Computing high-quality Lagrangian bounds of the stochastic mixed-integer programming problem

Prof. Andrew Eberhard

Mathematical Sciences
School of Sciences
Royal Melbourne Institute of Technology (RMIT) University
Melbourne, Victoria, Australia

AMSI Optimise
Melbourne
30th June 2017.

Team on this DP Project

Co-authors:

- At Georgia Tech:
  - Prof. Natashia Boland
- At RMIT University:
  - Dr. Brian Dandurand (Postdoc)
  - Mr. Jeffrey Christiansen (Ph.D. student)
  - Dr. Fabricio Oliveira (Postdoc)
- At University of Wisconsin-Madison:
  - Prof. Jeffrey Linderoth
  - Prof. James Luedtke

Funding:

- For Boland, Christiansen, Dandurand, Eberhard, Oliveira, and Linderoth: (ARC) Grant ARC DP140100985;
- For Linderoth and Luedtke: U.S. Department of Energy, Applied Mathematics program under contract number DE-AC02-06CH11357.

Outline

Stochastic Mixed-Integer Optimization: Background

Algorithm FW-PH

Computational Experiments

Generating Primal Solutions

Stochastic Mixed-Integer Optimization

- Provides a framework for modeling problems where decisions are made in stages.
- Between stages, some uncertainty in the problem parameters is unveiled, and decisions in subsequent stages may depend on the outcome of this uncertainty.
- When some decisions are modeled using discrete variables, the problem is known as a Stochastic Mixed-Integer Programming (SMIP) problem.
- However, the combination of both uncertainty with discreteness makes this class of problems particularly challenging from a computational perspective.

Stochastic Mixed-Integer Optimization

- Provides a framework for modeling problems where decisions are made in stages.
- Between stages, some uncertainty in the problem parameters is unveiled, and decisions in subsequent stages may depend on the outcome of this uncertainty.
- When some decisions are modeled using discrete variables, the problem is known as a Stochastic Mixed-Integer Programming (SMIP) problem.
- However, the combination of both uncertainty with discreteness makes this class of problems particularly challenging from a computational perspective.

Stochastic Mixed-Integer Optimization

- Provides a framework for modeling problems where decisions are made in stages.
- Between stages, some uncertainty in the problem parameters is unveiled, and decisions in subsequent stages may depend on the outcome of this uncertainty.
- When some decisions are modeled using discrete variables, the problem is known as a Stochastic Mixed-Integer Programming (SMIP) problem.
- However, the combination of both uncertainty with discreteness makes this class of problems particularly challenging from a computational perspective.

Stochastic Mixed-Integer Optimization

- Provides a framework for modeling problems where decisions are made in stages.
- Between stages, some uncertainty in the problem parameters is unveiled, and decisions in subsequent stages may depend on the outcome of this uncertainty.
- When some decisions are modeled using discrete variables, the problem is known as a Stochastic Mixed-Integer Programming (SMIP) problem.
- However, the combination of both uncertainty with discreteness makes this class of problems particularly challenging from a computational perspective.

Applications: SMIP models include

- unit commitment and hydro-thermal generation scheduling [NR00, TBL96],
- military operations [SWM09],
- vaccination planning [OPSR11, TSN08],
- air traffic flow management [AAAEP12],
- forestry management and forest fire response [BWW+15, NAS+12], and
- supply chain and logistics planning [LLM92, Lou86].

Two-stage SMIP Formulation

- A two-stage SMIP:

$$\zeta^{SMIP} := \min_{x} \left\{ c^{\top} x + \mathcal{Q}(x) \;:\; x \in X \right\}$$

  where
  - vector $c \in \mathbb{R}^{n_x}$ is known
  - $X \subset \mathbb{R}^{n_x}$ is a constraint set consisting of linear constraints and integer restrictions on some components of x
- Function $\mathcal{Q} : \mathbb{R}^{n_x} \mapsto \mathbb{R}$ outputs the expected recourse value

$$\mathcal{Q}(x) := \mathbb{E}_{\xi} \left[\min_{y} \left\{ q(\xi)^{\top} y \;:\; W(\xi) y = h(\xi) - T(\xi) x, y \in Y(\xi) \right\} \right]$$

  - each realization ξ_s of ξ, is called a *scenario* and encodes the realizations observed for each of the random elements $(q_s, h_s, W_s, T_s, Y_s)$ for each scenario index $s \in \mathcal{S}$
  - for each $s \in \mathcal{S}$, the set $Y_s \subset \mathbb{R}^{n_y}$ is a mixed-integer set containing both linear constraints and integrality constraints on a subset of the variables, y_s

Two-stage SMIP Formulation

- A two-stage SMIP:

$$\zeta^{SMIP} := \min_x \left\{ c^\top x + \mathcal{Q}(x) \ : \ x \in X \right\}$$

  where
  - vector $c \in \mathbb{R}^{n_x}$ is known
  - $X \subset \mathbb{R}^{n_x}$ is a constraint set consisting of linear constraints and integer restrictions on some components of x
- Function $\mathcal{Q} : \mathbb{R}^{n_x} \mapsto \mathbb{R}$ outputs the expected recourse value

$$\mathcal{Q}(x) := \mathbb{E}_\xi \left[\min_y \left\{ q(\xi)^\top y \ : \ W(\xi) y = h(\xi) - T(\xi) x, y \in Y(\xi) \right\} \right]$$

  - each realization ξ_s of ξ, is called a *scenario* and encodes the realizations observed for each of the random elements $(q_s, h_s, W_s, T_s, Y_s)$ for each scenario index $s \in \mathcal{S}$
  - for each $s \in \mathcal{S}$, the set $Y_s \subset \mathbb{R}^{n_y}$ is a mixed-integer set containing both linear constraints and integrality constraints on a subset of the variables, y_s

When we have a finite number of senarios with probabilities $\{p_s\}_{s \in \mathcal{S}}$ this problem SMIP may be reformulated as its *deterministic equivalent*

$$\zeta^{SMIP} = \min_{x,y} \left\{ \; c^\top x + \sum_{s \in \mathcal{S}} p_s q_s^\top y_s \; : \; (x, y_s) \in K_s, \forall s \in \mathcal{S} \; \right\},$$

where $K_s := \{(x, y_s) \; : \; W_s y_s = h_s - T_s x, x \in X, y_s \in Y_s\}$.

We assume throughout that

- problem SMIP is feasible, and
- the sets K_s, $s \in \mathcal{S}$ are bounded.
- These problems grow exponentially in size with the number of scenarios. This leads to the situation that they cannot be solved using normal methods.
- This leads to the search for methods that utilise the structure of the problem to enable decomposition into smaller problems that can be solved in parallel.

When we have a finite number of senarios with probabilities $\{p_s\}_{s\in\mathcal{S}}$ this problem SMIP may be reformulated as its *deterministic equivalent*

$$\zeta^{SMIP} = \min_{x,y} \left\{ \ c^\top x + \sum_{s\in\mathcal{S}} p_s q_s^\top y_s \ : \ (x, y_s) \in K_s, \forall s \in \mathcal{S} \ \right\},$$

where $K_s := \{(x, y_s) \ : \ W_s y_s = h_s - T_s x, x \in X, y_s \in Y_s\}$.

We assume throughout that

- problem SMIP is feasible, and
- the sets K_s, $s \in \mathcal{S}$ are bounded.
- These problems grow exponentially in size with the number of scenarios. This leads to the situation that they cannot be solved using normal methods.
- This leads to the search for methods that utilise the structure of the problem to enable decomposition into smaller problems that can be solved in parallel.

To induce a decomposable structure, scenario-dependent copies x_s for each $s \in S$ of the first-stage variable x are introduced to create the following *split-variable* reformulation of SMIP:

$$\zeta^{SMIP} = \min_{x,y,z} \left\{ \begin{array}{c} \sum_{s \in \mathcal{S}} p_s(c^\top x_s + q_s^\top y_s) \;: \\ (x_s, y_s) \in K_s, x_s = z, \; \forall \; s \in \mathcal{S}, z \in \mathbb{R}^{n_x} \end{array} \right\}.$$

- The constraints $x_s = z$, $s \in \mathcal{S}$, enforce *nonanticipativity* for the first-stage decisions.

To induce a decomposable structure, scenario-dependent copies x_s for each $s \in S$ of the first-stage variable x are introduced to create the following *split-variable* reformulation of SMIP:

$$\zeta^{SMIP} = \min_{x,y,z} \left\{ \begin{array}{c} \sum_{s \in \mathcal{S}} p_s(c^\top x_s + q_s^\top y_s) \ : \\ (x_s, y_s) \in K_s, x_s = z, \ \forall\ s \in \mathcal{S}, z \in \mathbb{R}^{n_x} \end{array} \right\}.$$

- The constraints $x_s = z$, $s \in \mathcal{S}$, enforce *nonanticipativity* for the first-stage decisions.

Lagrangian Dual Problem

Applying Lagrangian relaxation to the nonanticipativity constraints in the reformulated problem SMIP yields the *nonanticipative Lagrangian dual function*

$$\phi(\mu) := \min_{x,y,z} \left\{ \begin{array}{c} \sum_{s\in\mathcal{S}} p_s(c^\top x_s + q_s^\top y_s) + \mu_s^\top (x_s - z) \ : \\ (x_s, y_s) \in K_s, \ \forall s \in \mathcal{S}, z \in \mathbb{R}^{n_x} \end{array} \right\},$$

where $\mu := (\mu_1, \ldots, \mu_{|\mathcal{S}|}) \in \prod_{s\in\mathcal{S}} \mathbb{R}^{n_x}$ is the vector of multipliers associated with the relaxed constraints $x_s = z$, $s \in \mathcal{S}$.

- We assume the following *dual feasibility condition*: $\sum_{s\in\mathcal{S}} \mu^s = 0$.

By setting $\omega_s := \frac{1}{p_s}\mu_s$, the dual function ϕ may be rewritten as

$$\phi(\omega) := \min_{x,y,z} \left\{ \begin{array}{c} \sum_{s\in\mathcal{S}} p_s L_s(x_s, y_s, z, \omega_s) \ : \\ (x_s, y_s) \in K_s, \ \forall s \in \mathcal{S}, \ z \in \mathbb{R}^{n_x} \end{array} \right\},$$

where $L_s(x_s, y_s, z, \omega_s) := c^\top x_s + q_s^\top y_s + \omega_s^\top (x_s - z)$.

- In order for the Lagrangian function $\phi(\omega)$ to be bounded from below in z, the dual feasibility condition $\sum_{s\in\mathcal{S}} p_s\omega_s = 0$ must be satisfied.
- Under this assumption, the z term vanishes, and the Lagrangian dual function $\phi(\omega)$ decomposes into separable functions,

$$\phi(\omega) = \sum_{s\in\mathcal{S}} p_s \phi_s(\omega_s),$$

  where for each $s \in \mathcal{S}$,

$$\phi_s(\omega_s) := \min_{x,y} \left\{ (c + \omega_s)^\top x + q_s^\top y \ : \ (x, y) \in K_s \right\}.$$

By setting $\omega_s := \frac{1}{p_s}\mu_s$, the dual function ϕ may be rewritten as

$$\phi(\omega) := \min_{x,y,z} \left\{ \begin{array}{c} \sum_{s\in\mathcal{S}} p_s L_s(x_s, y_s, z, \omega_s) \ : \\ (x_s, y_s) \in K_s, \ \forall s \in \mathcal{S}, \ z \in \mathbb{R}^{n_x} \end{array} \right\},$$

where $L_s(x_s, y_s, z, \omega_s) := c^\top x_s + q_s^\top y_s + \omega_s^\top (x_s - z)$.

- In order for the Lagrangian function $\phi(\omega)$ to be bounded from below in z, the dual feasibility condition $\sum_{s\in\mathcal{S}} p_s \omega_s = 0$ must be satisfied.
- Under this assumption, the z term vanishes, and the Lagrangian dual function $\phi(\omega)$ decomposes into separable functions,

$$\phi(\omega) = \sum_{s\in\mathcal{S}} p_s \phi_s(\omega_s),$$

where for each $s \in \mathcal{S}$,

$$\phi_s(\omega_s) := \min_{x,y} \left\{ (c + \omega_s)^\top x + q_s^\top y \ : \ (x, y) \in K_s \right\}.$$

By setting $\omega_s := \frac{1}{p_s}\mu_s$, the dual function ϕ may be rewritten as

$$\phi(\omega) := \min_{x,y,z} \left\{ \begin{array}{c} \sum_{s\in\mathcal{S}} p_s L_s(x_s, y_s, z, \omega_s) \ : \\ (x_s, y_s) \in K_s, \ \forall s \in \mathcal{S}, \ z \in \mathbb{R}^{n_x} \end{array} \right\},$$

where $L_s(x_s, y_s, z, \omega_s) := c^\top x_s + q_s^\top y_s + \omega_s^\top (x_s - z)$.

- In order for the Lagrangian function $\phi(\omega)$ to be bounded from below in z, the dual feasibility condition $\sum_{s\in\mathcal{S}} p_s\omega_s = 0$ must be satisfied.
- Under this assumption, the z term vanishes, and the Lagrangian dual function $\phi(\omega)$ decomposes into separable functions,

$$\phi(\omega) = \sum_{s\in\mathcal{S}} p_s\phi_s(\omega_s),$$

where for each $s \in \mathcal{S}$,

$$\phi_s(\omega_s) := \min_{x,y} \left\{ (c + \omega_s)^\top x + q_s^\top y \ : \ (x, y) \in K_s \right\}.$$

For any choice of $\omega = (\omega_1, \ldots, \omega_{|\mathcal{S}|})$, it is well-known that the value of the Lagrangian provides a lower bound on the optimal solution to SMIP: $\phi(\omega) \le \zeta^{SMIP}$.

- The ability to compute high-quality Lagrangian bounds $\phi(\omega) \le \zeta^{SMIP}$ is useful for exact, enumerative approaches, such as those in [CS99, LMPS13].
- The problem of finding the best such lower bound is the *Lagrangian dual problem*:

$$\zeta^{LD} := \sup_{\omega} \left\{ \phi(\omega) \ : \ \sum_{s \in \mathcal{S}} p_s \omega_s = 0 \right\}.$$

- Computing the optimal Lagrangian bound cannot usually be carried out in closed form and so numerical techniques are need.

For any choice of $\omega = (\omega_1, \ldots, \omega_{|\mathcal{S}|})$, it is well-known that the value of the Lagrangian provides a lower bound on the optimal solution to SMIP: $\phi(\omega) \leq \zeta^{SMIP}$.

- The ability to compute high-quality Lagrangian bounds $\phi(\omega) \leq \zeta^{SMIP}$ is useful for exact, enumerative approaches, such as those in [CS99, LMPS13].
- The problem of finding the best such lower bound is the *Lagrangian dual problem*:

$$\zeta^{LD} := \sup_{\omega} \left\{ \phi(\omega) \ : \ \sum_{s \in \mathcal{S}} p_s \omega_s = 0 \right\}.$$

- Computing the optimal Lagrangian bound cannot usually be carried out in closed form and so numerical techniques are need.

Outline

Introducing Algorithm FW-PH

To this end, we contribute the method FW-PH:

- integrates Frank-Wolfe (FW) like iterations and progressive hedging (PH) iterations;
- parallelizable;
- theoretical support provides mild conditions under which (dual) optimal convergence is realized.
- can be easily extended to multi-stage problems.

To motivate our approach, we first consider the application of PH to the following well-known primal characterization of ζ^{LD}:

$$\zeta^{LD} = \min_{x,y,z} \left\{ \begin{array}{c} \sum_{s \in \mathcal{S}} p_s (c^\top x_s + q_s^\top y_s) \ : \\ (x_s, y_s) \in \text{conv}(K_s), \ x_s = z, \ \forall s \in \mathcal{S} \end{array} \right\},$$

where $\text{conv}(K_s)$ denotes the convex hull of K_s for each $s \in \mathcal{S}$. (See, for example, Theorem 6.2 of [NW88].)

- We refer to the above primal characterization problem as *Conv-SMIP*.
- The sequence of Lagrangian bounds $\{\phi(\omega^k)\}$ generated by the application of PH to Conv-SMIP is known to be convergent (see Rockafellar-Wetts et al).
- Thus, the value of the Lagrangian dual, ζ^{LD}, may, in theory, be approached by applying PH to Conv-SMIP.

However, an explicit polyhedral description of conv(K_s), $s \in \mathcal{S}$ is not available, and so this is not directly implementable.

To motivate our approach, we first consider the application of PH to the following well-known primal characterization of ζ^{LD}:

$$\zeta^{LD} = \min_{x,y,z} \left\{ \begin{array}{c} \sum_{s \in \mathcal{S}} p_s (c^\top x_s + q_s^\top y_s) \; : \\ (x_s, y_s) \in \text{conv}(K_s), \; x_s = z, \; \forall s \in \mathcal{S} \end{array} \right\},$$

where $\text{conv}(K_s)$ denotes the convex hull of K_s for each $s \in \mathcal{S}$. (See, for example, Theorem 6.2 of [NW88].)

- We refer to the above primal characterization problem as *Conv-SMIP*.
- The sequence of Lagrangian bounds $\{\phi(\omega^k)\}$ generated by the application of PH to Conv-SMIP is known to be convergent (see Rockafellar-Wetts et al).
- Thus, the value of the Lagrangian dual, ζ^{LD}, may, in theory, be approached by applying PH to Conv-SMIP.

However, an explicit polyhedral description of $\text{conv}(K_s)$, $s \in \mathcal{S}$ is not available, and so this is not directly implementable.

To motivate our approach, we first consider the application of PH to the following well-known primal characterization of ζ^{LD}:

$$\zeta^{LD} = \min_{x,y,z} \left\{ \begin{array}{c} \sum_{s \in \mathcal{S}} p_s(c^\top x_s + q_s^\top y_s) \ : \\ (x_s, y_s) \in \text{conv}(K_s), \ x_s = z, \ \forall s \in \mathcal{S} \end{array} \right\},$$

where $\text{conv}(K_s)$ denotes the convex hull of K_s for each $s \in \mathcal{S}$. (See, for example, Theorem 6.2 of [NW88].)

- We refer to the above primal characterization problem as *Conv-SMIP*.
- The sequence of Lagrangian bounds $\{\phi(\omega^k)\}$ generated by the application of PH to Conv-SMIP is known to be convergent (see Rockafellar-Wetts et al).
- Thus, the value of the Lagrangian dual, ζ^{LD}, may, in theory, be approached by applying PH to Conv-SMIP.

However, an explicit polyhedral description of conv(K_s), $s \in \mathcal{S}$ is not available, and so this is not directly implementable.

Progressive Hedging (PH)

In the following, we consider how to modify the PH algorithm to obtain the required implementability.

- The Augmented Lagrangian (AL) function:

$$L^\rho((x_s, y_s)_{s\in\mathcal{S}}, z, \omega) := \sum_{s\in\mathcal{S}} p_s L_s^\rho(x_s, y_s, z, \omega_s),$$

  where

$$L_s^\rho(x_s, y_s, z, \omega_s) := c^\top x_s + q_s^\top y_s + \omega_s^\top (x_s - z) + \frac{\rho}{2}\|x_s - z\|_2^2$$

  and $\rho > 0$ is a penalty parameter.
- The augmented Lagrangian method (a.k.a, the method of mulipliers) [Hes69, Pow69, Roc73, Ber82, EY14]: For $D_s = K_s$ we solve with a QMIP.
  - At iteration $k \geq 1$, compute:

    1) $((x_s^k, y_s^k)_{s\in\mathcal{S}}, z^k) \in$ $\text{argmin}_{x,y,z} \left\{ \begin{array}{c} L^\rho((x_s, y_s)_{s\in\mathcal{S}}, z, \omega^k) : \\ (x_s, y_s) \in D_s \ \forall s \in \mathcal{S}, \ z \in \mathbb{R}^n \end{array} \right\}$

    2) $\omega_s^{k+1} \leftarrow \omega_s^k + \rho(x_s^k - z^k)$ for all $s \in \mathcal{S}$

Progressive Hedging (PH)

In the following, we consider how to modify the PH algorithm to obtain the required implementability.

- The Augmented Lagrangian (AL) function:

$$L^{\rho}((x_s, y_s)_{s\in\mathcal{S}}, z, \omega) := \sum_{s\in\mathcal{S}} p_s L_s^{\rho}(x_s, y_s, z, \omega_s),$$

  where

$$L_s^{\rho}(x_s, y_s, z, \omega_s) := c^{\top} x_s + q_s^{\top} y_s + \omega_s^{\top}(x_s - z) + \frac{\rho}{2}\|x_s - z\|_2^2$$

  and $\rho > 0$ is a penalty parameter.

- The augmented Lagrangian method (a.k.a, the method of mulipliers) [Hes69, Pow69, Roc73, Ber82, EY14]: For $D_s = K_s$ we solve with a QMIP.
  - At iteration $k \geq 1$, compute:

    1) $((x_s^k, y_s^k)_{s\in\mathcal{S}}, z^k) \in$

    $$\operatorname{argmin}_{x,y,z} \left\{ \begin{array}{c} L^{\rho}((x_s, y_s)_{s\in\mathcal{S}}, z, \omega^k) : \\ (x_s, y_s) \in D_s \ \forall s \in \mathcal{S}, \ z \in \mathbb{R}^n \end{array} \right\}$$

    2) $\omega_s^{k+1} \leftarrow \omega_s^k + \rho(x_s^k - z^k)$ for all $s \in \mathcal{S}$

Progressive Hedging (PH)

In the following, we consider how to modify the PH algorithm to obtain the required implementability.

- The Augmented Lagrangian (AL) function:

$$L^{\rho}((x_s, y_s)_{s\in\mathcal{S}}, z, \omega) := \sum_{s\in\mathcal{S}} p_s L_s^{\rho}(x_s, y_s, z, \omega_s),$$

  where

$$L_s^{\rho}(x_s, y_s, z, \omega_s) := c^\top x_s + q_s^\top y_s + \omega_s^\top (x_s - z) + \frac{\rho}{2} \|x_s - z\|_2^2$$

  and $\rho > 0$ is a penalty parameter.

- The augmented Lagrangian method (a.k.a, the method of mulipliers) [Hes69, Pow69, Roc73, Ber82, EY14]: For $D_s = K_s$ we solve with a QMIP.
  - At iteration $k \geq 1$, compute:

    1) $((x_s^k, y_s^k)_{s\in\mathcal{S}}, z^k) \in$
    $\text{argmin}_{x,y,z} \left\{ \begin{array}{c} L^{\rho}((x_s, y_s)_{s\in\mathcal{S}}, z, \omega^k) : \\ (x_s, y_s) \in D_s \ \forall s \in \mathcal{S}, \ z \in \mathbb{R}^n \end{array} \right\}$

    2) $\omega_s^{k+1} \leftarrow \omega_s^k + \rho(x_s^k - z^k)$ for all $s \in \mathcal{S}$

Progressive Hedging (PH)

In the following, we consider how to modify the PH algorithm to obtain the required implementability.

- The Augmented Lagrangian (AL) function:

$$L^{\rho}((x_s, y_s)_{s\in\mathcal{S}}, z, \omega) := \sum_{s\in\mathcal{S}} p_s L_s^{\rho}(x_s, y_s, z, \omega_s),$$

  where

$$L_s^{\rho}(x_s, y_s, z, \omega_s) := c^\top x_s + q_s^\top y_s + \omega_s^\top (x_s - z) + \frac{\rho}{2} \|x_s - z\|_2^2$$

  and $\rho > 0$ is a penalty parameter.

- The augmented Lagrangian method (a.k.a, the method of mulipliers) [Hes69, Pow69, Roc73, Ber82, EY14]: For $D_s = K_s$ we solve with a QMIP.
  - At iteration $k \geq 1$, compute:

    1) $((x_s^k, y_s^k)_{s\in\mathcal{S}}, z^k) \in$
    $\text{argmin}_{x,y,z} \left\{ \begin{array}{c} L^{\rho}((x_s, y_s)_{s\in\mathcal{S}}, z, \omega^k) : \\ (x_s, y_s) \in D_s \; \forall s \in \mathcal{S}, \; z \in \mathbb{R}^n \end{array} \right\}$

    2) $\omega_s^{k+1} \leftarrow \omega_s^k + \rho(x_s^k - z^k)$ for all $s \in \mathcal{S}$

Progressive Hedging (PH)

In the following, we consider how to modify the PH algorithm to obtain the required implementability.

- The Augmented Lagrangian (AL) function:

$$L^{\rho}(x, y, z, \omega) := \sum_{s \in \mathcal{S}} p_s L^{\rho}_s(x_s, y_s, z, \omega_s),$$

  where

$$L^{\rho}_s(x_s, y_s, z, \omega_s) := c^{\top} x_s + q_s^{\top} y_s + \omega_s^{\top}(x_s - z) + \frac{\rho}{2} \|x_s - z\|_2^2$$

  and $\rho > 0$ is a penalty parameter.
- The k^{th} iteration, $k \geq 1$, of progressive hedging (PH):
  1) For all $s \in \mathcal{S}$, take
     $(x_s^k, y_s^k) \in \operatorname{argmin}_{x,y} \left\{ L^{\rho}_s(x, y, z^{k-1}, \omega_s^k) \; : \; (x, y) \in D_s \right\}$
  2) Compute $z^k \leftarrow \sum_{s \in \mathcal{S}} p_s x_s^k$
  3) Compute $\omega_s^{k+1} \leftarrow \omega_s^k + \rho(x_s^k - z^k)$ for all $s \in \mathcal{S}$
- PH is a specialization of the alternating direction method of multipliers (ADMM) [GM76, EB92, BPC+11] to the decomposed SMIP.

Progressive Hedging (PH)

In the following, we consider how to modify the PH algorithm to obtain the required implementability.

- The Augmented Lagrangian (AL) function:

$$L^\rho(x, y, z, \omega) := \sum_{s \in \mathcal{S}} p_s L_s^\rho(x_s, y_s, z, \omega_s),$$

  where

$$L_s^\rho(x_s, y_s, z, \omega_s) := c^\top x_s + q_s^\top y_s + \omega_s^\top (x_s - z) + \frac{\rho}{2} \|x_s - z\|_2^2$$

  and $\rho > 0$ is a penalty parameter.

- The k^{th} iteration, $k \geq 1$, of progressive hedging (PH):
  1) For all $s \in \mathcal{S}$, take
     $(x_s^k, y_s^k) \in \text{argmin}_{x,y} \left\{ L_s^\rho(x, y, z^{k-1}, \omega_s^k) \ : \ (x, y) \in D_s \right\}$
  2) Compute $z^k \leftarrow \sum_{s \in \mathcal{S}} p_s x_s^k$
  3) Compute $\omega_s^{k+1} \leftarrow \omega_s^k + \rho(x_s^k - z^k)$ for all $s \in \mathcal{S}$

- PH is a specialization of the alternating direction method of multipliers (ADMM) [GM76, EB92, BPC+11] to the decomposed SMIP.

Progressive Hedging (PH)

In the following, we consider how to modify the PH algorithm to obtain the required implementability.

- The Augmented Lagrangian (AL) function:

$$L^\rho(x, y, z, \omega) := \sum_{s \in \mathcal{S}} p_s L_s^\rho(x_s, y_s, z, \omega_s),$$

  where

$$L_s^\rho(x_s, y_s, z, \omega_s) := c^\top x_s + q_s^\top y_s + \omega_s^\top (x_s - z) + \frac{\rho}{2} \|x_s - z\|_2^2$$

  and $\rho > 0$ is a penalty parameter.
- The k^{th} iteration, $k \geq 1$, of progressive hedging (PH):
  1) For all $s \in \mathcal{S}$, take
     $(x_s^k, y_s^k) \in \text{argmin}_{x,y} \left\{ L_s^\rho(x, y, z^{k-1}, \omega_s^k) \ : \ (x, y) \in D_s \right\}$
  2) Compute $z^k \leftarrow \sum_{s \in \mathcal{S}} p_s x_s^k$
  3) Compute $\omega_s^{k+1} \leftarrow \omega_s^k + \rho(x_s^k - z^k)$ for all $s \in \mathcal{S}$
- PH is a specialization of the alternating direction method of multipliers (ADMM) [GM76, EB92, BPC+11] to the decomposed SMIP.

Proposition

Assume that problem Conv-SMIP is feasible with conv(K_s) bounded for each $s \in \mathcal{S}$, and let Algorithm PH be applied to problem Conv-SMIP (so that $D_s = conv(K_s)$ for each $s \in \mathcal{S}$) with tolerance $\epsilon = 0$ for each $k \geq 1$. Then, the limit $\lim_{k\to\infty} \omega^k = \omega^*$ exists, and furthermore,

1. $\lim_{k\to\infty} \sum_{s\in\mathcal{S}} p_s(c^\top x_s^k + q_s^\top y_s^k) = \zeta^{LD}$,
2. $\lim_{k\to\infty} \phi(\omega^k) = \zeta^{LD}$,
3. $\lim_{k\to\infty}(x_s^k - z^k) = 0$ *for each* $s \in \mathcal{S}$,

and each limit point $(((x_s^*, y_s^*)_{s\in\mathcal{S}}, z^*)$ *is an optimal solution for Conv-SMIP.*

Proof.

Since the constraint sets $D_s = \text{conv}(K_s)$, $s \in \mathcal{S}$, are bounded and polyhedral, and the objective is linear (and thus convex), problem Conv-SMIP has a saddle point $((x_s^*, y_s^*)_{s\in\mathcal{S}}, z^*, \omega^*)$ and the classical (continuous convex) convergence analysis applied.

□

Proposition

Assume that problem Conv-SMIP is feasible with conv(K_s) bounded for each $s \in \mathcal{S}$, and let Algorithm PH be applied to problem Conv-SMIP (so that $D_s = conv(K_s)$ for each $s \in \mathcal{S}$) with tolerance $\epsilon = 0$ for each $k \geq 1$. Then, the limit $\lim_{k\to\infty} \omega^k = \omega^*$ exists, and furthermore,

1. $\lim_{k\to\infty} \sum_{s\in\mathcal{S}} p_s(c^\top x_s^k + q_s^\top y_s^k) = \zeta^{LD}$,
2. $\lim_{k\to\infty} \phi(\omega^k) = \zeta^{LD}$,
3. $\lim_{k\to\infty}(x_s^k - z^k) = 0$ *for each* $s \in \mathcal{S}$,

and each limit point $(((x_s^*, y_s^*)_{s\in\mathcal{S}}, z^*)$ *is an optimal solution for Conv-SMIP.*

Proof.

Since the constraint sets $D_s = \text{conv}(K_s)$, $s \in \mathcal{S}$, are bounded and polyhedral, and the objective is linear (and thus convex), problem Conv-SMIP has a saddle point $((x_s^*, y_s^*)_{s\in\mathcal{S}}, z^*, \omega^*)$ and the classical (continuous convex) convergence analysis applied.

□

Remark:

PH can be applied to the original split-variable SMIP in practice where one solves MIP when the structure of conv K_s is not known.

- Although there is not guarantee of optimal convergence in theory or practice, reasonable (but very slow) *apparent* convergence can nevertheless be observed [WW11] for small values of penalty parameters ρ.
- As a means to generate (lower) Lagrangian bounds, PH can also be applied directly to the original split-variable SMIP [GHR$^+$15] to obtain bounds using a QMIP solve.
- However, optimal dual convergence is not realized, and *improvement in PH as a tool to generate strong Lagrangian bounds is needed.*

Remark:

PH can be applied to the original split-variable SMIP in practice where one solves MIP when the structure of conv K_s is not known.

- Although there is not guarantee of optimal convergence in theory or practice, reasonable (but very slow) *apparent* convergence can nevertheless be observed [WW11] for small values of penalty parameters ρ.
- As a means to generate (lower) Lagrangian bounds, PH can also be applied directly to the original split-variable SMIP [GHR$^+$15] to obtain bounds using a QMIP solve.
- However, optimal dual convergence is not realized, and *improvement in PH as a tool to generate strong Lagrangian bounds is needed*.

The Frank-Wolfe (FW) method and the simplicial decomposition method (SDM)

To use Algorithm PH to solve Conv-SMIP requires a method for solving the subproblem

$$(x_s^k, y_s^k) \in \operatorname*{argmin}_{x,y} \left\{ L_s^\rho(x, y, z^{k-1}, \omega_s^k) : (x, y) \in \operatorname{conv}(K_s) \right\}.$$

- We apply an iterative approach similar to the well-known Frank-Wolfe method [FW56], known as the simplicial decomposition method (SDM) [Hol74, VH77].
- The following summarizes iteration t of the SDM:

for $s \in \mathcal{S}$ **do**
1) Compute:
$$(\widehat{x}_s, \widehat{y}_s) \in \operatorname{argmin}_{x,y} \left\{ \begin{array}{c} \nabla_{(x,y)} L_s^\rho(x^{t-1}, y^{t-1}, z^{k-1}, \omega_s^k) \begin{bmatrix} x \\ y \end{bmatrix} : \\ (x, y) \in \mathcal{V}(\operatorname{conv}(K_s)) \end{array} \right\}$$
2) Construct: $V_s^t \leftarrow V_s^{t-1} \cup \{(\widehat{x}_s, \widehat{y}_s)\}$
3) Compute:
$$(x^t, y^t) \in \operatorname{argmin}_{x,y} \left\{ L_s^\rho(x, y, z^{k-1}, \omega_s^k) : (x, y) \in \operatorname{conv}(V_s^t) \right\}$$
end for

The Frank-Wolfe (FW) method and the simplicial decomposition method (SDM)

To use Algorithm PH to solve Conv-SMIP requires a method for solving the subproblem

$$(x_s^k, y_s^k) \in \underset{x,y}{\operatorname{argmin}} \left\{ L_s^\rho(x, y, z^{k-1}, \omega_s^k) : (x, y) \in \operatorname{conv}(K_s) \right\}.$$

- We apply an iterative approach similar to the well-known Frank-Wolfe method [FW56], known as the simplicial decomposition method (SDM) [Hol74, VH77].
- The following summarizes iteration t of the SDM:

for $s \in \mathcal{S}$ **do**

1) Compute:

$$(\widehat{x}_s, \widehat{y}_s) \in \operatorname{argmin}_{x,y} \left\{ \begin{array}{c} \nabla_{(x,y)} L_s^\rho(x^{t-1}, y^{t-1}, z^{k-1}, \omega_s^k) \begin{bmatrix} x \\ y \end{bmatrix} : \\ (x, y) \in \mathcal{V}(\operatorname{conv}(K_s)) \end{array} \right\}$$

2) Construct: $V_s^t \leftarrow V_s^{t-1} \cup \{(\widehat{x}_s, \widehat{y}_s)\}$

3) Compute:

$$(x^t, y^t) \in \operatorname{argmin}_{x,y} \left\{ L_s^\rho(x, y, z^{k-1}, \omega_s^k) : (x, y) \in \operatorname{conv}(V_s^t) \right\}$$

end for

The Frank-Wolfe (FW) method and the simplicial decomposition method (SDM)

To use Algorithm PH to solve Conv-SMIP requires a method for solving the subproblem

$$(x_s^k, y_s^k) \in \operatorname*{argmin}_{x,y} \left\{ L_s^\rho(x, y, z^{k-1}, \omega_s^k) : (x, y) \in \mathrm{conv}(K_s) \right\}.$$

- We apply an iterative approach similar to the well-known Frank-Wolfe method [FW56], known as the simplicial decomposition method (SDM) [Hol74, VH77].
- The following summarizes iteration t of the SDM:

for $s \in \mathcal{S}$ **do**

1) Compute:

$$(\widehat{x}_s, \widehat{y}_s) \in \mathrm{argmin}_{x,y} \left\{ \begin{array}{c} \nabla_{(x,y)} L_s^\rho(x^{t-1}, y^{t-1}, z^{k-1}, \omega_s^k) \begin{bmatrix} x \\ y \end{bmatrix} : \\ (x, y) \in \mathcal{V}(\mathrm{conv}(K_s)) \end{array} \right\}$$

2) Construct: $V_s^t \leftarrow V_s^{t-1} \cup \{(\widehat{x}_s, \widehat{y}_s)\}$

3) Compute:

$$(x^t, y^t) \in \mathrm{argmin}_{x,y} \left\{ L_s^\rho(x, y, z^{k-1}, \omega_s^k) : (x, y) \in \mathrm{conv}(V_s^t) \right\}$$

end for

Precondition: $V_s^0 \subset \text{conv}(K_s)$, $z \in \text{argmin}_z \left\{ \sum_{s \in \mathcal{S}} p_s \left\| x_s^0 - z \right\|_2^2 \right\}$

function SDM($V_s^0, x_s^0, \omega_s, z, t_{max}, \tau$)

  for $t = 1, \ldots, t_{max}$ **do**

    $\widehat{\omega}_s^t \leftarrow \omega_s + \rho(x_s^{t-1} - z)$

    $(\widehat{x}_s, \widehat{y}_s) \in \text{argmin}_{x,y} \left\{ \begin{array}{c} (c + \widehat{\omega}_s^t)^\top x + q_s^\top y \; : \\ (x, y) \in \mathcal{V}(K_s) \end{array} \right\}$

    if $t = 1$ **then**

      $\phi_s \leftarrow (c + \widehat{\omega}_s^t)^\top \widehat{x}_s + q_s^\top \widehat{y}_s$

    end if

    $\Gamma^t \leftarrow -[(c + \widehat{\omega}_s^t)^\top (\widehat{x}_s - x_s^{t-1}) + q_s^\top (\widehat{y}_s - y_s^{t-1})]$

    $V_s^t \leftarrow V_s^{t-1} \cup \{(\widehat{x}_s, \widehat{y}_s)\}$

    $(x_s^t, y_s^t) \in \text{argmin}_{x,y} \left\{ L_s^\rho(x, y, z, \omega_s) \; : \; (x, y) \in \text{conv}(V_s^t) \right\}$

    if $\Gamma^t \leq \tau$ **then**

      return $(x_s^t, y_s^t, V_s^t, \phi_s)$

    end if

  end for

  return $(x_s^{t_{max}}, y_s^{t_{max}}, V_s^{t_{max}}, \phi_s)$

end function

Precondition: $\sum_{s\in\mathcal{S}} p_s\omega_s^0 = 0$, $\rho > 0$

function FW-PH($(V_s^0)_{s\in\mathcal{S}}, \omega^0, \rho, \alpha, \epsilon, k_{max}, t_{max}, \{\tau_k\}$)

  $z^0 \leftarrow \sum_{s\in\mathcal{S}} p_s x_s^0$

  $\omega_s^1 \leftarrow \omega_s^0 + \rho(x_s^0 - z^0)$, for $s \in \mathcal{S}$

  for $k = 1, \ldots, k_{max}$ **do**

    for $s \in \mathcal{S}$ **do**

      $\widetilde{x}_s \leftarrow (1-\alpha)z^{k-1} + \alpha x_s^{k-1}$ (average the concesus and current primal value for $s \in \mathcal{S}$)

      $[x_s^k, y_s^k, V_s^k, \phi_s^k] \leftarrow \mathrm{SDM}(V_s^{k-1}, \widetilde{x}_s, \omega_s^k, z^{k-1}, t_{max}, \tau_k)$ (the FW - SDM step)

    end for

    $\phi^k \leftarrow \sum_{s\in\mathcal{S}} p_s\phi_s^k$ (the lower bound on the dual value)

    $z^k \leftarrow \sum_{s\in\mathcal{S}} p_s x_s^k$ (the G-S step - minimises the dispersion on z)

    if $\sum_{s\in\mathcal{S}} p_s \left\| x_s^k - z^{k-1} \right\|_2^2 < \epsilon$ **then**

      return $((x_s^k, y_s^k)_{s\in\mathcal{S}}, z^k, \omega^k, \phi^k)$

    end if

    $\omega_s^{k+1} \leftarrow \omega_s^k + \rho(x_s^k - z^k)$, for $s \in \mathcal{S}$

  end for

  return $\left((x_s^{k_{max}}, y_s^{k_{max}})_{s\in\mathcal{S}}, z^{k_{max}}), \omega^{k_{max}}, \phi^{k_{max}}\right)$

end function

Optimal convergence $\lim_{k\to\infty} \phi^k = \phi^*$ can be established when

- the augmented Lagrangian L_s^ρ is modified to be strongly convex (add a quadratic proximal term in y);
- the sequence $\{\tau_k\}$ of SDM convergence tolerances is generated a priori so that $\sum_{k=1}^{\infty} \sqrt{\tau_k} < \infty$.

It can be shown that these above two conditions imply the satisfaction of assumptions used in the convergence analysis of inexact ADMM in Theorem 8 of [EB92].

- However, requiring the SDM termination condition to be satisfied for each such $\tau_k \geq 0$ is not needed in practice (and is undesirable); nor will it be required in theory, as we shall soon see.

Precondition: $\sum_{s \in \mathcal{S}} p_s \omega_s^0 = 0$, $\rho > 0$

function FW-PH($(V_s^0)_{s \in \mathcal{S}}, \omega^0, \rho, \alpha, \epsilon, k_{max}, t_{max}, \{\tau_k\}$)

  $z^0 \leftarrow \sum_{s \in \mathcal{S}} p_s x_s^0$

  $\omega_s^1 \leftarrow \omega_s^0 + \rho(x_s^0 - z^0)$, for $s \in \mathcal{S}$

  for $k = 1, \ldots, k_{max}$ **do**

    for $s \in \mathcal{S}$ **do**

      $\widetilde{x}_s \leftarrow (1-\alpha) z^{k-1} + \alpha x_s^{k-1}$ (average the concesus and current primal value for $s \in \mathcal{S}$)

      $[x_s^k, y_s^k, V_s^k, \phi_s^k] \leftarrow \text{SDM}(V_s^{k-1}, \widetilde{x}_s, \omega_s^k, z^{k-1}, t_{max}, \tau_k)$ (the FW - SDM step)

    end for

    $\phi^k \leftarrow \sum_{s \in \mathcal{S}} p_s \phi_s^k$ (the lower bound on the dual value)

    $z^k \leftarrow \sum_{s \in \mathcal{S}} p_s x_s^k$ (the G-S step - minimises the dispersion on z)

    if $\sum_{s \in \mathcal{S}} p_s \left\| x_s^k - z^{k-1} \right\|_2^2 < \epsilon$ **then**

      return $((x_s^k, y_s^k)_{s \in \mathcal{S}}, z^k, \omega^k, \phi^k)$

    end if

    $\omega_s^{k+1} \leftarrow \omega_s^k + \rho(x_s^k - z^k)$, for $s \in \mathcal{S}$

  end for

  return $\left((x_s^{k_{max}}, y_s^{k_{max}})_{s \in \mathcal{S}}, z^{k_{max}}), \omega^{k_{max}}, \phi^{k_{max}} \right)$

end function

Optimal convergence $\lim_{k \to \infty} \phi^k = \phi^*$ can be established when

- the augmented Lagrangian L_s^ρ is modified to be strongly convex (add a quadratic proximal term in y);
- the sequence $\{\tau_k\}$ of SDM convergence tolerances is generated a priori so that $\sum_{k=1}^{\infty} \sqrt{\tau_k} < \infty$.

It can be shown that these above two conditions imply the satisfaction of assumptions used in the convergence analysis of inexact ADMM in Theorem 8 of [EB92].

- However, requiring the SDM termination condition to be satisfied for each such $\tau_k \geq 0$ is not needed in practice (and is undesirable); nor will it be required in theory, as we shall soon see.

```
Precondition: ∑_{s∈S} p_s ω_s^0 = 0, ρ > 0
function FW-PH((V_s^0)_{s∈S}, ω^0, ρ, α, ϵ, k_max, t_max, {τ_k})
    z^0 ← ∑_{s∈S} p_s x_s^0
    ω_s^1 ← ω_s^0 + ρ(x_s^0 − z^0), for s ∈ S
    for k = 1, . . . , k_max do
        for s ∈ S do
            x̃_s ← (1 − α)z^{k−1} + α x_s^{k−1}    (average the concesus and current primal value for s ∈ S)
            [x_s^k, y_s^k, V_s^k, φ_s^k] ← SDM(V_s^{k−1}, x̃_s, ω_s^k, z^{k−1}, t_max, τ_k)    (the FW - SDM step)
        end for
        φ^k ← ∑_{s∈S} p_s φ_s^k    (the lower bound on the dual value)
        z^k ← ∑_{s∈S} p_s x_s^k    (the G-S step - minimises the dispersion on z)
        if ∑_{s∈S} p_s ||x_s^k − z^{k−1}||_2^2 < ϵ then
            return ((x_s^k, y_s^k)_{s∈S}, z^k, ω^k, φ^k)
        end if
        ω_s^{k+1} ← ω_s^k + ρ(x_s^k − z^k), for s ∈ S
    end for
    return (((x_s^{k_max}, y_s^{k_max})_{s∈S}, z^{k_max}), ω^{k_max}, φ^{k_max})
end function
```

Optimal convergence $\lim_{k\to\infty} \phi^k = \phi^*$ can be established when

- the augmented Lagrangian L_s^ρ is modified to be strongly convex (add a quadratic proximal term in y);
- the sequence $\{\tau_k\}$ of SDM convergence tolerances is generated a priori so that $\sum_{k=1}^{\infty} \sqrt{\tau_k} < \infty$.

It can be shown that these above two conditions imply the satisfaction of assumptions used in the convergence analysis of inexact ADMM in Theorem 8 of [EB92].

- However, requiring the SDM termination condition to be satisfied for each such $\tau_k \geq 0$ is not needed in practice (and is undesirable); nor will it be required in theory, as we shall soon see.

Lemma

At each iteration, k, of Algorithm FW-PH, the value, $\phi^k = \sum_{s\in\mathcal{S}} p_s\phi_s^k$, is the value of the Lagrangian relaxation $\phi(\cdot)$ evaluated at a feasible Lagrangian dual feasible point $\tilde{\omega}^k$, and hence provides a lower bound on ζ^{LD}.

Proof.

- In iteration k, the problem solved, for each $s \in \mathcal{S}$, at Line 5 in the first iteration ($t = 1$) of Algorithm SDM, corresponds to the evaluation of the Lagrangian bound $\phi(\widetilde{\omega}^k)$, where

$$
\begin{aligned}
\widetilde{\omega}_s^k := \widehat{\omega}_s^1 &= \omega_s^k + \rho(\widetilde{x}_s - z^{k-1}) \\
&= \omega_s^k + \rho((1-\alpha)z^{k-1} + \alpha x_s^{k-1} - z^{k-1}) \\
&= \omega_s^k + \alpha\rho(x_s^{k-1} - z^{k-1}).
\end{aligned}
$$

- By construction, the points $((x_s^{k-1})_{s\in\mathcal{S}}, z^{k-1})$ always satisfy $\sum_{s\in\mathcal{S}} p_s(x_s^{k-1} - z^{k-1}) = 0$ and $\sum_{s\in\mathcal{S}} p_s\omega_s^k = 0$.
- Thus, $\sum_{s\in\mathcal{S}} p_s\widetilde{\omega}_s^k = 0$, so $\widetilde{\omega}^k$ is feasible for the Lagrangian dual problem, and $\phi(\widetilde{\omega}^k) = \sum_{s\in\mathcal{S}} p_s\phi_s^k \leq \zeta^{LD}$.

However we consider a convergence analysis of FW-PH that is *not* based on the optimal convergence (approximate or otherwise) of SDM.

- Convergence will depend instead on the SDM expansion of the inner approximations conv(V_s) “as needed”.

Lemma

For any given scenario $s \in \mathcal{S}$, let Algorithm SDM be applied to the iteration $k \geq 1$ PH subproblem

$$\min_{x,y} \left\{ L_s^{\rho}(x, y, z^{k-1}, \omega_s^k) \ : \ (x, y) \in conv(K_s) \right\} \tag{1}$$

For $1 \leq t < t_{max}$, if (x_s^t, y_s^t) is **not** *optimal for (1), then*

$$conv(V_s^{t+1}) \supset conv(V_s^t)$$

.

Optimal Convergence of FW-PH

Proposition

Let Algorithm FW-PH with $k_{max} = \infty$, $\epsilon = 0$, $\alpha \in \mathbb{R}$, and $t_{max} \geq 1$ be applied to the convexified separable deterministic equivalent SMIP, which is assumed to have an optimal solution. If either $t_{max} \geq 2$ or $\cap_{s \in \mathcal{S}} Proj_x(conv(V_s^0)) \neq \emptyset$ holds, then $\lim_{k\to\infty} \phi^k = \zeta^{LD}$.

Proof: (Basic ideas: See [BCD+16] for detail.)
First note that for any $t_{max} \geq 1$, the sequence of inner approximations $\text{conv}(V_s^k)$, $s \in \mathcal{S}$, will stabilize, in that, for some threshold $0 \leq \bar{k}_s$, we have for all $k \geq \bar{k}_s$

$$\text{conv}(V_s^k) =: \overline{D}_s \subseteq \text{conv}(K_s). \quad (2)$$

This follows due to $V_s^k \leftarrow V_s^{k-1} \cup \{(\widehat{x}_s, \widehat{y}_s)\}$, where $(\widehat{x}_s, \widehat{y}_s)$ is a vertex of $\text{conv}(K_s)$. Since each polyhedron $\text{conv}(K_s)$, $s \in \mathcal{S}$ has only a finite number of such vertices, the stabilization (2) must occur at some $\bar{k}_s < \infty$.

The stabilizations (2), $s \in \mathcal{S}$, are reached at some iteration $\bar{k} := \max_{s \in \mathcal{S}} \left\{ \bar{k}_s \right\}$. Noting that $\overline{D}_s = \text{conv}(V_s^k)$ for $k > \overline{k}$ we must have

$$(x_s^k, y_s^k) \in \operatorname*{argmin}_{x,y} \left\{ L_s^\rho(x, y, z^{k-1}, \omega_s^k) : (x, y) \in \text{conv}(K_s) \right\}. \quad (3)$$

Otherwise, due to Lemma 2, the call to SDM on Line 8 must return $V_s^k \supset V_s^{k-1}$, contradicting the stabilization (2).

Therefore, the $k \geq \bar{k}$ iterations of Algorithm FW-PH are identical to Algorithm PH iterations applied to Conv-SMIP, and so Proposition 1 implies that

1. $\lim_{k\to\infty} x_s^k - z^k = 0$, $s \in \mathcal{S}$, and
2. $\lim_{k\to\infty} \sum_{s \in \mathcal{S}} \phi_s(\omega_s^k + \alpha(x_s^{k-1} - z^{k-1})) = \lim_{k\to\infty} \sum_{s \in \mathcal{S}} \phi_s(\omega_s^k) = \lim_{k\to\infty} \phi(\omega^k) = \zeta^{LD}$ for all $\alpha \in \mathbb{R}$.

In the case $t_{max} = 1$ does need some consideration of extra issues (which we omit here).

The stabilizations (2), $s \in \mathcal{S}$, are reached at some iteration $\bar{k} := \max_{s \in \mathcal{S}} \{\bar{k}_s\}$. Noting that $\overline{D}_s = \text{conv}(V_s^k)$ for $k > \overline{k}$ we must have

$$(x_s^k, y_s^k) \in \underset{x,y}{\text{argmin}} \left\{ L_s^\rho(x, y, z^{k-1}, \omega_s^k) : (x, y) \in \text{conv}(K_s) \right\}. \quad (3)$$

Otherwise, due to Lemma 2, the call to SDM on Line 8 must return $V_s^k \supset V_s^{k-1}$, contradicting the stabilization (2).

Therefore, the $k \geq \bar{k}$ iterations of Algorithm FW-PH are identical to Algorithm PH iterations applied to Conv-SMIP, and so Proposition 1 implies that

1. $\lim_{k\to\infty} x_s^k - z^k = 0$, $s \in \mathcal{S}$, and
2. $\lim_{k\to\infty} \sum_{s\in\mathcal{S}} \phi_s(\omega_s^k + \alpha(x_s^{k-1} - z^{k-1})) = \lim_{k\to\infty} \sum_{s\in\mathcal{S}} \phi_s(\omega_s^k) = \lim_{k\to\infty} \phi(\omega^k) = \zeta^{LD}$ for all $\alpha \in \mathbb{R}$.

In the case $t_{max} = 1$ does need some consideration of extra issues (which we omit here).

Outline

Stochastic Mixed-Integer Optimization: Background

Algorithm FW-PH

Computational Experiments

Generating Primal Solutions

Computational Experiments

- We performed computations using a C++ implementation of Algorithms PH ($D_s = K_s$, $s \in \mathcal{S}$) and FW-PH using CPLEX 12.5 [IBM] as the solver.
- Computations run on the Raijin cluster:
  - high performance computing (HPC) environment;
  - maintained by Australia's National Computing Infrastructure (NCI) and supported by the Australian Government [NCI];
- In the experiments with Algorithms PH and FW-PH, we set the convergence tolerance $\epsilon = 10^{-3}$ and the maximum number of outer loop iterations at $k_{max} = 200$.
- For Algorithm FW-PH, we set $t_{max} = 1$.
- Also, for all experiments performed, we set $\omega^0 = 0$.

Computational Experiments

- We performed computations using a C++ implementation of Algorithms PH ($D_s = K_s$, $s \in \mathcal{S}$) and FW-PH using CPLEX 12.5 [IBM] as the solver.
- Computations run on the Raijin cluster:
  - high performance computing (HPC) environment;
  - maintained by Australia's National Computing Infrastructure (NCI) and supported by the Australian Government [NCI];
- In the experiments with Algorithms PH and FW-PH, we set the convergence tolerance $\epsilon = 10^{-3}$ and the maximum number of outer loop iterations at $k_{max} = 200$.
- For Algorithm FW-PH, we set $t_{max} = 1$.
- Also, for all experiments performed, we set $\omega^0 = 0$.

Computational Experiments

- We performed computations using a C++ implementation of Algorithms PH ($D_s = K_s$, $s \in \mathcal{S}$) and FW-PH using CPLEX 12.5 [IBM] as the solver.
- Computations run on the Raijin cluster:
  - high performance computing (HPC) environment;
  - maintained by Australia's National Computing Infrastructure (NCI) and supported by the Australian Government [NCI];
- In the experiments with Algorithms PH and FW-PH, we set the convergence tolerance $\epsilon = 10^{-3}$ and the maximum number of outer loop iterations at $k_{max} = 200$.
- For Algorithm FW-PH, we set $t_{max} = 1$.
- Also, for all experiments performed, we set $\omega^0 = 0$.

Computational Experiments

- We performed computations using a C++ implementation of Algorithms PH ($D_s = K_s$, $s \in \mathcal{S}$) and FW-PH using CPLEX 12.5 [IBM] as the solver.
- Computations run on the Raijin cluster:
  - high performance computing (HPC) environment;
  - maintained by Australia's National Computing Infrastructure (NCI) and supported by the Australian Government [NCI];
- In the experiments with Algorithms PH and FW-PH, we set the convergence tolerance $\epsilon = 10^{-3}$ and the maximum number of outer loop iterations at $k_{max} = 200$.
- For Algorithm FW-PH, we set $t_{max} = 1$.
- Also, for all experiments performed, we set $\omega^0 = 0$.

- Two sets of Algorithm FW-PH experiments correspond to variants considering $\alpha = 1$ and $\alpha = 0$.
- Computations were performed on four problems:
  1. the CAP (capacitated facility locations) instance 101 with the first 250 scenarios (CAP-101-250) [BDGL14],
  2. the DCAP (dynamic capacity allocations) instance DCAP233_500 with 500 scenarios,
  3. the SSLP (server location under uncertainty) instances SSLP5.25.50 with 50 scenarios (SSLP-5-25-50) and
  4. SSLP10.50.100 with 100 scenarios (SSLP-10-50-100).
- The latter three problems are described in detail in [Nta04, GTU] and accessible at [GTU].
- All computational experiments were allowed to run for a maximum of two hours in wall clock time.

	Percentage gap			# Iterations			Termination		
Penalty	PH	FW-PH $\alpha = 0$	FW-PH $\alpha = 1$	PH	FW-PH $\alpha = 0$	FW-PH $\alpha = 1$	PH	FW-PH $\alpha = 0$	FW-PH $\alpha = 1$
20	0.08%	0.10%	0.11%	466	439	430	T	T	T
100	0.01%	0.00%	0.00%	178	406	437	C	T	T
500	0.07%	0.00%	0.00%	468	92	93	T	C	C
1000	0.15%	0.00%	0.00%	516	127	130	T	C	C
2500	0.34%	0.00%	0.00%	469	259	274	T	C	C
5000	0.66%	0.00%	0.00%	33	431	464	C	T	T
7500	0.99%	0.00%	0.00%	28	18	19	C	C	C
15000	1.59%	0.00%	0.00%	567	28	33	T	C	C

Table: Result summary for CAP-101-250, with the absolute percentage gap based on the known optimal value 733827.3

	Percentage gap			# Iterations			Termination		
Penalty	PH	FW-PH $\alpha = 0$	FW-PH $\alpha = 1$	PH	FW-PH $\alpha = 0$	FW-PH $\alpha = 1$	PH	FW-PH $\alpha = 0$	FW-PH $\alpha = 1$
2	0.13%	0.12%	0.12%	1717	574	600	T	T	T
5	0.22%	0.09%	0.09%	2074	589	574	T	T	T
10	0.23%	0.07%	0.07%	2598	592	587	T	T	T
20	0.35%	0.07%	0.07%	1942	590	599	T	T	T
50	1.25%	0.06%	0.06%	2718	597	533	T	T	T
100	1.29%	0.06%	0.06%	2772	428	438	T	C	C
200	2.58%	0.06%	0.06%	2695	256	262	T	C	C
500	2.58%	0.07%	0.07%	2871	244	246	T	C	C

Table: Result summary for DCAP-233-500, with the absolute percentage gap based on the best known lower bound 1737.7.

	Percentage gap			# Iterations			Termination		
Penalty	PH	FW-PH $\alpha = 0$	FW-PH $\alpha = 1$	PH	FW-PH $\alpha = 0$	FW-PH $\alpha = 1$	PH	FW-PH $\alpha = 0$	FW-PH $\alpha = 1$
1	0.30%	0.00%	0.00%	105	115	116	C	C	C
2	0.73%	0.00%	0.00%	51	56	56	C	C	C
5	0.91%	0.00%	0.00%	25	26	27	C	C	C
15	3.15%	0.00%	0.00%	12	16	17	C	C	C
30	6.45%	0.00%	0.00%	12	18	18	C	C	C
50	9.48%	0.00%	0.00%	18	25	26	C	C	C
100	9.48%	0.00%	0.00%	8	45	45	C	C	C

Table: Result summary for SSLP-5-25-50, with the absolute percentage gap based on the known optimal value -121.6

	Percentage gap			# Iterations			Termination		
Penalty	PH	FW-PH $\alpha = 0$	FW-PH $\alpha = 1$	PH	FW-PH $\alpha = 0$	FW-PH $\alpha = 1$	PH	FW-PH $\alpha = 0$	FW-PH $\alpha = 1$
1	0.57%	0.22%	0.23%	126	234	233	T	T	T
2	0.63%	0.03%	0.03%	127	226	228	T	T	T
5	1.00%	0.00%	0.00%	104	219	220	C	T	T
15	2.92%	0.00%	0.00%	33	45	118	C	C	C
30	4.63%	0.00%	0.00%	18	21	22	C	C	C
50	4.63%	0.00%	0.00%	11	26	27	C	C	C
100	4.63%	0.00%	0.00%	9	43	45	C	C	C

Table: Result summary for SSLP-10-50-100, with the absolute percentage gap based on the known optimal value -354.2

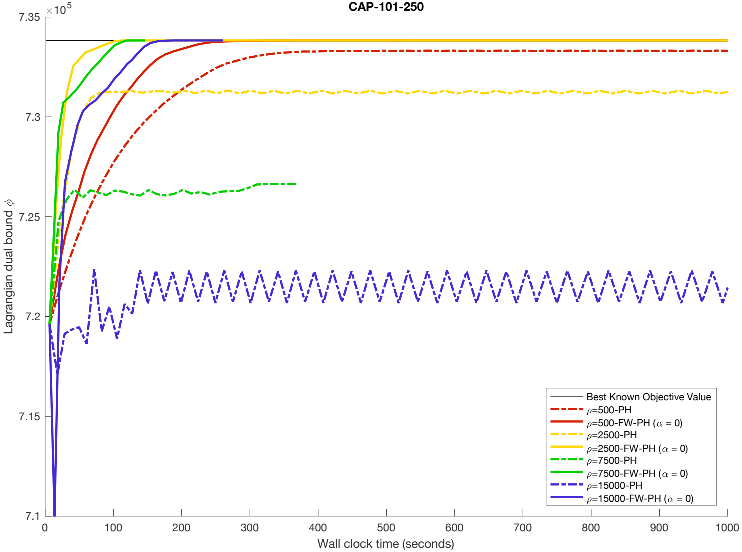

Figure: Convergence profile for CAP-101-250

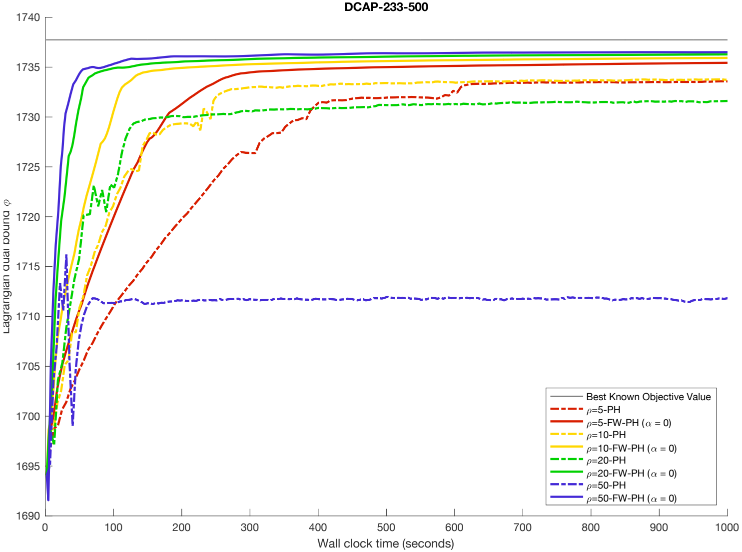

Figure: Convergence profile for DCAP-233-500

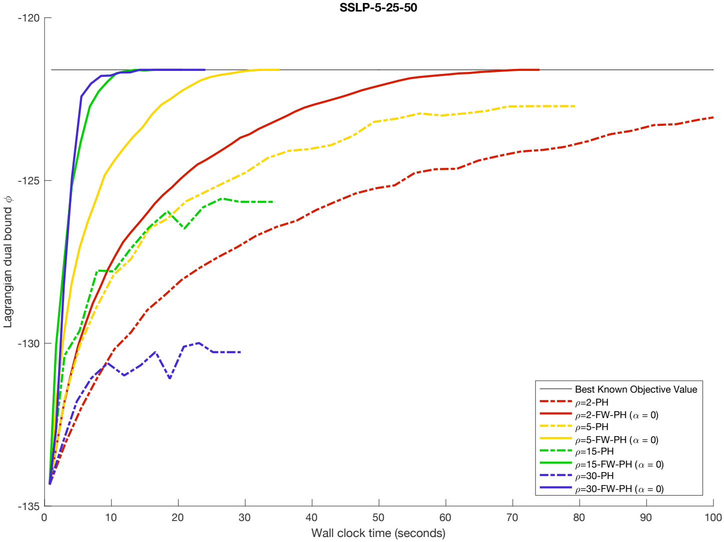

Figure: Convergence profile for SSLP-5-25-50 (chosen penalties match those of Figure 2 of [GHR+15])

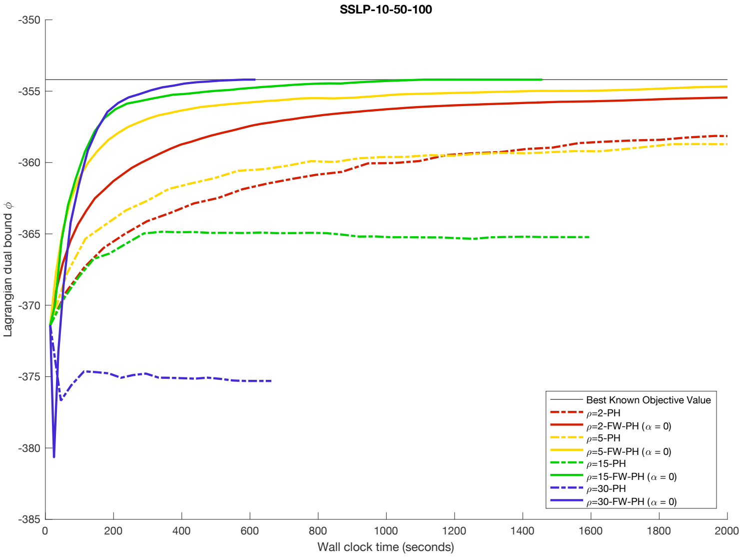

Figure: Convergence profile for SSLP-10-50-100

Outline

Generating Primal Solutions

- The FW-FW solves the problem involving the convexified feasible region. Thus the concensus z contains fractional values for integer variables. Thus we try three types of additional steps to extract an optimal solutuion.
- **H1**: Use the last x_s (i.e., when termination of FW-PH is observed) added to V_s, $s \in \mathcal{S}$, as candidate first-stage solutions, solve for the corresponding second stage variables and reporting that with the best objective value.
- **H2 and H3**: The second and third strategies consist of solving the MIQPs that would have been solved in PH for the current value of z obtained from FW-PH, which returns integral solutions that can be evaluated in the same manner as before.
- In **H2** we keep the penalty term unchanged while in **H3** we use for this last step the smallest ρ considered for each problem.

Generating Primal Solutions

- The FW-FW solves the problem involving the convexified feasible region. Thus the concensus z contains fractional values for integer variables. Thus we try three types of additional steps to extract an optimal solutuion.
- **H1**: Use the last x_s (i.e., when termination of FW-PH is observed) added to V_s, $s \in \mathcal{S}$, as candidate first-stage solutions, solve for the corresponding second stage variables and reporting that with the best objective value.
- **H2 and H3**: The second and third strategies consist of solving the MIQPs that would have been solved in PH for the current value of z obtained from FW-PH, which returns integral solutions that can be evaluated in the same manner as before.
- In **H2** we keep the penalty term unchanged while in **H3** we use for this last step the smallest ρ considered for each problem.

Generating Primal Solutions

- The FW-FW solves the problem involving the convexified feasible region. Thus the concensus z contains fractional values for integer variables. Thus we try three types of additional steps to extract an optimal solutuion.
- **H1**: Use the last x_s (i.e., when termination of FW-PH is observed) added to V_s, $s \in \mathcal{S}$, as candidate first-stage solutions, solve for the corresponding second stage variables and reporting that with the best objective value.
- **H2 and H3**: The second and third strategies consist of solving the MIQPs that would have been solved in PH for the current value of z obtained from FW-PH, which returns integral solutions that can be evaluated in the same manner as before.
- In **H2** we keep the penalty term unchanged while in **H3** we use for this last step the smallest ρ considered for each problem.

Generating Primal Solutions

- The FW-FW solves the problem involving the convexified feasible region. Thus the concensus z contains fractional values for integer variables. Thus we try three types of additional steps to extract an optimal solutuion.
- **H1**: Use the last x_s (i.e., when termination of FW-PH is observed) added to V_s, $s \in \mathcal{S}$, as candidate first-stage solutions, solve for the corresponding second stage variables and reporting that with the best objective value.
- **H2 and H3**: The second and third strategies consist of solving the MIQPs that would have been solved in PH for the current value of z obtained from FW-PH, which returns integral solutions that can be evaluated in the same manner as before.
- In **H2** we keep the penalty term unchanged while in **H3** we use for this last step the smallest ρ considered for each problem.

ρ	PH	FW-PH H1	FW-PH H2	FW-PH H3
20	0.14%	0.15%	0.15%	0.15%
100	0.01%	0.00%	0.00%	0.00%
500	0.07%	0.00%	0.00%	0.00%
1000	0.15%	0.00%	0.00%	0.00%
2500	0.34%	0.00%	0.00%	0.00%
5000	0.65%	0.00%	0.00%	0.00%
7500	0.98%	0.00%	0.00%	0.00%
15000	1.66%	0.00%	0.00%	0.00%

Table: CAP-101-250

ρ	PH	FW-PH		
		H1	H2	H3
2	0.64%	0.71%	0.31%	0.31%
5	0.63%	0.29%	0.59%	0.59%
10	0.64%	0.28%	0.49%	0.49%
20	0.51%	0.11%	0.48%	0.48%
50	1.23%	0.26%	0.48%	0.26%
100	1.68%	0.47%	0.48%	0.46%
200	2.92%	2.82%	0.48%	0.47%
500	2.51%	0.19%	0.48%	0.08%

Table: DCAP-233-500

ρ	PH	FW-PH		
		H1	H2	H3
1	0.31%	0.00%	0.00%	0.00%
2	0.73%	0.00%	0.00%	0.00%
5	0.92%	0.00%	0.00%	0.00%
15	3.25%	0.00%	0.00%	0.00%
30	6.90%	0.00%	0.00%	0.00%
50	10.48%	0.00%	0.00%	0.00%
100	12.91%	0.00%	0.00%	0.00%

Table: SSLP-5-25-50

ρ	PH	FW-PH H1	FW-PH H2	FW-PH H3
1	0.59%	0.25%	0.25%	0.25%
2	0.57%	0.03%	0.03%	0.03%
5	1.01%	0.00%	0.00%	0.00%
15	3.01%	0.00%	0.00%	0.00%
30	4.86%	0.00%	0.00%	0.00%
50	4.86%	0.00%	0.00%	0.00%
100	4.86%	0.00%	0.00%	0.00%

Table: SSLP-10-50-100

Thank you!

A. Agustín, A. Alonso-Ayuso, L. F. Escudero, and C. Pizarro.
On air traffic flow management with rerouting. Part II: Stochastic case.
European Journal of Operational Research, 219(1):167–177, 2012.

N.L. Boland, J. Christiansen, B. Dandurand, A. Eberhard, J. Linderoth, J. Luedtke, and F. Oliveira.
Progressive hedging with a Frank-Wolfe based method for computing stochastic mixed-integer programming Lagrangian dual bounds.
Optimization Online, 2016.

M. Bodur, S. Dash, O. Günlük, and J. Luedtke.
Strengthened benders cuts for stochastic integer programs with continuous recourse.
2014.
Last accessed on 13 January 2015.

D.P. Bertsekas.
Constrained Optimization and Lagrange Multiplier Methods.
Academic Press, 1982.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein.
Distributed optimization and statistical learning via the alternating direction method of multipliers.
Foundation and Trends in Machine Learning, 3(1):1–122, 2011.

Fernando Badilla Veliz, Jean-Paul Watson, Andres Weintraub, Roger J.-B. Wets, and David L. Woodruff.
Stochastic optimization models in forest planning: a progressive hedging solution approach.
Annals of Operations Research, 232:259–274, 2015.

Claus C Carøe and Rüdiger Schultz.
Dual decomposition in stochastic integer programming.
Operations Research Letters, 24(1):37–45, 1999.

J. Eckstein and D.P. Bertsekas.
On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators.
Mathematical Programming, 55(1-3):293–318, 1992.

J. Eckstein and W. Yao.

Understanding the convergence of the alternating direction method of multipliers: Theoretical and computational perspectives.
Technical report, Rutgers University, 2014.

M. Frank and P. Wolfe.
An algorithm for quadratic programming.
Naval Research Logistics Quarterly, 3:149–154, 1956.

D. Gade, G. Hackebeil, S.M. Ryan, J.-P. Watson, R. J-B Wets, and D.L. Woodruff.
Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs.
Technical report, Graduate School of Management, UC Davis, 2015.

D. Gabay and B. Mercier.
A dual algorithm for the solution of nonlinear variational problems via finite element approximation.
Computers and Mathematics with Applications, 2:17–40, 1976.

Last accessed 23 December, 2015.

M. R. Hestenes.
Multiplier and gradient methods.
Journal of Optimization Theory and Applications, pages 303–320, 1969.

C.A. Holloway.
An extension of the Frank and Wolfe method of feasible directions.
Mathematical Programming, 6(1):14–27, 1974.

IBM Corporation.
IBM ILOG CPLEX V12.5.
Last accessed 28 Jan 2016.

G. Laporte, F. V. Louveaux, and H. Mercure.
The vehicle routing problem with stochastic travel times.
Transportation Science, 26:161–170, 1992.

M. Lubin, K. Martin, C.G. Petra, and B. Sandıkçı.
On parallelizing dual decomposition in stochastic integer programming.
Operations Research Letters, 41(3):252–258, 2013.

F. V. Louveaux.
Discrete stochastic location models.
Annals of Operations Research, 6:23–34, 1986.

A. Løkketangen and D. Woodruff.
Progressive hedging and tabu search applied to mixed integer (0,1) multi-stage stochastic programming.
Journal of Heuristics, 2(2):111–128, 1996.

L. Ntaimo, J. A. Gallego Arrubla, C. Stripling, J. Young, and T. Spencer.
A stochastic programming standard response model for wildfire initial attack planning.
Canadian Journal of Forest Research, 42(6):987–1001, 2012.

Last accessed 20 April, 2016.

M. P. Nowak and W. Römisch.
Stochastic Lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty.
Annals of Operations Research, 100:251–272, 2000.

Lewis Ntaimo.
Decomposition Algorithms for Stochastic Combinatorial Optimization: Computational Experiments and Extensions.
PhD thesis, 2004.

G.L. Nemhauser and L.A. Wolsey.
Integer and combinatorial optimization.
Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1988.

O. Y. Özaltın, O. A. Prokopyev, A. J. Schaefer, and M. S. Roberts.
Optimizing the societal benefits of the annual influenza vaccine: A stochastic programming approach.
Operations Research, 59:1131–1143, 2011.

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

R.T. Rockafellar.

The multiplier method of hestenes and powell applied to convex programming.
Journal of Optimization Theory and Applications, 12(6):555–562, 1973.

J. Salmeron, R. K. Wood, and D. P. Morton.
A stochastic program for optimizing military sealift subject to attack.
Military Operations Research, 14:19–39, 2009.

S. Takriti, J. Birge, and E. Long.
A stochastic model for the unit commitment problem.
IEEE Transactions on Power Systems, 11(3):1497–1508, 1996.

M. W. Tanner, L. Sattenspiel, and L. Ntaimo.
Finding optimal vaccination strategies under parameter uncertainty using stochastic programming.
Mathematical Biosciences, 215(2):144–151, 2008.

B. Von Hohenbalken.
Simplicial decomposition in nonlinear programming algorithms.
Mathematical Programming, 13(1):49–68, 1977.

J.P. Watson and D. Woodruff.
Progressive hedging innovations for a class of stochastic resource allocation problems.
Computational Management Science, 8(4):355–370, 2011.