l'application de la PDS à des systèmes de petite taille.

Les travaux d'application de la PDS à des systèmes de petite taille se concentrent sur l'amélioration de la modélisation du processus stochastique. Cela permet, entre autres, de représenter convenablement les corrélations temporelles entre les apports hydriques, essentielles pour une bonne gestion des crues printanières. Par exemple, Kelman et al. (1990) proposent d'utiliser directement les scénarios afin de conditionner la règle de gestion. Turgeon (2005) développe une variable hydrologique basée sur un modèle auto-régressif à plusieurs retards. Desreumaux et al. (2014) démontrent que pour des systèmes fortement influencés par un couvert neigeux, l'utilisation de variables exogènes telles que l'équivalent en eau de la neige améliore la performance de la PDS. Ces améliorations jouent un rôle important sur la performance de la PDS pour l'évaluation de la politique d'opération.

Côté et Leconte (2015) comparent différentes méthodes pour la gestion du système SLSJ. Les résultats confirment que la PDS assure une gestion adéquate du système. Cependant, le processus s'avère coûteux en temps de calcul dans un contexte opérationnel.

Très peu d'études ont été proposées afin d'accélérer le temps de calcul de la PDS pour des petits systèmes. Parmi ces travaux, certains sont reliés aux techniques d'interpolation utilisées pour estimer la valeur de l'eau. Entre autre, il a été démontré qu'il est possible de réduire la taille de la grille de points de discrétisation tout en maintenant la performance de la politique de gestion par l'utilisation des polynômes d'Hermite (Foufoula-Georgiou et Kitandis, 1988) ou de spline cubique (Johnson et al., 1993). Récemment, des stratégies de parallélisation de la PDS ont été proposées afin de répartir la résolution des sous-problèmes sur plusieurs processeurs (Zhang et al., 2013; Li et al., 2014). Bien que ces améliorations aient été exploitées dans le logiciel développé par Rio Tinto, elles ne parviennent pas à obtenir un temps de résolution acceptable pour une utilisation opérationnelle.

Dans ce contexte, sachant que le temps de calcul provient majoritairement de la résolution des sous-problèmes, le choix de l'algorithme d'optimisation pourrait fortement influencer la rapidité d'exécution de la PDS. Comme le présentera la prochaine section, il existe de nombreuses méthodes d'optimisation non linéaire. Les travaux de ce mémoire proposent d'analyser et de comparer plusieurs de ces méthodes dans le but de réduire le temps de calcul de la PDS, sans perte significative de la qualité de la politique d'opération.

2.2 Méthodes d'optimisation non linéaire avec contraintes

En toute généralité, un sous-problème non linéaire de la PDS peut être défini de la façon suivante :

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{Minimiser}} \quad & f(x), && \text{(2.3a)}\\ \text{sujet à} \quad & g(x) \geq 0, && \text{(2.3b)}\\ & h(x) = 0. && \text{(2.3c)} \end{aligned}$$

où $f\colon \mathbb{R}^n \to \mathbb{R}$, $g\colon \mathbb{R}^n \to \mathbb{R}^{m_I}$ et $h\colon \mathbb{R}^n \to \mathbb{R}^{m_E}$ sont des fonctions de classe $\mathcal{C}^2$. On note $\nabla f(x) \in \mathbb{R}^n$ le gradient de f en x, et $\nabla g(x) \in \mathbb{R}^{m_I \times n}$ et $\nabla h(x) \in \mathbb{R}^{m_E \times n}$ sont les matrices jacobiennes de g et h en x, respectivement.

Soit x un point satisfaisant les contraintes de (2.3), on dit que la condition de qualification d'indépendance linéaire (LICQ) est vérifiée si l'ensemble

$$\{\nabla g_i(x) | i \in \mathcal{A}(x)\} \cup \{\nabla h_i(x) | i = 1, \ldots, m_E\},$$

est un ensemble de vecteurs linéairement indépendants, où $\mathcal{A}(x) = \{i | g_i(x) = 0\}$ correspond à l'ensemble des contraintes d'inégalité actives en x.

Si x^* est un minimum local de (2.3), et que la LICQ est vérifiée en x^*, alors il existe des vecteurs $\lambda \in \mathbb{R}^{m_I}$ et $\mu \in \mathbb{R}^{m_E}$ tel que

$$\begin{aligned} \nabla f(x^*) - \nabla g(x^*)^T\lambda - \nabla h(x^*)^T\mu &= 0, && \text{(2.4a)}\\ \lambda^T g(x^*) &= 0, && \text{(2.4b)}\\ g(x^*) &\geq 0, && \text{(2.4c)}\\ h(x^*) &= 0, && \text{(2.4d)}\\ \lambda &\geq 0. && \text{(2.4e)} \end{aligned}$$

où λ et μ sont aussi connus sous le nom de multiplicateurs de Lagrange associés respectivement aux contraintes d'inégalité et d'égalité. Ces conditions sont couramment appelées les conditions de Karush-Kuhn-Tucker (KKT).

La majorité des méthodes pour résoudre (2.3) suivent un processus itératif dans lequel l'itéré courant x_k est mis à jour à chaque itération par un pas Δx tel que $x_{k+1} = x_k + \Delta x$. Dans le cadre de la programmation non linéaire, le pas est généralement obtenu par la résolution de sous-problèmes d'optimisation, ou systèmes linéaires, plus facile à résoudre. On distingue

dans la littérature trois familles de méthodes pour la résolution de problèmes non linéaires avec contraintes (Bazaraa et al., 2005; Nocedal et Wright, 2006) : les méthodes du lagrangien augmenté, les méthodes séquentielles et les méthodes de points intérieurs. Chacune de ces méthodes est décrite dans les prochaines sections.

2.2.1 Méthodes de lagrangien augmenté

L'une des premières idées proposées pour la résolution de (2.3) consiste à résoudre à chaque itération un sous-problème d'optimisation sans contrainte dans lequel le carré de la violation des contraintes est inclus dans la fonction objectif :

$$\underset{x\in\mathbb{R}^n}{\text{Minimiser}} \qquad f(x) + \rho_k \sum_{i=0}^{m_E} h_i(x)^2 + \rho_k \sum_{i=0}^{m_I} \max\{-g_i(x), 0\}^2, \tag{2.5}$$

où $\rho_k > 0$ correspond à un paramètre de pénalité à l'itération k.

Cette méthode fut proposée par Courant (1943) sous le nom de méthode par pénalisation quadratique.

Bien que cette méthode ait d'importantes limitations, dues entre autres à un mauvais conditionnement, elle fut à la base de la très populaire méthode du lagrangien augmenté initialement proposée par Hestenes (1969) et Powell (1969).

On définit une fonction lagrangienne par

$$\mathcal{L}(x, \mu) := f(x) + \mu^T h(x), \tag{2.6}$$

où μ est une estimation des multiplicateurs de Lagrange associés aux contraintes d'égalité. Les contraintes d'inégalité sont généralement redéfinies comme contraintes d'égalité par l'ajout de variables d'écart ($s \in \mathbb{R}^{m_I}$) tel que :

$$g(x) - s = 0, \quad s \geq 0, \tag{2.7}$$

La fonction du lagrangien augmenté est définie par

$$\Phi(x, \mu, \rho) := \mathcal{L}(x, \mu) + \tfrac{1}{2}\rho\|h(x)\|_2^2, \tag{2.8}$$

La méthode du lagrangien augmenté est alors composée de deux étapes couramment appelées l'itération externe et interne. Dans l'itération externe, la méthode formule à chaque itération

le sous-problème suivant :

$$\min_{x\in\mathbb{R}^n} \Phi(x, \mu_k, \rho_k) \quad \text{sujet à} \quad x_l \leq x \leq x_u, \tag{2.9}$$

où μ_k, ρ_k sont fixes à chaque itération et les bornes x_l et x_u était implicitement contenues dans les contraintes $g(x)$.

Le problème (2.9) est alors résolu dans l'itération interne en utilisant une méthode d'optimisation pour des problèmes avec des contraintes de bornes ou des contraintes linéaires. Une fois le sous-problème interne résolu, les coefficients μ_k et ρ_k sont mis à jour et le cycle se répète jusqu'à satisfaction des critères d'arrêt du problème principal. L'algorithme 1 résume les étapes importantes de la méthode du lagrangien augmenté.

Algorithme 1 Méthode du lagrangien augmenté

Étape 0. Initialisation :
Choisir un point initial (x_0, μ_0), paramètre de pénalité initial $\rho_0 > 0$
Initialiser le compteur d'itération $k = 0$

Étape 1. Vérification des conditions d'optimalité :
Si le critère de convergence est satisfait, alors STOP

Étape 2. Recherche d'une nouvelle solution x_{k+1} :
Résoudre le problème (2.9)

Étape 3. Mise à jour des paramètres :
Définir des nouveaux multiplicateurs de Lagrange μ_{k+1} et coefficient de pénalité ρ_{k+1}
Incrémenter le compteur $k \leftarrow k + 1$ et retourner à l'étape 1

Les logiciels MINOS (Murtagh et Saunders, 1978, 2003) et LANCELOT (Conn et al., 1992; Gould et al., 2003) sont considérés comme les implémentations les plus établies de la méthode du lagrangien augmenté. Au lieu de résoudre à chaque itération (2.9), MINOS a la particularité de définir un problème avec des contraintes linéaires. LANCELOT conserve quant à lui la non-linéarité des contraintes. Le sous-problème interne est alors formulé tel (2.9) et résolu. Par conséquent, MINOS est plus efficace lorsque le problème est composé de contraintes linéaires (Bongartz et Toint, 1997).

2.2.2 Optimisation linéaire et quadratique séquentielle

Les méthodes séquentielles ont la particularité de résoudre (2.3) par une succession de sous-problèmes linéaires ou quadratiques issus d'une approximation du problème principal.

Cette philosophie est introduite par Griffith et Stewart (1961) du groupe de recherche et

développement de la compagnie pétrolière Shell. À cette époque, Griffith propose de résoudre (2.3) par la méthode de Programmation Linéaire Séquentielle (PLS). La méthode PLS consiste à minimiser une suite de sous-problèmes linéaires définis par l'approximation de Taylor d'ordre 1 du problème principal autour de l'itéré courant x_k dans une région de confiance de rayon Δ_k. La région de confiance permet d'avoir un certain contrôle sur la linéarisation du modèle. Tout d'abord, elle permet de garantir qu'une solution du sous-problème linéaire est finie. Dans le cas où le sous-problème linéarisé est proche de (2.3), la région de confiance permet d'agrandir le domaine de linéarisation pour favoriser la convergence de l'algorithme. Dans le cas contraire, ce domaine est alors rétréci pour éviter de trop grands écarts entre les deux modèles.

Peu de temps après, Wilson (1963) propose, durant ses travaux de doctorat, de résoudre (2.3) par la méthode de Programmation Quadratique Séquentielle (PQS). En gardant le même principe que la méthode PLS, la méthode PQS minimise une suite de sous-problèmes quadratiques cette fois-ci définis par l'approximation de Taylor d'ordre 2.

Il existe deux approches différentes pour résoudre ces problèmes : une recherche linéaire ou une région de confiance. Étant donné les similitudes avec la méthode PLS, on introduit dans un premier temps la résolution de la méthode PQS par région de confiance. À l'itération k, la méthode PQS par région de confiance obtient un pas Δx en résolvant le sous-problème quadratique

$$\begin{aligned}
\underset{\Delta x \in \mathbb{R}^n}{\text{Minimiser}} \quad & f_k + \nabla f_k^T \Delta x + \tfrac{1}{2} \Delta x^T \nabla_{xx} \mathcal{L}_k \Delta x, && \text{(2.10a)} \\
\text{sujet à} \quad & g_k + \nabla g_k \Delta x \geq 0, && \text{(2.10b)} \\
& h_k + \nabla h_k \Delta x = 0, && \text{(2.10c)} \\
& \|\Delta x\|_\infty \leq \Delta_k. && \text{(2.10d)}
\end{aligned}$$

où $\nabla_{xx} \mathcal{L}_k$ représente la matrice hessienne du lagrangien.

La solution est alors évaluée afin de déterminer si le pas est accepté pour la résolution du problème principal. Si le pas est accepté, alors le nouvel itéré est obtenu par $x_{k+1} := x_k + \Delta x$ et le rayon de confiance peut être agrandi. Dans le cas contraire, le pas est rejeté et le rayon est diminué. Il se peut qu'il soit trop coûteux d'évaluer $\nabla_{xx} \mathcal{L}_k$. Une stratégie consiste à utiliser une approximation symétrique et définie positive (B_k) du hessien par des méthodes quasi-Newton telles que BFGS ou SR1 (Nocedal et Wright, 2006, Chapitre 6). B_k est mise à jour à chaque fois qu'un pas est accepté. L'algorithme 2 résume les étapes principales de la méthode PQS avec une région de confiance.

Algorithme 2 Méthode PQS avec une région de confiance

Étape 0. Initialisation :
Choisir un point initial (x_0, λ_0) et un rayon de la région de confiance $\Delta_0 > 0$
Initialiser le compteur d'itération $k = 0$

Étape 1. Vérification des conditions d'optimalité :
Si le critère de convergence est satisfait, alors STOP

Étape 2. Recherche d'un pas Δx :
Résoudre le problème (2.10)

Étape 3. Mise à jour de la région de confiance :
Évaluer le rapport entre la prédiction du modèle quadratique et modèle principale.
Si le rapport est suffisamment élevé :
    Mettre à jour les multiplicateurs de Lagrange (λ_k,μ_k) et poser $x_{k+1} := x_k + \Delta x$
    Augmenter le rayon de la région de confiance Δ_k
    Si $\nabla_{xx}\mathcal{L}_k$ est approximé :
        Mettre à jour l'approximation B_k
Sinon :
    Diminuer le rayon de la région de confiance Δ_k
Incrémenter le compteur $k \leftarrow k + 1$ et retourner à l'étape 1

La méthode PQS avec recherche linéaire diffère de la méthode de région confiance au niveau de la formulation du sous-problème. Celle-ci utilise les conditions d'optimalité du premier ordre (KKT) pour formuler un système d'équations non linéaires. Ce système est alors résolu par la méthode de Newton pour déterminer un pas. Avant de mettre à jour l'itéré x_k, la méthode réalise une recherche linéaire, telle que la recherche linéaire d'Armijo, pour déterminer une longueur de pas α_k afin de garantir une convergence. La solution est alors mise à jour par $x_{k+1} := x_k + \alpha_k \Delta x$.

En pratique, la méthode PQS a été utilisée avec succès pour la résolution de problèmes de grande taille. En général, les méthodes PQS sont moins coûteuses en nombre d'évaluations de fonction que les méthodes du lagrangien augmenté, grâce à l'utilisation d'une approximation du problème. Les logiciels les plus connus de la PQS sont FilterSQP (Fletcher et Leyffer, 2002) et SNOPT (Gill et al., 2005). FilterSQP suit une approche par région de confiance en utilisant un filtre pour accepter le pas. Quant à SNOPT, l'implémentation suit une approche par recherche linéaire où le sous-problème quadratique est défini par les conditions d'optimalité et une recherche linéaire est réalisée pour évaluer la taille du pas.

Finalement, Fletcher et de la Maza (1989) proposent de combiner l'utilisation des sous-problèmes linéaires et quadratiques dans la méthode Programmation Linéaire Quadratique Séquentielle (PLQS). L'idée générale de cette méthode est d'estimer l'ensemble des contraintes

actives, par la résolution d'un sous-problème linéaire, pour n'avoir qu'un problème quadratique avec contraintes d'égalité, plus facile à résoudre. Ces deux sous-problèmes sont formulés dans deux régions de confiance distinctes. Le pas obtenu par PLQS est alors un compromis entre les pas obtenus dans les deux sous-problèmes. Cette méthode est implémentée dans la librairie KNITRO par Byrd et al. (2003).

2.2.3 Méthodes de points intérieurs

On termine cette revue des méthodes d'optimisation non linéaires par la présentation des méthodes de points intérieurs grandement utilisées de nos jours pour résoudre (2.3). Les premières idées de ces méthodes ont été proposées par Frisch (1955). La méthode de points intérieurs résout le problème principal par une succession de sous-problèmes dans lesquels des barrières logarithmiques sont utilisées pour traiter la violation des contraintes. Le sous-problème logarithmique est alors formulé par

$$\underset{x\in\mathbb{R}^n, s\in\mathbb{R}^{m_I}}{\text{Minimiser}} \quad f(x) - \rho_k \sum_{i=0}^{m_I} \log(s_i), \tag{2.11a}$$

$$\text{sujet à} \quad g(x) - s = 0, \tag{2.11b}$$

$$h(x) = 0. \tag{2.11c}$$

où s est le vecteur des variables d'écart et $\rho_k > 0$ est le paramètre de barrière.

Tout comme la méthode du SQP, (2.11) peut être résolue par une approche de région de confiance ou de recherche linéaire. On propose cette fois-ci d'introduire dans un premier temps l'approche par recherche linéaire. Comme il a été évoqué dans la section 2.2.2, la formulation des sous-problèmes par recherche linéaire est issue des conditions d'optimalité de premier ordre. Les conditions de KKT de (2.11) sont alors de la forme suivante :

$$\nabla f(x) - \nabla g(x)^T\lambda - \nabla h(x)^T\mu = 0, \tag{2.12a}$$

$$S\lambda - \rho e = 0, \tag{2.12b}$$

$$g(x) - s = 0, \tag{2.12c}$$

$$h(x) = 0, \tag{2.12d}$$

$$\lambda \geq 0. \tag{2.12e}$$

où $e = [1, ..., 1]^T$ et $S = \text{diag}(s_1, ..., s_{m_I})$ est une matrice diagonale des variables d'écart.

L'approche consiste à résoudre (2.12) en appliquant la méthode de Newton pour une séquence de ρ_k convergeant vers zéro. Le système est alors donné par

$$\begin{bmatrix} \nabla_{xx}\mathcal{L} & 0 & \nabla h(x)^T & \nabla g(x)^T \\ 0 & \lambda & 0 & S \\ \nabla h(x) & 0 & 0 & 0 \\ \nabla g(x) & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta \mu \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} \nabla f(x) - \nabla g(x)^T\lambda - \nabla h(x)^T\mu \\ S\lambda - \rho e \\ h(x) \\ g(x) - s \end{bmatrix}, \tag{2.13}$$

Le système (2.13) est couramment appelé dans la littérature le système primal-dual. La matrice du système est généralement de grande taille et très creuse. De ce fait, l'utilisation de méthodes de factorisation performantes est primordiale pour résoudre efficacement le système afin d'obtenir les directions : $\Delta x, \Delta s, \Delta\mu$ et $\Delta\lambda$.

Ensuite, une recherche linéaire est effectuée pour calculer les longueurs de pas α_s, α_λ et mettre à jour les variables du problème par

$$x_{k+1} := x_k + \alpha_s \Delta x \qquad s_{k+1} := s_k + \alpha_s \Delta s, \tag{2.14a}$$

$$\mu_{k+1} := \mu_k + \alpha_\lambda \Delta \mu \qquad \lambda_{k+1} := \lambda_k + \alpha_\lambda \Delta \lambda, \tag{2.14b}$$

Comme pour PQS, il est possible d'utiliser une approximation B_k au lieu d'utiliser la matrice hessienne $\nabla_{xx}\mathcal{L}$. L'algorithme est alors résumé comme suit

Algorithme 3 Méthode de points intérieurs avec une recherche linéaire

Étape 0. Initialisation :
Choisir un point initial $(x_0, s_0, \mu_0, \lambda_0)$ et le paramètre pénalité de barrière ρ_0.
Initialiser le compteur d'itération $k = 0$

Étape 1. Vérification des conditions d'optimalité :
Si le critère de convergence est satisfait, alors STOP

Étape 2. Recherche du nouvel itéré :
Résoudre le problème (2.13) pour trouver les directions $\Delta x, \Delta s, \Delta\mu$ et $\Delta\lambda$.
Recherche des longueurs de pas α_s, α_λ

Étape 3. Mise à jour des paramètres :
Mettre à jour les $x_{k+1}, s_{k+1}, \mu_{k+1}, \lambda_{k+1}$ par (2.14)
Si $\nabla_{xx}\mathcal{L}_k$ est approximé :
    Mettre à jour l'approximation B_k
Définir le nouveau paramètre de barrière ρ_{k+1}
Incrémenter le compteur $k \leftarrow k + 1$ et retourner à l'étape 1

Les logiciels les plus connus ayant implémenté cette méthode sont IPOPT (Wächter et Biegler, 2006) et KNITRO (Waltz et al., 2006).

Dans la résolution par région de confiance, (2.11) est résolu par une méthode PQS spécifique à la structure du problème. Le sous-problème quadratique cherche à obtenir les pas $\Delta x, \Delta s$ en résolvant le problème suivant

$$\begin{aligned}
\underset{\Delta x \in \mathbb{R}^n, \Delta s \in \mathbb{R}^{m_I}}{\text{Minimiser}} \quad & f_k + \nabla f_k^T \Delta x + \tfrac{1}{2} \Delta x^T \nabla_{xx} \mathcal{L}_k \Delta x - \rho e^T S_k^{-1} \Delta s + \tfrac{1}{2} \Delta s^T S_k^{-1} Z_k \Delta s, && \text{(2.15a)} \\
\text{sujet à} \quad & g_k - s + \nabla g_k \Delta x - \Delta s = 0, && \text{(2.15b)} \\
& h_k + \nabla h_k \Delta x = 0, && \text{(2.15c)} \\
& \|\Delta x, S^{-1} \Delta s\|_\infty \leq \Delta_k. && \text{(2.15d)}
\end{aligned}$$

Une fois les pas obtenus, l'algorithme suit le même processus que la méthode PQS par région de confiance, où le rayon de confiance, les multiplicateurs de Lagrange et le paramètre de pénalité sont mis à jour. Cette approche est implémentée dans le second algorithme de points intérieurs du logiciel KNITRO (Byrd et al., 1999).

2.3 Remarques sur l'utilisation des méthodes non linéaires dans le contexte de gestion de réservoirs

Labadie (2004) rapporte dans sa revue des méthodes de gestion de réservoirs, l'utilisation de plusieurs algorithmes d'optimisation non linéaire tels que la PLS, la PQS et le lagrangien augmenté. Cependant, les méthodes de gestion diffèrent de la PDS et les problèmes à résoudre sont généralement de plus grande taille. Parmi ces exemples, le logiciel MINOS ressort dans de nombreux travaux (Tejada-Guibert et al., 1990; Peng et Buras, 2000; Barros et al., 2003).

Comme il a été discuté dans la section précédente, certaines implémentations avantagent beaucoup un type de formulation du problème. Par exemple, MINOS est conçu pour être plus performant lorsque les contraintes sont linéaires. Par ailleurs, l'évaluation de la fonction objectif des sous-problèmes nécessite un certain temps de calcul à cause de l'utilisation de splines. Les méthodes séquentielles devraient alors avoir un certain avantage, car elles nécessitent en général moins d'évaluations de la fonction objectif.

Par conséquent, il est difficile d'estimer à l'avance quelles seront les implémentations les plus performantes pour la résolution des sous-problèmes de PDS.

CHAPITRE 3 ORGANISATION DU MÉMOIRE

L'objectif principal de ce mémoire est d'améliorer la rapidité d'exécution de l'algorithme de Programmation Dynamique Stochastique (PDS) pour la gestion du système hydrique du Saguenay-Lac-Saint-Jean. Comme il a été discuté dans les chapitres précédents, l'utilisation de la PDS dans un contexte opérationnel est coûteuse en temps de calcul, en raison du nombre important de sous-problèmes d'optimisation à résoudre pour l'évaluation de la politique de gestion.

En considérant les différentes méthodes d'optimisation non linéaire avec contraintes présentées dans le chapitre 2, le chapitre 4 présente une étude de la performance de la PDS, en fonction du choix de l'algorithme d'optimisation l'assistant au cours de son processus. Les logiciels LANCELOT, MINOS, SNOPT, IPOPT et KNITRO ont été retenus pour présenter leurs méthodes respectives. Étant donné la formulation des sous-problèmes, il est possible que l'algorithme de Programmation Linéaire Séquentielle (PLS) soit un bon choix d'algorithme non linéaire pour assister la résolution de la PDS. Or, peu d'implémentation de cette méthode ont été trouvées dans la littérature. Une implémentation de cet algorithme est présentée au chapitre 4. Tous ces logiciels sont alors comparés dans un banc d'essai sur deux formulations d'un ensemble de sous-problèmes de la PDS. La première de ces formulations considère la non-linéarité des courbes de production des centrales, alors que la seconde approche ces fonctions par une approximation linéaire par morceaux. À l'issue de ces résultats, une étude plus approfondie est réalisée sur le comportement de certains logiciels au cours de la résolution de la PDS dans un contexte opérationnel. Ces travaux ont été publiés dans un cahier du GERAD (G-2017-64) et un article a été soumis à la revue *Optimization and Engineering*. La synthèse des résultats est présentée au chapitre 5 et une conclusion au chapitre 6.

CHAPITRE 4 ARTICLE 1: NUMERICAL METHODS FOR STOCHASTIC DYNAMIC PROGRAMMING WITH APPLICATION TO HYDROPOWER OPTIMIZATION

Cet article a été soumis à la revue *Optimization and Engineering*. L'article est actuellement en processus de révision. Un rapport technique a été publié à la revue suivante: P. Côté, K. Demeester, D. Orban et M. Towhidi, Numerical Methods for Stochastic Dynamic Programming with Application to Hydropower Optimization, Cahiers du GERAD, (G-2017-64).

Abstract: Stochastic Dynamic Programming (SDP) is a powerful approach applicable to nonconvex and stochastic stagewise problems. We investigate the impact of the formulation of the subproblems and of the choice of optimization method used to solve them on the overall performance of SDP in the context of hydropower reservoir management. We report numerical results on real systems and compare a number of state-of-the-art solvers. In one set of tests, subproblems feature nonlinear hydropower production constraints, while in a second set, the latter are approximated with linear constraints. Our results show that, when linear hydropower constraints are acceptable during policy computation, a Sequential Linear Programming (SLP) approach requires the lowest number of function evaluations while being the most robust in terms of number of subproblems solved successfully. On the other hand, IPOPT exhibits the best performance during the simulation phase in the presence of nonlinear hydropower constraints, where SLP is less reliable. A combination of SLP for policy computation using linear hydropower constraints with IPOPT in the simulation phase using nonlinear hydropower constraints results in an improvement of the average annual cumulative cost and reduces the total run time by a factor of about 3.3 compared to IPOPT alone using nonlinear hydropower constraints during both phases.

Key Words: hydropower optimization, stochastic dynamic programming, sequential linear programming

4.1 Introduction

A common problem in multi-reservoir systems consists in determining the optimal volume of water to release from each reservoir at each time step in order to maximize water use over a certain time horizon. Our study focuses on the mid term management problem which defines the time horizon as one year. The problem is difficult for two reasons: Natural inflows to the reservoirs cannot be forecast in advance and the process may require solving numerous

nonlinear and nonconvex problems to obtain reliable solution.

Many optimization methods were designed to solve this problem (Labadie, 2004). When a hydropower system is composed of few reservoirs, say at most four, the Stochastic Dynamic Programming (SDP) algorithm is one of the most powerful methods for dealing with stochasticity and nonlinearity (Turgeon, 2006; Mujumdar et Nirmala, 2007; Anvari et al., 2014; Davidsen et al., 2014; Desreumaux et al., 2014). The Rio Tinto system is composed of five power houses and three reservoirs, and is managed by SDP (Côté et Leconte, 2015). For larger systems, others approaches have been proposed in the literature. Stochastic Dual Dynamic Programming (SDDP) (Pereira et Pinto, 1985; Tilmant et al., 2002) is similar to SDP except that the cost-to-go is approximated using Benders cuts. Carpentier et al. (2013) and Zéphyr et al. (2014) propose the Progressive Hedging Algorithm (PHA), which decomposes the problem by scenarios instead of decomposing it by time period. Major advantages of SDDP and PHA are that they do not require discretizing the state space and that the subproblems are linear. In our case, we use SDP so as to be able to exploit the nonlinearity and nonconvexity of the problem.

SDP searches an optimal operating policy for the system by maximizing the expected benefit. At each period, the expected future benefit function is estimated from the current period until the end of the horizon using a reverse-time procedure. This function is evaluated by discretizing the decision domain and solving a sequence of small (possibly nonlinear) subproblems. Therefore, the overall SDP computation time depends significantly on the time spent to solve each subproblem. For background material on SDP, we refer the reader to (Bertsekas, 2005).

To the best of our knowledge, the choice of algorithm used to solve SDP subproblems is not discussed in the literature. In this paper, we compare the efficiency of optimization methods to solve SDP subproblems. Our test cases arise from the management of hydropower plants owned and operated by Rio Tinto, a metal and mining corporation, with the aim of powering its aluminium smelters. Our main objective is to decrease SDP computation time with no significant loss of accuracy. We assess and compare state-of-the-art optimization solvers on two formulations of the reservoir management problem. The first formulation features nonlinear hydropower production constraints, while in the second, those nonlinear constraints are approximated by a set of linear constraints. We illustrate the advantages of the linearly-constrained formulation with our own implementation of a Sequential Linear Programming (SLP) solver. Our results show that when using linear hydropower constraints, active-set optimization methods, including sequential linear and quadratic optimization methods, are the most robust in the policy computation phase and require the fewest objective evaluations. In

particular, our SLP implementation dominates all other solvers tested, including commercial solvers. During the simulation phase, where nonlinear hydropower constraints are desirable, SLP is less reliable and interior-point methods, including the open-source IPOPT package, are the fastest and most robust. We determine that combinations of solvers yield the best results. In particular, using MINOS, of the augmented-Lagrangian family, during the policy computation phase with linear hydropower constraints combined with IPOPT during the simulation phase with nonlinear hydropower constraints yields the lowest run time and second best average annual cumulative cost (AACC). Combining SLP and IPOPT similarly yields the best AACC and a total run time improvement of about 3.3 compared to using IPOPT in both phases with nonlinear hydropower constraints.

The paper is organized as follows. Section 4.2 summarizes the SDP approach in the context of hydropower management. In Section 4.3, we describe the case study of the Saguenay-Lac-Saint-Jean (SLSJ) hydroelectric power system operated by Rio Tinto. Section 4.4 presents an overview of nonlinear methods and a detailed description of the SLP process. Finally, numerical results and discussions are presented in Section 4.5.

4.2 Multireservoir management optimization

In hydropower reservoir management, operators attempt to determine the best use of water in a multi-reservoir system considering future uncertainty, such as hydrological events. Decisions occur over a finite time horizon and must be tradeoffs between the present benefit and the expected future benefit associated to the water stored at the end of the period. The stagewise optimization problem is formulated naturally as a Stochastic Dynamic Programming Problem (SDP). Bertsekas (2005) describes dynamic programming as a general approach for sequential optimization under uncertainty. A basic dynamic problem is composed of two kinds of variables: state and control variables. In a system with n reservoirs, state variables are the vector of reservoir storage levels at the beginning of period t: $\mathbf{s}_t := (s_{1,t}, s_{2,t}, \ldots, s_{n,t})$, and the control variables are the vector of water releases, or discharge, at the beginning of period t: $\mathbf{u}_t := (u_{1,t}, u_{2,t}, \ldots, u_{n,t})$.

The state variables are updated from period t to $t+1$ using mass balance conditions stated as the dynamic law

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{q}_t - \mathbf{C}\,\mathbf{u}_t, \tag{4.1}$$

where $\mathbf{q}_t := (q_{1,t}, q_{2,t}, \ldots, q_{n,t})$ are the natural inflows to the reservoirs during the period t and C is a connectivity matrix that describes the topology of the system. All diagonal elements of $\mathbf{C}$ are equal to $+1$ and the off-diagonal element $\mathbf{C}_{i,j}$ is set to -1 if reservoir i is

directly downstream of reservoir j. All other elements $\mathbf{C}_{i,j}$ are set to zero. Note that $\mathbf{C}$ is not symmetric. The law (4.1) implicitly assumes that the time period is such that all the water released from a reservoir at time t reaches the reservoirs located immediately downstream at time $t+1$.

The objective function is the sum of a terminal time cost $J_{T+1}(\mathbf{s}_{T+1})$ with stagewise benefit functions $G_t(\mathbf{s}_t, \mathbf{u}_t)$. The latter are mainly defined by the way the hydropower is used, including maximizing revenue from selling energy, minimizing energy purchase for sustaining a load, and minimizing the risk of flooding.

We thus consider the reservoir management problem formulated as

$$\begin{aligned}
\underset{\mathbf{u}_t}{\text{minimize}} \quad & \sum_{t=1}^{T} G_t(\mathbf{s}_t, \mathbf{u}_t) + J_{T+1}(\mathbf{s}_{T+1}) && && \text{(4.2a)}\\
\text{subject to} \quad & \mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{q}_t - \mathbf{C}\,\mathbf{u}_t, && t = 1, \ldots, T, && \text{(4.2b)}\\
& s_{\min} \leq \mathbf{s}_t \leq s_{\max}, && t = 1, \ldots, T+1, && \text{(4.2c)}\\
& 0 \leq \mathbf{u}_t \leq u_{\max}, && t = 1, \ldots, T, && \text{(4.2d)}
\end{aligned}$$

where $s_{\min} \in \mathbb{R}^n$, $s_{\max} \in \mathbb{R}^n$ are given by reservoir limitations, and $u_{\max} \in \mathbb{R}^n$ is the maximum capacity of water discharge for each reservoir.

The water value function at time $t = 1, \ldots, T$ is defined according to the **?** principle of dynamic programming as

$$\begin{aligned}
J_t(\mathbf{s}_t) := \underset{\mathbf{u}_t}{\text{minimize}} \quad & G_t(\mathbf{s}_t, \mathbf{u}_t) + J_{t+1}(\mathbf{s}_{t+1}), && \text{(4.3a)}\\
\text{subject to} \quad & \mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{q}_t - \mathbf{C}\,\mathbf{u}_t, && \text{(4.3b)}\\
& s_{\min} \leq \mathbf{s}_{t+1} \leq s_{\max}, && \text{(4.3c)}\\
& 0 \leq \mathbf{u}_t \leq u_{\max}. && \text{(4.3d)}
\end{aligned}$$

The notation in (4.3) means that $J_t(\mathbf{s}_t)$ is the optimal value of the constrained optimization problem. In stochastic dynamic programming, it is customary to call *policy* the optimal control law corresponding to a given state at a given stage. More precisely, a policy is a vector $\Pi := (\pi_1, \ldots, \pi_T)$ where each function π_t is defined as

$$\pi_t(\mathbf{s}_t) \in \arg\min_{\mathbf{u}_t} \ (4.3).$$

Note that each evaluation of J_t requires the solution of (4.3) at time steps $t, \ldots, T$. The SDP algorithm naturally proceeds backwards, first solving (4.3) at time T, i.e., with objective

function $J_{T+1}(\mathbf{s}_{T+1})$, and recursively until time t. In many applications (Côté et al., 2011; Turgeon, 2005; Zhao et al., 2012), J_t is sampled on a grid Ω contained in the box $[\mathbf{s}_{\min}, \mathbf{s}_{\max}]$. Interpolation techniques then allow us to approximate J_t outside of Ω. Various interpolation techniques could be used, e.g., cubic splines (Johnson et al., 1993), which also provide approximations of first and second derivatives.

The natural inflows $\mathbf{q}_t$ are random parameters that are often strongly correlated over time and must be estimated using statistical models. Several approaches have been proposed in the literature to model uncertainty related to the inflows (Desreumaux et al., 2014). The Rio Tinto software provided for this study uses a Markov model from inflow measurements during the previous period, $\mathbf{q}_{t-1}$. In such an approach, $\mathbf{q}_{t-1}$ becomes a state variable and its domain is added to the grid Ω. The counterpart of (4.3) can be written

$$\begin{aligned}
\tilde{J}_t(\mathbf{s}_t, \mathbf{q}_{t-1}) := \underset{\mathbf{u}_t}{\text{minimize}} \quad & \underset{\mathbf{q}_t|\mathbf{q}_{t-1}}{\mathbf{E}} \Big\{ G_t(\mathbf{s}_t, \mathbf{u}_t) + \tilde{J}_{t+1}(\mathbf{s}_{t+1}, \mathbf{q}_t) \Big\} && \text{(4.4a)} \\
\text{subject to} \quad & \mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{q}_t - \mathbf{C}\,\mathbf{u}_t, && \text{(4.4b)} \\
& s_{\min} \leq \mathbf{s}_{t+1} \leq s_{\max}, && \text{(4.4c)} \\
& 0 \leq \mathbf{u}_t \leq u_{\max}, && \text{(4.4d)}
\end{aligned}$$

where $\underset{\mathbf{q}_t|\mathbf{q}_{t-1}}{\mathbf{E}}$ represents the conditional expectation of $\mathbf{q}_t$ given the event $\mathbf{q}_{t-1}$.
To evaluate the expectation in (4.4a), one possibility is to discretize the domain of $\mathbf{q}_t$ into M points $\mathbf{q}_t^1, \ldots, \mathbf{q}_t^M$ and use the approximation

$$\sum_{j=1}^{M} \Big(G_t(\mathbf{s}_t, \mathbf{u}_t) + \tilde{J}_{t+1}(\mathbf{s}_{t+1}, \mathbf{q}_t^j) \Big)\ \mathbf{P}(\mathbf{q}_t^j \mid \mathbf{q}_{t-1}^i), \tag{4.5}$$

where $\mathbf{P}(\mathbf{q}_t^j \mid \mathbf{q}_{t-1}^i)$ is the probability of observing $\mathbf{q}_t^j$ at time t given that $\mathbf{q}_{t-1}^i$ was observed at time $t-1$. A common approach to estimate these probabilities is with a least-squares technique using historical data (Faber, 2001; Côté et Leconte, 2015).

Tilmant et al. (2002) explain that in order to take into account the yearly periodicity of an optimal policy and the infinite horizon of (4.2), an optimal solution is obtained when release decisions generated by the SDP algorithm reach a steady state. The model is said to converge when no change in the policy is observed for two consecutive passes. For that reason, we initially set $\tilde{J}_{T+1}$ to zero and solve (4.4) with objective (4.5) in a loop in which $\tilde{J}_{T+1}$ is set to the $\tilde{J}_1$ of the previous pass.

Once a policy is obtained, SDP begins a simulation process in order to evaluate the quality of the policy in presence of uncertainty. Using a second set of inflows scenarios—different

from those used in the approximation process—SDP simulates the evolution of the system in a forward process by solving (4.4) with objective (4.5) to determine the volume of water to release. The procedure is summarized in Algorithm 4.2.1.

The main computational cost of SDP resides in the efficient solution of (4.4) for each $\mathbf{s}_t \in \Omega$, and that cost increases exponentially with the number of reservoirs. In the next section, we describe the context of our industrial partner in this research.

Algorithm 4.2.1 Stochastic Dynamic Programming (SDP)

Initialization
  Set the terminal cost $\tilde{J}_{T+1} = 0$,
  Compute the transition probabilities $\mathbf{P}(\mathbf{q}_t^j \mid \mathbf{q}_{t-1}^i)$ for $i, j = 1, \ldots, M$ and $t = 1, \ldots, T$.
Approximation *// Approximation of the operational policy*
  repeat
    for $t = T, T-1, \ldots, 1$ **do**
      for all $\mathbf{s}_t \in \Omega$ **do**
        Solve (4.4) in which the objective function is replaced with (4.5).
      Obtain $\tilde{J}_t()$ via interpolation.
    $\tilde{J}_{T+1} \leftarrow \tilde{J}_1$.
  until policy converges
Simulation *// Simulation process*
  for a set of inflow scenarios **do**
    for $t = 1, 2, \ldots, T$ **do**
      Solve (4.4) in which the objective function is replaced with (4.5)

4.3 The Saguenay-Lac-Saint-Jean (SLSJ) reservoir

4.3.1 Context

Rio Tinto is a multinational metal and mining corporation that operates a hydroelectric system in the Saguenay-Lac-Saint-Jean region of Québec to power aluminum smelters. The water resources management group at Rio Tinto recently developed an SDP solver for the management of the SLSJ system. The system, illustrated in Figure 4.1, is composed of five generating stations and three reservoirs, for an installed capacity of 3100 MW.

Their implementation computes the current benefit function as the operating cost of the system, i.e., the cost of purchasing energy to compensate for a deficit in generation and the ability to sell excess energy depending on a certain demand known in advance, d_t. Therefore, information regarding hydropower production for each powerhouse must be considered in the model.

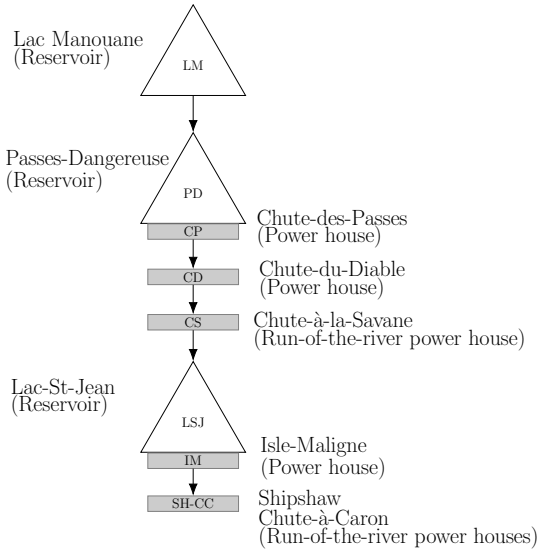

Figure 4.1 Saguenay-Lac-Saint-Jean hydroelectric system.

4.3.2 Rio Tinto problem: modelling

The power produced by a turbine can be approximated by

$$P(\mathbf{u}_t, \bar{\mathbf{s}}_t) = \mathbf{u}_t \, \eta(\mathbf{u}_t) \, h(\bar{\mathbf{s}}_t) \, g, \tag{4.6}$$

where $\bar{\mathbf{s}}_t$ is the average reservoir levels between periods t and $t+1$, η is the turbine efficiency curve, h is the turbine net head function and g is the gravitational acceleration expressed in $\mathrm{m/s^2}$.

A powerhouse is composed of multiple turbines, each with its own efficiency curve η. It is possible to formulate the problem of maximizing the total power produced by the powerhouse as a unit commitment problem in which we seek an optimal discharge dispatch among the turbines. This unit commitment problem can be solved efficiently using dynamic programming (Yi et al., 2003; Breton et al., 2004). Like (4.6), the hydropower function, i.e., the total power produced by a powerhouse, depends on $\bar{\mathbf{s}}_\mathbf{t}$ and $\mathbf{u}_t$. The dashed line in Figure 4.2 represents the value of the hydropower function divided by the discharge with 3, 4 and 5 turbines as a function of the discharge for fixed values of $\bar{\mathbf{s}}_\mathbf{t}$.

As illustrated in Figure 4.2, the hydropower function is typically nonconvex. Instead of using the actual hydropower function, we propose to use its envelope, which is smooth, concave and exhibits a linear behavior over large portions of the domain. Points on the graph of the envelope function are generated by precomputing the convex hull of the points used to determine the actual (dashed) efficiency curve. Using the envelope in place of the actual

efficiency curve in day-to-day operation is not problematic because the efficiency suggested by the envelope using a constant discharge can be achieved by operating fewer turbines at a lower discharge for a portion of the time and more turbines at a higher discharge for the rest of the time. For example, the point labeled "convex combination" is located halfway between the points labeled "pt #1" and "pt #2" on the envelope sketched in Figure 4.2, and corresponds to a discharge of 135. The points "pt #1" and "pt #2" correspond to the discharges that yield the highest efficiency for 3 and 4 turbines, i.e., about 115 and 155, respectively. The efficiency suggested by the envelope can be achieved by operating 3 turbines at a discharge of 115 half of the time and 4 turbines at a discharge 155 the other half of the time. In practice, a maintenance schedule is used to obtain the number of turbines available at each powerhouses during the period t.

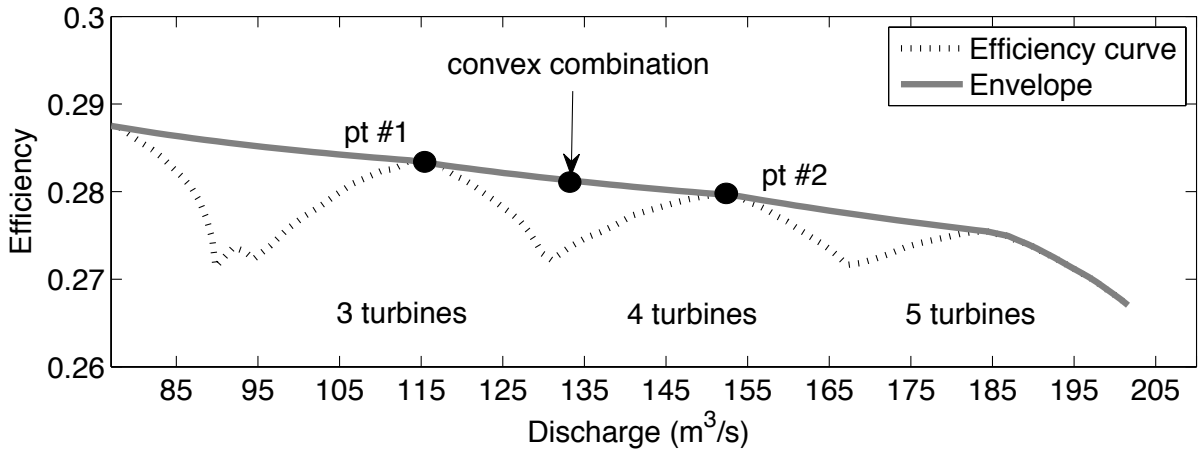

Figure 4.2 Powerhouse efficiency curve for a specific net head.

We use two different approaches to interpolate the envelope. In the first, we use cubic splines, which also supply approximations to the first derivatives. However, in our experience, and as the results of §4.5.2 illustrate, they can make the SDP process substantially more difficult to solve. The second interpolation technique is a piecewise linear approximation that bounds the envelope of the hydropower production function by the set of K hyperplanes

$$p_{i,t} \leq \alpha_{i,k}\bar{s}_{i,t} + \beta_{i,k}u_{i,t} + \gamma_{i,k}, \qquad i = 1,\ldots,5,\ k = 1,\ldots,K. \tag{4.7}$$

where $p_{i,t}$ is the power produced by the i-th powerhouse at time t, and $\alpha_{i,k}$, $\beta_{i,k}$ and $\gamma_{i,k} \in \mathbb{R}$ are determined using a Gauss-Newton heuristic (Magnani et Boyd, 2009). Compared to cubic splines, the second approach increases the number of constraints significantly but it allows us to work with linear constraints. Table 4.1 summarizes the problem dimensions in each formulation.

The SDP subproblem is defined as

$$\tilde{J}_t(\mathbf{s}_t, \mathbf{q}_{t-1}) := \underset{\mathbf{u}_t}{\text{minimize}} \quad \underset{\mathbf{q}_t|\mathbf{q}_{t-1}}{\mathbf{E}} \left\{ G_t(\mathbf{s}_t, \mathbf{u}_t) + \tilde{J}_{t+1}(\mathbf{s}_{t+1}, \mathbf{q}_t) \right\}, \tag{4.8a}$$

$$\text{subject to } \mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{q}_t - \mathbf{C}\mathbf{u}_t, \tag{4.8b}$$

$$p_{i,t} = P_i(u_{i,t}, \bar{s}_{i,t}), \qquad i = 1, \ldots, 5, \tag{4.8c}$$

$$p_{\min} \leq \sum_{i=1}^{5} p_{i,t}, \tag{4.8d}$$

$$s_{\min} \leq \mathbf{s}_{t+1} \leq s_{\max}, \tag{4.8e}$$

$$0 \leq \mathbf{u}_t \leq u_{\max}, \tag{4.8f}$$

where $p_{\min}$ corresponds to the minimum energy required to maintain the aluminum smelter operational. When the splines are used, P_i represents the spline interpolation of the hydropower production envelope at the i-th powerhouse. The second formulation is similar to (4.8) except that (4.8c) is formulated as (4.7).

Table 4.1 Size of SDP subproblems depending of the interpolation technique

Interpolation technique	Variables	Constraints		
		Linear	Nonlinear	Total
Cubic spline	24	13	5	18
Piecewise linear	24	163	0	163

In the next section, we describe the methodology behind the state of the art implementation that we benchmark in §4.5 to compare both formulation.

4.4 Brief description of optimization methods considered

4.4.1 Overview

The subproblems generated in the course of the SDP process can be expressed in the general constrained form

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f(x) \quad \text{subject to } \; g_l \leq g(x) \leq g_u, \; h(x) = 0, \; l \leq x \leq u, \tag{4.9}$$

where $g\colon \mathbb{R}^n \to \mathbb{R}^{m_I}$ and $h\colon \mathbb{R}^n \to \mathbb{R}^{m_E}$ are smooth and differentiable, g_l, $g_u \in \mathbb{R}^{m_I}$ and l, $u \in \mathbb{R}^n$.

The numerical methods for (4.9) that we consider fit in three categories: augmented Lagrangian methods, sequential linear or quadratic optimization, and interior-point methods (Bazaraa

et al., 2005; Nocedal et Wright, 2006). We summarize each in turn in the next three sections.

Augmented Lagrangian methods

The augmented Lagrangian method, also known as the method of multipliers, was proposed by Powell (1969) and Hestenes (1969) as a penalty approach for problems with equality constraints that, contrary to the quadratic penalty method, does not require an unbounded penalty parameter. The MINOS (Murtagh et Saunders, 1978) and LANCELOT (Conn et al., 1992) packages were the first available implementations and they remain reference implementations.

For the simplicity of discussion, let us assume temporarily that (4.9) does not contain general inequality constraints. The method is composed of outer and inner iterations. The k-th outer iteration of LANCELOT consists in solving the subproblem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \;\; f(x) + \mu_k^T h(x) + \tfrac{1}{2}\rho_k \|h(x)\|_2^2 \quad \text{subject to } \; l \leq x \leq u, \tag{4.10}$$

where $\mu_k \in \mathbb{R}^{m_E}$ is an approximation to the optimal Lagrange multipliers associated to the equality constraints and $\rho_k > 0$ is a penalty parameter that may be updated at every outer iteration. The solution of (4.10) requires a method for bound-constrained problems. The iterations of the latter are called the inner iterations corresponding to the k-th outer iteration. An approximate solution is returned to the outer iteration, where μ_k and/or ρ_k is updated. The process is repeated until satisfaction of relaxed first-order optimality conditions for (4.9).

The k-th outer iteration of MINOS is similar, except that a linearization of $h(x) = 0$ about the current iterate is included in the constraints of (4.10) and the departure from linearity, i.e., the difference between $h(x)$ and its linearization, is used in the objective in place of $h(x)$. Each subproblem is thus linearly constrained.

When general inequality constraints are present, LANCELOT introduces slack variables to transform them into equality constraints, and penalizes the latter similarly to $h(x) = 0$. The bound constraints on the slack variables are treated identically to those on x.

Other implementations exist, including some that penalize inequality constraints without introducing slack variables, but we do not consider them in our study.

Sequential linear and quadratic optimization

Sequential optimization methods aim to solve a nonlinear problem by solving a sequence of subproblems defined by Taylor-like approximations.

Bazaraa et al. (2005) credit Griffith et Stewart (1961) for proposing to solve (4.9) using Sequential Linear Programming (SLP). The idea of SLP is straightforward. At every iteration, the method solves a linear subproblem defined by the first-order Taylor expansion of the objective and constraints. A trust region defined in a polyhedral norm, typically the ℓ_∞-norm, is added to keep the subproblem from being unbounded and to maintain linearity of the constraints. The subproblem at iteration k has the form

$$\begin{aligned}
\underset{\Delta x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x_k) + \nabla f(x_k)^T \Delta x && \text{(4.11a)}\\
\text{subject to} \quad & g_l \leq g(x_k) + \nabla g(x_k) \Delta x \leq g_u, && \text{(4.11b)}\\
& h(x_k) + \nabla h(x_k) \Delta x = 0, && \text{(4.11c)}\\
& l \leq x_k + \Delta x \leq u, && \text{(4.11d)}\\
& \|\Delta x\|_\infty \leq \Delta_k, && \text{(4.11e)}
\end{aligned}$$

where ∇f is the gradient of f, ∇g and ∇h are the gradient matrices, i.e., the transposed Jacobians, of g and h, respectively, and $\Delta_k > 0$ is the trust-region radius at iteration k. The advantage of SLP is that state-of-the-art linear optimization software can be used to compute steps. Its main disadvantage is that the curvature of strongly nonlinear problems is not captured adequately. Surprisingly few implementations of SLP are available. The only one we are aware of is included in the FICO Xpress Optimization suite [1]. We describe our own in §4.4.2.

Wilson (1963) suggested Sequential Quadratic Programming (SQP) as an improvement over SLP. SQP solves (4.9) by a sequence of quadratic programs (QP). Each QP is similar to (4.11) except that (4.11a) is replaced with the second-order expansion

$$f(x_k) + \nabla f(x_k)^T \Delta x + \tfrac{1}{2} \Delta x^T B_k \Delta x, \tag{4.12}$$

where $B_k = B_k^T$ is an approximation of the Hessian of the Lagrangian of (4.9) for certain estimates of the optimal multipliers.

SQP methods can be of the trust-region kind and solve a subproblem with constraints (4.11b)–(4.11e) with the exception that the trust region is typically, though not always, defined by an elliptical norm. Globalization may also be promoted using a linesearch or a filter.

The SNOPT (Gill et al., 2005) and FilterSQP (Fletcher et Leyffer, 2002) packages are considered as the state-of-the-art in this regard. SNOPT uses a linesearch strategy together with an augmented Lagrangian merit function to judge of the quality of a trial step. SNOPT

1. http://www.fico.com/en/products/fico-xpress-optimization-suite

does not use exact second derivatives and maintains instead a dense BFGS approximation. FilterSQP follows a trust region approach combined with a filter to determine acceptance of a trial step.

We close with the Sequential Linear Quadratic Programming (SLQP) approach (Nocedal et Wright, 2006, Chapter 18.5). The method proceeds in two steps. In the first step, a linear problem similar to (4.11) is solved to obtain an active set estimate. A step is then obtained by solving an equality-constrained quadratic problem defined by the active set estimate. Therefore, this second subproblem may reduce significantly the size of the quadratic subproblem in case of large problem. The overall step is defined as a combination of the steps from the linear and quadratic subproblems. SLQP is implemented in KNITRO/ACTIVE by Byrd et al. (2003).

Interior-Point methods

In the last decades, interior-point methods proved to be some of the most powerful numerical methods for large scale nonlinear optimization. Briefly, an interior-point method solves (4.9) by a succession of subproblems using a logarithmic barrier to prevent violation of the inequality constraints and bounds:

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) - \rho_k \sum_{i=0}^{m_I} \left(\log(g_i(x) - g_{l_i}) + \log(g_{u_i} - g_i(x))\right) \\ & - \rho_k \sum_{i=0}^{n} \left(\log(x_i - x_{l_i}) + \log(x_{u_i} - x_i)\right) \\ \text{subject to} \quad & h(x) = 0. \end{aligned} \tag{4.13}$$

If the problem functions remain well defined at values of x that violate the inequality constraints, it is typical to introduce slack variables to convert the latter to equality constraints. In this case, only simple bounds appear in the barrier terms. As for SQP, both trust-region and linesearch mechanisms can be used to promote global convergence.

IPOPT (Wächter et Biegler, 2006) and KNITRO/DIRECT (Waltz et al., 2006) are prime examples of linesearch implementations. Byrd et al. (1999) propose a trust-region implementation in KNITRO/CG. KNITRO implementation details are given by Byrd et al. (2006).

4.4.2 Sequential Linear Programming

According to the discussion of §4.3 and to Figure 4.2, hydropower functions are linear on large portions of their domain so that an SLP approach seems appropriate to assist SDP. In

this section, we give a detailed description of the method that follows (Bazaraa et al., 2005).

Because SLP works with linearized constraints (4.11b)–(4.11c), the constraints of (4.9) may be violated during the iterations. In addition, (4.11) could be infeasible even though (4.9) is feasible, either because the linearizations describe disjoint sets, or because the trust region does not intersect the linearized feasible set. A common approach to avoiding infeasible subproblems while at the same time providing a mechanism by which to assess progress towards the feasible set of (4.9) is to introduce an ℓ_1-penalty term into the objective of (4.9). The exact ℓ_1 penalty function is

$$\phi(x;\rho_k) := f(x) + \rho_k \theta(x),$$

where $\rho_k > 0$ is the penalty parameter at iteration k and

$$\theta(x) := \sum_{i=1}^{m_I} (\max\{0, g_i(x) - g_{u_i}\} + \max\{0, g_{l_i} - g_i(x)\}) + \sum_{i=1}^{m_E} |h_i(x)|$$

is the ℓ_1 infeasibility measure for (4.9). Note that SLP ensures feasibility at each iteration with respect to linear constraints that might be present in (4.9). For that reason, the penalty function need only be applied to nonlinear constraints. We introduce elastic variables $p \in \mathbb{R}^{m_I}$ and $q \in \mathbb{R}^{m_E}$ so as to obtain the equivalent smooth reformulation of the penalty problem

$$\begin{aligned} \underset{x,p,q}{\text{minimize}} \quad & f(x) + \rho_k \sum_{i=0}^{m_I} p_i + \rho_k \sum_{i=0}^{m_E} q_i \\ \text{subject to} \quad & g_l - p \le g(x) \le g_u + p, \\ & -q \le h(x) \le q, \\ & b_l \le Ax \le b_u, \quad l \le x \le u, \end{aligned}$$

where we have stated the linear constraints separately.

The k-th SLP subproblem is now given defined by

$$\begin{aligned} \underset{x,p,q}{\text{minimize}} \quad & \nabla f(x_k)^T x + \rho_k \sum_{i=0}^{m_I} p_i + \rho_k \sum_{i=0}^{m_E} q_i \\ \text{subject to} \quad & g_l - p \le g(x_k) + \nabla g(x_k)(x - x_k) \le g_u + p, \\ & -q \le h(x_k) + \nabla h(x_k)(x - x_k) \le q, \\ & b_l \le Ax \le b_u, \quad l \le x \le u, \\ & \|x - x_k\|_\infty \le \Delta_k, \end{aligned} \tag{4.14}$$

where we removed constant terms from the objective.

A solution $(\hat{x}, \hat{p}, \hat{q})$ of (4.14) is a candidate $(x_{k+1}, p_{k+1}, q_{k+1})$. The decision to accept $(\hat{x}, \hat{p}, \hat{q})$ uses a classical ratio of achieved versus predicted reduction, as found in trust-region methods. Define the step $\Delta x = \hat{x} - x_k$. The achieved reduction in the penalty function is

$$\Delta\phi_k := \phi(x_k; \rho_k) - \phi(x_k + \Delta x; \rho_k).$$

Consider the piecewise linear model of $\phi(x; \rho_k)$ about x_k

$$\begin{aligned}\ell_k(x) := f(x_k) &+ \nabla f(x_k)^T(x - x_k) \\ &+ \rho_k \sum_{i=0}^{m_I} \max\{0, g_i(x_k) + \nabla g_i(x_k)(x - x_k) - g_{u_i}\} \\ &+ \rho_k \sum_{i=0}^{m_I} \max\{0, g_{l_i} - g_i(x_k) + \nabla g_i(x_k)(x - x_k)\} \\ &+ \rho_k \sum_{i=0}^{m_E} |h_i(x_k) + \nabla h_i(x_k)(x - x_k)|.\end{aligned}$$

Predicted reduction is defined as

$$\Delta\ell_k := \ell_k(x_k) - \ell_k(x_k + \Delta x).$$

The step Δx is deemed acceptable if sufficient progress towards feasibility has been achieved, i.e.,

$$\theta(\hat{x}) \leq \epsilon_a,$$

and

$$R_k := \Delta\phi_k / \Delta\ell_k > \eta_1 \quad \text{where} \quad 0 < \eta_1 < 1.$$

If $\theta(\hat{x}) \leq \epsilon_a$ and the more demanding condition $R_k > \eta_2$ is satisfied, where $\eta_1 < \eta_2 < 1$, the trust-region radius is expanded by a factor $\gamma_1 \geq 1$. On the other hand, if $\theta(\hat{x}) > \epsilon_a$ or $R_k \leq \eta_1$, the step is rejected and the trust region is shrunk by a factor $\gamma_2 < 1$.

Before solving the first subproblem (4.14), we check whether the linear constraints of (4.9) are satisfied at the initial guess x_0. If they are not satisfied, we project x_0 into the linear constraints by solving

$$\underset{x}{\text{minimize}} \|x - x_0\|_1 \quad \text{subject to} \quad b_l \leq Ax \leq b_u, \quad l \leq x \leq u.$$

We follow (Bazaraa et al., 2005, Theorem 10.3.1) and report $\hat{x}$ as stationary for (4.9) if

$$\|\Delta x\|_\infty \leq \epsilon_d \quad \text{and} \quad \theta(\hat{x}) \leq \epsilon_p,$$

where ϵ_d, $\epsilon_p > 0$ are the dual and primal infeasibility tolerance, respectively.

As is typical with the ℓ_1-penalty approach, Bazaraa et al. (2005) indicate that ρ_k must be at least as large as the ℓ_∞-norm of the optimal vector of Lagrange multipliers associated with the nonlinear constraints of (4.9). Our update has the form

$$\rho_{k+1} = \min\{\, \|\lambda_k\|_\infty,\, \rho_{\max}\} \tag{4.15}$$

where $\rho_{\max} := 10^3$ is the maximal value of the penalty parameter and λ_k is the vector of multipliers associated to the linearized constraints of (4.14) by the LP solver. In addition, we also follow (Nocedal et Wright, 2006, Algorithm 18.5) and only update ρ_k when $p > 0$ or $q > 0$ in (4.14), i.e., the linearized constraints of (4.9) are violated.

Algorithm 4.4.1 summarizes the sequential linear programming approach.

Algorithm 4.4.1 Sequential Linear Programming (SLP)

0: Initialization: Choose $x_0 \in \mathbb{R}^n$, trust region parameters $\Delta_0 = \min\{\, 10,\, 0.1\|\nabla f(x_0)\|_\infty\}$, $0 < \eta_1 < \eta_2 < 1$, $\gamma_2 < 1 \leq \gamma_1$, penalty parameter $0 < \rho_0 \leq \rho_{\max}$, stopping tolerances ϵ_p, ϵ_a, and $\epsilon_d > 0$.
1: Solve (4.14) to compute an improving candidate $(\hat{x}, \hat{p}, \hat{q})$ and define $\Delta x = \hat{x} - x_k$.
2: **if** $\|\Delta x\|_\infty \leq \epsilon_d$ and $\theta(\hat{x}) \leq \epsilon_p$ **then**
3: terminate with the solution $\hat{x}$. Otherwise continue to 4.
4: Compute $R_k = \Delta\phi_k / \Delta\ell_k$.
5: **if** $R_k > \eta_1$ and $\theta(\hat{x}) \leq \epsilon_a$ **then**
6: $(x_{k+1}, p_{k+1}, q_{k+1}) \leftarrow (\hat{x}, \hat{p}, \hat{q})$
7: **if** $R_k > \eta_2$ **then**
8: $\Delta_{k+1} \leftarrow \gamma_1 \Delta_k$
9: **else**
10: $(x_{k+1}, p_{k+1}, q_{k+1}) \leftarrow (x_k, p_k, q_k)$
11: $\Delta_{k+1} \leftarrow \gamma_2 \|d_k\|_\infty$
12: Optionally perform a backtracking Armijo linesearch.
13: **if** $\hat{p} > 0$ or $\hat{q} > 0$ **then**
14: Update ρ_k using (4.15).
15: Set $k \leftarrow k + 1$ go to 1.

4.5 Implementation and numerical results

4.5.1 Implementation

We implemented Algorithm 4.4.1 in the Python programming language as part of the NLP.py Python development environment for linear and nonlinear optimization (Arreckx et al., 2016). Because (4.14) is a linear program, we can solve it using an LP solver. In our implementation, we decide to compare SLP efficiency using the open-source linear solver COIN-OR's CLP [2] via the CyLP interface (Towhidi et Orban, 2016) as well as the commercial linear solver included in the library FICO Xpress Optimization suite. Our implementation, named SLP, is available from `github.com/kenjydem/SLP.py`.

In order to tune the performance of our implementation, we use the OPAL library of Audet et al. (2014) to set locally optimal values of the trust-region parameters η_1, η_2, γ_1, and γ_2. OPAL identifies parameter values that maximize a performance measure of a given implementation by way of black-box optimization. In our case, OPAL is applied to a small subset of subproblems from SDP to find the parameter values minimizing the total number of iterations necessary to satisfy the stopping criteria. No surrogate model was used in our experiments. The parameter values suggested by OPAL are $\eta_1 = 0.11$, $\eta_2 = 0.49$, $\gamma_1 = 2.10$, $\gamma_2 = 0.79$. Note that those differ substantially from the ones given by Audet et Orban (2006) in a trust-region context for unconstrained optimization.

Because the methods discussed previously are implemented in low-level programming languages such as Fortran and C++, some wrappers were necessary in order to be able to call them from Python. Only KNITRO provides a Python wrapper. SNOPT is accessible in Python via the pyOpt interface (Perez et al., 2012), and IPOPT via a Cython wrapper [3]. Cython is a superset of the Python language that makes interfacing C and C++ libraries convenient. Moreover, it is a compiled language and can be expected to run at speeds comparable to native C. We developed our own Python wrappers for MINOS, Xpress and LANCELOT. We use the LANCELOT implementation included in GALAHAD (Gould et al., 2003). In our comparative study, LANCELOT is used via its simplified LANCELOT_simple interface (Gould et al., 2008).

We compare the efficiency of the various solvers in the next two sections. In §4.5.2, we perform benchmarks to estimate the computational effort required to solve each formulation of SDP subproblems. In §4.5.3, we compare the performance of the operating policies found by SDP during a simulation process.

2. https://projects.coin-or.org/Clp
3. https://pypi.python.org/pypi/ipopt

4.5.2 Benchmarks

We use a set of 3,750 random subproblems from SDP with the formulation (4.8). We order to limit the computational effort, we impose a maximum of 700 objective function evaluations.

We report our numerical results using the performance and data profiles of Moré et Wild (2009) with the convergence test

$$r_k \leq r_{\text{best}} + \tau(r_0 - r_{\text{best}}), \tag{4.16}$$

where $\tau > 0$ is a tolerance, r_k is the first-order optimality residual at iteration k, r_0 is the initial guess residual, and r_{best} is the smallest optimality residual found by any solver.

With the exception of SLP, the limitations of the Python interfaces and solver APIs are such that we do not have access to local information computed by the solver at each iteration, and in particular, multiplier estimates. Thus, we do not have access to r_k from the solver. Instead, we intercept each evaluation of the objective function requested by the solver and compute our own multiplier estimates by solving the linear problem (4.11a)–(4.11d), where x_k is the iterate for which the solver requests $f(x_k)$. Because x_k is a stationary point if and only if $\Delta x = 0$ in (4.11a)–(4.11d) under a standard constraint qualification condition, we are able to terminate each solver using the same stopping criterion. We set the stopping tolerances of every solver to 1.0e−16 to ensure that runs do not terminate before our conditions are satisfied. [4]

We do not use exact second derivatives in our experiments for two reasons. The first is that spline interpolation is performed using an in-house library that does not supply second derivatives. The second is that SNOPT does not make provision for exact second derivatives and uses a BFGS approximation instead. In addition, LANCELOT only uses dense quasi-Newton updates. KNITRO, MINOS and SNOPT all offer a dense quasi-Newton option, but IPOPT only offers a limited-memory BFGS option. Thus in an attempt to perform a fair comparison, we instruct all solvers to employ a dense BFGS Hessian approximation and IPOPT to use a memory of 500 in its limited-memory approximation.

Finally, we deactivate the following features on all solvers to ensure a fair baseline comparison: problem scaling, presolve, warm start, derivative checker and backtracking. We also deactivate the parallelism option in KNITRO. Because each objective evaluation is expensive, we set the `Linesearch tolerance` option to 0.99 in MINOS and SNOPT, and deactivate the backtracking linesearch in Algorithm 4.4.1.

4. Certain solvers, including as IPOPT, do not accept zero as a tolerance.

While our approach allows us to estimate correctly the Lagrange multipliers for all methods, it increases considerably the computational time for all solvers. Hence, we do not use computational time as a benchmark measure in our comparisons.

Performance profiles reveal the proportion of problems satisfying (4.16) as a function of the performance of each method relative to the others. In the present case, by *performance*, we mean the number of objective function evaluations required to satisfy (4.16). Figure 4.3 shows the performance profiles with a factor $\tau = 10^{-3}$.

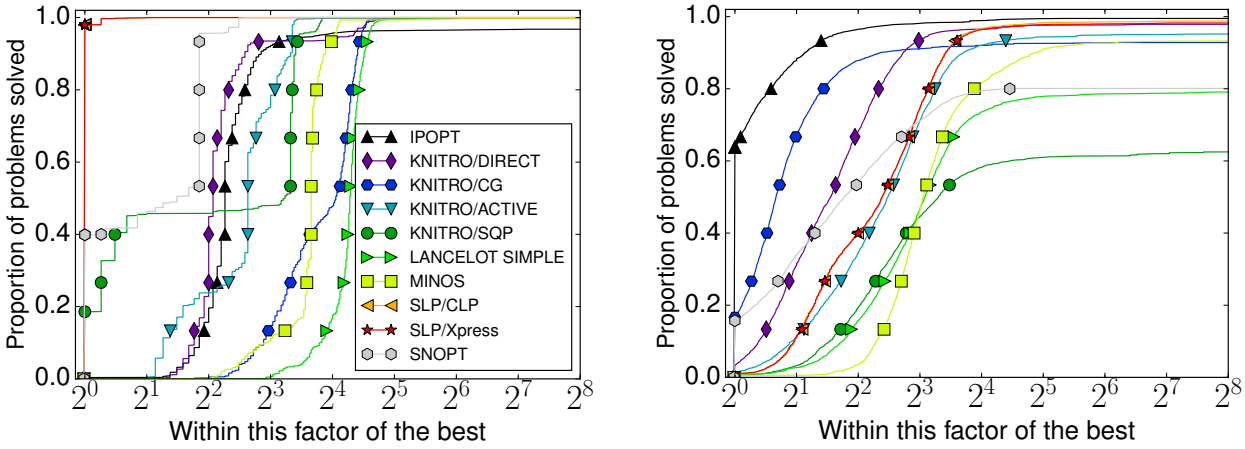

Figure 4.3 Performance profiles (logarithmic scale) of the number of objective function evaluations required to satisfy (4.16) with $\tau = 10^{-3}$ for $3,750$ subproblems from SDP. The left plot shows results for the formulation with linear constraints (hyperplanes) and the right plot those for the formulation with nonlinear constraints using cubic splines.

The two profiles tell different stories. In the formulation with the linear constraints, almost all algorithms are able to satisfy the convergence test for all subproblems. Sequential linear and quadratic optimization methods are the most efficient when constraints are linear. In particular, SLP dominates all other solvers followed by SNOPT. They both dominate all other solvers. KNITRO/SQP initially exhibits the same behavior but is soon dominated by the interior-point solvers IPOPT and KNITRO/DIRECT. KNITRO/ACTIVE shows a performance on par with the interior-point methods.

Using nonlinear hydropower constraints appears to increase the level of difficulty because some solvers are no longer able to satisfy the stopping criteria for all problems. This time, the interior-point methods dominate all other solvers. Although SNOPT does not lag far behind in terms of efficiency, its robustness is substantially lower at about 80%. SLP on the other hand may not be as efficient but shows levels of robustness comparable to the interior-point solvers. Note that in both profiles, the curves SLP/CLP and SLP/Xpress are superposed.

As a complement to performance profiles, data profiles (Moré et Wild, 2009) illustrate the proportion of problems satisfying (4.16) within a given budget of objective evaluations. Figure 4.4 shows the data profiles for $\tau = 10^{-3}$ with both problem formulations.

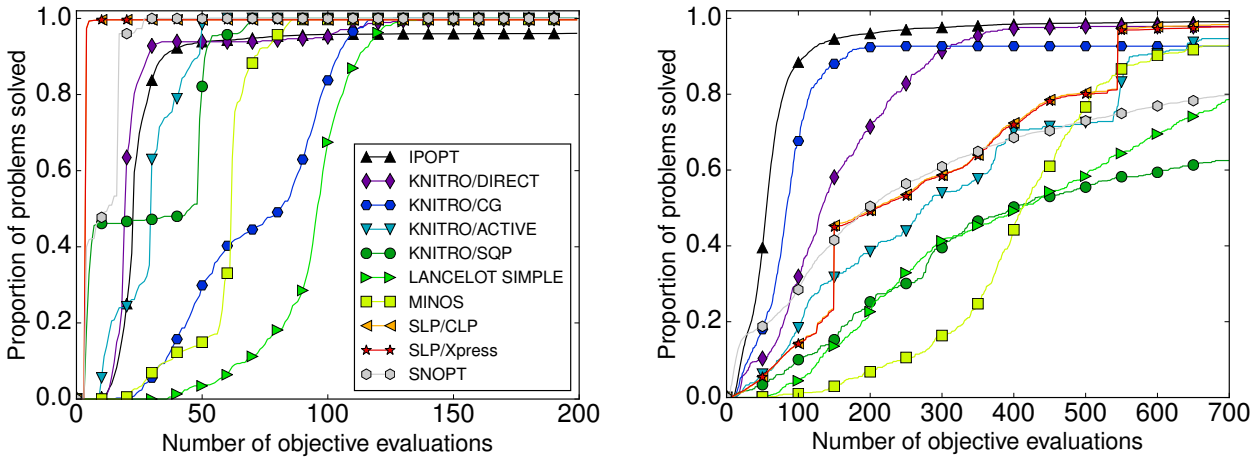

Figure 4.4 Data profiles with $\tau = 10^{-3}$ for $3,750$ subproblems from SDP. The left plot shows results for the formulation with linear constraints (hyperplanes) and the right plot those for the formulation with nonlinear constraints using cubic splines.

The data profiles show that SLP and SNOPT attain (4.16) with few objective evaluations when all constraints are linear, followed closely by the interior-point methods. Again, SLP dominates all other solvers, and almost all solvers attain (4.16) within 200 objective evaluations. A larger budget of objective evaluations is necessary when using nonlinear constraints, but there again, the interior-point solvers dominate. Multiple solvers fail to attain (4.16) within 700 objective evaluations. SNOPT is one of them, while SLP is not, though it requires over 500 evaluations.

4.5.3 Comparison of optimal operation policies

We now compare several solvers during the simulation phase, i.e., Step 11 of Algorithm 4.2.1. The simulation evaluates the operating policy identified by SDP in Step 3 of Algorithm 4.2.1. Throughout this section, we use a set $\mathcal{Q}$ composed of 25 annual inflow scenario from historical data. The simulation process will give rise to new reservoirs levels. In our implementation, we perform the simulation by solving the optimization problems again instead of interpolating the policy resulting from the approximation phase (Tejada-Guibert et al., 1993). We assess the performance of the solvers by measuring the quality of the operating policy in simulation and the time to solve all subproblems. Our own external optimality test described in §4.5.2 is deactivated. We set the stopping tolerance of each solver to 10^{-5} and set a budget of

700 objective evaluations. Note that times should be taken as indicative because IPOPT, MINOS and SNOPT are all implemented in low-level languages while SLP is implemented in Python, and each solver implements slightly different stopping criteria, possibly involving different scalings. We believe the times reported remain meaningful nonetheless because all computationally expensive tasks taking place in SLP occur either in the Cython communication layer or in CLP, which is itself implemented in a low-level language. Because we minimize the operational cost at each period, the quality of the policies is given by the annual cumulative cost.

The state space of reservoir levels is discretized into a grid Ω of 7 values for each of the three reservoirs. The random natural inflows are discretized separately into 5 values. We use $T = 122$ time steps and apply the repeat loop at Step 4 of Algorithm 4.2.1 three times. Thus, SDP solves a total of $7^3 \cdot 5 \cdot 122 \cdot 3 = 627,690$ subproblems to compute a policy approximation. Because of the high number of subproblems, parallelism is essential. Exploiting parallelism during policy approximation is simple because of the grid Ω. However, there is no possibility to parallelize the model in simulation because of the recursion. In our results, the approximation is parallelized over 20 cores.

We compare IPOPT, SLP and SNOPT because of their performance in the previous analysis. Despite its performance in §4.5.2, we retain MINOS as a representative of the family of augmented-Lagrangian methods in the present assessment. In our experiments, we always use nonlinear constraints during the simulation process to better reflect the reality of hydropower functions. Table 4.2 summarizes our results.

Table 4.2 Summary of the average annual cumulative cost (AACC) using IPOPT, MINOS, SLP and SNOPT on the formulation with linear (left) and nonlinear constraints (right) during the approximation process. The simulation process always uses nonlinear constraints. The columns "Simul" and "Approx" refer to the simulation and approximation steps of Algorithm 4.2.1.

Solver	AACC	Time (min)			AACC	Time (min)		
	Simul	Approx	Simul	Total	Simul	Approx	Simul	Total
IPOPT	140.3	157.9	3.6	161.5	134.0	94.9	3.4	98.3
MINOS	154.3	12.4	55.7	68.1	9,584.9	2,502.8	105.2	2,608.0
SLP	222.2	26.0	23.3	49.3	5,298.3	285.8	16.8	302.6
SNOPT	127.9	28.9	26.7	55.6	146.9	2,033.3	28.1	2,061.4

The results of Table 4.2 show that despite parallelism, the computational time using nonlinear constraints is excessive for all solvers except IPOPT. MINOS is the fastest solver to approximate the policy when constraints are linear, possibly because it is able to exploit those linear constraints. Surprisingly, IPOPT is the only method that is slower using linear constraints. However, IPOPT outperforms all solvers during the simulation, where the hydropower constraints are nonlinear. SLP and MINOS produce larger values of the annual cumulative cost during the simulation than the other solvers. In order to understand why, we compute the evolution of the average cost over all scenarii in $\mathcal{Q}$ for each of the four solvers and plot the results in Figure 4.5.

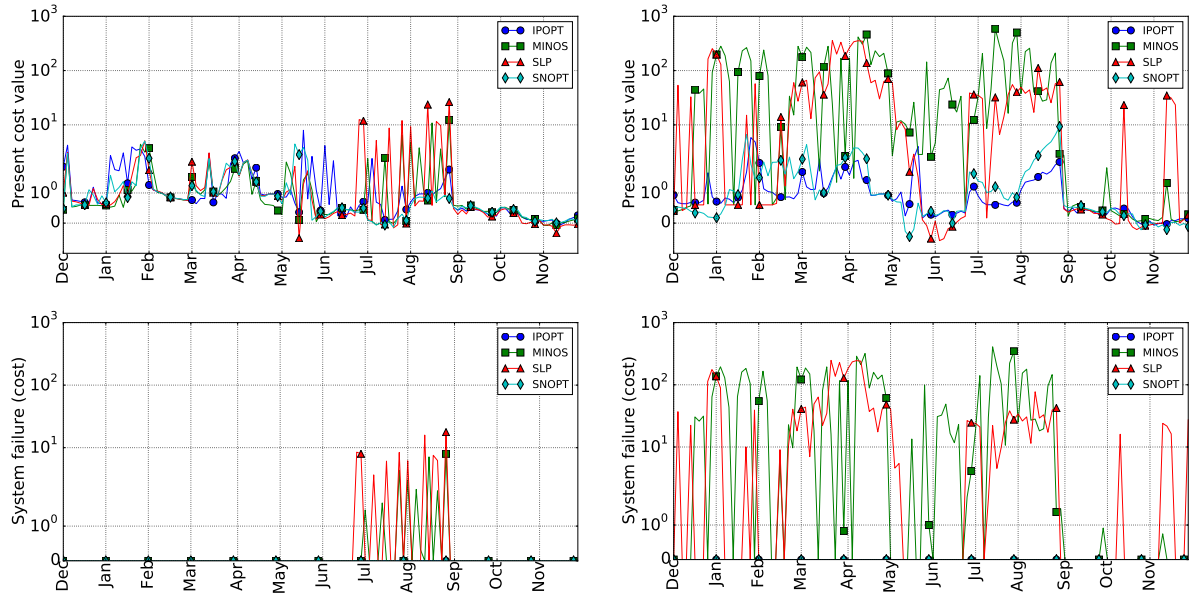

Figure 4.5 Top: Evolution of the average cost over 25 inflow scenarii. Bottom: Evolution of the average cost over the same 25 inflow scenarii related to system failures. The left plots correspond to linear constraints during the approximation step and the right plots to nonlinear constraints.

The top row of Figure 4.5 reveals that using linear constraints during the approximation step (Step 3) of Algorithm 4.2.1 mostly keeps costs down over all 25 inflows scenarii. However, MINOS and SLP induce unnecessarily high costs between July and September. The same solvers induce high costs through most of the year when using nonlinear hydropower constraints. The origin of those costs is as follows. If the system is not able to provide sufficient energy at a certain period, extra energy is purchased to compensate the deficit. Because the amount of energy available to purchase is limited, the system can be forced to shut down if the deficit persists, and such a system failure is severely penalized.

The bottom row of Figure 4.5 illustrates the evolution of the average cost caused by system failures over all scenarii in $\mathcal{Q}$ for each of the four solvers and confirms that failures occur and are related to MINOS and SLP. A detailed look at the behavior of MINOS and SLP during the SDP process reveals that they both perform well during the approximation but often attain the maximum number of objective evaluations during the simulation step. Therefore, the decisions returned are likely suboptimal, which leads to the system failures.

A promising implementation results from combining the efficiency of SLP during the approximation step (Step 3) of Algorithm 4.2.1 with that of IPOPT during the simulation step (Step 11). For purposes of comparison, we combine each of SLP, MINOS and SNOPT used for the policy approximation during Step 3 with IPOPT for the simulation in Step 11. Policy approximation uses linearly-constrained problems while simulation uses nonlinear hydropower constraints. The results appear in Table 4.3 and confirm that a combination of solvers is more efficient than any one solver used in both phases. In particular, the MINOS/IPOPT combination produces the lowest run times, which represent a reduction of a factor of about 6.1 compared to IPOPT alone using nonlinear constraints—see Table 4.2, and the second lowest average annual cumulative cost. The SLP/IPOPT combination yields the lowest average annual cumulative cost with a total run time improvement of about 3.3 compared to IPOPT alone. Both improve the average annual cumulative cost.

Table 4.3 Summary of the average annual cumulative cost (AACC) by combining MINOS, SLP, SNOPT with IPOPT using linear and nonlinear hydropower constraints in Step 3 ("Approx") and Step 11 ("Simul"), respectively.

Solver	AACC	Time (min)		
	Simul	Approx	Simul	Total
MINOS/IPOPT	133.0	12.4	3.7	16.1
SLP/IPOPT	131.7	26.0	3.6	29.6
SNOPT/IPOPT	134.1	29.1	3.7	32.8

We close this section with a comparison of the policies identified by each solver. We wish to determine whether the optimal policies identified using linear and nonlinear constraints in the approximation phase are statistically equivalent in terms of the annual costs obtained during the simulation process. Let us call IPOPT-N/IPOPT the variant where IPOPT is employed for both the approximation and simulation and where the approximation phase uses nonlinear constraints. IPOPT-N/IPOPT corresponds to the first row of the right-hand side of Table 4.2 and is used as our reference. We compare the combinations of Table 4.3 to IPOPT-N/IPOPT. We also add IPOPT-L/IPOPT to the comparison, which corresponds to the first row of the left-hand side of Table 4.2.

For each of the four combinations, we compute the annual cost difference e_i with IPOPT-N/IPOPT for all 25 scenarii in $\mathcal{Q}$. Those differences can be visualized as the empirical cumulative distribution functions in Figure 4.6, obtained by sorting the cost differences $e_1, \ldots, e_{25}$ and plotting the function $\alpha \in \mathbb{R} \mapsto \#\{i \mid e_i \leq \alpha\}/25$, where $\#$ indicates cardinality.

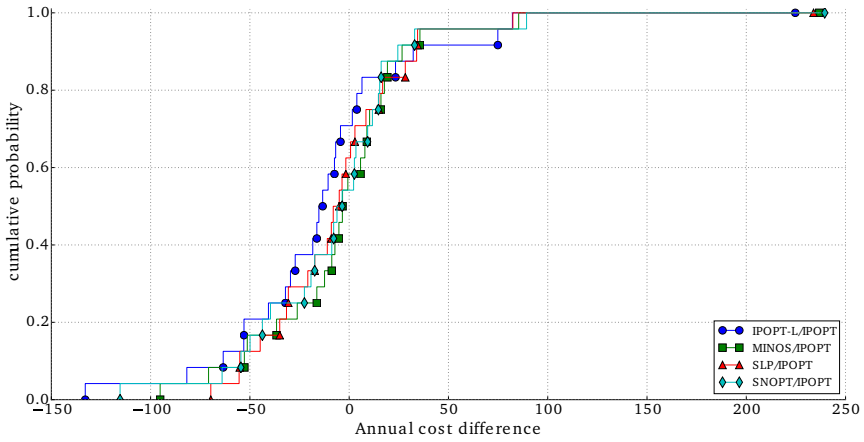

Figure 4.6 Empirical cumulative distribution functions of the difference between the annual cost generated by IPOPT-N/IPOPT and the solver combinations of Table 4.3.

Figure 4.6 suggests that the distributions are approximately normal with zero mean, and nearly coincide. To confirm those conclusions, we use the two-sample t-test, a statistical test to determine whether two population means are statistically equivalent. The first population comprises the 25 annual costs obtained by IPOPT-N/IPOPT and the second the 25 annual costs found by each solver combination of Table 4.3 in turn. Let μ_1 and μ_2 denote the mean of each population, and s_1 and s_2 their standard deviation. The test statistic T is defined by

$$T = \frac{\mu_1 - \mu_2}{\sqrt{s_1^2/N + s_2^2/N}}.$$

The hypothesis $\mu_1 \approx \mu_2$ is rejected based on a 95% confidence interval if

$$|T| \geq t_{1-\alpha/2,\,v},$$

where $t_{1-\alpha/2,\,v}$ is the critical value of the distribution associated with a significance level $\alpha = 0.05$ and a degree of freedom $v = N - 1 = 24$.

Table 4.4 gives the result of the statistical test.

Table 4.4 Results from the two-sample t-test

Solver	Test statistic	p-value	Critical value		$\mu_1 \approx \mu_2$?
			Lower tail	Upper tail	
IPOPT-L/IPOPT	-0.479	0.636	-33.32	20.77	Yes
MINOS/IPOPT	0.264	0.794	-21.85	28.26	Yes
SLP/IPOPT	0.199	0.844	-21.84	26.49	Yes
SNOPT/IPOPT	0.029	0.977	-25.70	26.44	Yes

Table 4.4 indicates that the lower and upper tails are different, which suggests that our normality assumption is not quite satisfied. The differences, however, are not substantial, except in the case of IPOPT-L/IPOPT. In each case, the hypothesis that $\mu_1 \approx \mu_2$ is accepted.

4.6 Conclusions

Subproblem formulation in SDP has a decisive impact on the overall efficiency of the process. Our comparison of optimization solvers focuses on the modeling of hydropower constraints during the estimation and simulation phases. Active-set methods, including sequential linear and quadratic optimization methods are among the most efficient when using linear hydropower constraints, although interior-point methods are not far behind. In particular, our Python implementation of SLP dominates all other solvers. However, interior-point methods dominate when using nonlinear hydropower constraints. Because using nonlinear hydropower constraints is more realistic in a practical setting, we propose a combination of subproblems with linear constraints in the approximation phase and nonlinear constraints in the simulation phase. The solver combination MINOS/IPOPT is the most effective in terms of run time with a final average annual cumulative cost close to the smallest value that we were able to obtain. The SLP/IPOPT combination is competitive in terms of run time and produces the lowest final average annual cumulative cost. It is also particularly attractive because it consists only of open-source solvers. Given the performance of piecewise linear approximations of the production functions, it would be interesting to consider a Stochastic Dual Dynamic Programming, where the water value function is also piecewise linear. A comparison of both approaches is in progress. However, because SDDP is by definition an approximation of SDP, we expect that the process will be faster but less accurate. SLP can clearly be improved on problems with nonlinear constraints. In particular, a better control of infeasibility would improve the robustness on individual problems and possibly avoid high penalties due to system failures in simulation.

CHAPITRE 5 DISCUSSION GÉNÉRALE

Ce chapitre présente quelques éléments d'information supplémentaires qui ne sont pas discutés au chapitre 4.

5.1 Justification du choix du langage de programmation Python

Comme il a été évoqué dans la section 4.5.1, le projet a été entièrement conçu dans le langage de programmation Python. Étant donné que la problématique du projet concerne le temps de calcul de la PDS, le choix d'un tel langage nécessite quelques explications.

Python est un langage de programmation interprété en accès libre couramment utilisé par la communauté scientifique. Son accès libre a grandement contribué à son évolution, et permis le développement de nombreuses librairies telles que : Numpy [1] et Scipy [2] conçues pour les opérations usuelles d'algèbre linéaire numérique. Le fait que Python soit un langage interprété facilite aussi son utilisation. De part la simplicité de sa syntaxe, très peu de lignes de code sont alors nécessaires pour réaliser de complexes implémentations. Pour ces raisons, le langage Python est maintenant devenu un standard autant dans le milieu académique qu'industriel.

Par ailleurs, Python offre la fonctionnalité de communiquer avec des langages de bas niveau notamment grâce à son extension Cython [3]. La principale différence entre les langages Python et Cython provient du fait que ce dernier est compilé au lieu d'être interprété. C'est cette compilation qui permet entre autres de faire le lien entre le Python et les langages de bas niveau tels que le C ou le Fortran. Les méthodes développées en Cython sont accessibles directement depuis Python. L'utilisation des logiciels MINOS, LANCELOT et XPRESS depuis Python a ainsi été réalisée grâce aux développements d'interfaces Cython. À noter que le logiciel LANCELOT, programmé en Fortran 90, a nécessité un traitement particulier. L'interface LANCELOT simple (Gould et al., 2008) a donc été modifiée pour y inclure le module ISO_C_BINDING afin d'assurer la communication avec Cython.

5.2 Implémentation de la PLS

Afin de pouvoir rivaliser avec les autres logiciels d'optimisation en terme de temps de calcul, plusieurs stratégies ont été exploitées dans l'implémentation de la PLS.

1. www.numpy.org
2. www.scipy.org
3. www.cython.org

Tout d'abord, on a pu constater que l'implémentation n'est pas entièrement en Python puisqu'elle repose sur une interface Cython. Au lieu de résoudre les sous-problèmes linéaires en pur Python, celle-ci transfère cette tâche à des méthodes de bas niveau. Ainsi, un logiciel en accès libre (CLP) et un logiciel commercial (XPRESS) ont été sélectionnés lors de l'analyse de performance. Le logiciel XPRESS a été choisi car il est actuellement utilisé pour l'opération chez Rio Tinto. L'interface XPRESS a été développée à l'intérieur de ce projet. Le logiciel CLP est utilisé grâce aux travaux de Towhidi et Orban (2016). L'interface CyLP est aussi réalisée en partie en Cython. Son implémentation est cependant plus riche et offre une plus grande flexibilité d'utilisation.

Pour limiter les manipulations matricielles, les contraintes linéaires sont traitées différemment au cours de la résolution du problème. En développant l'approximation de Taylor des contraintes linéaires (5.1), on constate que la recherche de Δx comme solution nécessite une mise à jour des contraintes à chaque fois que le pas est accepté.

$$Ax_k + A\Delta x \leq b \tag{5.1}$$

Toutefois, sachant que $\Delta x = x_{k+1} - x_k$, cette mise à jour peut être évitée en recherchant la solution x_{k+1} telle que :

$$Ax_{k+1} \leq b \tag{5.2}$$

Les contraintes linéaires n'ont alors pas besoin d'être modifiées durant la résolution.

Finalement, pour réduire le nombre d'appels au problème principal, l'implémentation de la PLS conserve temporairement chacune des informations du problème grâce à l'usage des propriétés et des décorateurs en Python. Ces propriétés permettent de définir plusieurs attributs à une variable ce qui est idéal pour la gestion de ces informations. Par conséquent, peu d'évaluations sont nécessaires durant son processus, ce qui est avantageux lorsqu'il est coûteux d'évaluer ces informations.

C'est ainsi qu'en dépit de l'utilisation de Python, l'implémentation de PLS est compétitive par rapport aux autres logiciels en terme de temps de calcul. Étant donné la confidentialité des logiciels commerciaux, seule l'implémentation de PLS avec CyLP est disponible sur `github.com/kenjydem/SLP.py`.

5.3 Étude de démarrage à chaud

En dehors des travaux présentés dans le chapitre 4, une étude de démarrage à chaud a été réalisée afin d'exploiter des informations provenant des sous-problèmes déjà résolus. Lors

du passage d'un sous-problème à l'autre, la PDS modifie uniquement la valeur de certaines bornes des contraintes. De ce fait, la formulation des sous-problèmes reste la même. Compte tenu de ces similarités, de précieuses informations peuvent alors être transmises à la suite de la résolution du problème précédant. Une analyse a alors été menée dans le but de déterminer s'il pouvait être avantageux de donner un point de départ basé sur la solution optimale du sous-problème précédent ou issu d'une autre stratégie. Trois stratégies ont été développées.

La première stratégie consiste simplement à utiliser les stratégies de démarrage à chaud fournies par les librairies utilisées. Ces stratégies permettent généralement de conserver de nombreuses informations numériques favorisant la résolution des problèmes.

La seconde stratégie conserve la solution du problème précédemment résolu, et utilise une heuristique afin de l'adapter au nouveau problème. Lorsque la PDS modifie l'un de ces sous-problèmes, les équations de balance de l'eau ne sont alors plus respectées. L'heuristique permet d'équilibrer le système par un processus de soutirage de l'eau en commençant par le réservoir en amont jusqu'à celui en aval.

Finalement, la dernière stratégie propose de résoudre un problème linéaire issu de la formulation des sous-problèmes. Étant donné qu'une partie de la fonction objectif et la majorité des contraintes de la PDS sont linéaires, le point initial provenant de cette stratégie est estimé par la résolution de ce petit problème linéaire. Cette stratégie ne bénéficie donc pas des informations des problèmes précédemment résolus.

Ces stratégies ont été testées sur le même banc d'essai que la comparaison des logiciels. Cependant, les résultats n'ont pas été très concluants, car aucune de ces stratégies ne s'est révélée dominante sur la plupart des problèmes. Une explication probable à ce résultat provient de la transition des problèmes. En effet, lors du passage d'un sous-problème à un autre, certaines transitions provoquent de brusques changements au niveau des variables d'état, ce qui peut perturber le démarrage à chaud.

CHAPITRE 6 CONCLUSION

6.1 Synthèse des travaux

La compagnie Rio Tinto, producteur d'aluminium dans la région du Saguenay-Lac-Saint-Jean, dispose des droits d'opération d'un système hydrique pour subvenir à ses besoins énergétiques. Pour gérer efficacement ce système, le groupe de ressources hydriques utilise l'algorithme de Programmation Dynamique Stochastique (PDS). Cependant, dans un contexte opérationnel, l'utilisation de la PDS pour l'évaluation de la politique de gestion du système est trop coûteuse en temps de calcul, notamment à cause d'un nombre important de sous-problèmes non linéaires à résoudre. Les travaux effectués dans ce mémoire ont donc pour mandat de réduire le temps de calcul de la PDS sans perte significative de la qualité de la politique. Pour parvenir à cet objectif, deux stratégies sont exploitées lors de l'évaluation de la politique de gestion. La première stratégie consiste à déterminer la méthode d'optimisation non linéaire la plus performante assistant la PDS au cours de sa résolution. La seconde utilise une approximation linéaire par morceaux des fonctions de production du modèle afin d'obtenir un modèle avec contraintes linéaires uniquement. Les résultats numériques démontrent que la combinaison de l'algorithme de Programmation Linéaire Séquentielle (PLS) et d'un algorithme de points intérieurs comme le logiciel IPOPT s'avère être un bon compromis entre l'amélioration de la vitesse d'exécution et la précision des résultats.

6.2 Limitations de la solution proposée

L'utilisation d'une interface pour la manipulation des logiciels d'optimisation possède de nombreux aspects intéressants. En effet, cela a permis de faciliter grandement leur manipulation et de centraliser les travaux sur le langage Python. Cependant, comme l'indique Gould et al. (2008), les interfaces simplifient souvent l'utilisation du logiciel, ce qui peut avoir un impact sur les performances des algorithmes. Par exemple, LANCELOT est conçu pour exploiter différents groupes de fonctions dans le but de structurer convenablement le problème. Or, dans l'interface utilisée, la fonction objectif est simplement formulée en toute généralité par une fonction non linéaire.

Tel qu'illustré par les résultats numériques au chapitre 4, la PLS présente un manque de robustesse par rapport aux autres méthodes lors de la résolution de problèmes avec des contraintes non linéaires. Ce manque de performance s'explique par la difficulté qu'apporte la gestion de la violation des contraintes non linéaires. Par conséquent, il n'est pas recommandé

de prendre pour acquis que la PLS est le meilleur algorithme pour toutes les formulations du problème.

Concernant l'utilisation des logiciels commerciaux, l'exploitation de stratégies de parallélisation pose aussi un problème au niveau opérationnel. En effet, lorsque le processus est parallélisé, une licence est requise sur chacun des coeurs utilisés. Par conséquent, les frais associés au prix d'un logiciel commercial peuvent alors s'avérer dispendieux.

6.3 Améliorations futures

Le développement d'un algorithme est une tâche continue. Sans cesse, des améliorations peuvent être apportées au comportement de l'algorithme. L'implémentation de la PLS proposée dans l'étude n'échappe pas à cette règle. Tout d'abord, il serait important d'essayer quelques stratégies améliorant la performance de l'algorithme en présence de contraintes non linéaires. L'une des possibilités serait d'implémenter des techniques de correction de second ordre ou d'exploiter des stratégies nonmonotone. Par la suite, il serait intéressant d'analyser les performances de la PLS avec l'utilisation d'un mécanisme de filtrage lors de l'acceptation d'un pas. À la différence de l'utilisation de la fonction mérite, cette stratégie évalue la qualité d'un pas en considérant distinctement la réduction de la fonction objective et la violation des contraintes. Le pas est alors accepté si celui-ci améliore l'un de ces deux aspects par rapport aux pas précédemment déjà évalués. Finalement, en considérant la proximité de la PLS avec celle de la PQS, une extension de l'implémentation pourrait s'orienter vers la résolution de sous-problème quadratique grâce à l'utilisation d'une approximation BFGS pour estimer la matrice hessienne.

Par rapport aux travaux sur la PDS, il serait judicieux de poursuivre l'étude des stratégies de démarrage à chaud. Compte tenu des résultats préliminaires obtenus à ce sujet, l'élaboration d'une stratégie d'ordonnancement des sous-problèmes permettrait d'obtenir une meilleure continuité pour leur résolution. Ceci serait certainement bénéfique afin d'améliorer encore une fois la vitesse de résolution.

RÉFÉRENCES

S. Anvari, S. J. Mousavi, et S. Morid, “Sampling/stochastic dynamic programming for optimal operation of multi-purpose reservoirs using artificial neural network-based ensemble streamflow predictions”, *Journal of Hydroinformatics*, vol. 16, no. 4, pp. 907–921, 2014. DOI : 10.2166/hydro.2013.236

S. Arreckx, D. Orban, et N. van Omme, “NLP.py : An object-oriented environment for large-scale optimization”, *Cahier du GERAD G-2016-42, GERAD, Montréal, QC, Canada*, 2016. DOI : 10.13140/RG.2.1.2846.6803

C. Audet et D. Orban, “Finding optimal algorithmic parameters using derivative-free optimization”, *SIAM Journal on Optimization*, vol. 17, no. 3, pp. 642–664, 2006. DOI : 10.1137/040620886

C. Audet, K.-C. Dang, et D. Orban, “Optimization of algorithms with OPAL”, *Mathematical Programming Computation*, vol. 6, no. 3, pp. 233–254, 2014.

M. T. Barros, F. T. Tsai, S.-l. Yang, J. E. Lopes, et W. W. Yeh, “Optimization of large-scale hydropower system operations”, *Journal of Water Resources Planning and Management*, vol. 129, no. 3, pp. 178–188, 2003. DOI : 10.1061/(ASCE)0733-9496(2003)129:3(178)

M. S. Bazaraa, H. D. Sherali, et C. M. Shetty, “Methods of feasible directions”, dans *Nonlinear Programming*, 3e éd. John Wiley & Sons, Inc., 2005, pp. 537–653. DOI : 10.1002/0471787779.ch10

R. Bellman, *Dynamic Programming*, 1er éd. Princeton, NJ, USA : Princeton University Press, 1957.

D. P. Bertsekas, *Dynamic programming and optimal control*, 3e éd. Athena Scientific Belmont, MA, 2005, vol. 1.

I. Bongartz et P. L. Toint, “A numerical comparison between the LANCELOT and MINOS packages for large-scale nonlinear optimization : the complete results”, Technical Report 97/14, Department of Mathematics, Namur, Belgium, 1997, Rapp. tech., 1997.

M. Breton, S. Hachem, et A. Hammadia, “Accounting for losses in the optimization of production of hydroplants”, *IEEE Transactions on Energy Conversion*, vol. 19, no. 2, pp. 346–351, 2004. DOI : 10.1109/TEC.2004.827043

R. H. Byrd, M. E. Hribar, et J. Nocedal, “An interior point algorithm for large-scale nonlinear programming”, *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, 1999. DOI : 10.1137/S1052623497325107

R. H. Byrd, N. I. Gould, J. Nocedal, et R. A. Waltz, “An algorithm for nonlinear optimization using linear programming and equality constrained subproblems”, *Mathematical Programming*, vol. 100, no. 1, pp. 27–48, 2003. DOI : 10.1007/s10107-003-0485-4

R. H. Byrd, J. Nocedal, et R. A. Waltz, “KNITRO : An integrated package for nonlinear optimization”, *Large-Scale Nonlinear Optimization*, vol. 83, pp. 35–59, 2006.

P.-L. Carpentier, M. Gendreau, et F. Bastin, “Long-term management of a hydroelectric multireservoir system under uncertainty using the progressive hedging algorithm”, *Water Resources Research*, vol. 49, no. 5, pp. 2812–2827, 2013. DOI : 10.1002/wrcr.20254

A. R. Conn, N. I. M. Gould, et P. L. Toint, “Lancelot : A FORTRAN package for large-scale nonlinear optimization (release A)”, *Springer-Verlag New York, Inc.*, 1992.

P. Côté et R. Leconte, “Comparison of stochastic optimization algorithms for hydropower reservoir operation with ensemble streamflow prediction”, *Journal of Water Resources Planning and Management*, vol. 142, no. 2, p. 04015046, 2015. DOI : 10.1061/(ASCE)WR.1943-5452.0000575

P. Côté, D. Haguma, R. Leconte, et S. Krau, “Stochastic optimisation of hydro-quebec hydropower installations : a statistical comparison between sdp and ssdp methods”, *Canadian Journal of Civil Engineering*, vol. 38, no. 12, pp. 1427–1434, 2011. DOI : 10.1139/l11-101

R. Courant, “Variational methods for the solution of problems of equilibrium and vibrations”, *Bulletin of the American Mathematical Society*, vol. 49, pp. 1–23, 1943.

C. Davidsen, S. J. Pereira-Cardenal, S. Liu, X. Mo, D. Rosbjerg, et P. Bauer-Gottwein, “Using stochastic dynamic programming to support water resources management in the ziya river basin, china”, *Journal of Water Resources Planning and Management*, vol. 141, no. 7, p. 04014086, 2014. DOI : 10.1061/(ASCE)WR.1943-5452.0000482

Q. Desreumaux, P. Côté, et R. Leconte, “Role of hydrologic information in stochastic dynamic programming : a case study of the kemano hydropower system in british columbia”, *Canadian Journal of Civil Engineering*, vol. 41, no. 9, pp. 839–844, 2014. DOI : 10.1139/cjce-2013-0370

D. Etkin, P. Kirshen, D. Watkins, C. Roncoli, M. Sanon, L. Some, Y. Dembele, J. Sanfo, J. Zoungrana, et G. Hoogenboom, “Stochastic programming for improved multiuse reservoir operation in burkina faso, west africa”, *Journal of Water Resources Planning and Management*, vol. 141, no. 3, p. 04014056, 2015. DOI : 10.1061/(ASCE)WR.1943-5452.0000396

B. A. Faber, “Real-time reservoir optimization using ensemble streamflow forecasts”, Thèse de doctorat, Cornell University, 2001.

R. Fletcher et E. S. de la Maza, “Nonlinear programming and nonsmooth optimization by successive linear programming”, *Mathematical Programming*, vol. 43, no. 1, pp. 235–256, 1989.

R. Fletcher et S. Leyffer, “Nonlinear programming without a penalty function”, *Mathematical programming*, vol. 91, no. 2, pp. 239–269, 2002. DOI : 10.1007/s101070100244

E. Foufoula-Georgiou et P. Kitandis, “Gradient dynamic programming for stochastic optimal control of multidimensional water resources systems”, *Water Resources Research*, vol. 24, no. 8, pp. 1345–1359, 1988. DOI : 10.1029/WR024i008p01345

K. R. Frisch, “The logarithmic potential method of convex programming”, University Institute of Economics, Oslo, Rapp. tech., 1955.

P. E. Gill, W. Murray, et M. A. Saunders, “SNOPT : An SQP algorithm for large-scale constrained optimization”, *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005. DOI : 10.1137/S0036144504446096

Q. Goor, R. Kelman, et A. Tilmant, “Optimal multipurpose-multireservoir operation model with variable productivity of hydropower plants”, *Journal of Water Resources Planning and Management*, vol. 137, no. 3, pp. 258–267, 2010. DOI : 10.1061/(ASCE)WR.1943-5452.0000117

N. I. M. Gould, D. Orban, et P. L. Toint, “GALAHAD, a library of thread-safe fortran 90 packages for large-scale nonlinear optimization”, *ACM Transactions on Mathematical Software (TOMS)*, vol. 29, no. 4, pp. 353–372, 2003.

——, “LANCELOT_simple : A simple interface for LANCELOT B”, *Cahier du GERAD, G-2008-11, GERAD, Montréal, QC, Canada*, 2008.

R. E. Griffith et R. A. Stewart, “A nonlinear programming technique for the optimization of continuous processing systems”, *Management science*, vol. 7, no. 4, pp. 379–392, 1961. DOI : 10.1287/mnsc.7.4.379

M. R. Hestenes, “Multiplier and gradient methods”, *Journal of optimization theory and applications*, vol. 4, no. 5, pp. 303–320, 1969.

S. Johnson, J. R. Stedinger, C. Shoemaker, Y. Li, et J. Tejada-Guibert, “Numerical solution of continuous-state dynamic prgramming using linear and spline interpolation”, *Operations research*, vol. 41, no. 3, 1993. DOI : 10.1287/opre.41.3.484

J. Kelman, J. R. Stedinger, L. A. Cooper, E. Hsu, et S.-Q. Yuan, “Sampling stochastic dynamic programming applied to reservoir operation”, *Water Resources Research*, vol. 26, no. 3, pp. 447–454, 1990. DOI : 10.1029/WR026i003p00447

Y.-O. Kim, H.-I. Eum, E.-G. Lee, et I. H. Ko, “Optimizing operational policies of a korean multireservoir system using sampling stochastic dynamic programming with ensemble streamflow prediction”, *Journal of Water Resources Planning and Management*, vol. 133, no. 1, pp. 4–14, 2007. DOI : 10.1061/(ASCE)0733-9496(2007)133:1(4)

J. W. Labadie, “Optimal operation of multireservoir systems : State-of-the-art review”, *Journal of water resources planning and management*, vol. 130, no. 2, pp. 93–111, 2004. DOI : 10.1061/(ASCE)0733-9496(2004)130:2(93)

X. Li, J. Wei, T. Li, G. Wang, et W. W.-G. Yeh, “A parallel dynamic programming algorithm for multi-reservoir system optimization”, *Advances in Water Resources*, vol. 67, pp. 1–15, 2014. DOI : 10.1016/j.advwatres.2014.01.002

A. Magnani et S. P. Boyd, “Convex piecewise-linear fitting”, *Optimization and Engineering*, vol. 10, no. 1, pp. 1–17, 2009. DOI : 10.1007/s11081-008-9045-3

J. J. Moré et S. M. Wild, “Benchmarking derivative-free optimization algorithms”, *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 172–191, 2009. DOI : 10.1137/080724083

P. P. Mujumdar et B. Nirmala, “A bayesian stochastic optimization model for a multireservoir hydropower system”, *Water resources management*, vol. 21, no. 9, pp. 1465–1485, 2007. DOI : 10.1007/s11269-006-9094-3

B. A. Murtagh et M. A. Saunders, “Large-scale linearly constrained optimization”, *Mathematical programming*, vol. 14, no. 1, pp. 41–72, 1978. DOI : 10.1007/BF01588950

——, “Minos 5.51 user’s guide”, Technical report SOL 83-20R, Rapp. tech., 2003.

J. Nocedal et S. J. Wright, *Numerical optimization*, 2e éd. Springer Science & Business Media, 2006.

C.-s. Peng et N. Buras, “Dynamic operation of a surface water resources system”, *Water resources research*, vol. 36, no. 9, pp. 2701–2709, 2000. DOI : 10.1029/2000WR900169

M. Pereira et L. Pinto, “Stochastic optimization of a multireservoir hydroelectric system : a decomposition approach”, *Water resources research*, vol. 21, no. 6, pp. 779–792, 1985. DOI : 10.1029/WR021i006p00779

R. E. Perez, P. W. Jansen, et J. R. R. A. Martins, “pyOpt : A Python-based object-oriented framework for nonlinear constrained optimization”, *Structures and Multidisciplinary Optimization*, vol. 45, no. 1, pp. 101–118, 2012. DOI : 10.1007/s00158-011-0666-3

M. J. D. Powell, “A method for nonlinear constraints in minimization problems”, dans *Optimization*, R. Fletcher, éd. New York : Academic Press, 1969, pp. 283–298.

L. Raso, P.-O. Malaterre, et J.-C. Bader, “Effective streamflow process modeling for optimal reservoir operation using stochastic dual dynamic programming”, *Journal of water resources planning and management*, vol. 143, no. 4, p. 04017003, 2017. DOI : 10.1061/(ASCE)WR.1943-5452.0000746

C. Rougé et A. Tilmant, “Using stochastic dual dynamic programming in problems with multiple near-optimal solutions”, *Water Resources Research*, vol. 52, no. 5, pp. 4151–4163, 2016. DOI : 10.1002/2016WR018608

J. A. Tejada-Guibert, J. R. Stedinger, et K. Staschus, “Optimization of value of cvp’s hydropower production”, *Journal of Water Resources Planning and Management*, vol. 116, no. 1, pp. 52–70, 1990. DOI : 10.1061/(ASCE)0733-9496(1990)116:1(52)

J. A. Tejada-Guibert, S. A. Johnson, et J. R. Stedinger, “Comparison of two approaches for implementing multireservoir operating policies derived using stochastic dynamic programming”, *Water resources research*, vol. 29, no. 12, pp. 3969–3980, 1993. DOI : 10.1029/93WR02277

A. Tilmant, M. Vanclooster, L. Duckstein, et E. Persoons, “Comparison of fuzzy and nonfuzzy optimal reservoir operating policies”, *Journal of Water Resources Planning and Management*, vol. 128, no. 6, pp. 390–398, 2002. DOI : 10.1061/(ASCE)0733-9496(2002)128:6(390)

M. Towhidi et D. Orban, “Customizing the solution process of coin-or’s linear solvers with python”, *Mathematical Programming Computation*, vol. 8, no. 4, pp. 377–391, 2016.

A. Turgeon, “Solving a stochastic reservoir management problem with multilag autocorrelated inflows : solving a stochastic reservoir management problem”, *Water Resources Research*, vol. 41, no. 12, pp. n/a–n/a, 2005. DOI : 10.1029/2004WR003846

——, “Solving daily reservoir management problems with dynamic programming”, *Cahier du Gerad*, no. G-2006-22, 2006.

A. Wächter et L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”, *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006. DOI : 10.1007/s10107-004-0559-y

R. A. Waltz, J. L. Morales, J. Nocedal, et D. Orban, “An interior algorithm for nonlinear optimization that combines line search and trust region steps”, *Mathematical programming*, vol. 107, no. 3, pp. 391–408, 2006. DOI : 10.1007/s10107-004-0560-5

R. B. Wilson, *A simplicial algorithm for concave programming*. Graduate School of Business Administration, George F. Baker Foundation, Harvard University, 1963.

J. Yi, J. W. Labadie, et S. Stitt, “Dynamic optimal unit commitment and loading in hydropower systems”, *Journal of water resources planning and management*, vol. 129, no. 5, pp. 388–398, 2003. DOI : 10.1061/(ASCE)0733-9496(2003)129:5(388)

L. Zéphyr, P. Lang, et B. F. Lamond, “Adaptive monitoring of the progressive hedging penalty for reservoir systems management”, *Energy Systems*, vol. 5, no. 2, pp. 307–322, 2014. DOI : 10.1007/s12667-013-0110-4

Z. Zhang, S. Zhang, Y. Wang, Y. Jiang, et H. Wang, “Use of parallel deterministic dynamic programming and hierarchical adaptive genetic algorithm for reservoir operation optimization”, *Computers & Industrial Engineering*, vol. 65, no. 2, pp. 310–321, 2013. DOI : 10.1016/j.cie.2013.02.003

T. Zhao, J. Zhao, et D. Yang, “Improved dynamic programming for hydropower reservoir operation”, *Journal of Water Resources Planning and Management*, vol. 140, no. 3, pp. 365–374, 2012. DOI : 10.1061/(ASCE)WR.1943-5452.0000343