Les Cahiers du GERAD ISSN: 0711–2440

Numerical methods for stochastic dynamic programming with application to hydropower optimization

P. Côté, K. Demeester,
D. Orban, M. Towhidi

G–2017–64

August 2017

Numerical methods for stochastic dynamic programming with application to hydropower optimization

Pascal Côté[a]

Kenjy Demeester[b]

Dominique Orban[b]

Mehdi Towhidi[c]

[a] *GERAD & Rio Tinto, Saguenay (Québec) Canada, G7S 4R5*

[b] *GERAD & Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7*

[c] *Kronos Canadian Systems, Montréal (Québec) Canada, H3V 1H8*

kenjy.demeester@gerad.ca
pascal.cote@riotinto.ca
dominique.orban@gerad.ca
mehdi.towhidi@kronos.com

August 2017

Les 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 reduces the total run time by a factor of about 3.3 compared to IPOPT alone using nonlinear hydropower constraints during both phases.

Keywords: Hydropower optimization, stochastic dynamic programming, sequential linear programming

Résumé : La Programmation Dynamique Stochastique (PDS) est une puissante méthode applicable aux problèmes mutli-étapes non-convexes et stochastiques. Nous étudions l'impact de la formulation des sous-problèmes et du choix de la méthode d'optimisation utilisée pour leurs résolutions sur la performance générale de la PDS dans le contexte de la gestion de réservoirs hydroélectrique. Nous rapportons les résultats numériques sur un système opérationnel et comparons plusieurs logiciels de pointe en optimisation. Dans un premier ensemble de tests, les sous-problèmes d'optimisation comportent des contraintes de production hydroélectrique non linéaires, tandis qu'un second ensemble utilise une approximation linéaire de ces contraintes. Nos résultats montrent que, lorsque les contraintes hydroélectriques linéaires sont acceptées au cours de l'évaluation de la politique, la Programmation Linéaire Séquentielle (PLS) nécessite le plus petit nombre d'évaluations de fonctions tout en étant la plus robuste en termes de nombre de sous-problèmes résolus avec succès. D'autre part, IPOPT offre la meilleure performance au cours de la phase de simulation en présence de contraintes hydroélectriques non linéaires, où la PLS est moins fiable. Une combinaison de la PLS pour l'évaluation de la politique en utilisant des contraintes hydroélectriques linéaires avec IPOPT dans la phase de simulation en utilisant des contraintes hydroélectriques non linéaires entraîne une amélioration du coût cumulatif annuel moyen et réduit le temps d'exécution total d'un facteur d'environ 3.3 par rapport à IPOPT seul en utilisant les contraintes hydroélectriques non linéaires au cours des deux phases.

Mots clés : Optimisation hydroélectrique, programmation dynamique stochastique, programmation linéaire séquentielle

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 for the entire 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 and Nirmala, 2007; Anvari, Mousavi, and Morid, 2014; Davidsen, Pereira-Cardenal, Liu, Mo, Rosbjerg, and Bauer-Gottwein, 2014; Desreumaux, Côté, and Leconte, 2014). The Rio Tinto system is composed of five power houses and three reservoirs, and is managed by SDP (Côté and Leconte, 2015).

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, 1995).

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 2 summarizes the SDP approach in the context of hydropower management. In Section 3, we describe the case study of the Saguenay-Lac-Saint-Jean (SLSJ) hydroelectric power system operated by Rio Tinto. Section 4 presents an overview of nonlinear methods and a detailed description of the SLP process. Finally, numerical results and discussions are presented in Section 5.

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 (1995) 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

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

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, i.e., whose elements are either ± 1 or 0.

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

$$\begin{aligned}
\underset{\mathbf{u}_t}{\text{minimize}} \quad & \sum_{t=1}^{T} G_t(\mathbf{s}_t, \mathbf{u}_t) + J_{T+1}(\mathbf{s}_{T+1}) && && \text{(2a)}\\
\text{subject to} \quad & \mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{q}_t - C\,\mathbf{u}_t, && t = 1, \ldots, T, && \text{(2b)}\\
& s_{\min} \le \mathbf{s}_t \le s_{\max}, && t = 1, \ldots, T+1, && \text{(2c)}\\
& 0 \le \mathbf{u}_t \le u_{\max}, && t = 1, \ldots, T, && \text{(2d)}
\end{aligned}$$

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 Bellman (1957) principle of dynamic programming as

$$\begin{aligned}
J_t(\mathbf{s}_t) := \underset{\mathbf{u}_t}{\text{minimize}} \quad & G_t(\mathbf{s}_t, \mathbf{u}_t) + J_{t+1}(\mathbf{s}_{t+1}), && \text{(3a)}\\
\text{subject to} \quad & \mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{q}_t - C\,\mathbf{u}_t, && \text{(3b)}\\
& s_{\min} \le \mathbf{s}_{t+1} \le s_{\max}, && \text{(3c)}\\
& 0 \le \mathbf{u}_t \le u_{\max}. && \text{(3d)}
\end{aligned}$$

The notation in (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

$$\pi_t(\mathbf{s}_t) \in \arg\min_{\mathbf{u}_t} \ (3).$$

Note that each evaluation of J_t requires the solution of (3) at time steps $t, \ldots, T$. The SDP algorithm naturally proceeds backwards, first solving (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, Haguma, Leconte, and Krau, 2011; Turgeon, 2005; Zhao, Zhao, and Yang, 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, Stedinger, Shoemaker, Li, and Tejada-Guibert, 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. The model of Tejada-Guibert, Johnson, and Stedinger (1995) estimates $\mathbf{q}_t$ as a linear function of $\mathbf{q}_{t-1}$. In such an approach, $\mathbf{q}_{t-1}$ temporarily becomes a state variable and its domain is added to the grid Ω. The counterpart of (3) can be written

$$\begin{aligned}
\tilde{J}_t(\mathbf{s}_t, \mathbf{q}_{t-1}) := \underset{\mathbf{u}_t}{\text{minimize}} \quad & \underset{\mathbf{q}_t|\mathbf{q}_{t-1}}{\mathbf{E}} \Big\{ G_t(\mathbf{s}_t, \mathbf{u}_t) + \tilde{J}_{t+1}(\mathbf{s}_{t+1}, \mathbf{q}_t) \Big\} && \text{(4a)}\\
\text{subject to} \quad & \mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{q}_t - C\,\mathbf{u}_t, && \text{(4b)}\\
& s_{\min} \le \mathbf{s}_{t+1} \le s_{\max}, && \text{(4c)}\\
& 0 \le \mathbf{u}_t \le u_{\max}, && \text{(4d)}
\end{aligned}$$

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 (4a), one possibility is to discretize the domain of $\mathbf{q}_t$ into M points $\mathbf{q}_t^1, \dots, \mathbf{q}_t^M$ and use the approximation

$$\sum_{j=1}^{M} \Big(G_t(\mathbf{s}_t, \mathbf{u}_t) + \tilde{J}_{t+1}(\mathbf{s}_{t+1}, \mathbf{q}_t^j) \Big)\ \mathbf{P}(\mathbf{q}_t^j \mid \mathbf{q}_{t-1}^i), \tag{5}$$

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 and Leconte, 2015).

Tilmant, Vanclooster, Duckstein, and Persoons (2002) explain that in order to take into account the yearly periodicity of an optimal policy and the infinite horizon of (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) with objective (5) in a loop in which $\tilde{J}_{T+1}$ is set to the $\tilde{J}_1$ of the previous pass. The procedure is summarized in Algorithm 2.1.

The main computational cost of SDP resides in the efficient solution of (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 2.1 Stochastic Dynamic Programming (SDP)

```
0: Initialization
1:     Set the terminal cost $\tilde{J}_{T+1} = 0$,
2:     Compute the transition probabilities $\mathbf{P}(\mathbf{q}_t^j \mid \mathbf{q}_{t-1}^i)$ for $i, j = 1, \dots, M$ and $t = 1, \dots, T$.
3: Approximation                                   // Approximation of the operational policy
4:     repeat
5:         for $t = T, T-1, \dots, 1$ do
6:             for all $\mathbf{s}_t \in \Omega$ do
7:                 Solve (4) in which the objective function is replaced with (5).
8:             Obtain $\tilde{J}_t()$ via interpolation.
9:         $\tilde{J}_{T+1} \leftarrow \tilde{J}_1$.
10:    until policy converges
11: Simulation                                     // Simulation process
12:    for a set of inflow scenarios do
13:        for $t = 1, 2, \dots, T$ do
14:            Solve (4) in which the objective function is replaced with (5)
```

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

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 1, is composed of five generating stations and three reservoirs, for an installed capacity of 3100 MW.

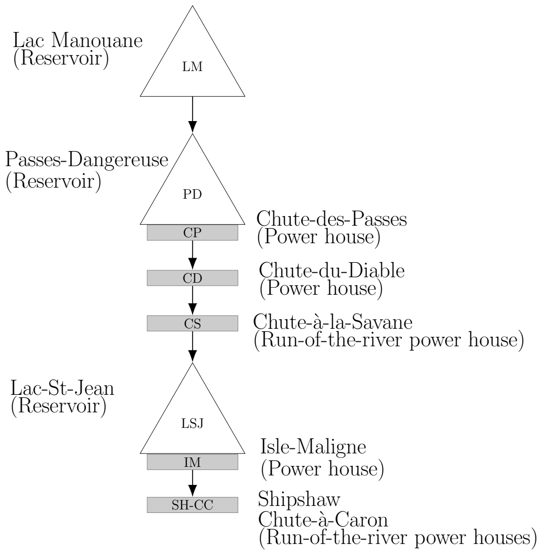

Figure 1: Saguenay-Lac-Saint-Jean hydroelectric system.

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.

3.2 Rio Tinto problem: modelling

By definition, the power produced by a turbine can be approximated by

$$P(\mathbf{u}_t, \bar{\mathbf{s}}_t) = \mathbf{u}_t \, \eta(\mathbf{u}_t) \, h(\bar{\mathbf{s}}_t) \, g, \tag{6}$$

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 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, Labadie, and Stitt, 2003; Breton, Hachem, and Hammadia, 2004). Like (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 2 represents the hydropower function 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 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 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.

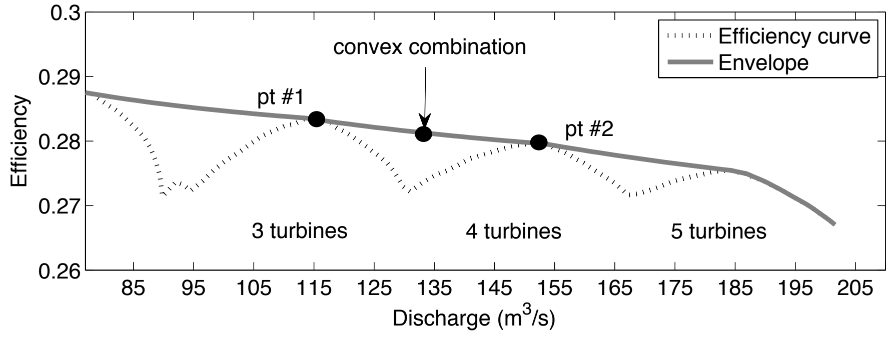

Figure 2: Powerhouse efficiency curve for a specific net head.

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 §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

$$p_{i,t} \le \alpha_{i,k}\bar{s}_{i,t} + \beta_{i,k}u_{i,t} + \gamma_{i,k}, \qquad i = 1,\dots,5,\ k = 1,\dots,K. \tag{7}$$

where $\alpha_{i,k}, \beta_{i,k}, \gamma_{i,k} \in \mathbb{R}$ are determined using a Gauss-Newton heuristic (Magnani and 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 1 summarizes the problem dimensions in each formulation.

The SDP subproblem is defined as

$$\tilde{J}_t(\mathbf{s}_t, \mathbf{q}_{t-1}) := \underset{\mathbf{u}_t}{\text{minimize}} \quad \underset{\mathbf{q}_t|\mathbf{q}_{t-1}}{\mathbf{E}} \left\{ G_t(\mathbf{s}_t, \mathbf{u}_t) + \tilde{J}_{t+1}(\mathbf{s}_{t+1}, \mathbf{q}_t) \right\}, \tag{8a}$$

$$\text{subject to } \mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{q}_t - C\mathbf{u}_t, \tag{8b}$$

$$p_{i,t} = P_i(u_{i,t}, \bar{s}_{i,t}), \qquad i = 1,\dots,5, \tag{8c}$$

$$p_{\min} \le \sum_{i=1}^{5} p_{i,t}, \tag{8d}$$

$$s_{\min} \le \mathbf{s}_{t+1} \le s_{\max}, \tag{8e}$$

$$0 \le \mathbf{u}_t \le u_{\max}, \tag{8f}$$

where $p_{\min}$ corresponds to the minimum energy required to maintain the aluminum smelter operational. When the splines are used, P_i represent the spline interpolation of the hydropower production envelope at the i-th powerhouse. The second formulation is similar to (8) except that (8c) is formulated as (7).

Table 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 §5 to compare both formulation.

4 Brief description of optimization methods considered

4.1 Overview

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

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\; f(x) \quad \text{subject to}\; g_l \le g(x) \le g_u,\; h(x) = 0,\; l \le x \le u, \tag{9}$$

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 (9) that we consider fit in three categories: augmented Lagrangian methods, sequential linear or quadratic optimization, and interior-point methods (Bazaraa, Sherali, and Shetty, 2005; Nocedal and Wright, 2006). We summarize each in turn in the next three sections.

4.1.1 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 and Saunders, 1978) and LANCELOT (Conn, Gould, and Toint, 1992) packages were the first available implementations and they remain reference implementations.

For the simplicity of discussion, let us assume temporarily that (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

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\; f(x) + \mu_k^T h(x) + \tfrac{1}{2}\rho_k \|h(x)\|_2^2 \quad \text{subject to}\; l \le x \le u, \tag{10}$$

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 (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 (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 (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.

4.1.2 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 and Stewart (1961) for proposing to solve (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

$$
\begin{aligned}
\underset{\Delta x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x_k) + \nabla f(x_k)^T \Delta x && \text{(11a)}\\
\text{subject to} \quad & g_l \le g(x_k) + \nabla g(x_k) \Delta x \le g_u, && \text{(11b)}\\
& h(x_k) + \nabla h(x_k) \Delta x = 0, && \text{(11c)}\\
& l \le x_k + \Delta x \le u, && \text{(11d)}\\
& \|\Delta x\|_\infty \le \Delta_k, && \text{(11e)}
\end{aligned}
$$

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 §4.2.

Wilson (1963) suggested Sequential Quadratic Programming (SQP) as an improvement over SLP. SQP solves (9) by a sequence of quadratic programs (QP). Each QP is similar to (11) except that (11a) is replaced with the second-order expansion

$$
f(x_k) + \nabla f(x_k)^T \Delta x + \tfrac{1}{2} \Delta x^T B_k \Delta x, \tag{12}
$$

where $B_k = B_k^T$ is an approximation of the Hessian of the Lagrangian of (9) for certain estimates of the optimal multipliers.

SQP methods can be of the trust-region kind and solve a subproblem with constraints (11b)–(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, Murray, and Saunders, 2005) and FilterSQP (Fletcher and 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 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 and Wright, 2006, Chapter 18.5). The method proceeds in two steps. In the first step, a linear problem similar to (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, Gould, Nocedal, and Waltz (2003).

4.1.3 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 (9) by a succession of subproblems using a logarithmic barrier to prevent violation of the inequality constraints and bounds:

$$
\begin{aligned}
\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) - \rho_k \sum_{i=0}^{m_I} \left(\log(g_i(x) - g_{l_i}) + \log(g_{u_i} - g_i(x))\right)\\
& \qquad - \rho_k \sum_{i=0}^{n} \left(\log(x_i - x_{l_i}) + \log(x_{u_i} - x_i)\right) \qquad \text{(13)}\\
\text{subject to} \quad & h(x) = 0.
\end{aligned}
$$

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

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 and Biegler, 2006) and KNITRO/DIRECT (Waltz, Morales, Nocedal, and Orban, 2006) are prime examples of linesearch implementations. Byrd, Hribar, and Nocedal (1999) propose a trust-region implementation in KNITRO/CG. KNITRO implementation details are given by Byrd, Nocedal, and Waltz (2006).

4.2 Sequential Linear Programming

According to the discussion of §3 and to Figure 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 (11b)–(11c), the constraints of (9) may be violated during the iterations. In addition, (11) could be infeasible even though (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 (9) is to introduce an ℓ_1-penalty term into the objective of (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

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

is the ℓ_1 infeasibility measure for (9). Note that SLP ensures feasibility at each iteration with respect to linear constraints that might be present in (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

$$\begin{aligned} \underset{x,p,q}{\text{minimize}} \quad & \nabla f(x_k)^T x + \rho_k \sum_{i=0}^{m_I} p_i + \rho_k \sum_{i=0}^{m_E} q_i \\ \text{subject to} \quad & g_l - p \le g(x_k) + \nabla g(x_k)(x - x_k) \le g_u + p, \\ & -q \le h(x_k) + \nabla h(x_k)(x - x_k) \le q, \\ & b_l \le Ax \le b_u, \quad l \le x \le u, \\ & \|x - x_k\|_\infty \le \Delta_k, \end{aligned} \tag{14}$$

where we removed constant terms from the objective.

A solution $(\hat{x}, \hat{p}, \hat{q})$ of (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.,

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

and

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

If $\theta(\hat{x}) \le \epsilon_a$ 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 $\gamma_1 \ge 1$. On the other hand, if $\theta(\hat{x}) > \epsilon_a$ or $R_k \le \eta_1$, the step is rejected and the trust region is shrunk by a factor $\gamma_2 < 1$.

Before solving the first subproblem (14), we check whether the linear constraints of (9) are satisfied at the initial guess x_0. If they are not satisfied, we project x_0 into the linear constraints by solving

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

We follow (Bazaraa et al., 2005, Theorem 10.3.1) and report $\hat{x}$ as stationary for (9) if

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

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 (9). Our update has the form

$$\rho_{k+1} = \min\{\, \|\lambda_k\|_\infty,\, \rho_{\max}\} \tag{15}$$

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 (14) by the LP solver. In addition, we also follow (Nocedal and Wright, 2006, Algorithm 18.5) and only update ρ_k when $p > 0$ or $q > 0$ in (14), i.e., the linearized constraints of (9) are violated.

Algorithm 4.1 summarizes the sequential linear programming approach.

Algorithm 4.1 Sequential Linear Programming (SLP)

```
0: Initialization: Choose $x_0 \in \mathbb{R}^n$, trust region parameters $\Delta_0 = \min\{10, 0.1\|\nabla f(x_0)\|_\infty\}$, $0 < \eta_1 < \eta_2 < 1$, $\gamma_2 < 1 \le \gamma_1$,
   penalty parameter $0 < \rho_0 \le \rho_{\max}$, stopping tolerances $\epsilon_p$, $\epsilon_a$, and $\epsilon_d > 0$.
1: Solve (14) to compute an improving candidate $(\hat{x}, \hat{p}, \hat{q})$ and define $\Delta x = \hat{x} - x_k$.
2: if $\|\Delta x\|_\infty \le \epsilon_d$ and $\theta(\hat{x}) \le \epsilon_p$ 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 $\theta(\hat{x}) \le \epsilon_a$ 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 $\rho_k$ using (15).
15: Set $k \leftarrow k + 1$ go to 1.
```

5 Implementation and numerical results

5.1 Implementation

We implemented Algorithm 4.1 in the Python programming language as part of the NLP.py Python development environment for linear and nonlinear optimization (Arreckx, Orban, and van Omme, 2016). Because (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 and 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, Dang, and Orban (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 and 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, Jansen, and Martins, 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, Orban, and Toint, 2003). In our comparative study, LANCELOT is used via its simplified LANCELOT_simple interface (Gould, Orban, and Toint, 2008).

We compare the efficiency of the various solvers in the next two sections. In §5.2, we perform benchmarks to estimate the computational effort required to solve each formulation of SDP subproblems. In §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

5.2 Benchmarks

We use a set of $3,750$ subproblems from SDP with the formulation (8). I 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 and Wild (2009) with the convergence test

$$r_k \leq r_{\text{best}} + \tau(r_0 - r_{\text{best}}), \tag{16}$$

where $\tau > 0$ is a tolerance, x_0 is the initial guess, r_k is the first-order optimality residual at iteration k 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 (11a)–(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 (11a)–(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 4.1.

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 (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 (16). Figure 3 shows the performance profiles with a factor $\tau = 10^{-3}$.

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.

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

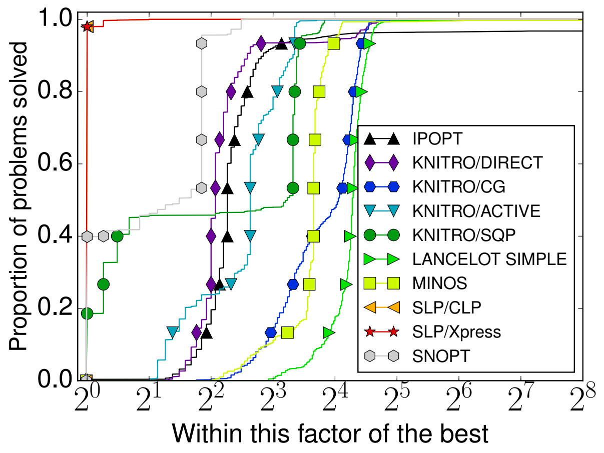
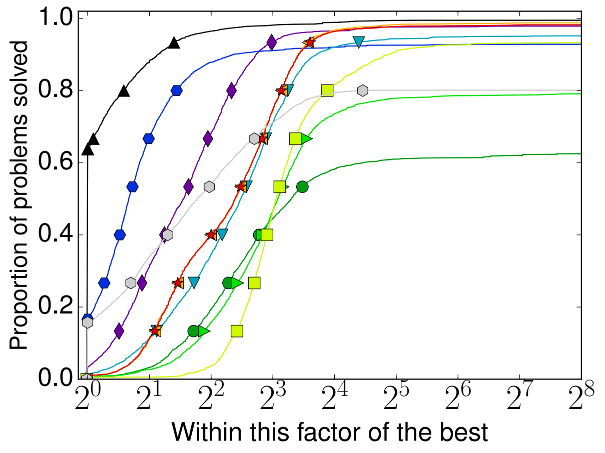

Figure 3: Performance profiles (logarithmic scale) of the number of objective function evaluations required to satisfy (16) with $\tau = 10^{-3}$ for $3,750$ subproblems from SDP. The left plot shows results for the formulation with linear constraints (hyperplanes) and the right plot those for the formulation with nonlinear constraints using cubic splines.

As a complement to performance profiles, data profiles (Moré and Wild, 2009) illustrate the proportion of problems satisfying (16) within a given budget of objective evaluations. Figure 4 shows the data profiles for $\tau = 10^{-3}$ with both problem formulations.

The data profiles show that SLP and SNOPT attain (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 (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 (16) within 700 objective evaluations. SNOPT is one of them, while SLP is not, though it requires over 500 evaluations.

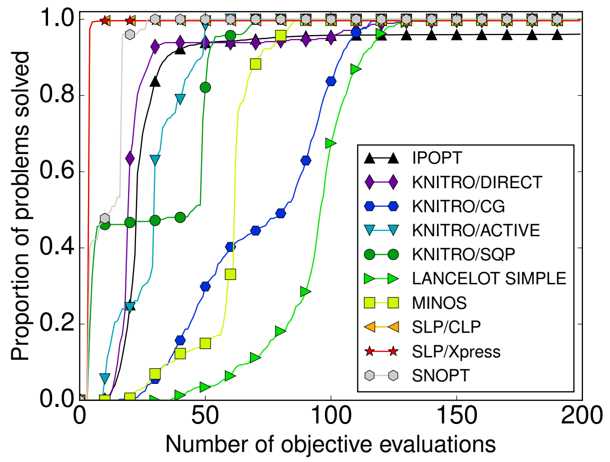
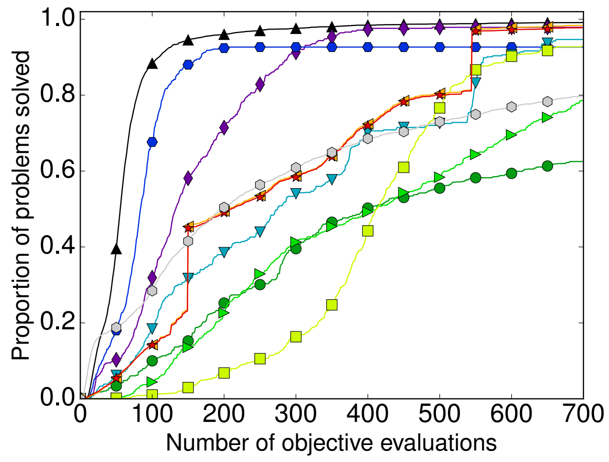

Figure 4: Data profiles with $\tau = 10^{-3}$ for $3,750$ subproblems from SDP. The left plot shows results for the formulation with linear constraints (hyperplanes) and the right plot those for the formulation with nonlinear constraints using cubic splines.

5.3 Comparison of optimal operation policies

We now compare several solvers during the simulation phase, i.e., Step 11 of Algorithm 2.1. The simulation evaluates the operating policy identified by SDP in Step 3 of Algorithm 2.1. Throughout this section, we use a set $\mathcal{Q}$ composed of 25 annual inflow scenario from historical data. 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 §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 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 §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 2 summarizes our results.

Table 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.

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 simulation process always uses nonlinear constraints. The columns “Simul” and “Approx” refer to the simulation and approximation steps of Algorithm 2.1.

The results of Table 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 5.

The top row of Figure 5 reveals that using linear constraints during the approximation step (Step 3) of Algorithm 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 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.

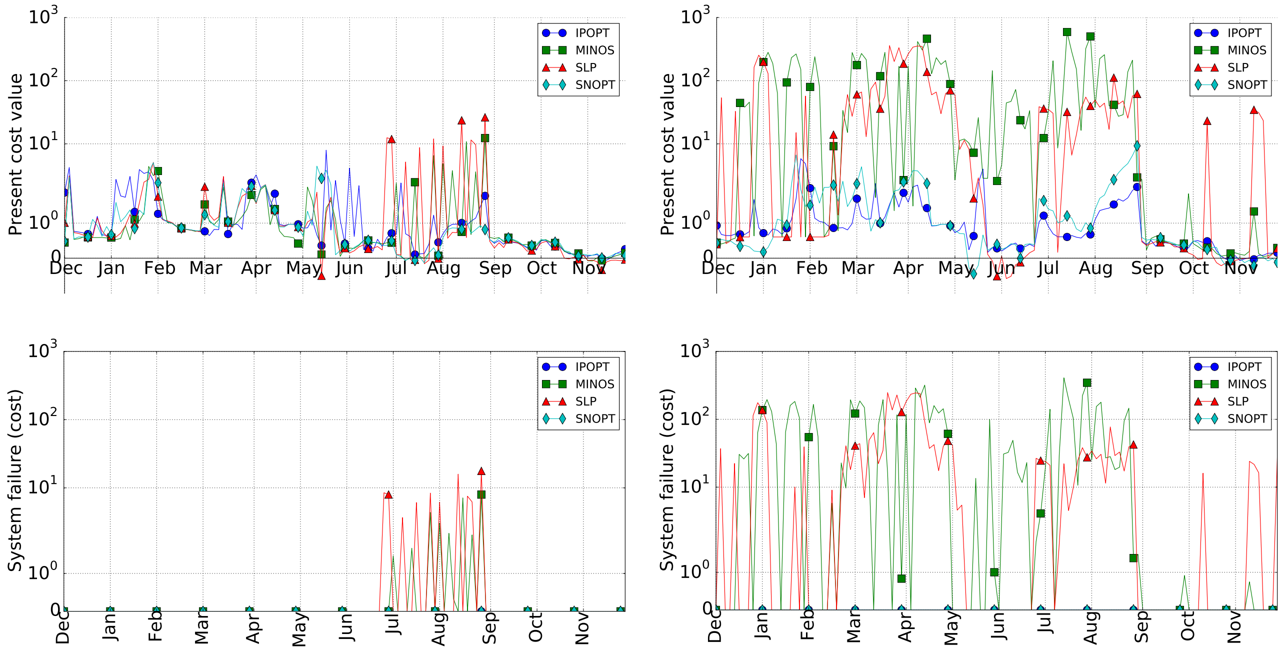

Figure 5: Top: Evolution of the average cost over 25 inflow scenarii. Bottom: Evolution of the average cost over the same 25 inflow scenarii related to system failures. The left plots correspond to linear constraints during the approximation step and the right plots to nonlinear constraints.

A promising implementation results from combining the efficiency of SLP during the approximation step (Step 3) of Algorithm 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 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 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 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 2 and is used as our reference. We compare the combinations of Table 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 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 probability density functions in Figure 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.

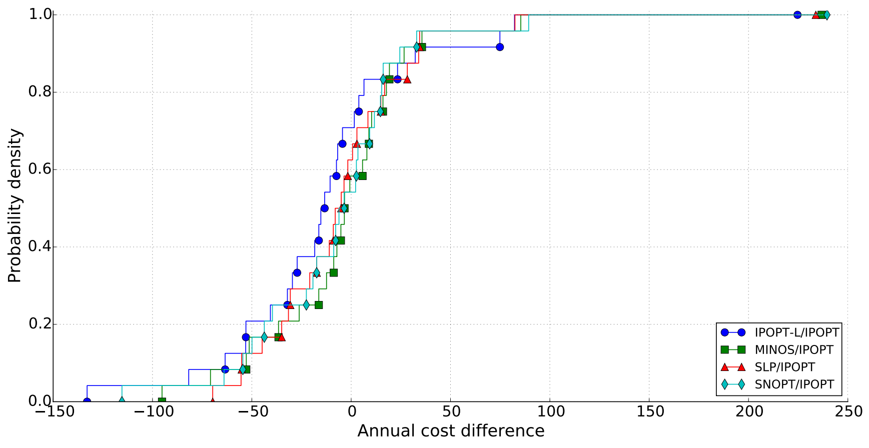

Figure 6: Probability density functions of the difference between the annual cost generated by IPOPT-N/IPOPT and the solver combinations of Table 3.

Figure 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 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 gives the result of the statistical test.

Table 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.

Table 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

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.

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.

References

S. Anvari, S. J. Mousavi, and S. Morid. Sampling/stochastic dynamic programming for optimal operation of multi-purpose reservoirs using artificial neural network-based ensemble streamflow predictions. Journal of Hydroinformatics, 16(4):907–921, 2014. DOI:10.2166/hydro.2013.236.

S. Arreckx, D. Orban, and N. van Omme. NLP.py: An object-oriented environment for large-scale optimization. Cahier du GERAD G-2016-42, GERAD, Montréal, (Québec), Canada, 2016. DOI:10.13140/RG.2.1.2846.6803.

C. Audet and D. Orban. Finding optimal algorithmic parameters using derivative-free optimization. SIAM Journal on Optimization, 17(3):642–664, 2006. DOI:10.1137/040620886.

C. Audet, K.-C. Dang, and D. Orban. Optimization of algorithms with OPAL. Mathematical Programming Computation, 6(3):233–254, 2014.

M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. Nonlinear Programming. John Wiley & Sons, Inc., third edition, 2005. DOI:10.1002/0471787779.

R. Bellman. Dynamic Programming. Princeton University Press, Princeton, NJ, USA, 1957.

D. P. Bertsekas. Dynamic programming and optimal control, volume 1. Athena Scientific Belmont, MA, 1995.

M. Breton, S. Hachem, and A. Hammadia. Accounting for losses in the optimization of production of hydroplants. IEEE Transactions on Energy Conversion, 19(2):346–351, 2004. DOI:10.1109/TEC.2004.827043.

R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large-scale nonlinear programming. SIAM Journal on Optimization, 9(4):877–900, 1999. DOI:10.1137/S1052623497325107.

R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. An algorithm for nonlinear optimization using linear programming and equality constrained subproblems. Mathematical Programming, 100(1):27–48, 2003. DOI:10.1007/s10107-003-0485-4.

R. H. Byrd, J. Nocedal, and R. A. Waltz. Knitro: An integrated package for nonlinear optimization. Large-Scale Nonlinear Optimization, 83:35–59, 2006. DOI:10.1007/0-387-30065-1-4.

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

P. Côté and R. Leconte. Comparison of stochastic optimization algorithms for hydropower reservoir operation with ensemble streamflow prediction. Journal of Water Resources Planning and Management, 142(2):04015046, 2015. DOI:10.1061/(ASCE)WR.1943-5452.0000575.

P. Côté, D. Haguma, R. Leconte, and S. Krau. Stochastic optimisation of hydro-quebec hydropower installations: a statistical comparison between sdp and ssdp methods. Canadian Journal of Civil Engineering, 38(12):1427–1434, 2011. DOI:10.1139/l11-101.

C. Davidsen, S. J. Pereira-Cardenal, S. Liu, X. Mo, D. Rosbjerg, and 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, 141(7):04014086, 2014. DOI:10.1061/(ASCE)WR.1943-5452.0000482.

Q. Desreumaux, P. Côté, and 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, 41(9):839–844, 2014. DOI:10.1139/cjce-2013-0370.

B. A. Faber. Real-time reservoir optimization using ensemble streamflow forecasts. PhD thesis, Cornell University, 2001.

R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. Mathematical programming, 91(2): 239–269, 2002. DOI:10.1007/s101070100244.

P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM review, 47(1):99–131, 2005. DOI:10.1137/S0036144504446096.

N. I. M. Gould, D. Orban, and P. h. L. Toint. GALAHAD, a library of thread-safe FORTRAN 90 packages for large-scale nonlinear optimization. ACM Transactions on Mathematical Software (TOMS), 29(4):353–372, 2003.

N. I. M. Gould, D. Orban, and P. h. L. Toint. LANCELOT_simple: A simple interface for LANCELOT B. Cahier du GERAD, G-2008-11, GERAD, Montréal, (Québec), Canada, 2008.

R. E. Griffith and R. A. Stewart. A nonlinear programming technique for the optimization of continuous processing systems. Management science, 7(4):379–392, 1961. DOI:10.1287/mnsc.7.4.379.

M. R. Hestenes. Multiplier and gradient methods. Journal of optimization theory and applications, 4(5):303–320, 1969. DOI:10.1007/BF00927673.

S.A. Johnson, J. R. Stedinger, C.A. Shoemaker, Y. Li, and J.A. Tejada-Guibert. Numerical solution of continuous-state dynamic programming using linear and spline interpolation. Operations research, 41(3), 1993. DOI:10.1287/opre.41.3.484.

J. W. Labadie. Optimal operation of multireservoir systems: State-of-the-art review. Journal of water resources planning and management, 130(2):93–111, 2004. DOI:10.1061/(ASCE)0733-9496(2004)130:2(93).

A. Magnani and S. P. Boyd. Convex piecewise-linear fitting. Optimization and Engineering, 10(1):1–17, 2009. DOI:10.1007/s11081-008-9045-3.

J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. SIAM Journal on Optimization, 20 (1):172–191, 2009. DOI:10.1137/080724083.

P. P. Mujumdar and B. Nirmala. A bayesian stochastic optimization model for a multi-reservoir hydropower system. Water resources management, 21(9):1465–1485, 2007. DOI:10.1007/s11269-006-9094-3.

B. A. Murtagh and M. A. Saunders. Large-scale linearly constrained optimization. Mathematical programming, 14(1): 41–72, 1978. DOI:10.1007/BF01588950.

J. Nocedal and S. J. Wright. Numerical optimization. Springer Science & Business Media, 2 edition, 2006.

R. E. Perez, P. W. Jansen, and J. R. R. A. Martins. pyOpt: A Python-based object-oriented framework for nonlinear constrained optimization. Structures and Multidisciplinary Optimization, 45(1):101–118, 2012. DOI:10.1007/s00158-011-0666-3.

M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, Optimization, pages 283–298. Academic Press, New York, 1969.

J. A. Tejada-Guibert, S. A. Johnson, and J. R. Stedinger. The value of hydrologic information in stochastic dynamic programming models of a multireservoir system. Water resources research, 31(10):2571–2579, 1995. DOI:10.1029/95WR02172.

A. Tilmant, M. Vanclooster, L. Duckstein, and E. Persoons. Comparison of fuzzy and nonfuzzy optimal reservoir operating policies. Journal of Water Resources Planning and Management, 128(6):390–398, 2002. DOI:10.1061/(ASCE)0733-9496(2002)128:6(390).

M. Towhidi and D. Orban. Customizing the solution process of COIN-ORs linear solvers with Python. Mathematical Programming Computation, 8(4):377–391, 2016. DOI:10.1007/s12532-015-0094-2.

A. Turgeon. Solving daily reservoir management problems with dynamic programming. Cahier du Gerad, G-2006-22, GERAD, Montréal, (Québec), Canada, 2006.

A. Turgeon. Solving a stochastic reservoir management problem with multilag autocorrelated inflows. Water Resources Research, 41(12), 2005. DOI:10.1029/2004WR003846.

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming, 106(1):25–57, 2006. DOI:10.1007/s10107-004-0559-y.

R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban. An interior algorithm for nonlinear optimization that combines line search and trust region steps. Mathematical programming, 107(3):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, and S. Stitt. Dynamic optimal unit commitment and loading in hydropower systems. Journal of water resources planning and management, 129(5):388–398, 2003. DOI:10.1061/(ASCE)0733-9496(2003)129:5(388).

T. Zhao, J. Zhao, and D. Yang. Improved dynamic programming for hydropower reservoir operation. Journal of Water Resources Planning and Management, 140(3):365–374, 2012. DOI:10.1061/(ASCE)WR.1943-5452.0000343.