New rigorous decomposition methods for mixed-integer linear and nonlinear programming

by

EMMANUEL OGBE

A thesis submitted to the
Department of Chemical Engineering
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen's University
Kingston, Ontario, Canada
December 2016

Copyright © EMMANUEL OGBE, 2016

Abstract

Process systems design, operation and synthesis problems under uncertainty can readily be formulated as two-stage stochastic mixed-integer linear and nonlinear (nonconvex) programming (MILP and MINLP) problems. These problems, with a scenario based formulation, lead to large-scale MILPs/MINLPs that are well structured.

The first part of the thesis proposes a new finitely convergent cross decomposition method (CD), where Benders decomposition (BD) and Dantzig-Wolfe decomposition (DWD) are combined in a unified framework to improve the solution of scenario based two-stage stochastic MILPs. This method alternates between DWD iterations and BD iterations, where DWD restricted master problems and BD primal problems yield a sequence of upper bounds, and BD relaxed master problems yield a sequence of lower bounds. A variant of CD, which includes multiple columns per iteration of DW restricted master problem and multiple cuts per iteration of BD relaxed master problem, called multicolumn-multicut CD is then developed to improve solution time. Finally, an extended cross decomposition method (ECD) for solving two-stage stochastic programs with risk constraints is proposed. In this approach, a CD approach at the first level and DWD at a second level is used to solve the original problem to optimality. ECD has a computational advantage over a bilevel decomposition strategy or solving the monolith problem using an MILP solver.

The second part of the thesis develops a joint decomposition approach combining Lagrangian decomposition (LD) and generalized Benders decomposition (GBD), to efficiently solve stochastic mixed-integer nonlinear nonconvex programming problems to global optimality, without the need for explicit branch and bound search. In this approach, LD subproblems and GBD subproblems are systematically solved in a single framework. The relaxed master problem obtained from the reformulation of the original problem, is solved only when necessary. A convexification of the relaxed master problem and a domain reduction procedure are integrated into the decomposition framework to improve solution efficiency. Using case studies taken from renewable resource and fossil-fuel based application in process systems engineering, it can be seen that these novel decomposition approaches have significant benefit over classical decomposition methods and state-of-the-art MILP/MINLP global optimization solvers.

Co-Authorship

This thesis includes manuscripts co-authored with Dr Xiang Li which have either been published or submitted for publication. The full list of manuscipts and chapters where they appear are detailed as follows:

Chapter 2 has been published as: A new cross decomposition method for stochastic mixed-integer linear programming, European Journal of Operational Research, 256 (2017), pp. 287-299.

Chapter 3 is written based on the conference paper: Multicolumn-multicut cross decomposition for stochastic mixed-integer linear programming, Computer Aided Chemical Engineering, 37 (2015) pp. 737-742.

Chapter 4 is the following submitted manuscript: Extended cross decomposition method for stochastic mixed-integer linear programs with strong and weak linking constraints, Computers & Chemical Engineering.

Chapter 5 has been submitted as the following manuscript: Joint decomposition method for multiscenario mixed-integer nonlinear nonconvex programs, Journal of Global Optimization.

In loving memory of my dad, ASP. John Otene Ogbe

Acknowledgements

I would like to sincerely thank my advisor, Professor Xiang Li, for his outstanding tutelage, excellent guidance and support all through the course of my program. Furthermore, I want to specially thank members of my thesis committee comprising of Professor Martin Guay, Professor James McLellan, Professor Bahman Gharesifard, and Professor Christopher Swartz. I am grateful to the discovery grant (RGPIN 418411-13) and the collaborative research and development grant (CRDPJ 485798-15) from Natural Sciences and Engineering Research Council of Canada (NSERC) for funding this research.

Finally, I also acknowledge the help and support of my mother, Mrs. Fidelia John Ogbe, my brothers, Adah, Raph, Tony, Pete and sisters, Florence, Bless and Joy, colleagues, all friends and well wishers in Canada, Nigeria, and beyond, who have supported me throughout my Ph.D. studies, and to God Almighty.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Model-based decision making via mathematical programming

Mathematical programming, as a field of applied mathematics, has evolved and seen tremendous application in a wide range of areas in engineering decision making; it has been a major area of *process systems engineering* (PSE) over the last four decades. It is concerned primarily with making optimal decision in the use of scarce resources to meet some desired objectives. Application of mathematical programming, also called mathematical optimization, to PSE ranges from modeling and process development to process synthesis and design and then to process operations, control, planning and scheduling [1] [2] [3]. Example of design and synthesis problems include chemical reactor and heat exchanger networks synthesis, distillation sequencing, process flowsheeting [4]. A major reason for the numerous application of mathematical programming is the fact that in many of these problems, it is often not easy to find the optimal solution amongst the set of the very many alternative solutions. Moreso, in many cases, finding an optimum can lead to significant economic savings, a greater energy efficiency and a cleaner environment.

The key aspect of mathematical programming for PSE is the availability of models (specifically mathematical models). Models are necessary in order to define the objective function and the constraints sets. Models in PSE can come from mass, momentum and energy balance equations, kinetic relationships, mass transfer relations, equilibrium relationships, etc. Model-based decision making, which involves application of modeling for system analysis, design, verification and validation, is the core of this thesis. Accuracy and reliability of models is therefore paramount, and can lead to more realistic decisions. A general optimization problem can be cast in the following form:

$$\begin{aligned} \min_{x} \ & f(x) \\ s.t \ \ & g(x) \leq 0, \\ & x \in D \subset \mathbb{R}^n, \end{aligned} \tag{P*}$$

where $D = \{x \in \mathbb{R}^n : Ax \leq d\}$.

The functions $f : D \to \mathbb{R}$ and $g : D \to \mathbb{R}^m$ are continuous, and parameters $A \in \mathbb{R}^{n \times m}$ and $d \in \mathbb{R}^m$. A minima to Problem (P*) exist if the set $D \cap \{x \in \mathbb{R}^n : g(x) \leq 0\}$ is nonempty and compact (Weierstrass extreme value theorem [5]).

Depending on how f and g are defined, Problem (P*) can be the following:

1. Linear program (LP), if f and g are affine.

2. Nonlinear program (NLP), if at least one of f or $g_j, \forall j = 1, ..., m$ is nonlinear. A nonlinear program can be,

   (a) convex, if f and g are convex functions.

(b) nonconvex, if either f or $g_j, \forall j = 1, ..., m$ is a nonconvex function.

Another class of optimization problems is when the variables in Problem (P*) include some integer values. Problem (P*) could be:

1. mixed-integer linear program (MILP), if f and g are affine and some $x_i, \forall i = 1, ..., n$ are integer variables, or,

2. mixed-integer nonlinear program (MINLP), if f or $g_j, \forall j = 1, ..., m$ is nonlinear and some $x_i, \forall i = 1, ..., n$ are integer variables.

An excellent book to review convex programming is [6]. The nice feature of a convex program is that *every local solution is a global solution.* Nonconvexities, in general, can be caused by either nonconvex functions and/or sets (as mentioned earlier) or the presence of integer variables. Hence, the following classes of problems are nonconvex; (a) MILP, (b) nonconvex NLP, (c) convex and nonconvex MINLP. This clearly means that alot of practical problems in PSE are nonconvex problems. Nonconvex programming problems can be solved by a special class of optimization methods that can solve Problem (P*) to ϵ-optimality called *deterministic global optimization* [7] [8]. Deterministic global optimization is discussed in the next section.

Application of MILP in PSE include production planning and scheduling [9], optimization of supply chains [10], determining minimum number of matches in heat exchanger synthesis [11], heat integration in sharp distillation column sequences [12], scheduling of batch processes [13], etc. MINLP typically arises in synthesis, design and scheduling problems [1] [3], e.g. synthesis of distillation-based seperation systems [14], design of complex reactor networks [15], design and scheduling of multipurpose

plants [16], finding optimum pump configuration [17], etc. They also give much greater flexibility for modeling a wide variety of problems [1].

1.2 Large-scale optimization problems with decomposable structures

The different application areas described in section 1.1, require that decisions are made in the presence of uncertainty [18]. Uncertainty for instance affects the price of fuel, demand of electricity, supply of raw materials, etc, in these applications. One approach to handle uncertainty in mathematical programming is *stochastic programming*. In the two-stage stochastic programming approach, decisions are divided into first-stage or *here and now* decisions, and second-stage or *wait and see* decisions [19] [20] [21]. Let us consider Problem (P*) with $x = (x_1, x_2)$, $f = (f_1, f_2)$ and $g = (g_1, g_2)$, a sample space Ω, with element $\omega \in \Omega$. We denote ζ as a finite dimensional random vector and $\mathbb{E}_\zeta$, the mathematical expectation with respect to ζ. The two-stage stochastic program for Problem (P*) is then:

$$\min_{x_1 \in X_1} \; f_1(x_1) + \mathbb{E}_\zeta Q(x_1, \zeta) \tag{TSSP0}$$

where,

$$\begin{aligned} Q(x_1, \zeta) = \min_{x_2} \; & f_2(x_2, \zeta) \\ s.t \quad & g_1(x_1) + g_2(x_2, \zeta) \leq 0, \\ & x_2 \in X_2(\zeta), \end{aligned}$$

where x_1 and x_2 are the first and second-stage decision variables respectively. In the first-stage, decisions are made without realization of uncertainty. Examples of such kinds of decisions in PSE are design decisions, which are to determine the number or capacity of process units, whether or not to include a process unit, etc. In the second-stage, decisions called recourse decisions are made after the realization of uncertainty. Second-stage decisions are typically operational decisions such as determining material flowrates, pressures, concentrations in a process unit. A typical approach to solve Problem (TSSP0) is to approximate the uncertainty set Ξ by a finite subset $\tilde{\Xi}$ that follow a discrete distribution with finite support $\tilde{\Xi} = \{\zeta_1, ..., \zeta_s\} \subset \Xi$. Each realization (called scenarios) $\omega \in \{1, ..., s\} \in \Omega$, has a associated probability p_ω. Discrete distributions have a lot of applications, either directly or empirically, as approximations to the underlying probability distribution [22]. Problem (TSSP0) can then be restated as the so-called *deterministic equivalent program* [21]:

$$\begin{aligned} &\min_{x_1, x_{2,1}, ..., x_{2,s}} \sum_{\omega=1}^{s} p_\omega [f_1(x_1) + f_2(x_{2,\omega}, \zeta_\omega)] \\ s.t \quad &g_1(x_1) + g_2(x_{2,\omega}, \zeta_\omega) \leq 0, \quad \forall \omega \in \{1, ..., s\} \\ &x_1 \in X_1, x_2 \in X_2(\zeta_\omega), \quad \forall \omega \in \{1, ..., s\} \end{aligned}$$

or more conveniently as;

$$\begin{aligned} &\min_{x_1, x_{2,1}, ..., x_{2,s}} \sum_{\omega=1}^{s} p_\omega [f_1(x_1) + f_{2,\omega}(x_{2,\omega})] \\ &s.t \quad g_1(x_1) + g_{2,\omega}(x_{2,\omega}) \leq 0, \quad \forall \omega \in \{1, ..., s\} \\ &\qquad x_1 \in X_1, x_{2,\omega} \in X_{2,\omega}, \quad \forall \omega \in \{1, ..., s\} \end{aligned} \tag{TSSP}$$

The above two-stage formulation implicitly assumes that $\forall \omega \in \{1, ..., s\}$,

$$x_1 \in P(\omega) = \{x_1 \in X_1 : \exists x_{2,\omega} \in X_{2,\omega}, g_1(x_1) + g_{2,\omega}(x_{2,\omega}) \leq 0\}.$$

An alternative to the above is to introduce duplicate variables $x_{1,1}, ..., x_{1,s}$ for x_1 and rewrite Problem (TSSP) as the following:

$$\begin{aligned} &\min_{\substack{x_1, x_{1,1}, ..., x_{1,s} \\ x_{2,1}, ..., x_{2,s}}} \sum_{\omega=1}^{s} p_\omega [f_1(x_{1,\omega}) + f_{2,\omega}(x_{2,\omega})] \\ &s.t \quad x_{1,\omega} = x_{1,\omega+1}, \quad \forall \omega \in \{1, ..., s-1\}, \\ &\qquad g_1(x_{1,\omega}) + g_{2,\omega}(x_{2,\omega}) \leq 0, \quad \forall \omega \in \{1, ..., s\} \\ &\qquad x_{1,\omega} \in X_1, x_{2,\omega} \in X_{2,\omega}, \quad \forall \omega \{1, ..., s\} \end{aligned} \tag{TSSP1}$$

where $x_{1,1} = x_{1,2} = ... = x_{1,s}$ are the *nonanticipativity constraints* [23] [24] [25]. The goal of stochastic programming is to minimize the expected cost. Considering expected cost alone however, usually leads to a risk-neutral formulation [21] [26]. When the effect of risk is considered in the two-stage stochastic programming formulation, such as the value-at-risk or the condition value-at-risk variability metric, a more realistic risk-averse stochastic program results [27] [19]. Another approach to modeling

uncertainty in stochastic programming is the use of *probabilistic* or *chance constrained* programming [28]. Probabilistic programming ensures that the probability of an undesirable outcome is limited by a specified threshold [29]. This is a useful approach when ensuring feasibility over all realization of uncertainty is difficult.

Two-stage stochastic programming with or without risk/chance constraints lead to large-scale optimization problems when s is large. These problems are large-scale in the sense that the number of variables and constraints are large. However, these problems have special structure because the constraint coefficient matrix are usually quiet sparse. The problem structure can either be angular, dual-angular [30] [31], or hybrid-angular structure. These different block structures are shown in Figure 1.1. Dual-angular programs contain variables that links across all rows in the block structure e.g. Problem (TSSP). These variables (i.e. x_0 in Problem (TSSP)) are called *linking variables* and the associated constraints can be referred to as *weak linking constraints*. Angular programs contain *strong linking constraints* (referred to as *global constraints* in [31]) that links all variables in the associated blocks. The constraints $x_{1,1} = x_{1,2} = ... = x_{1,s}$ in Problem (TSSP1) are strong linking constraints. Hybrid-angular structures contain both strong and weak linking constraints, e.g. two-stage stochastic program with embedded chance or risk constraints, the chance or risk constraints take the form $h(x_1, x_{2,1}, ..., x_{2,\omega}) \leq 0$. More discussion on risk constraints is in Chapter 4.

Other kinds of problems with special structure such as: (i) *multiperiod optimization* [32], where decisions are made over finite time periods, or (ii) modeling dynamically decoupled subsystems with linking constraints, e.g. a boiler-turbine system

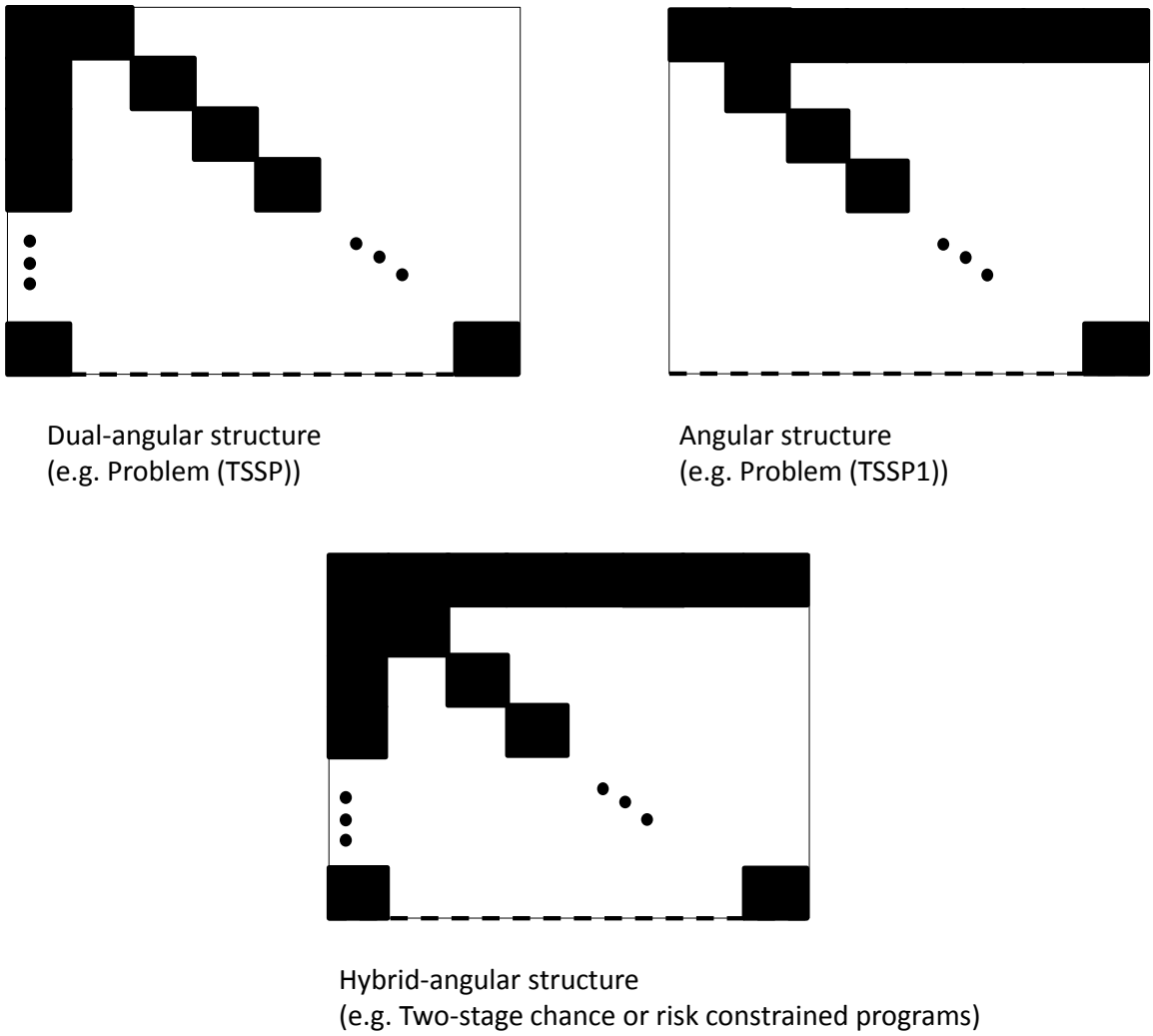


Figure 1.1: Dual-angular, angular and hybrid-angular structures

producing high pressure, middle pressure and low pressure steam as well as electricity; the boilers being the dynamically decoupled subsystems while the demand for various steam qualities and electrical power constitute the linking constraints [33]. These lead to large-scale optimization problems but they are however not considered in this thesis.

1.3 Decomposition based Approach for large-scale optimization problems

Approaches for solving large-scale mathematical programs may be divided broadly into two classes: (i) monolith (direct) methods, which involves solving the overall problems as a generic class of optimization problem such as LP, MILP, nonlinear convex or nonconvex programming, etc, and (ii) decomposition or partitioning techniques [30].

Decomposition based approaches are characterized by a splitting of the original problem into smaller independent subproblems and a coordinating problem, and solving these subproblems iteratively until convergence. The key to the validity of decomposition based approaches is that after solving the smaller subproblems for a finite number of times, the coordinating problem attains the optimal solution of the original problem; either precisely or within the prescribed tolerance, in most cases. The classical decomposition methods are Benders decomposition [34]/generalized Benders decomposition [35] (a variant of GBD is outer approximation by Duran and Grossmann [36]), Dantzig-Wolfe decomposition [37], Lagrangian decomposition [38] (variants include augmented Lagrangian method [39] , Progressive hedging [40] and the alternating direction of multiplier method [41] [42]).

J.F. Benders first introduced a decomposition technique, now called Benders decomposition (BD), to solve large-scale mixed-integer programming problems [34]. In stochastic programming literature, Van Slyke and Wets [43] [21] applied BD to dual-angular problems and called it L-shaped method (because of the L-shaped structure of angular programs). A recent review paper containing a list of application is [44]. Geoffrion developed the generalized Benders decomposition (GBD) for mixed-integer nonlinear programming problems [35], by using Lagrangian duality theory. In

BD/GBD, Problem (TSSP) is reformulated to the following problem by projection onto the space of x_1 thus:

$$\begin{aligned} &\min_{x_1} \ v(x_1) \\ &s.t \quad x_1 \in X_1 \cap V \end{aligned} \tag{Pproj*}$$

where,

$$\begin{aligned} v(x_1) = \inf_{x_{2,1},\ldots,x_{2,s}} &\sum_{\omega=1}^{s} p_\omega [f_1(x_1) + f_{2,\omega}(x_{2,\omega})] \\ s.t \quad & g_1(x_1) + g_{2,\omega}(x_{2,\omega}) \leq 0, \quad \forall \omega \in \{1, ..., s\} \\ & x_{2,\omega} \in X_{2,\omega}, \quad \forall \omega \in \{1, ..., s\} \end{aligned}$$

and,

$$V = \{x_1 \in X_1 : \exists x_2 \in X_2, g_1(x_1) + g_2(x_{2,\omega}) \leq 0, \quad \forall \omega \in \{1, ..., s\}\}.$$

The master problem, equivalent to Problem (TSSP), is constructed by applying Lagrangian duality on Problem (Pproj*). The relaxed master problem (the coordinating problem in this case), obtained by removing some of the constraints or cuts in the master problem, is updated by cuts derived by solving subproblems called primal problems. These cuts (or cutting planes [45]) are added iteratively to the relaxed master problem until it converges to the master problem. Further discussion of BD/GBD include multicut Benders decomposition [46] [47] and cut strengthening [48]. In GBD, dual information of the subproblems is required to construct the relaxed

master problem. In outer approximation, a variant of GBD, where primal information of the subproblems and differentiability property of the participating functions are required to construct the relaxed master problem [36].

Dantzig-Wolfe decomposition (DWD) was first developed for linear programs with angular structure by G. Dantzig and P. Wolfe [37]. Several applications, especially under the term column generation, appear in the literature [49] [50] [51]. DWD is based on the principle of convex combination. We can write a polyhedral set X_2 as:

$$X_2 = \{x_2 \in \mathbb{R}^{n_{x_2}} : x_2 = \sum_{j \in J} \theta^{\mathsf{j}} x_2^{\mathsf{j}}, \theta^{\mathsf{j}} \geq 0, \forall j \in J\}.$$

where x_2^{j} corresponds to the extreme points (called columns) in X_2 and J is the set containing indices of extreme points in X_2. A master problem, containing a convex hull representation of X_2 given above, is constructed. The extreme points or columns are generated by solving subproblems called pricing problems. Generation of columns iteratively refines the description of the restricted master problem (containing a subset of columns in the convex hull of X_2), ultimately leading to the convergence of the procedure. There is close relationship between BD and DWD, as BD applied to a problem is equivalent to applying DWD on the dual of the problem (only true for linear programming) [30] [39].

Lagrangian decomposition or relaxation (LD) (also closely related to DWD [52]) can be applied to large-scale programs with angular structure as well [53] [38]. LD are applicable to problems without a dual gap, which includes linear and convex nonlinear programming problems. In the LD procedure, the following dual problem for Problem (TSSP1) is obtained by dualizing the nonanticipativity constraints:

$$\max_{\lambda_1,...,\lambda_{s-1}\in\mathbb{R}^m} \quad D(\lambda_1,...,\lambda_{s-1}) \tag{DP*}$$

where $\lambda_1,...,\lambda_{s-1}$ are the Lagrange multipliers associated with the nonanticipativity constraints. The dual function D is given by,

$$\begin{aligned} D(\lambda_1,...,\lambda_{s-1}) = \min_{\substack{x_1,x_{1,1},...,x_{1,s}\\ x_{2,1},...,x_{2,s}}} \quad & L(x_1,x_{1,1}...,x_{1,s},x_{2,1}...,x_{2,s},\lambda_1,...,\lambda_{s-1}) \\ s.t \quad & g_1(x_{1,\omega})+g_{2,\omega}(x_{2,\omega})\leq 0, \quad \forall\omega\in\{1,...,s\}, \\ & x_1\in X_1, x_{2,\omega}\in X_{2,\omega}, \quad \forall\omega\in\{1,...,s\}, \end{aligned} \tag{DF*}$$

and the Lagrangian function L is,

$$L(x_1,x_{1,1}...,x_{1,s},x_{2,1}...,x_{2,s},\lambda_1,...,\lambda_{s-1}) = \sum_{\omega=1}^{s} p_\omega[f_1(x_1)+f_{2,\omega}(x_{2,\omega})] + \sum_{\omega=1}^{s-1}\lambda_\omega^{\mathrm{T}}(x_\omega - x_{1,\omega})$$

Problem (DP*) provides lower bounds on the original problem, but it is not solved directly. Problem (DF*), with Lagrange multipliers $\lambda_1,...,\lambda_{s-1}$ fixed, called Lagrangian subproblem is solved. $\lambda_1,...,\lambda_{s-1}$ can be generated by: (i) cutting plane methods, i.e. solving a master problem [54], or (ii) using heuristic based subgradient method [24] [39]. The issues in the application of Lagrangian decomposition are that; (a) the solution obtained from the Lagrangian subproblem may violate the constraints that have been relaxed, (b) the dual function, D, is seldom differentiable and (c) it may be difficult to obtain feasible solution to the original problem [55] [39]. To ensure convergence to the optimal solution for problems with a duality gap (e.g.

mixed-integer linear programs), Lagrangian decomposition is applied in the branch-and-bound framework [56]. One way to avoid numerical instability in LD is the augmented Lagrangian method [39], [57], where the augmented Lagrangian function, L_{aug}, has a stabilization term to penalize violation of infeasible solution, i.e.,

$$L_{aug}(x_1, x_{1,1}, ..., x_{1,s}, x_{2,1}, ..., x_{2,s}, \lambda_1, ..., \lambda_{s-1}) = \sum_{\omega=1}^{s} [p_\omega(f_1(x_1) + f_{2,\omega}(x_{2,\omega})) + \lambda_\omega^{\mathrm{T}}(x_\omega - x_{1,\omega}) + \frac{c}{2}||(x_\omega - x_{1,\omega})||^2$$

and,

$$D_{aug}(\lambda_1, ..., \lambda_{s-1}) = \min_{\substack{x_1, x_{1,1}, ..., x_{1,s} \\ x_{2,1}, ..., x_{2,s}}} L_{aug}(x_1, x_{1,1}, ..., x_{1,s}, x_{2,1} ..., x_{2,s}, \lambda_1, ..., \lambda_{s-1})$$
$$s.t \quad g_1(x_{1,\omega}) + g_{2,\omega}(x_{2,\omega}) \leq 0, \quad \forall \omega \in \{1, ..., s\},$$
$$x_1 \in X_1, x_{2,\omega} \in X_{2,\omega}, \quad \forall \omega \in \{1, ..., s\},$$

where c is a positive parameter, and $||.||$ is typically the $2-$norm. Progressive hedging introduced by R. Wets [40] and alternating direction of multiplier method (ADMM) [41] [42] are other decomposition methods based on the augmented Lagrangian function above and are not explored further. More discussions on BD/GBD, DWD and LD are in chapter 2 and 5. The key advantage of BD/GBD approach over the other decomposition methods is that, with the appropriate problem reformulation, it can guarantee rigorous global solutions for nonconvex optimization problems.

It is well known that BD is a cutting plane method [30]. All BD cuts form convex constraints in the BD master problem. As mentioned earlier, DWD on a

problem is equivalent to performing BD on the dual of the problem and therefore a cutting plane method [58]. The idea of cutting plane method for convex programs dates back to Kelly [59], which in turn has some connection to Gomory cuts [60] in integer programming. It has been proven that for convex programs, if the cuts are properly selected, the cutting plane method has geometric convergence rate [45] [58]. Geometric convergence is defined as the following:

$$||x^k - x^\star|| \leq c\delta^k, \quad \forall k,$$

where x^k, is the value at a particular iteration k, $x^\star$ is the optimal solution and c and δ are constants.

Rigorous proofs of geometric convergence for BD/GBD or DWD/LD is not known till date. This is because for convex programs, the standard implementation of BD and DWD may not yield proper cuts to achieve a geometric convergence rate, and for nonconvex programs the current proofs are not valid (as convexity of the feasible set is needed for the proof). Two important notes about BD/GBD or DWD/LD are the following:

1. Although theoretical rate of convergence of BD/GBD or DWD/LD is not known, no observations have been made in the literature to suggest that the number of BD/GBD or DWD/LD iterations could grow exponentially with the size of the overall problem (actually in the case studies to be presented in the subsequent chapters, no significant change in number of iterations can be seen for decomposition as the problem size increases).

2. Slow rate of convergence of BD/GBD or DWD/LD (called "tailing off effect")

have been widely recognized [51] [61] [62].

1.4 Global Optimization

Nonconvex optimization problems have been important throughout history in many engineering applications. Nevertheless, solutions to nonconvex problems had largely remained unexplored because of the huge amount of computational effort required. However, the work of G.P. McCormick in the late 1970s, led to a surge in activity in global optimization. The surge was partly due to the advancement of computer hardwares [63], and partly due to the realization that existing local optimization methods may find local solutions which are far away from global optimal solutions [18]. Global optimization of a nonconvex optimization problem (e.g. Problem (P*)) entails finding at least one point $x^\star \in D$ satisfying $f(x^\star) \leq f(x)$, $\forall x \in X$, where $X = D \cap \{x \in \mathbb{R}^n : g(x) \leq 0\}$ or show that Problem (P*) is infeasible. There are possibly two major difficulties in solving nonconvex optimization problems:

1. gradient-based search strategies for convex (local) optimization cannot guarantee global solution.

2. MILP belong to the class of $\mathcal{NP}$-complete problems [64] [65] and nonconvex continuous NLP is in class of $\mathcal{NP}$-hard [66], therefore in general, nonconvex optimization belong to the class of $\mathcal{NP}$-hard problems [67]. This means that in the worse case, the computational time grows exponentially with problem size.

A lot of practical engineering problems, however, can be solved rather efficiently. Global optimization methods refers to class of solution methods for nonconvex optimization problems. They can generally be categorized as stochastic or deterministic.

Stochastic methods rely on physical analogy by generating trial points which mimic the approach to an equilibrium point [1]. Examples include genetic algorithm, tabu search, particle swarm algorithms, simulated annealing, etc. These methods cannot guarantee that a global solution is obtained. Deterministic approaches on the other hand, operate by generating rigorous upper and lower bounds on the problem, that ultimately converge to an ϵ-optimal global solution. They typically provide mathematical guarantees for convergence to ϵ-global minimum in finite number of steps and is the focus of this thesis. General deterministic solution strategy for nonconvex optimization are based on the following key ingredients: branch-and-bound search, convex relaxation and relaxation strengthening.

In branch-and-bound [68] methods, the feasible set is relaxed and subsequently split into parts (branching) over which lower (and often upper) bounds of the objective function value can be determined (bounding). The details of the above step for Problem (P*), is as follows [66]:

1. start with a relaxed feasible set $M \supset X$ and partition M into finitely many subsets, $M_i, i \in I$.

2. for each subset M_i, determine lower bounds $\alpha(M_i)$ and possibly upper bounds $\beta(M_i)$, that satisfies

$$\alpha(M_i) \leq \min_{x \in M_i \cap X} f(x) \leq \beta(M_i).$$

Then $LBD = \min_{i \in I} \alpha(M_i)$ and $UBD = \min_{i \in I} \beta(M_i)$, where

$$LBD \leq \min_{x \in X} f(x) \leq UBD$$

3. if $UBD - LBD \leq \epsilon$, where ϵ is a small tolerance, then return the current solution as the ϵ-optimal global solution and stop.

4. otherwise select some of the subsets M_i, and partition these chosen subsets in order to obtain a refined partition of M (by fathoming some portions). Determine new bounds and the new partitions and repeat the process.

Convex relaxation (specifically continuous relaxation for MILP [69]) is one of the important tools in global optimization, and an essential aspect of the branch-and-bound process. Convex relaxations for a nonconvex problem (P*) are obtained by replacing nonconvex functions f, g by convex relaxation functions $\hat{f}$ and $\hat{g}$, where $\hat{f} \leq f$, $\hat{g} \leq g$ and the nonconvex set X by a convex relaxation $\hat{X} \supset X$. Because every local optimum is a global optimum for convex programs, solving the convex relaxation,

$$\min_{x \in \hat{X}} \hat{f}(x),$$

with a local optimizer will obtain a global minimum of the above problem, which is a lower bound to Problem (P*). Several techniques available for generating convex relaxations include McCormick relaxation [70], piecewise linear relaxation [71] [72], outer linearization [73] for factorable nonconvex functions and αBB relaxation for twice-differentiable nonconvex functions [74]. A hybrid relaxation approach is discussed in [75]. Consider Figure 1.2, for a univariate nonconvex function $f(x) = -x^2$ on the interval $[-1 \quad 2]$, a convex relaxation of the nonconvex function derived via αBB relaxation is shown. The convex function derived in this case, i.e., $\hat{f}(x) = -x - 2$, provides a lower bound to f and is infact the convex envelope of f (tightest possible convex relaxation).

The convex relaxation can often be very loose [2], and can be improved by strengthening. Typical examples of ways to strengthen convex relaxation are; addition of cutting planes [76] and domain reduction (either optimization based domain reduction or bound tightening [74], or marginal based domain reduction [77]). Figure 1.3 shows how a tighter convex relaxation of $f(x) = -x^2$, i.e. $\hat{f}(x) = -1.5x - 1$, is generated by fathoming part of the interval that does not contain the optimum (the shaded portion on the left), in other words, by reducing the range of x from $[-1 \quad 2]$ to $[-0.5 \quad 2]$, the convex relaxation of f is tightened.

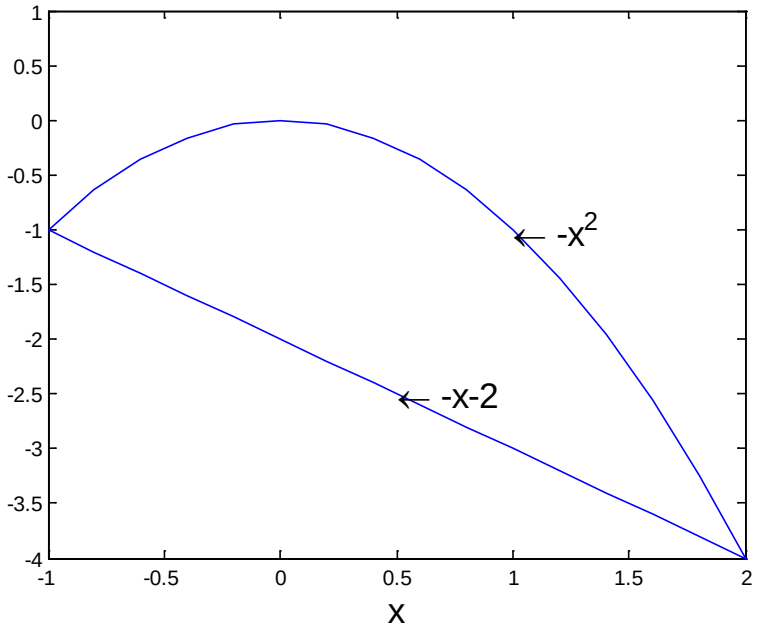


Figure 1.2: convex relaxation of $-x^2$ for $x \in [-1 \quad 2]$

Note that because global optimization methods rely on branch-and-bound search for solution, in the worse case, the solution time varies exponentially with the number of variables to be branched on. However, the difference between monolith and decomposition based approaches lie in the following; (i) standard monolith approaches

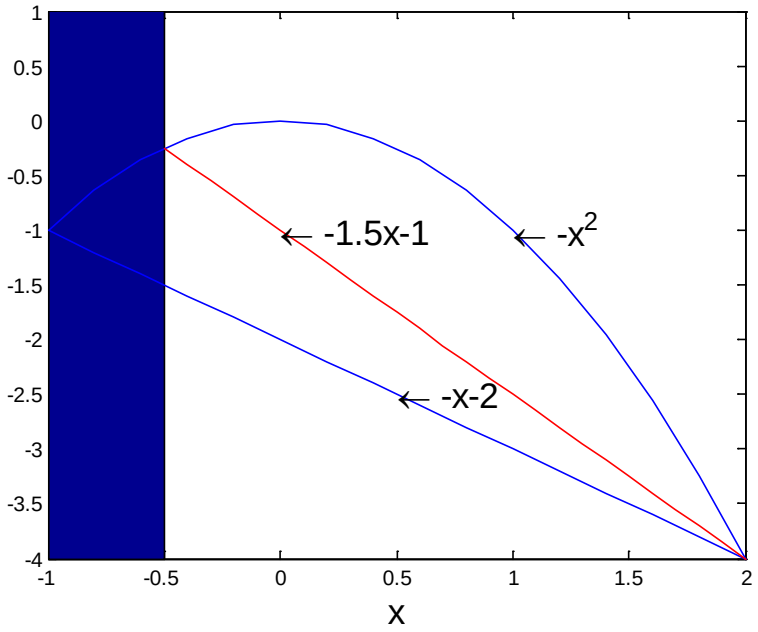


Figure 1.3: convex relaxation of $-x^2$ for $x \in [-1 \quad 2]$ and $x \in [-0.5 \quad 2]$

for solving nonconvex problems exhibit exponential computational complexity ($\mathcal{NP}$-hard) [66] with respect to the full problem size, (ii) decomposition based optimization, on the other hand, is also $\mathcal{NP}$-hard (because nonconvex problems are often solved) but with respect to the size of the smaller nonconvex subproblems. This partially explains why monolith approaches for stochastic programs exhibit exponential growth in solution time as the number of scenario increases, while decomposition based methods tend to exhibit a linear (at most polynomial) time behavior with increase in number of scenarios. The behavior is observed in the case studies in chapter 2, 3, 4 and 5. The other good feature of decomposition based approaches is that decomposition subproblems can readily be solved in parallel to improve computational time of the overall problem. Therefore, applying parallel computing architectures to decomposition algorithms can greatly improve solution time. Consequently, decomposition

approaches, in conjunction with global optimization techniques on smaller subproblems can potentially improve efficiency for large-scale optimization problems.

The current rigorous decomposition methods for global optimization of nonconvex programs are the following:

1. Lagrangian based branch and cut algorithm by Karuppiah and Grossmann [78]. This approach combines Lagrangian decomposition, branch-and-bound search and cutting planes to solve mixed-integer nonconvex programming problem.

2. Nonconvex generalized Benders decomposition (NGBD) by Li and Barton [79]. This approach is an extension of GBD whereby convex relaxations (based on McCommick relaxations [70]) and canonical cuts [80] are combined with GBD to solve mixed-integer nonconvex programs.

3. A most recently developed decomposition approach by Kannan and Barton [81] combines nonconvex generalized Benders decomposition, Lagrangian decomposition and branch-and-bound to solve mixed-integer nonconvex programs.

The methods presented above either require explicit branch-and-bound search at the decomposition level e.g. (1) and (3), or require that the linking variables, i.e. x_1 in Problem (TSSP) are integer variables e.g. (2).

1.5 Objective and Contribution of Thesis

The objective of the thesis is to develop new rigorous decomposition based methods that can, (i) improve performance of BD/GBD, and (ii) do not require explicit branch-and-bound search at the decomposition level, for global optimization of mixed-integer linear and nonlinear nonconvex optimization problems.

The contributions in this thesis is two-fold. First, a cross decomposition (CD) framework, a blend of BD and LD, is developed to solve angular and dual-angular mixed-integer linear programs. The CD in this thesis is different from CD in the literature [82] [83] [84] [54] due to the following reasons (i) it generates extra upper bounding problems for the original problem, (ii) feasibility issues are addressed extensively and (iii) new heuristic to avoid solving unnecessary subproblems are developed. A variant of CD composed of multicolumn and multicut restricted and relaxed master problem respectively, called multicolumn-multicut (MCMC) cross decomposition is also developed. Moreso, a novel extension of CD to handle hybrid-angular structures called extended cross decomposition (ECD), is proposed. Case studies of a bioproduct and a chemical supply chain optimization problems are used to demonstrate the performance of the new decomposition strategies.

Secondly, a novel decomposition technique for mixed-integer nonconvex programs referred to as "joint decomposition" is developed. This method is different from current decomposition techniques in the literature because, (i) it does not require explicit branch-and-bound search at the decomposition level, (ii) it can guarantee rigorous global solutions for MINLPs. In addition, domain reduction schemes are applied for the first time to joint decomposition to improve solution efficiency. Case studies include a pooling problem and a natural gas network design and operation problem.

1.6 Organization of Thesis

The thesis is organized as follows. In chapter 2, a new cross decomposition method for two-stage stochastic mixed integer programming is presented.

Thereupon chapter 3 discusses multicolumn-multicut extensions of cross decomposition method for two-stage stochastic mixed integer programming.

The extended cross decomposition to solve two-stage stochastic programs with strong and weak linking constraints (with the strong linking constraints coming from conditional value-at-risk constraints, CVaR) is described in chapter 4.

Chapter 5 discusses a novel decomposition technique to solve two-stage stochastic nonlinear nonconvex optimization problems. This method is called ‘joint decomposition‘.

Chapter 6 draw conclusions and outlines future research directions.

Chapter 2

A New Cross Decomposition Method for Stochastic Mixed-Integer Linear Programming *

2.1 Introduction

Mixed-integer linear programming (MILP) paradigm has been applied to a host of problems in process systems engineering, including but not limited to problems in supply chain optimization [85], oil field planning [86], gasoline blending and scheduling [87], expansion of chemical plants [32]. In such applications, there may be parameters in the model that are not known with certainty at the decision making stage. These parameters can be customer demands, material prices, yields of the plant, etc. One way of explicitly addressing the model uncertainty is to use the following scenario-based two-stage stochastic programming formulation:

*This chapter has been published in Ogbe E, Li X, A new cross decomposition method for stochastic mixed-integer linear programming, European Journal of Operational Research, 256 (2017), pp. 287-299. The equations, assumptions, propositions, theorems, symbols and notations defined in this chapter are self-contained.

$$
\begin{aligned}
\min_{\substack{x_0 \\ x_1,\ldots,x_s}} \ & \sum_{\omega=1}^{s} [c_{0,\omega}^T x_0 + c_\omega^T x_\omega] \\
\text{s.t.} \ & A_{0,\omega} x_0 + A_\omega x_\omega \le b_{0,\omega}, \quad \forall \omega \in \{1,...,s\}, \\
& x_\omega \in X_\omega, \quad \forall \omega \in \{1,...,s\}, \\
& x_0 \in X_0,
\end{aligned}
\tag{SP}
$$

Here x_0 includes the first-stage variables, which include n_i integer variables and n_c continuous variables. Set $X_0 = \{x_0 \in \mathbb{Z}^{n_i} \times \mathbb{R}^{n_c} : B_0 x_0 \le d_0\}$, x_ω includes the second-stage variables for scenario ω and set $X_\omega = \{x_\omega \in \mathbb{R}^{n_x} : B_\omega x_\omega \le d_\omega\}$. Parameter $b_{0,\omega} \in \mathbb{R}^m$, and other parameters in problem (SP) have conformable dimensions. Note that the second-stage variables in (SP) are all continuous.

Usually a large number of scenarios are needed to fully capture the characteristics of uncertainty; as a result, Problem (SP) becomes a large-scale MILP, for which solving the monolith (full model) using commercial solvers (such as CPLEX) may fail to return a solution or return a solution quickly enough. However, Problem (SP) exhibits a nice block structure that can be exploited for efficient solution. Figure 5.1 illustrates this structure. The structure of the first group of constraints in Problem (SP) is shown by part (1) of the figure, and the structure of the last two groups is shown by part (2).

There exist two classical ideas to exploit the structure of Problem (SP). One is that, if the constraints in part (1) are dualized, Problem (SP) can then be decomposed over the scenarios and therefore it becomes a lot easier to solve. With this idea, the first group of constraints in Problem (SP) are viewed as *linking constraints*.

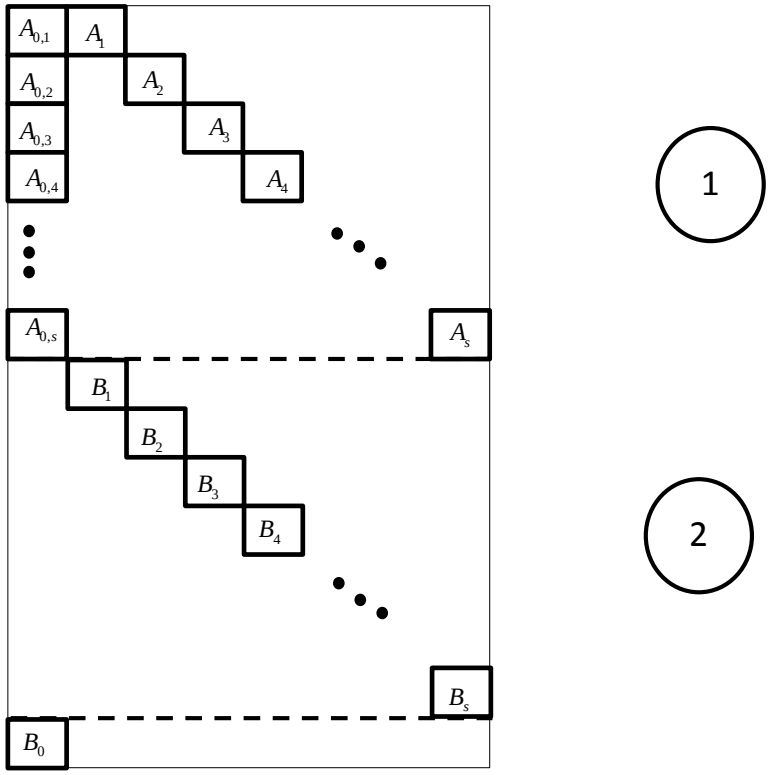


Figure 2.1: The block structure of Problem (SP)

Dantzig-Wolfe decomposition (DWD) [37] or column generation [49] [51] is one classical approach following this idea. In this approach, constraints in part (1) are dualized to form a pricing problem. The optimal solution of the pricing problem not only leads to a lower bound for Problem (SP), but also provides a point, or called a *column*,

which is used to construct a restriction of set $\prod_{\omega=1}^{s} X_\omega$. With this restriction, a (restricted) master problem is solved, and the solution gives an upper bound for Problem (SP) and new dual multipliers for constructing the next pricing problem. Another approach following the same idea is Lagrangian decomposition/relaxation [53] [88] [89], where a lower bounding Lagrangian subproblem is solved at each iteration and the Lagrange multipliers for the subproblems can be generated by solving the non-smooth Lagrangian dual problem or by some heuristics. Since this idea relies on the fact that the dualization of the constraints in part (1) is not subject to a duality gap, these methods can finitely find an optimal solution of Problem (SP). If integer variables are present, the methods have to be used in a branch-and-bound framework to ensure finite termination with an optimal solution [90] [89] [91], such as the branch-and-price method [92] [50] [90] [93] [94] [95] [96] [97] and the branch-price-cut method [98].

The other idea to exploit the structure is based on the fact that, if the value of x_0 is fixed, then the block column $A_{0,1}, \cdots, A_{0,s}$ in part (1) no longer links the different scenarios and therefore Problem (SP) becomes decomposable over the scenarios. With this idea, the first-stage variables are viewed as *linking variables*. Benders Decomposition (BD) [34] or L-shaped method [43] is a classical approach following this idea. In this approach, through the principle of projection and dualization, Problem (SP) is equivalently reformulated into a master problem, which includes a large but finite number of constraints, called *cuts*. A relaxation of master problem that includes a finite subset of the cuts can be solved to yield a lower bound for Problem (SP) as well as a value for x_0. Fixing x_0 to this value yields a decomposable upper bounding problem for Problem (SP). One important advantage of BD over DWD or Lagrangian decomposition is that, finite termination with an optimal solution is guaranteed, no

matter whether x_0 includes integer variables. However, when x_0 is fixed for some problems, the primal problem can have degenerate solutions[48] [83] [99], resulting in redundant Bender cuts and slow convergence of the algorithm [100] [101].

It is natural to consider synergizing the two aforementioned ideas for a unified decomposition framework that not only guarantees convergence for mixed-integer x_0, but also leads to improved convergence rate. Van Roy proposed a cross decomposition method, which solves BD and Lagrangian relaxation subproblems iteratively for MILPs with decomposable structures [82]. The computational advantage of the method was demonstrated through application to capacitated facility location problems [83]. Further discussions on the method, including generalization for convex nonlinear programs was done by Holmberg [102] [84]. One important assumption of this cross decomposition method is that, the (restricted or relaxed) master problems from BD and Lagrangian relaxation are difficult to solve and should be avoided as much as possible. However, this is usually not the case for Problem (SP). Therefore, Mitra, Garcia-Herreros and Grossmann recently proposed a different cross decomposition method [103] [54], which solves subproblems from BD and Lagrangian decomposition equally frequently. They showed that their cross decomposition method was significantly faster than BD and the monolith approach for a two-stage stochastic programming formulation of a resilient supply chain with risk of facility disruption [104] [105]. Both Van Roy and Mitra et al. assumed that all the subproblems solved are feasible.

In this chapter, we propose a new cross decomposition method which has two major differences from the cross decomposition methods in the literature. First, we combine BD and DWD instead of BD and Lagrangian decomposition in the method.

Second, we solve the subproblems in a different order. In addition, we include in the method a mechanism so that the algorithm will not be stuck with infeasible subproblems. In order to simplify our discussion, we rewrite Problem (SP) into the following form:

$$\begin{aligned} \min_{x_0,x} \ & c_0^T x_0 + c^T x \\ \text{s.t.} \ & A_0 x_0 + A x \leq b_0, \\ & x \in X, \\ & x_0 \in X_0, \end{aligned} \tag{P}$$

where $x = (x_1, \cdots, x_s)$, $X = \prod_{\omega}^{s} X_\omega$, $c_0 = (c_{0,1}, \cdots, c_{0,s})$, $c = (c_1, \cdots, c_s)$, $b_0 = (b_{0,1}, \cdots, b_{0,s})$, $A_0 = (A_{0,1}, \cdots, A_{0,s})$, $A = diag(A_1, \cdots, A_s)$. Remember that due to the problem structure shown in Figure 2.1, when x_0 is fixed or/and the first group of constraints are dualized, Problem (P) becomes much easier. Since for most real-world applications, the values of decision variables are naturally bounded, we make the following assumption.

Assumption 2.1. *X_0 and X are non-empty and compact.*

In fact, this assumption is not vital for the proposed method, but with it the discussion is more convenient.

The remaining part of the article is organized as follows. In section 2.2, we discuss classical decomposition methods and their relationships. Then, in section 2.3, we present the new cross decomposition method, give the subproblems and discuss the properties of the subproblems. In section 2.4, we give further discussions on the new method, including warm starting the algorithm with a Phase I procedure

and adaptively alternate between DWD and BD iterations. Two case studies; a bio-product supply chain optimization problem and an industrial chemical supply chain problem, are presented to demonstrate the computational advantage of the new cross decomposition method in section 2.5. The article ends with conclusions in section 2.5.3.

2.2 Classical Decomposition Methods

In this section we discuss three classical decomposition methods, which are BD, DWD, and Lagrangian decomposition. We review some theoretical results that are important for understanding how and why our new cross decomposition method works. The results either have been proved in literature or are easy to prove. Proofs of all propositions in the chapter are provided in Appendix A.

2.2.1 Benders decomposition

We first explain BD from the Lagrangian duality perspective, such as in [35], rather than from the linear programming (LP) duality perspective. There are two benefits in doing this here. One is the convenience for associating BD to DWD and Lagrangian decomposition, the other is the convenience for future extension of the cross decomposition to convex nonlinear problems. In BD, an alternative problem that is equivalent to Problem (P) is considered. We call this problem Benders Master Problem (BMP)

in this chapter, and it is given below:

$$\begin{aligned}
\min_{x_0,\eta} \quad & \eta \\
\text{s.t.} \quad & \eta \geq \inf_{x\in X} \left(c^T x + \lambda^T A x\right) + (c_0^T + \lambda^T A_0)x_0 - \lambda^T b_0, \quad \forall \lambda \in \Lambda_{opt}, \\
& 0 \geq \inf_{x\in X} \lambda^T A x + \lambda^T (A_0 x_0 - b_0), \quad \forall \lambda \in \Lambda_{feas}, \\
& x_0 \in X_0.
\end{aligned} \tag{BMP}$$

This problem can be equivalently transformed from Problem (P) via projection and dualization. The reformulation procedure is explained in [35]. The first group of constraints in Problem (BMP) are called *optimality cuts*, and the second group of constraints are called *feasibility cuts*. λ represents Lagrange multipliers from dualizing the linking constraints, and the different optimality or feasibility cuts are differentiated by the different multipliers involved. The multipliers in the optimality cuts are optimal dual solutions of the following Benders Primal Problem (BPP), which is constructed at Benders iteration k by fixing x_0 to constant x_0^k:

$$\begin{aligned}
\min_{x} \quad & c_0^T x_0^k + c^T x \\
\text{s.t.} \quad & A_0 x_0^k + A x \leq b_0, \\
& x \in X.
\end{aligned} \tag{BPP^k}$$

If Problem (BPP^k) is infeasible for $x_0 = x_0^k$, then the following Benders Feasibility Problem (BFP^k) is solved and its optimal dual solution is a multiplier for a feasibility

cut:

$$\begin{aligned} \min_{x,z} \quad & ||z|| \\ \text{s.t.} \quad & A_0 x_0^k + Ax \leq b_0 + z, \\ & x \in X, \\ & z \geq 0, \end{aligned} \tag{BFP^k}$$

where $||\cdot||$ can be any norm function. The 1-norm is used in the case studies of this chapter. Note that Problem (BFP^k) is always feasible and has a finite objective value if X is nonempty.

Proposition 2.1. *Problem* (BMP) *is equivalent to Problem* (P) *if* Λ_{opt} *is* $\{\lambda \in \mathbb{R}^m : \lambda \geq 0\}$ *or the set of all extreme dual multipliers of Problem* (BPP^k)*, and* Λ_{feas} *is* $\{\lambda \in \mathbb{R}^m : \lambda \geq 0\}$ *or the set of all extreme dual multipliers of Problem* (BFP^k)*. Here extreme dual multipliers of a LP problem refer to extreme points of the feasible set of the LP dual of the problem.*

In BD, a relaxation of Problem (BMP) that includes part of the optimality and feasibility cuts, instead of Problem (BMP) itself, is solved at each iteration. This problem can be called Benders Relaxed Master Problem (BRMP). We will further discuss BRMP in the next section.

2.2.2 Dantzig-Wolfe decomposition

DWD considers a different master problem, which is constructed by representing the bounded polyhedral set X in Problem (P) with a convex combination of all its extreme points. We call this problem Dantzig-Wolfe Master Problem (DWMP) and give it

below:

$$\begin{aligned} \min_{x_0,x} \quad & c_0^T x_0 + c^T x \\ \text{s.t.} \quad & A_0 x_0 + A x \leq b_0, \\ & x_0 \in X_0, \\ & x \in \left\{ x : x = \sum_{i=1}^{n_F} \theta^i x^i,\ \sum_{i=1}^{n_F} \theta^i = 1,\ \theta^i \geq 0\ (i = 1, \cdots, n_F) \right\}. \end{aligned} \tag{DWMP}$$

The next proposition shows that, for Problem (DWMP) being equivalent to Problem (P), the points x^i in Problem (DWMP) do not have to all be the extreme points.

Proposition 2.2. *Problem* (DWMP) *is equivalent to Problem* (P) *if* $E(X) \subset \{x^1, \cdots, x^{n_F}\} \subset X$*, where* $E(X)$ *denotes the set of all extreme points of* X*.*

In DWD, a restriction of Problem (DWMP) that includes part of the extreme points of X, called Dantzig-Wolfe Restricted Master Problem (DWRMP), is solved at each iteration, yielding an upper bound for Problem (P). The extreme points are selected from the solutions of the following Dantzig-Wolfe Pricing Problem (DWPP):

$$\begin{aligned} \min_{x} \quad & (c^T + (\lambda^k)^T A) x \\ \text{s.t.} \quad & x \in X, \end{aligned} \tag{DWPP^k}$$

where the λ^k denotes Lagrange multipliers of the linking constraints for the previously solved DWRMP. We will further discuss on DWRMP in the next section.

2.2.3 Lagrangian decomposition

Lagrangian decomposition considers the following dual problem of (P):

$$\max_{\lambda \geq 0} \min_{x_0 \in X_0, x \in X} \quad c_0^T x_0 + c^T x + \lambda^T (A_0 x_0 + A x - b_0) \tag{LD}$$

Note that Problem (LD) is equivalent to Problem (P) only when there is no duality gap. However, this is generally not the case for MILPs. In iteration k of Lagrangian decomposition, the following relaxation of Problem (P), called Lagrangian subproblem, is solved:

$$\begin{aligned} \min_{x_0, x} \quad & c_0^T x_0 + c^T x + (\lambda^k)^T (A_0 x_0 + A x - b_0) \\ \text{s.t.} \quad & x_0 \in X_0, \\ & x \in X. \end{aligned} \tag{LS^k}$$

It is not trivial to generate λ^k for (LS^k) at each iteration. Several approaches have been used in the literature for multiplier generation, including solving the nonsmooth Lagrangian dual problem via some nonsmooth optimization methods such as subgradient methods [106] [38] [107], and solving restricted Lagrangian dual problems [82] [103]. The restricted Lagrangian dual master problem is given as:

$$\begin{aligned} \max_{\eta_0, \lambda} \quad & \eta_0 \\ \text{s.t.} \quad & \eta_0 \leq c_0^T x_0^i + c^T x^i + (\lambda)^T (A_0 x_0^i + A x^i - b_0), i \in I^k \\ & \lambda \geq 0, \end{aligned} \tag{RLD^k}$$

where x_0^i are extreme points of X_0 that are generated from previous iterations.

Obviously, Problem (LS^k) can be decomposed into a subproblem including x_0 only and a subproblem including x only. The latter one is actually Problem (DWPP^k). Thus we can view DWD as a variant of Lagrangian decomposition, which, in addition to what a regular Lagrangian decomposition approach does, also provides a systematic mechanism to generate Lagrange multipliers and problem upper bounds.

The next proposition states that, for a group of given Lagrange multipliers, a Benders optimality cut can be constructed either from the solution of a BD subproblem or from a DWD subproblem.

Proposition 2.3. *Let λ^k represent Lagrange multipliers for the linking constraints in Problem* (BPP^k) *and Problem* (DWPP^k)*, then*

$$\begin{aligned}
&\inf_{x\in X}\left(c^T x+\left(\lambda^k\right)^T Ax\right)+\left(c_0^T+(\lambda^k)^T A_0\right)x_0-(\lambda^k)^T b_0\\
=&obj_{BPP^k}+\left(c_0^T+(\lambda^k)^T A_0\right)\left(x_0-x_0^k\right)\\
=&obj_{DWPP^k}+\left(c_0^T+(\lambda^k)^T A_0\right)x_0-(\lambda^k)^T b_0,
\end{aligned}$$

where obj_{BPP^k}, obj_{DWPP^k} denote the optimal objective values of Problems (BPP^k), (DWPP^k)*, respectively, and x^k denotes an optimal solution of Problem* (BPP^k) *associated with λ^k.*

The next proposition states that a Benders feasibility cut can be constructed from the solution of Problem (BFP^k).

Proposition 2.4. *Let λ^k represent Lagrange multipliers for the linking constraints*

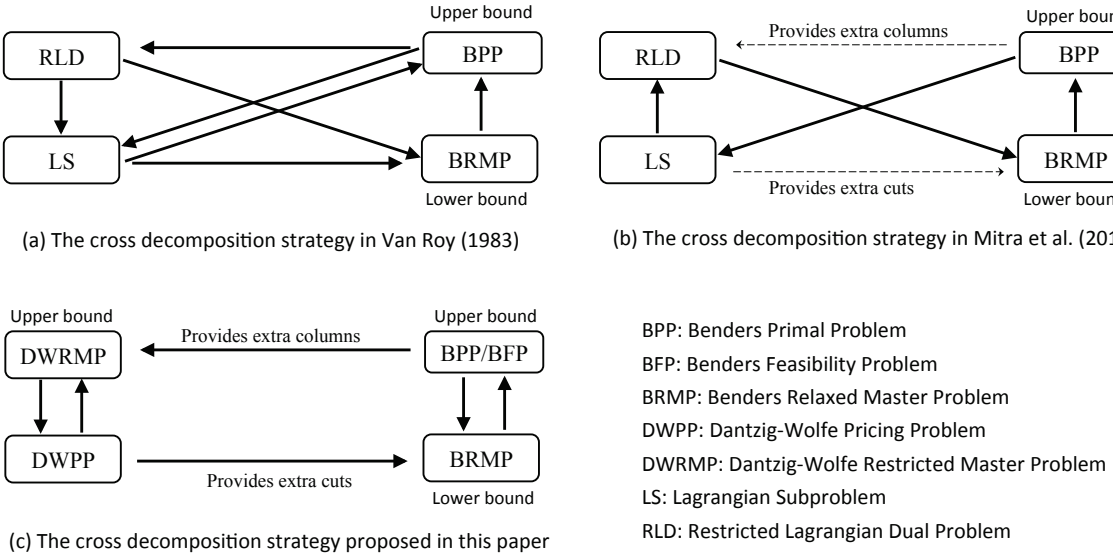


Figure 2.2: Three cross decomposition strategies

in Problem (BFP^k)*, then*

$$\begin{aligned}&\inf_{x\in X}(\lambda^k)^T Ax + (\lambda^k)^T(A_0x_0 - b_0)\\ =&obj_{BFP^k} + (\lambda^k)^T A_0\left(x_0 - x_0^k\right),\end{aligned}$$

where obj_{BFP^k} *denotes the optimal objective values of Problems* (BFP^k) *and* x^k *denotes an optimal solution of Problem* (BFP^k)*.*

2.3 The New Cross Decomposition Method

2.3.1 Different Cross Decomposition Strategies

Figure 2.2 illustrates the diagrams of three cross decomposition strategies proposed by Van Roy [82], Mitra et al. [54], and this chapter. Van Roy's cross decomposition

includes BD subproblems BPP and BRMP, and Lagrangian decomposition subproblems RLD and LS. Here RLD stands for restricted Lagrangian dual problem, which results from restricting set X in Problem (LD) to the convex hull of a number of extreme points of X. Since this cross decomposition method is designed for applications in which the master problems RLD and BRMP are much more difficult than problems BPP and LS, so it mostly solves BPP and LS iteratively and only solves RLD and BRMP when necessary.

The cross decomposition proposed by Mitra et al. includes the same subproblems, but the order in which the subproblems are solved is different. As the method is designed for stochastic MILPs in which the master problems RLD and BRMP are usually easier than subproblems LS and BPP, it does not avoid solving RLD and BRMP as much as possible. Instead, it solves each of the four subproblems equally frequently. In addition, solutions of BPP are used to yield extra columns to enhance RLD and the solutions of LS are used to yield extra cuts to enhance BRMP. Although the extra cuts and columns make the master problems larger and more time consuming to solve, they also tighten the relaxed master problems and reduce the number of iterations needed for convergence.

The cross decomposition method proposed in this chapter was initially motivated by the complementary features of DWD and BD. So this method includes DWD iterations that solve DWRMP and DWPP and BD iterations that solve BPP/BFP and BRMP. The method alternates between several DWD iterations and several BD iterations. Just like in the cross decomposition proposed by Mitra et al., the solutions of BPP and DWPP are used to generate extra columns and cuts to enhance master problems DWRMP and BRMP. Compared to the other two cross decomposition

methods, we believe that there are two major advantages of the method proposed in this chapter:

1. The DWD restricted master problem DWRMP provides a rigorous upper bound for Problem (P), while the restricted Lagrangian dual RLD does not. Actually, according to Van Roy (1983), RLD is a dual of DWRMP. On the other hand, DWPP is similar to LS and either one can provide a cut to BRMP (according to the discussion in the previous section). Therefore, using DWD instead of Lagrangian decomposition in the cross decomposition framework is likely to achieve a better convergence rate.

2. Feasibility issues are addressed systematically. When BPP is infeasible, a Benders feasibility problem BFP is solved to allow the algorithm to proceed. In addition, a Phase I procedure is introduced to avoid infeasible DWRMP.

In the next two subsections, we will give details for the subproblems solved in the proposed new cross decomposition method, and the sketch of the basic algorithm. In section 2.4, we will propose a Phase I procedure to avoid solving infeasible DWRMP and also discuss how to adaptively alternate between DWD and BD iterations.

2.3.2 Subproblems in the New Cross Decomposition Method

In the new cross decomposition method, we call either a BD iteration (i.e., the solution of one BPP/BFP and one BRMP) or a DWD iteration (i.e. the solution of one DWRMP and one DWPP) a CD iteration. At the kth CD iteration, subproblem BPP/BFP or DWPP to be solved is same to Problem (BPP^k)/(BFP^k) or (DWPP^k) given in section 2.2. The BRMP problem solved in the kth CD iteration can be

formulated as follows:

$$
\begin{aligned}
\min_{x_0,\eta} \quad & \eta \\
s.t. \quad & \eta \geq obj_{BPP^i} + \left(c_0^T + (\lambda^i)^T A_0\right)\left(x_0 - x_0^i\right), \quad \forall i \in T_{opt}^k, \\
& 0 \geq obj_{BFP^i} + \left(\lambda^i\right)^T A_0 (x_0 - x_0^i), \quad \forall i \in T_{feas}^k, \\
& \eta \geq obj_{DWPP^i} + \left(c_0^T + (\lambda^i)^T A_0\right) x_0 - (\lambda^i)^T b_0, \quad \forall i \in U_{opt}^k, \\
& x_0 \in X_0,
\end{aligned}
\tag{BRMP$_r^k$}
$$

where T_{opt}^k includes the indices of all previous iterations in which Problem (BPP^k) is solved and feasible, T_{feas}^k includes the indices of all previous iterations in which Problem (BFP^k) is solved, U_{opt}^k includes the indices of all previous iterations in which Problem (DWPP^k) is solved.

Proposition 2.5. *Problem* (BRMP$_r^k$) *is a valid lower bounding problem for Problem* (P)

The DWRMP problem solved in the kth CD iteration can be formulated as follows:

$$
\begin{aligned}
\min_{x_0,\theta^i} \quad & c_0^T x_0 + c^T \left(\sum_{i \in I^k} \theta^i x^i\right) \\
s.t. \quad & A_0 x_0 + A \left(\sum_{i \in I^k} \theta^i x^i\right) \leq b_0, \\
& \sum_{i \in I^k} \theta^i = 1, \\
& \theta^i \geq 0, \quad \forall i \in I^k, \\
& x_0 \in X_0,
\end{aligned}
\tag{DWRMP^k}
$$

where the index set $I^k \subset \left(T_{opt}^k \cup T_{feas}^k \cup U_{opt}^k\right)$, in other words, the columns considered

in Problem (DWRMP^k) come from the solutions of some previously solved BPP/BFP and DWPP subproblems. Note that here we assume that I^k is nonempty and it is such that Problem (DWRMP^k) is feasible. In the next section, we will discuss how to ensure this through a Phase I procedure.

Proposition 2.6. *Problem* (DWRMP^k) *is a valid upper bounding problem for Problem* (P).

2.3.3 Sketch of the New Cross Decomposition Algorithm

A sketch of the new cross decomposition algorithm is given in Table 2.1. With the following assumption, the finiteness of the algorithm can be easily proved.

Assumption 2.2. *The primal and dual optimal solutions of an LP returned by an optimization solver are extreme points and extreme dual multipliers.*

This assumption is needed to prevent the generation of an infinite number of Lagrange multipliers that lead to the same feasible solution of Problem (P). This is a mild assumption as most commercial solvers (such as CPLEX) return extreme optimal primal and dual solutions for LPs using 'primal simplex' and 'dual simplex' algorithm options respectively. If an 'interior point' algorithm is used, a 'cross over' from an 'interior point' solution to a 'basic feasible solution' (a default option in CPLEX) [108], ensures that extreme points solution is generated. With this assumption, Problem (BPP^k) or (BFP^k) can only yield a finite number of Lagrange multipliers.

Theorem 2.1. *If there are at most a finite number of DWD iterations between two BD iterations, then the cross decomposition algorithm described in Table 2.1 terminates in a finite number of steps with an optimal solution of Problem* (P) *or an indication that Problem* (P) *is infeasible.*

Proof. This proof is based on the finite convergence property of the BD method. It is well known that (e.g., [34], [30]), Problem (BPP^k) will not yield the same Lagrange multipliers (for constructing cuts in Problem (BRMP$_r^k$)) twice unless the Lagrange multipliers are the ones associated with an optimal solution of Problem (P). And upon generation of optimal Lagrange multipliers for a second time, the upper bound from Problem (BPP^k) and the lower bound from Problem (BRMP$_r^k$) will coincide, leading to termination with an optimal solution of Problem (P). This procedure is finite as (a) only a finite number of Lagrange multipliers can be generated by Problem (BPP^k) and (b) there are at most a finite number of DWD iterations between two BD iterations.

If Problem (P) is infeasible, then the master problem (BMP) is infeasible. Note that according to Proposition 2.1, we can consider Problem (BMP) to involve a finite number of extreme dual multipliers. So Problem (BRMP$_r^k$) needs only a finite number of steps to grow into Problem (BMP), and therefore after a finite number of steps, it must be infeasible which indicates the infeasibility of Problem (P). □

2.4 Further Discussions

2.4.1 Phase I Procedure

Problem (DWRMP^k) is feasible only when at least one convex combination of the columns $\{x^i\}_{i\in I^k}$ is feasible for Problem (P). Here we introduce a Phase I procedure as a systematic way to generate a group of columns that enable the feasibility of Problem (DWRMP^k) or indicate the infeasibility of Problem (P). Similar to the Phase I procedure in simplex algorithm, this proposed Phase I procedure is to solve

Table 2.1: Sketch of the New Cross Decomposition Algorithm

Initialization:

(a) Give set I^1 that includes indices of a number of points in set X such that Problem (DWRMP^k) is feasible. Give termination tolerance ϵ.

(b) Let index sets $U^1_{opt} = T^1_{opt} = T^1_{feas} = \emptyset$. Iteration counter $k = 1$, upper bound $UBD = +\infty$, lower bound $LBD = -\infty$.

Step 1 (DWD iterations):

Execute the DWD iteration described below several times:

(1.a) Solve Problem (DWRMP^k). Let x^k_0, $\{\theta^{i,k}\}_{i\in I^k}$ be the optimal solution obtained, and λ^k be Lagrange multipliers for the linking constraints. If $obj_{DWRMP^k} < UBD$, update $UBD = obj_{DWRMP^k}$ and the incumbent solution $(x^*_0, x^*) = (x^k_0, \sum_{i\in I^k} \theta^{i,k} x^i)$. If $UBD \leq LBD + \epsilon$, terminate and the incumbent solution (x^*_0, x^*) is an optimal solution for Problem (P).

(1.b) Solve Problem (DWPP^k), let x^k be the optimal solution obtained.

(1.c) Generate I^{k+1}, U^{k+1}_{opt} for adding columns and cuts to the master problems. Update $k = k + 1$.

Step 2 (BD iterations):

Execute the BD iteration described below several times, and then go to step 1.

(2.a) Solve Problem (BRMP^k_r). Let (η^k, x^k_0) be the optimal solution obtained. If $\eta^k > LBD$, update $LBD = \eta^k$. If $UBD \leq LBD + \epsilon$, terminate and the incumbent solution (x^*_0, x^*) is an optimal solution for Problem (P).

(2.b) Solve Problem (BPP^k). If Problem (BPP^k) is feasible and $obj_{BPP^k} < UBD$, update $UBD = obj_{BPP^k}$ and the incumbent solution $(x^*_0, x^*) = (x^k_0, x^k)$. If Problem (BPP^k) is infeasible, solve Problem (BFP^k). No matter which problem is solved, let x^k, λ^k be an optimal solution and the related Lagrange multipliers for the linking constraints.

(2.c) Generate I^{k+1}, T^{k+1}_{opt}, T^{k+1}_{feas} for adding columns and cuts to the master problems. If Problem (BPP^k) is feasible, $T^{k+1}_{opt} = T^k_{opt} \cup \{k\}$, $T^{k+1}_{feas} = T^k_{feas}$; otherwise, $T^{k+1}_{feas} = T^{k+1}_{feas} \cup \{k\}$, $T^{k+1}_{opt} = T^{k+1}_{opt}$. Update $k = k + 1$.

the following Feasibility Problem instead of Problem (P) itself:

$$\begin{aligned} \min_{x_0,x,z} \quad & ||z|| \\ s.t. \quad & A_0x_0 + Ax \leq b_0 + z, \\ & x_0 \in X_0, \\ & x \in X, \\ & z \geq 0, \end{aligned} \tag{FP}$$

where $||\cdot||$ denotes any norm function, and the 1-norm is used for the case studies in this chapter. In the Phase I procedure, Problem (FP) is solved via the proposed cross decomposition method, and this procedure is illustrated in Figure 2.3. In this

procedure, each DWD iteration solves a restricted DWD master problem for Problem (FP), DWFRMP, and a DWD pricing problem for Problem (FP), DWFP. Each BD iteration solves a primal for Problem (FP), BFP, and a BD relaxed master problem for Problem (FP). BFP at the kth CD iteration is same to Problem (BFP^k), so we are to give the other three subproblems.

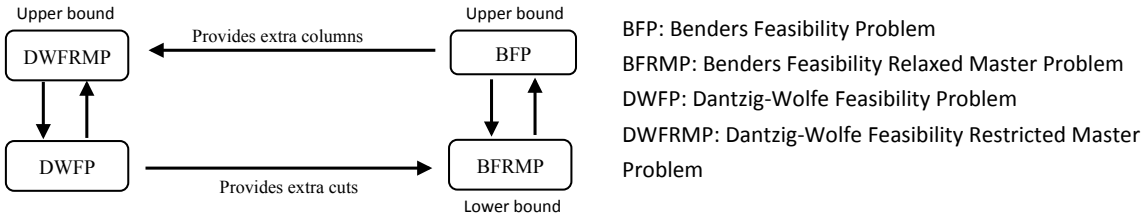


Figure 2.3: Diagram of the Phase I procedure

Problem BFRMP solved at the kth CD iteration is:

$$\begin{aligned} \min_{x_0,\eta} \quad & \eta \\ s.t. \quad & \eta \geq obj_{BFP^i} + \left(\lambda^i\right)^T A_0(x_0 - x_0^i), \quad \forall i \in T_{feas}^k, \\ & \eta \geq obj_{DWFP^i} + (\lambda^i)^T(A_0x_0 - b_0), \quad \forall i \in U_{feas}^k, \\ & x_0 \in X_0, \end{aligned} \tag{BFRMP^k}$$

where T_{feas}^k includes the indices for all previous BD iterations and U_{feas}^k includes the indices for all previous DWD iterations. λ^i denotes Lagrange multipliers for the linking constraints for Problem (BFP^i) or for Problem (DWFP^i), and x_0^i denotes the fixed x_0 value for Problem (BFP^i). As a result of Proposition 2.5, Problem (BFRMP^k) is a valid lower bounding problem for Problem (FP).

Problem DWFRMP solved at the kth CD iteration is:

$$\begin{aligned} \min_{x_0,z,\theta^i} \quad & ||z|| \\ s.t. \quad & A_0x_0 + A\left(\sum_{i\in I^k}\theta^i x^i\right) \leq b_0 + z, \\ & \sum_{i\in I^k}\theta^i = 1, \\ & \theta^i \geq 0, \quad \forall i \in I^k, \\ & x_0 \in X_0, \\ & z \geq 0, \end{aligned} \tag{DWFRMP^k}$$

where $I^k = T^k_{feas} \cup U^k_{feas}$. As a results of Proposition 2.6, Problem (DWFRMP^k) is a valid upper bounding problem for Problem (FP). Note that the Phase I procedure starts with solving Problem (DWFRMP^k) (for k=1), in which the index I^1 has to include at least one column. In order to generate an initial column, called x^0, for I^1, the following initial pricing problem is solved:

$$\begin{aligned} \min_{x} \quad & c^T x \\ s.t. \quad & x \in X. \end{aligned} \tag{IPP}$$

Note that if Problem (IPP) is infeasible, then set X is empty and therefore Problem (P) is infeasible.

Problem DWFP solved at the kth iteration is:

$$\begin{aligned} \min_{x} \quad & (\lambda^k)^T Ax \\ s.t. \quad & x \in X, \end{aligned} \tag{DWFP^k}$$

where λ^k includes Lagrange multipliers for the linking constraints in Problem (DWFRMP^k).

The following theorem results from applying Theorem 2.1 to Problem (FP).

Theorem 2.2. *The Phase I procedure illustrated in Figure 2.3 terminates finitely with an optimal solution of Problem* (FP) *or an indication that Problem* (FP) *is infeasible. If the optimal objective value of Problem* (FP) *is greater than 0, then Problem* (P) *is infeasible; otherwise, Problem* (P) *is feasible and Problem* (DWRMP^k) *is feasible with the columns generated in the Phase I procedure.*

Note that the optimal value of Problem (FP) cannot be negative, so the Phase I procedure can terminate when the current upper bound is 0 (no matter what the current lower bound is). In this case, the 0 upper bound comes from either the optimal value of Problem (BFP^k) or the optimal value of Problem (DWFRMP^k), and therefore the solution of one of the problems provides a feasible column, with which Problem (DWRMP^k) is feasible.

After completing the Phase I procedure, the algorithm starts the Phase II procedure that solves Problem (P) using the cross decomposition strategy. In the Phase II procedure, the iteration counter k continues to increase from its value at the end of the Phase I procedure, and the index sets I^k, T^k_{feas} also grows from the ones at the end of the Phase I procedure. The index set U^k_{feas} remains the same as the one at the end of the Phase I procedure because Problem (DWFP^k) is never solved in the Phase II procedure.

With the results from the Phase I procedure, the BD relaxed master problem in

Phase II can be updated as:

$$
\begin{aligned}
\min_{x_0,\eta} \quad & \eta \\
s.t. \quad & \eta \geq obj_{BPP^i} + \left(c_0^T + (\lambda^i)^T A_0\right)\left(x_0 - x_0^i\right), \quad \forall i \in T_{opt}^k, \\
& 0 \geq obj_{BFP^i} + \left(\lambda^i\right)^T A_0(x_0 - x_0^i), \quad \forall i \in T_{feas}^k, \\
& \eta \geq obj_{DWPP^i} + \left(c_0^T + (\lambda^i)^T A_0\right) x_0 - (\lambda^i)^T b_0, \quad \forall i \in U_{opt}^k, \\
& 0 \geq obj_{DWFP^i} + (\lambda^i)^T (A_0 x_0 - b_0), \quad \forall i \in U_{feas}^k, \\
& x_0 \in X_0.
\end{aligned}
\tag{BRMP^k}
$$

Note that the cuts in this problem come from the subproblems solved in both the Phase I procedure and the Phase II procedure. The following proposition states the validity of the cuts.

Proposition 2.7. *Problem* (BRMP^k) *is a valid lower bounding problem for Problem* (P).

Now we give a cross decomposition algorithm that combines the Phase I and the Phase II procedures to systematically solve Problem (P). In either phase, the algorithm alternates between one DWD iteration and one BD iteration. The solutions of subproblems in the DWD iterations are all used to construct extra cuts to enhance Problem (BFRMP^k)/(BRMP^k), while the solutions of subproblems in the BD iterations are all used to construct extra columns to enhance Problem (DWFRMP^k)/(DWRMP^k). The details of the algorithm is given in Table 2.2. According to Theorems 2.1 and 2.2, the algorithm has finite convergence property.

Corollary 2.1. *Cross Decomposition Algorithm 1 given in Table 2.2 terminates in a finite number of steps with an optimal solution for Problem* (P) *or an indication that*

Problem (P) *is infeasible.*

Table 2.2: Cross Decomposition Algorithm 1

Initialization

(a) Select a point from set X (e.g., by solving Problem (IPP)). Let x^0 be the selected point and $I^1 = \{0\}$. If X is empty, then Problem (P) is infeasible.

(b) Give termination tolerance ϵ. Let index sets $U^1_{opt} = T^1_{opt} = T^1_{feas} = \emptyset$, iteration counter $k = 1$, bounds for Problem (FP) $UBDF = +\infty$, $LBDF = -\infty$, bounds for Problem (P) $UBD = +\infty$, $LBD = -\infty$.

Cross Decomposition Phase I

Step 1 (DWD iteration):

(1.a) Solve Problem (DWFRMP^k). Let λ^k be the obtained Lagrange multipliers for the linking constraints. If $obj_{DWFRMP^k} < UBDF$, update $UBDF = obj_{DWFRMP^k}$.

(1.b) Solve Problem (DWFP^k), let x^k be the optimal solution obtained.

(1.c) Update $I^{k+1} = I^k \cup \{k\}$, $U^{k+1}_{feas} = U^k_{feas} \cup \{k\}$, $U^{k+1}_{opt} = U^k_{opt}$, $k = k + 1$. If $UBDF \leq \epsilon$, end Phase I and go to Phase II.

Step 2 (BD iteration):

(2.a) Solve Problem (BFRMP^k), update $LBDF = obj_{BFRMP^k}$. If $LBDF > \epsilon$, terminate and Problem (P) is infeasible.

(2.b) Solve Problem (BFP^k), and let x^k, λ^k be the obtained optimal solution and the related Lagrange multipliers for the linking constraints. If $obj_{BFP^k} < UBDF$, update $UBDF = obj_{BFP^k}$

(2.c) Update $I^{k+1} = I^k \cup \{k\}$, $T^{k+1}_{feas} = T^k_{feas} \cup \{k\}$, $T^{k+1}_{opt} = T^k_{opt}$, $k = k + 1$. If $UBDF \leq \epsilon$, end Phase I and go to Phase II; otherwise, go to step (1.a).

Cross Decomposition Phase II

Step 1 (DWD iteration):

(1.a) Solve Problem (DWRMP^k). Let x^k_0, $\{\theta^{i,k}\}_{i \in I^k}$ be the optimal solution obtained, and λ^k be Lagrange multipliers for the linking constraints. If $obj_{DWRMP^k} < UBD$, update $UBD = obj_{DWRMP^k}$ and the incumbent solution $(x^*_0, x^*) = (x^k_0, \sum_{i \in I^k} \theta^{i,k} x^i)$. If $UBD \leq LBD + \epsilon$, terminate and (x^*_0, x^*) is an optimal solution for Problem (P).

(1.b) Solve Problem (DWPP^k), let x^k be the optimal solution obtained.

(1.c) Update $I^{k+1} = I^k \cup \{k\}$, $U^{k+1}_{opt} = U^k_{opt} \cup \{k\}$, $U^{k+1}_{feas} = U^k_{feas}$, $k = k + 1$.

Step 2 (BD iteration):

(2.a) Solve Problem (BRMP^k), update $LBD = obj_{BRMP^k}$. If $UBD \leq LBD + \epsilon$, terminate and (x^*_0, x^*) is an optimal solution for Problem (P).

(2.b) Solve Problem (BPP^k). If Problem (BPP^k) is feasible and $obj_{BPP^k} < UBD$, update $UBD = obj_{BPP^k}$ and the incumbent solution $(x^*_0, x^*) = (x^k_0, x^k)$. If Problem (BPP^k) is infeasible, solve Problem (BFP^k). No matter which problem is solved, let x^k, λ^k be an optimal solution and the related Lagrange multipliers for the linking constraints.

(2.c) If Problem (BPP^k) is feasible, $T^{k+1}_{opt} = T^k_{opt} \cup \{k\}$, $T^{k+1}_{feas} = T^k_{feas}$; otherwise, $T^{k+1}_{feas} = T^k_{feas} \cup \{k\}$, $T^{k+1}_{opt} = T^k_{opt}$. Update $I^{k+1} = I^k \cup \{k\}$, $k = k + 1$. Go to step (1.a).

2.4.2 Adaptive Alternation Between DWD and BD Iterations

Cross Decomposition Algorithm 1 shown in Table 2.2 alternates between one DWD iteration and one BD iteration. However, in some cases it may be better to perform several DWD iterations or BD iterations in a row. For example, if the solution of a DWD restricted master problem decreases the current upper bound significantly, then the DWD pricing problem is likely to generate a good column for another DWD iteration to further reduce the gap; in this case, the algorithm should go to another DWD iteration rather than going to a BD iteration. If the solution of a BD primal problem cannot decrease the current upper bound, the column from the solution is not likely to enhance the DWD restricted master problem for further decrease of the upper bound; in this case, the algorithm should go to another BD iteration rather than going to a DWD iteration.

As a result, we introduce a different cross decomposition algorithm, which adaptively determines the type of the next iteration according to the following rules:

1. After a DWD iteration, solve Problem (BFRMP^k) (for Phase I) or (BRMP^k) (for Phase II). If the decrease of the upper bound in the DWD iteration is more than the increase of the lower bound (resulting from the solution of (BFRMP^k)/(BRMP^k)), then the algorithm will go to another DWD iteration. Otherwise, the algorithm will go to a BD iteration.

2. After a BD iteration, solve Problem (DWFRMP^k) (for Phase I) or (DWRMP^k) (for Phase II). If the optimal value of the problem is better than the current upper bound, then the algorithm will go to a DWD iteration. Otherwise, the algorithm will go to another BD iteration.

The details of this different algorithm is given in Table 2.3.

Table 2.3: Cross Decomposition Algorithm 2

Initialization

(a) Select a point from set X (e.g., by solving Problem (IPP)). Let x^0 be the selected point and $I^1 = \{0\}$. If X is empty, then Problem (P) is infeasible.

(b) Set termination tolerance ϵ. Let index sets $U^1_{opt} = T^1_{opt} = T^1_{feas} = \emptyset$, iteration counter $k = 1$, bounds for Problem (FP) $UBDF = +\infty$, $LBDF = -\infty$, bounds for Problem (P) $UBD = +\infty$, $LBD = -\infty$.

Cross Decomposition Phase I

Step 1 (DWD iteration):

(1.a) Solve Problem (DWFRMP^k). if $obj_{DWFRMP^k} > UBDF - \epsilon$, $\Delta_{DW} = 0$ go to step (2.a); otherwise calculate $\Delta_{DW} = UBDF - obj_{DWFRMP^k}$, update $UBDF = obj_{DWFRMP^k}$, and let λ^k be the obtained Lagrange multipliers for the linking constraints.

(1.b) Solve Problem (DWFP^k), let x^k be the optimal solution obtained.

(1.c) Update $U^{k+1}_{feas} = U^k_{feas} \cup \{k\}$, $U^{k+1}_{opt} = U^k_{opt}$, $I^{k+1} = I^k \cup \{k\}$. If $UBDF \leq \epsilon$, $k = k + 1$, end Phase I and go to Phase II

Step 2 (BD iteration):

(2.a) Solve Problem (BFRMP^k). If $obj_{BFRMP^k} > \epsilon$, terminate and Problem (P) is infeasible. Calculate $\Delta_{BD} = obj_{BFRMP^k} - LBDF$, update $LBDF = obj_{BFRMP^k}$. If $\Delta_{BD} \geq \Delta_{DW}$, go to step (2.b); otherwise, go to step (1.a).

(2.b) Solve Problem (BFP^k). Let x^k, λ^k be an optimal solution and the related Lagrange multipliers for the linking constraints.

(2.c) Update $T^{k+1}_{feas} = T^k_{feas} \cup \{k\}$, $T^{k+1}_{opt} = T^k_{opt}$. If $obj_{BFP^k} < UBDF - \epsilon$, $I^{k+1} = I^k \cup \{k\}$; otherwise, $I^{k+1} = I^k$. If $\min\{UBDF, obj_{BFP^k}\} \leq \epsilon$, $k = k + 1$, end Phase I and go to Phase II.

(2.d) If $obj_{BFP^k} < UBDF$, $UBDF = obj_{BFP^k}$; set $k = k + 1$ and go to step (1.a).

Cross Decomposition Phase II

Step 1 (DWD iteration):

(1.a) Solve Problem (DWRMP^k). Let x^k_0, $\{\theta^{i,k}\}_{i \in I^k}$ be the optimal solution obtained, and λ^k be the related Lagrange multipliers for the linking constraints. If $obj_{DWRMP^k} > UBD - \epsilon$, go to step (2.a). Otherwise, calculate $\Delta_{DW} = UBD - obj_{DWRMP^k}$, update $UBD = obj_{DWRMP^k}$, and the incumbent solution $(x^*_0, x^*) = (x^k_0, \sum_{i \in I^k} (\theta^i)^k x^i)$.

(1.b) Solve Problem (DWPP^k), let x^k be the optimal solution obtained.

(1.c) Update $U^{k+1}_{opt} = U^k_{opt} \cup \{k\}$, $U^{k+1}_{feas} = U^k_{feas}$, $I^{k+1} = I^k \cup \{k\}$.

Step 2 (BD iteration):

(2.a) Solve Problem (BRMP^k). If $UBD \leq obj_{BRMP^k} + \epsilon$, terminate and (x^*_0, x^*) is an optimal solution for Problem (P). Calculate $\Delta_{BD} = obj_{BRMP^k} - LBD$, update $LBD = obj_{BRMP^k}$. If $\Delta_{BD} \geq \Delta_{DW}$, go to step (2.b); otherwise, go to step (1.a).

(2.b) Solve Problem (BPP^k). If Problem (BPP^k) is feasible and $obj_{BPP^k} < UBD$, the incumbent solution $(x^*_0, x^*) = (x^k_0, x^k)$. If Problem (BPP^k) is infeasible, solve Problem (BFP^k). No matter which problem is solved, let x^k, λ^k be an optimal solution and the related Lagrange multipliers for the linking constraints.

(2.c) If Problem (BPP^k) is feasible, $T^{k+1}_{opt} = T^k_{opt} \cup \{k\}$, $T^{k+1}_{feas} = T^k_{feas}$. Update $UBD = min\{UBD, obj_{BPP^k}\}$. Then set $I^{k+1} = I^k \cup \{k\}$, $k = k + 1$ and go to step (1.a).

(2.d) If Problem (BPP^k) is infeasible, $T^{k+1}_{feas} = T^k_{feas} \cup \{k\}$, $T^{k+1}_{opt} = T^k_{opt}$, $k = k + 1$, go to step (1.a).

Proposition 2.8. *In Cross Decomposition Algorithm 2 shown in Table 2.3, there*

cannot be an infinite number of DWD iterations between two BD iterations.

According to Theorems 2.1, 2.2 and Proposition 2.8, the algorithm also has the finite convergence property.

Corollary 2.2. *Cross Decomposition Algorithm 2 given in Table 2.3 terminates in a finite number of steps with an optimal solution for Problem* (P) *or an indication that Problem* (P) *is infeasible.*

2.5 Case Study Results

2.5.1 Case Study Problems

We demonstrate the advantages of the proposed CD methods using two case study problems. Case study A is a bio-product supply chain optimization (SCO) problem, which was originally studied in [10] but modified into a two-stage stochastic MILP problem in [109]. The supply chain has four echelons involving different operations such as preprocessing, conversion and product distribution. The goal of the strategic SCO is to determine the optimal configuration of the supply chain network and the technologies used in the processing plants, such that the total profit is maximized and the customer demands at the demand locations are satisfied. Two uncertain parameters, demand for electricity and corn stover yield are considered. They are assumed to be independent of each other and follow uniform distributions. The first-stage decisions are whether or not specific units or technologies are to be included in the supply chain and these are represented by binary variables. The second-stage decision variables are material or product flows that are determined by the operation of the supply chain, and they are represented by continuous variables. The model

contains 18 binary variables, $2376s+7$ continuous variables and $3192s+10$ constraints, where s is the number of scenarios.

Case study B is a two-stage stochastic MILP formulated by McLean et al. [110] for optimization of an industrial chemical supply chain. The supply chain involves different grades of Primary Raw Material (PRM) that is converted in 5 manufacturing plants for onward delivery to customers. The aim of the SCO problem is to determine the optimal capacities of the plants, such that the total profits are maximized and customer demands are satisfied. The uncertainty considered is minimum demands, and it is modeled using two uniformly distributed random variables described in [110]. The first-stage decisions are the capacities of plants represented by integer variables. The second-stage decision variables are material or product flows that are determined by the operation of the supply chain, and they are represented by continuous variables. Consequently, the model contains 5 positive integer variables (all bounded from above by 20), $8210s + 6$ continuous variables and $14770s + 11$ constraints.

2.5.2 Implementation

The two case studies were performed on a virtual machine created on a computer allocated with a 3.4GHz CPU and 4GB RAM. The virtual machine runs Linux operating system (Ubuntu 14.04). All decomposition algorithms and the subproblems were implemented on GAMS 24.6.1 [111]. CPLEX 12.6.3 [108] was used as the LP and MILP solver for all algorithms. Four solution approaches were compared in the case studies, namely, monolith, BD, CD1, and CD2. Here monolith refers to solving the problem directly using CPLEX, CD1 and CD2 refer to the first and the second

CD algorithms, respectively. A GAMS extension, based on the principle of 'Gather-Update-Solve-Scatter' or GUSS [112], was utilized in all of the three decomposition methods (with default GUSS options), in order to achieve efficient model generation and solution for the decomposed scenario problems.

The relative tolerance used for all approaches was 10^{-3}. For monolith approach, the initial point was generated by CPLEX via its preprocessing procedure. For BD, the initial values for all first-stage variables were 0. CD1 and CD2 generated the initial columns x^0 by solving Problem (IPP). For case study A, (IPP) is very easy to solve so the CD methods obtained an optimal solution of (IPP) as the initial column. For case study B, however, solving (IPP) to optimality is very time-consuming and therefore only a feasible solution of (IPP) was obtained and used as the initial column.

In addition, CPLEX was set to use interior point method for the LP/MILP subproblems for case study B, as it can significantly reduce the solution time. But interior point method does not have significant benefit for case study A subproblems, so the default solution method option of CPLEX was used for case study A.

2.5.3 Results and Discussion

The computational results for case study A with the four solution approaches are summarized in Tables 2.4, 2.5, 2.6 and 2.7, and those for case study B in Tables 2.8, 2.9, 2.10 and 2.11. For both case studies, all approaches lead to the same optimal objective values (within the specified tolerance). The monolith approach is faster than the decomposition approaches for small numbers of scenarios, but with the increase of number of scenarios, its performance deteriorates rapidly because it does not exploit the decomposable structure present in the problem. For case study A,

CD1 and CD2 are faster than BD, as they both can significantly reduce the number of BD iterations with the DWD iterations. In addition, CD2 is faster than CD1 for the cases in which it can reduce the number of BD iterations with much fewer DWD iterations. Note that CD2 is not always better than CD1, as the rules it follows to determine the next iteration is only likely (and cannot guarantee) to avoid ineffective DWD iterations. For case study B, the performance of BD is very good as no more than 50 BD iterations are needed for convergence. For this case study, CD1 is worse than BD, because it does not significantly reduce the number of BD iterations but needs a relatively large number of DWD iterations. On the other hand, CD2 can reduce the number of BD iterations at the expense of only a few DWD iterations. As a result, CD2 requires fewer total iterations than BD (except for the 361 scenario case where BD requires fewer total iterations but more total solution time). These results indicate that CD2 is a better choice than CD1 if we need an approach that can consistently outperform BD.

Note that each of the tables for the decomposition approaches shows two different total times. "Total solver time" refers to the total time for CPLEX to solve all the LP/MILP subproblems. "Total run time" refers to the wall time for solving the problem, including the total solver time as well as the computing overhead. The computing overhead mainly comes from the frequent loading of scenario data and generation of scenario subproblems in the GAMS environment. If the decomposition approaches had been implemented with a platform/language that incurs little computing overhead (such as C++), the total run times could be significantly reduced. But even with the large computing overhead, the total run times of the three decomposition approaches are still much less than the monolith approach for large problems

cases.

The advantages of the proposed decomposition methods can also be seen from how the bounds of each decomposition method change at each iteration. Figure 2.4 shows for case study A how the upper and lower bounds (UBD and LBD) in the decomposition methods improve over the iterations. It can be seen that, compared to BD, CD1 and CD2 both generate better upper bounds and reach the optimal solution much faster. Although BD may generate better lower bounds at the first several iterations, CD1 and CD2 start to generate similarly good lower bounds fairly quickly. This is because a good upper bounding problem (DWRMP or BPP) solution is likely to yield a good BD cut, and CD1 and CD2 apparently can generate such solution earlier. This can be seen more clearly from case study B bound evolution curves shown in Figure 2.5. For this case study, CD1 and CD2 again generate better upper bounds at the first several iterations, and CD2 also generate better lower bounds at the beginning (which is due to the better upper bounding solutions used to construct the BD cuts). Since CD1 needs to follow a large number of DWD iterations that cannot improve the bounds, it has very slow convergence. But CD2 improves the bounds much more efficiently than CD1, and it requires fewer number of iterations than BD (except for the 361 scenario case in which BD requires 3 less iterations). These results indicate that, the main advantage of CD1/CD2 over BD is the generation of better upper bounds, and the other advantage is that the better upper bounding solutions can sometimes lead to better lower bounds. Compared to CD1, CD2 is able to consistently exploit these advantages to outperform BD, as it avoids solving DWD subproblems that are unlikely to improve the bounds.

Table 2.4: Results for case study A - Monolith (Unit for time: sec)

Number of scenarios	9	49	121	225	289	361
Optimal obj. (Million $)	-73.44	-75.53	-75.81	-75.88	-75.92	-75.93
Total solver time	70	1133	2559	8553	20990	43503
Total run time	71	1142	2571	8988	21105	43646

Table 2.5: Results for case study A - BD (Unit for time: sec)

Number of scenarios	9	49	121	225	289	361
Number of iterations	312	436	398	470	411	389
Optimal obj. (Million $)	-73.44	-75.53	-75.80	-75.90	-75.91	-75.94
Time for BPP/BFP	414	3127	6829	15990	14311	20593
Time for BRMP	41	91	95	124	35	74
Total solver time	455	3218	6924	16114	14393	20668
Total run time	793	4010	7930	17879	15882	22609

Table 2.6: Results for case study A - CD1 (Unit for time: sec)

Number of scenarios	9	49	121	225	289	361
Num. of BD iterations	73	134	102	140	137	118
Num. of DWD iterations	73	134	102	140	137	118
Total num. of iter.	146	268	204	280	274	236
Optimal obj. (Million $)	-73.44	-75.53	-75.81	-75.86	-75.92	-75.94
Time for IPP/DWPP/DWFP	75	770	1571	4119	5141	5631
Time for DWRMP/DWFRMP	17	475	869	3510	3021	3442
Time for BPP/BFP	67	626	1147	3213	3765	3976
Time for BRMP/BFRMP	5	8	4	9	10	19
Total solver time	164	1879	3591	10851	11938	12957
Total run time	291	2759	4922	15301	16064	18734

Table 2.7: Results for case study A - CD2 (Unit for time: sec)

Number of scenarios	9	49	121	225	289	361
Num. of BD iterations	233	128	130	131	98	79
Num. of DWD iterations	8	5	3	5	5	3
Total num. of iter.	241	133	133	136	103	82
Optimal obj. (Million $)	-73.44	-75.53	-75.80	-75.88	-75.91	-75.94
Time for IPP/DWPP/DWFP	9	25	38	128	159	105
Time for DWRMP/DWFRMP	11.1	13.8	35	71	77	68
Time for BPP/BFP	213	654	1413	3251	2609	2699
Time for BRMP/BFRMP	16	6	5	5	4	2
Total solver time	249	699	1489	3455	2848	2872
Total run time	454	923	1901	4650	3783	3762

Table 2.8: Results for case study B - Monolith (Unit for time: sec)

Number of scenarios	9	25	49	121	225	361
Optimal obj. (Million $)	-21592	-21613	-21615	– †	–‡	–‡
Total solver time	1258	6457	17454	–	–	–
Total run time	1263	6473	17524	–	–	–

†: Out of memory with 4GB RAM. With 8GB RAM, no integer solution returned within 36000 seconds.
‡: Out of memory with 8GB RAM.

Table 2.9: Results for case study B - BD (Unit for time: sec)

Number of scenarios	9	25	49	121	225	361
Number of iterations	39	46	47	45	40	46
Optimal obj. (Million $)	-21602	-21610	-21612	-21614	-21615	-21615
Time for BPP/BFP	161	306	618	1641	2563	5243
Time for BRMP	0.3	0.3	0.3	0.4	0.3	0.3
Total solver time	162	306	618	1642	2563	5244
Total run time	272	452	835	2081	3186	6375

Table 2.10: Results for case study B - CD1 (Unit for time: sec)

Number of scenarios	9	25	49	121	225	361
Num. of BD iterations	34	34	31	36	35	43
Num. of DWD iterations	35	35	32	37	36	44
Total num. of iter.	69	69	63	73	71	87
Optimal obj. (Million $)	-21608	-21613	-21615	-21616	-21616	-21616
Time for IPP/DWPP/DWFP	135	250	437	1046	1849	3603
Time for DWRMP/DWFRMP	0.6	1.2	2.9	6.0	10.0	35.7
Time for BPP/BFP	64	154	284	848	1842	4090
Time for BRMP/BFRMP	0.6	0.7	0.5	0.4	0.3	0.4
Total solver time	201	406	725	1901	3701	7729
Total run time	390	651	1049	2615	4992	10776

Table 2.11: Results for case study B - CD2 (Unit for time: sec)

Number of scenarios	9	25	49	121	225	361
Num. of BD iterations	29	33	35	33	33	40
Num. of DWD iterations	4	8	8	7	5	9
Total num. of iter.	33	41	43	40	38	49
Optimal obj. (Million $)	-21608	-21613	-21615	-21616	-21616	-21616
Time for IPP/DWPP/DWFP	42	109	222	458	731	1437
Time for DWRMP/DWFRMP	0.4	1.0	1.9	5.0	8.1	25
Time for BPP/BFP	46	170	352	770	1580	3202
Time for BRMP/BFRMP	0.3	0.3	0.4	0.3	0.2	0.4
Total solver time	89	281	576	1233	2319	4665
Total run time	168	433	814	1678	3014	6182

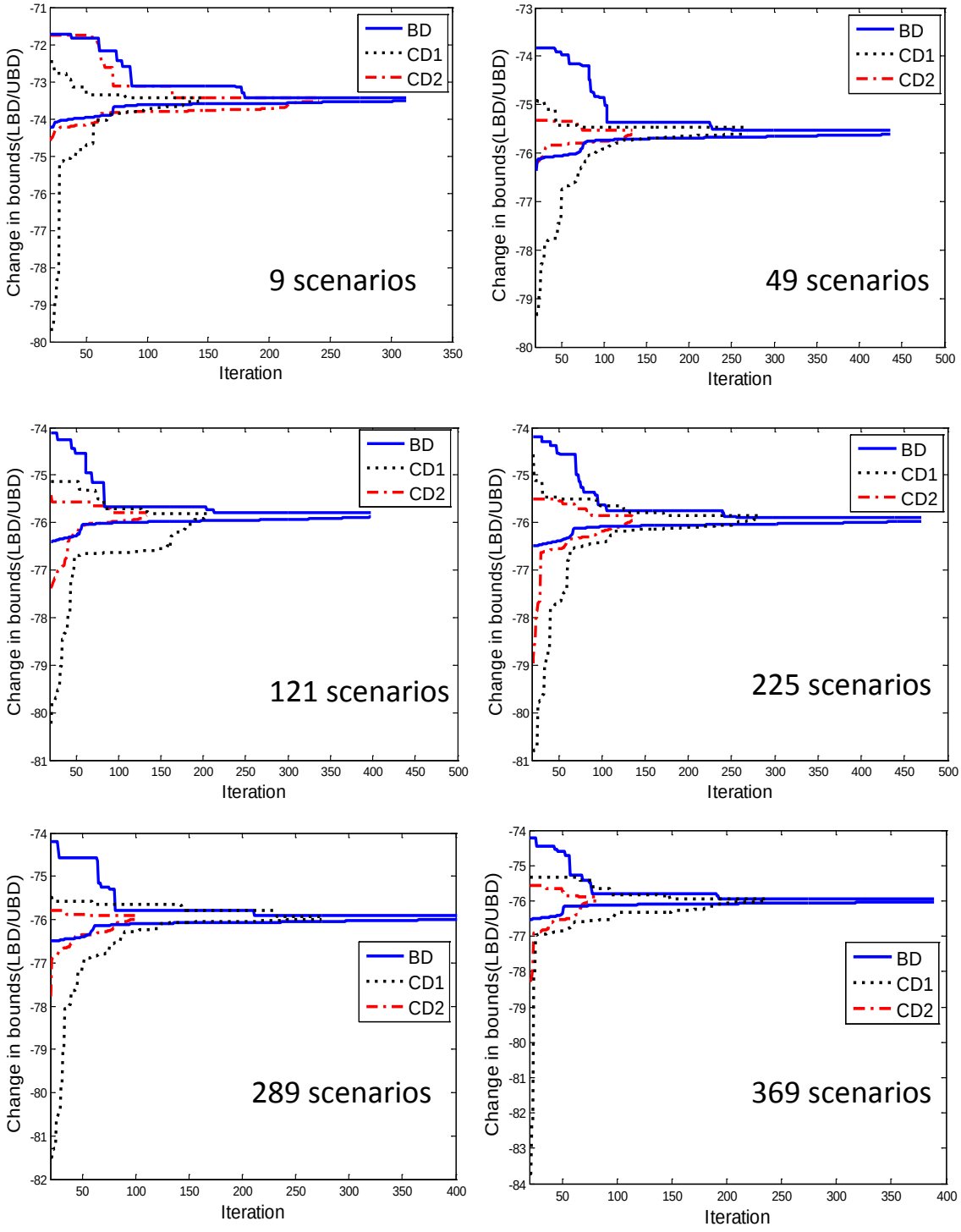


Figure 2.4: Comparison of bound evolution in different decomposition methods (case study A)

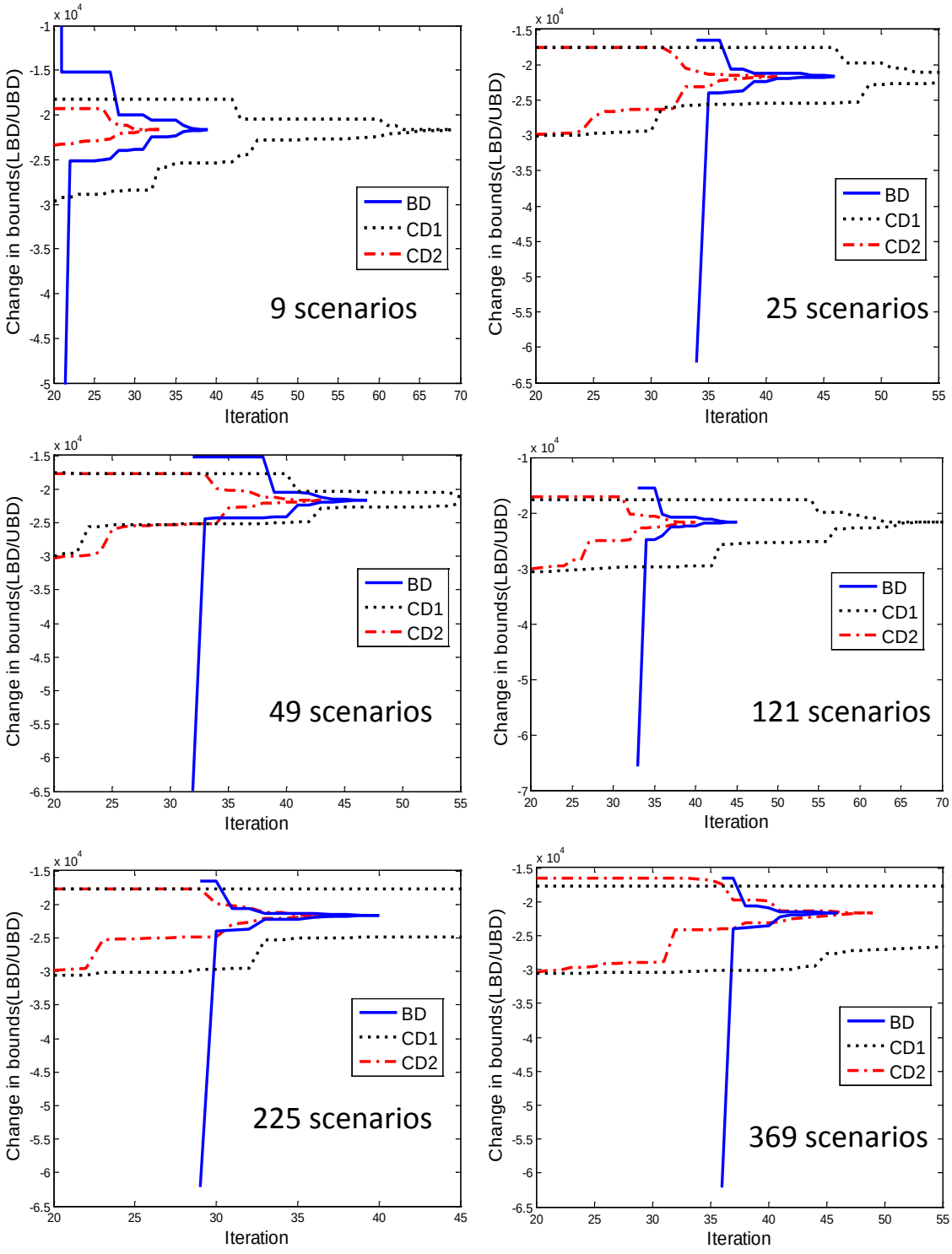


Figure 2.5: Comparison of bound evolution in different decomposition methods (case study B)

2.6 Conclusions

This chapter proposes a new cross decomposition framework for solving Problem (P). Different from the existing cross decomposition methods in the literature, this framework exploits the synergy between BD and DWD (rather than Lagrangian decomposition) to achieve improved solution efficiency. In this framework, a sequence of upper bounds for Problem (P) are generated via solving BD primal problems and DWD restricted master problems, and a sequence of lower bounds are generated via solving BD relaxed master problems, where some BD cuts are generated via solving DWD pricing problems. A phase 1 procedure to warm start the solution procedure is also developed, so the framework can deal with infeasible problems or problems for which initial feasible solutions are difficult to find. With this new framework, two cross decomposition algorithms, CD1 and CD2, are developed. CD1 alternates between one BD iteration and one DWD iteration, while CD2 determines the type of the next iteration adaptively.

The performance of the new CD approaches is demonstrated in comparison with the monolith and BD approaches, via case study of a bio-product SCO problem and an industrial chemical SCO problem. In both cases, the three decomposition methods outperform the monolith approach significantly when the number of scenarios is large. In the first case study where BD convergences slowly, both CD1 and CD2 require much fewer iterations and therefore less total solver times; when the number of scenarios is 361, CD2 reduces the solution time by more than 80% over BD (and 90% over the monolith approach). In the second case study where BD converges quickly, CD2 is still faster than BD but CD1 is slower due to the many ineffective DWD iterations. These results indicate that, for problems for which BD is not efficient enough (e.g.,

due to the "tailing off effect"), the proposed CD methods are good alternatives for exploiting the problem structure. In addition, for problems for which BD is already efficient, CD2 may still be a better alternative, but its advantage over BD may not be very significant. For example, if a two-stage MILP problem has a tight LP relaxation, BD is likely to be efficient and the advantages of the proposed CD methods may not be significant. But if this problem has a weak LP relaxation, BD is not likely to be efficient and the proposed CD methods can be much better alternatives.

The proposed CD framework applies to two-stage MILPs where the second-stage variables are continuous. Obviously, it also applies to two-stage LP problems. Extension of the current CD framework to MILPs with second-stage integer variables and MINLPs is a potential future research direction. This extension can be developed based on generalized versions of BD [35] [79] and DWD methods [113]. Furthermore, while we only discuss the application of CD to two-stage stochastic programs in this chapter, the application to multi-stage programs is viable and it is another potential future research direction.

Chapter 3

A Multicolumn-multicut Cross Decomposition Method for Stochastic Mixed-integer Linear Programming *

3.1 Introduction

Mixed-integer linear programming (MILP) paradigm has been applied to a host of problems in Process Systems Engineering (PSE) literature. Typical applications include supply chain optimization, process network design and operation, production

*This chapter is based off of the conference paper; Ogbe E, Li X, Multicolumn-multicut cross decomposition for stochastic mixed-integer linear programming, Computer Aided Chemical Engineering, 37 (2015) pp. 737-742. The equations, assumptions, propositions, theorems, symbols and notations defined in this chapter are self-contained. The following changes were made to the paper to increase the implementation performance:

a. The MATLAB/GAMS implementation platform was avoided to reduce overhead time.

b. Newer version of GAMS and CPLEX, GAMS 24.6.1 and CPLEX 12.6.3, were used to increase subproblem solution performance.

c. The GAMS utility, Gather-Update-Solve-Scatter (GUSS) [112], was utilized to efficiently solve decomposable subproblems.

d. The cross decomposition cited in this section is the first submission of the now published paper, Ogbe E, Li X, A new cross decomposition method for stochastic mixed-integer linear programming, European Journal of Operational Research, 256 (2017), pp. 287-299".

planning and scheduling, etc. These applications often involve factors that are usually not known with certainty before some decisions are made, which result in uncertain parameters in the MILP model. Using the classical scenario approach, the stochastic MILP problem can be formulated into a two-stage stochastic programming problem [21] as follows:

$$\begin{aligned} \min_{\substack{x_0,\\ x_1,\ldots,x_s}} \quad & c_0^{\mathrm{T}} x_0 + \sum_{\omega\in\Omega} c_\omega^{\mathrm{T}} x_\omega \\ \text{s.t.} \quad & A_{0,\omega} x_0 + A_\omega x_\omega \le b_{0,\omega}, \quad \forall \omega \in \Omega, \\ & x_\omega \in X_\omega, \quad \forall \omega \in \Omega, \\ & x_0 \in X_0, \end{aligned} \tag{P}$$

where $x_0 \in X_0 = \{x_0 = (x_{0,b}, x_{0,c}) \in \{0,1\}^{n_{x_{0,b}}} \times \mathbb{R}^{n_{x_{0,c}}} : B_0 x_0 \le d_0\}$ denotes the first-stage decisions, $x_\omega \in X_\omega = \{x_\omega \in \mathbb{R}^{n_x} : B_\omega x_\omega \le d_\omega\}$ denotes the second-stage decisions, and the subscript $\omega \in \Omega = \{1, 2, ..., s\}$ indexes each scenario. We assume that sets X_0 and x_ω are nonempty and bounded.

Problem (P) is computationally challenging when the number of scenarios involved is large, but its structure can be exploited by a decomposition strategy for efficient solution. Classical decomposition methods for Problem (P) include Dantzig-Wolfe decomposition (DWD) [37], Benders decomposition (BD) [34], Lagrangian decomposition (LD) [89], and cross decomposition (CD) [82] [114].

Recently, a new CD method has been developed through the integration of the classical DWD and BD methods, which has significant advantages over the classical DWD and BD methods for solving Problem (P) [115]. In this CD method, only a single column or a single cut is added to the DWD or Benders master problem

in one iteration. This chapter proposes a variant of the new CD that adds multiple columns or cuts in one iteration to achieve an improved convergence rate, as it is well-known that the multicolumn and multicut strategies can accelerate the convergence of classical DWD and BD [30].

The remaining part of the article is organized as follows. In section 3.2, we briefly introduce the new CD method. Then in section 3.3, the multicolumn-multicut CD is presented together with its convergence property. Case study results for a bio-product supply chain problem is presented in section 3.4 to demonstrate computational advantage of the proposed method. The article ends with conclusions in section 3.5.

3.2 The cross decomposition method

The CD method recently developed by the authors synergizes DWD and BD by solving the subproblems from each decomposition method in a unified framework [115]. On the one hand, two subproblems from DWD, called **DWD restricted master problem** and **DWD pricing problem** in this chapter, are solved in the CD. They are constructed through vertex representations of bounded polyhedral sets. Specifically, set $X = \prod_{\omega \in \Omega} X_\omega$ in Problem (P) can be represented as:

$$X = \{x = (x_1, ..., x_s) \in \mathbb{R}^{s \cdot n_x} : x_\omega = \sum_{j \in J} \theta^j x_\omega^j, \omega \in \Omega, \sum_{j \in J} \theta^j = 1, \theta^j \geq 0, \forall j \in J\}, \tag{3.1}$$

where set J includes indexes for all extreme points of X and possibly other points in X as well [115]. Each point used for defining X is called a *column*. The following set,

used in the lth DWD iteration in the CD method, is a subset of X,

$$X^l = \{x = (x_1, ..., x_s) \in \mathbb{R}^{s \cdot n_x} : x_\omega = \sum_{j \in J^l} \theta^{\mathrm{j}} x_\omega^{\mathrm{j}}, \omega \in \Omega, \sum_{j \in J^l} \theta^j = 1, \theta^j \geq 0, \forall j \in J^l\}, \tag{3.2}$$

where $J^l \subset J$. When using X^l instead of X, Problem (P) is restricted into the DWD restricted master problem, which can be written in the following form:

$$\begin{aligned} obj_{DWRMP^l} = \min_{x_0, \theta^j} \; & c_0^{\mathrm{T}} x_0 + \sum_{\omega \in \Omega} c_\omega^{\mathrm{T}} \left(\sum_{j \in J^l} \theta^{\mathrm{j}} x_\omega^{\mathrm{j}} \right) \\ \text{s.t.} \; & A_{0,\omega} x_0 + A_\omega \left(\sum_{j \in J^l} \theta^{\mathrm{j}} x_\omega^{\mathrm{j}} \right) \leq b_{0,\omega}, \quad \forall \omega \in \Omega, \\ & \sum_{j \in J^l} \theta^{\mathrm{j}} = 1, \quad \theta^j \geq 0, \quad \forall j \in J^l, \\ & x_0 \in X_0. \end{aligned} \tag{DWRMP^l}$$

As a result, Problem (DWRMP^l) provides an upper bound for Problem (P). Let π_ω^l be Lagrangian multipliers for the first group of constraints in Problem (DWRMP^l), then a DWD pricing problem can be solved to generate an extra point for set X^l. This problem can be decomposed over the scenarios; for scenario ω, the subproblem is:

$$\begin{aligned} obj_{DWPP_\omega^l} = \min_{x_\omega} \; & (c_\omega^{\mathrm{T}} + \left(\pi_\omega^l\right)^{\mathrm{T}} A_\omega) x_\omega \\ & x_\omega \in X_\omega. \end{aligned} \tag{DWPP$_\omega^l$}$$

On the other hand, two subproblems from BD, called **BD primal problem** and **BD relaxed master problem** in this chapter, are also solved in CD. The BD primal

problem is constructed at the kth BD iteration by fixing $x_0 = x_0^k$. The resulting problem provides an upper bound for Problem (P), and it can be decomposed over the scenarios. For scenario ω, the subproblem can be written as:

$$\begin{aligned} obj_{\mathrm{BPP}_\omega^k} = \min_{x_\omega} \ & c_0^{\mathrm{T}} x_0^k + c_\omega^{\mathrm{T}} x_\omega \\ \text{s.t.} \ & A_{0,\omega} x_0^k + A_\omega x_\omega \le b_{0,\omega}, \\ & x_\omega \in X_\omega. \end{aligned} \tag{BPP_ω^k}$$

Using the principles of projection and dualization [35], Problem (P) can be equivalently formulated into a master problem that includes a finite number of duality-based constraints called *cuts*. When including a subset of the cuts, the problem becomes the following BD relaxed master problem:

$$\begin{aligned} \min_{x_0,\eta} \ & \eta \\ \text{s.t.} \quad & \eta \ge \sum_{\omega\in\Omega} obj_{BPP_\omega^j} + \sum_{\omega\in\Omega} \left(c_0^{\mathrm{T}} + (\lambda_\omega^j)^{\mathrm{T}} A_{0,\omega}\right)\left(x_0 - x_0^j\right), \quad \forall j \in T^k, \\ & \eta \ge \sum_{\omega\in\Omega} obj_{DWPP_\omega^j} + \sum_{\omega\in\Omega} \left(c_0^{\mathrm{T}} + \left(\pi_\omega^j\right)^{\mathrm{T}} A_{0,\omega}\right) x_0 - \left(\pi_\omega^j\right)^{\mathrm{T}} b_{0,\omega}, \quad \forall j \in U^l, \\ & x_0 \in X_0, \end{aligned} \tag{BRMP^k}$$

where T^k includes indexes of Lagrangian multipliers generated from previously solved Problem (BPP_ω^k) before the kth BD iteration, and U^l includes indexes of Lagrangian multipliers generated from previously solved Problem (DWPP_ω^l).

The CD method solves the above four subproblems iteratively to yield a sequence of upper bounds and lower bounds of Problem (P), and the algorithm can converge

in a finite number of iterations to an optimal solution. Figure 3.1 illustrates the algorithmic framework. More details about the CD algorithm can be found in [115].

Remark 3.1. *The CD method is advantageous over the classical BD method, because it generates better upper and lower bounds via using (a) the better one of the upper bounds provided by DWD and BD upper bounding problems and (b) the better one of the lower bounds from DWD and BD lower bounding problems. Note that in Problem (BRMP^k), the cuts generated by DWD subproblems do not necessarily dominate the Benders cuts, and vice versa.*

3.3 The multicolumn-multicut cross decomposition method

In the multicolumn-multicut (MCMC) CD, a multicolumn DW restricted master problem instead of Problem (DWRMP^l), and a multicut BD relaxed master problem instead of Problem (BRMP^k), are solved. The multicolumn DW restricted master problem restricts set X using the following set X^l_{MC} instead of X^l:

$$X^l_{MC} = \{x = (x_1, ..., x_s) \in \mathbb{R}^{s \cdot n_x} : x_\omega = \sum_{j \in J^l} \theta^j_\omega x^j_\omega, \sum_{j \in J^l} \theta^j_\omega = 1, \theta^j_\omega \geq 0, \forall j \in J^l, \forall \omega \in \Omega\}. \tag{3.3}$$

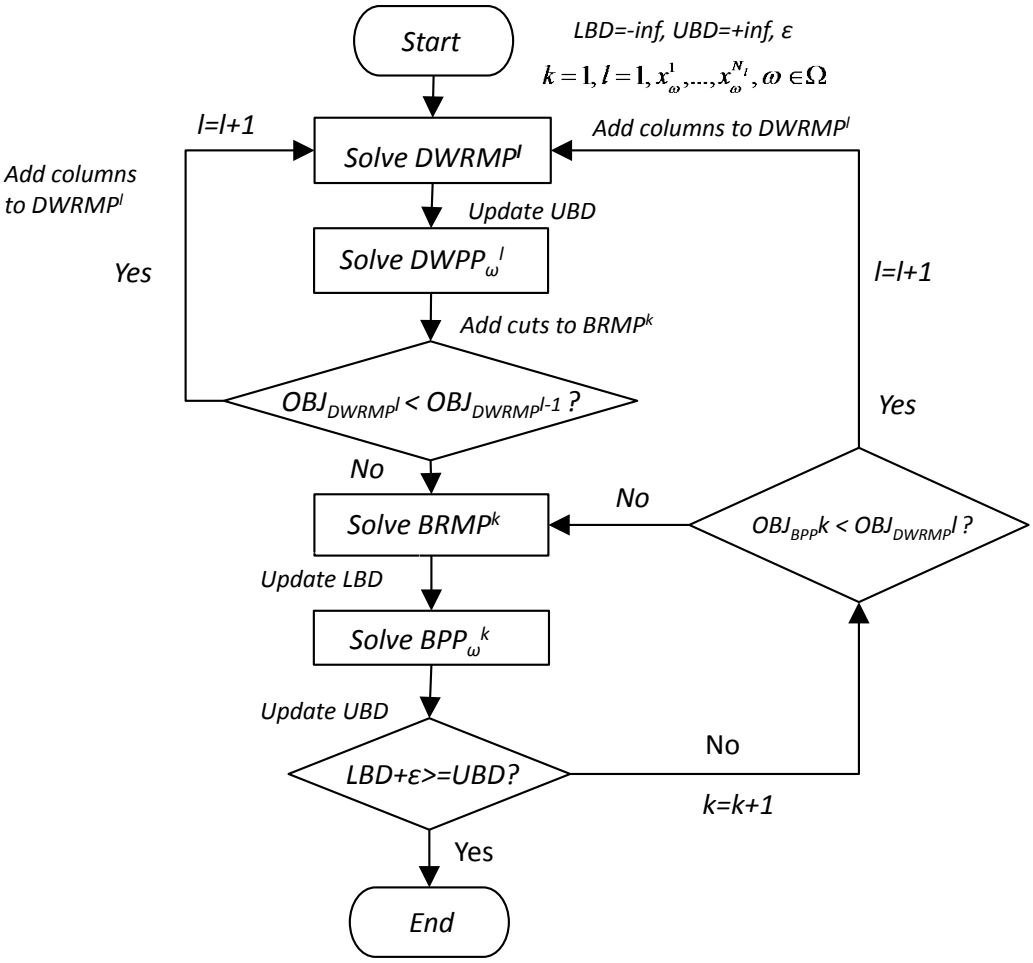


Figure 3.1: The cross decomposition algorithm flowchart

As a result, the problem can be written as:

$$
\begin{aligned}
obj_{DWRMP-MC^l} = \min_{x_0,\theta_\omega^j} \; & c_0^{\mathrm{T}} x_0 + \sum_{\omega\in\Omega} c_\omega^{\mathrm{T}} \left(\sum_{j\in J^l} \theta_\omega^{\mathrm{j}} x_\omega^{\mathrm{j}} \right) \\
\text{s.t.} \; & A_{0,\omega} x_0 + A_\omega \left(\sum_{j\in J^l} \theta_\omega^{\mathrm{j}} x_\omega^{\mathrm{j}} \right) \le b_{0,\omega}, \quad \forall \omega \in \Omega, \\
& \sum_{j\in J^l} \theta_\omega^{\mathrm{j}} = 1, \; \theta_\omega^j \ge 0, \quad \forall \omega \in \Omega, \; \forall j \in J^l, \\
& x_0 \in X_0.
\end{aligned}
\tag{DWRMP-MC^l}
$$

The following definition relates to the idea of a better restriction/relaxation of a problem.

Definition 3.1. *Problem (DWRMP-MC^l) is a better restriction of Problem* (P) *compared to Problem (DWRMP^l) if the optimal objective generated by Problem (DWRMP-MC^l) is closer to the solution of Problem* (P) *compared to that of Problem (DWRMP^l). Similarly, Problem (BRMP-MC^k) is a better relaxation of Problem* (P) *compared to Problem (BRMP^k) if the optimal objective generated by Problem (BRMP-MC^k) is closer to the solution of Problem* (P) *compared to that of Problem (DWRMP^l)*

Proposition 3.1. *Problem (DWRMP-MC^l) is a better restriction of Problem (P) compared to Problem (DWRMP^l).*

Proof. $\forall x = (x_1, \ldots, x_s) \in X^l_{MC}$, consider x_ω in this vector ($\forall \omega \in \Omega$). As x_ω is a convex combination of points in the convex set X_ω (according to Eq. (3.1)), $x_\omega \in X_\omega$. So $x \in \prod_{\omega\in\Omega} X_\omega = X$. Therefore, $X^l_{MC} \subset X$, and Problem (DWRMP-MC^l) is a restriction of Problem (P).

On the other hand, $\forall x = (x_1, \ldots, x_s) \in X^l$, according to Eq. (3.2), $\exists \theta^j \geq 0$, x^j_ω ($j \in J^l$, $\omega \in \Omega$) such that $\sum_{j\in J^l} \theta^j = 1$, $x_\omega = \sum_{j\in J^l} \theta^j x^j_\omega$. According to Eq. (3.3) this implies that $x \in X^l_{MC}$. So $X^l \subset X^l_{MC}$, which means that the feasible set of Problem (DWRMP-MC^l) is closer to the feasible set of Problem (P) and therefore Problem (DWRMP-MC^l) is a better restriction of Problem (P). □

The multicut BD relaxed master problem for the CD can be written as:

$$\begin{aligned}
&\min_{x_0,\eta_1,\ldots,\eta_s} \sum_{\omega\in\Omega} \eta_\omega \\
&\text{s.t.} \quad \eta_\omega \geq obj_{\mathrm{BPP}_\omega}(x_0^j) + \left(c_0^{\mathrm{T}} + (\lambda_\omega^j)^{\mathrm{T}} A_{0,\omega}\right)\left(x_0 - x_0^j\right), \quad \forall j \in T^k,\ \forall \omega \in \Omega, \\
&\qquad \eta_\omega \geq obj_{DWPP_\omega^j} + \left(c_0^{\mathrm{T}} + \left(\pi_\omega^j\right)^{\mathrm{T}} A_{0,\omega}\right) x_0 - \left(\pi_\omega^j\right)^{\mathrm{T}} b_{0,\omega}, \quad \forall j \in U^l,\ \forall \omega \in \Omega, \\
&\qquad x_0 \in X_0.
\end{aligned} \tag{BRMP-MC^k}$$

Proposition 3.2. *Problem (BRMP-MC^k) is a better (tighter) relaxation of Problem (P) compared to Problem (BRMP^k).*

Proof. This has been proved (in the context of multicut Benders decomposition) in the literature [46] [116]. □

Theorem 3.1. *if Problem (P) is feasible, and all the subproblems can be solved to ϵ-optimality in a finite number of steps, the MCMC CD algorithm terminates in a finite number of steps with an ϵ-optimal solution of Problem (P).*

Proof. The CD algorithm in proved to be finitely convergent in [115]. Propositions 1 and 2 show that the multicolumn and the multicut reformulations of the master problems even improves the upper and lower bounds generated at each iteration, so the MCMC CD method is finitely convergent. □

Note that, to simplify the presentation, here we assume that Problem (BPP$_\omega^k$) and Problem (DWRMP^l) (or Problem (DWRMP-MC^l) are always feasible. This assumption can actually be relaxed with appropriate changes to the algorithm. Readers are referred to [115] for more details.

3.4 Case Study

3.4.1 Case Study Problem and Implementation

We compare BD, multicut BD (MC BD), CD and MCMC CD through a bio-product supply chain optimization problem. This problem was originally presented in [10], and later modified into a two-stage stochastic MILP formulation by [109]. The stochastic MILP model contains 18 binary variables and $2376s + 7$ continuous variables, where s is the number of scenarios.

The case study was implemented on a virtual machine setup running Ubuntu 16.04 on a computer allocated with a 2.4 GHz CPU and 4 GB of memory. The decomposition algorithm and subproblems were modeled on GAMS 24.6.1 [111] with CPLEX 12.6.3 [108] (with default options) as the LP and MILP solver for the algorithm. GUSS [112], a GAMS extension, was utilized in all decomposition methods (with default GUSS options), for efficient model generation and solution of the decomposed scenario problems.

3.4.2 Results and Discussion

Figure 3.2 summarizes the times used to solve the case study problem with different numbers of scenarios using the four approaches. It can be seen that BD is the slowest among the four, and with the multicut formulation, the performance of BD is significantly improved. The multicolumn-multicut formulation also significantly improves the efficiency of CD, and the MCMC CD method achieves the best performance.

Why MCMC CD is the fastest algorithm among the four can be explained from the bound progression curves in Figures 3.3, 3.4 and 3.5. This figures illustrates how the bounds obtained by the four algorithms change during the solution procedure for

scenario instances of 81, 121 and 169. It can be seen that the bounds generated by MCMC CD converge the fastest, which is because MCMC CD not only updates the bounds using subproblems from both DWD and BD, but also uses better restricted and relaxed subproblems (through multicolumn and multicut formulations). The number of iterations for MCMC CD to converge is just slightly better than MC BD for 81 scenarios but performance gets better as the number of iterations increases; it has 20 % less iterations for the 361 scenarios. Additionally, MCMC CD is less than half of that for CD, and less than a fourth of that for BD. The fact that MCMC CD is faster than MCMC BD may be due to two reasons. One is that in MCMC CD, the upper bound is the better one of the upper bounds yielded from BD and DWD upper bounding problems. The other is that the solutions of DWD pricing problems provide extra cuts for the Benders relaxed master problem, some of which can be better than the Benders cuts. This is seen from the lower bound curves in Figures 3.3, 3.4 and 3.5.

3.5 Conclusions

A MCMC CD algorithm is developed in this chapter to solve stochastic MILPs in form of Problem (P). Tighter upper and lower bounds are derived for the MCMC CD through the multicolumn and multicut formulations and the new formulation does not hurt the finite convergence of the algorithm.

Case study of a bio-product supply chain optimization problem demonstrates the computational advantage of the proposed algorithm. The MCMC CD is faster than the classical BD method by an order of magnitude when the number of scenarios is large, and it is also significantly faster than CD and MC BD for all cases.

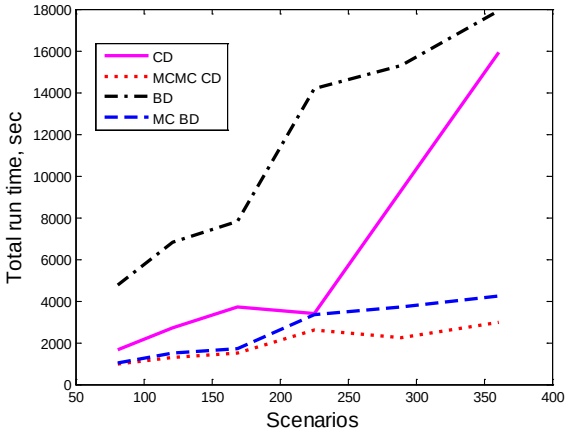


Figure 3.2: Summary of computational times

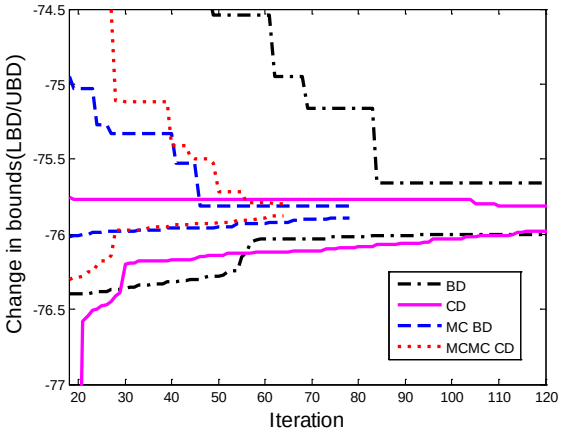


Figure 3.3: Bound evolution (for 81 scenarios)

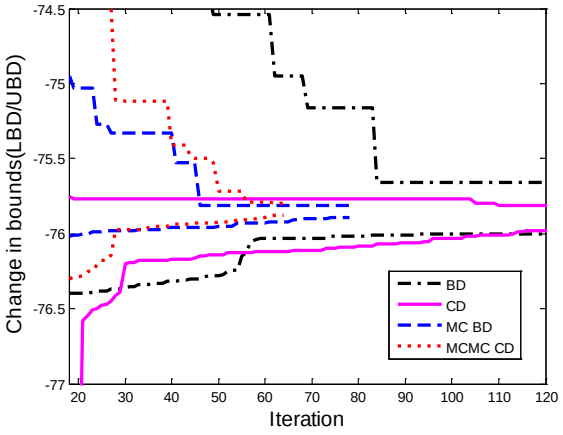


Figure 3.4: Bound evolution (for 121 scenarios)

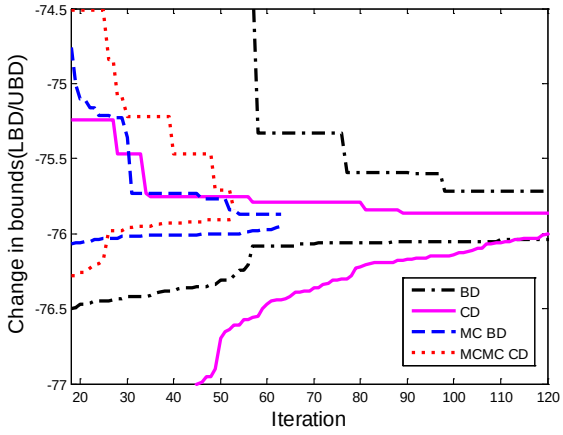


Figure 3.5: Bound evolution (for 169 scenarios)

Chapter 4

Extended Cross Decomposition Method for Mixed-integer Linear Programs with Strong and Weak Linking Constraints *

4.1 Introduction

This chapter aims at developing an efficient decomposition method to solve large-scale mixed-integer linear programming (MILP) problems in the following form:

$$\begin{aligned} \min_{x_0,x} \ & c_0^T x_0 + c^T x \\ \text{s.t.} \ & B_1 x_0 + A_1 x \leq b_1, \\ & B_2 x_0 + A_2 x \leq b_2, \\ & x \in X, \\ & x_0 \in X_0, \end{aligned} \tag{P}$$

*This chapter has been submitted for publication as Ogbe E, Li X, Extended cross decomposition method for stochastic mixed-integer linear programs with strong and weak linking constraints, Computers & Chemical Engineering. The equations, assumptions, propositions, theorems, symbols and notations defined in this chapter are self-contained.

where variables $x_0 \in \{(x_i, x_c) : x_i \in \mathbb{Z}^{n_i},\ x_c \in \mathbb{R}^{n_c}\}$, $x \in \mathbb{R}^{n_x}$, parameters $b_1 \in \mathbb{R}^{m_1}$, $b_2 \in \mathbb{R}^{m_2}$, and other parameters have conformable dimensions. The first two groups of constraints in (P) are called *linking constraints* in the sense that without them (P) can be decomposed into a number of small problems that are much easier to solve. In other words, linking constraints hinder the decomposition of (P). Among the linking constraints, the second group of constraints are called *weak linking constraints* in the sense that they do not hinder the decomposition of (P) if x_0 has fixed value, and the first group of constraints are called *strong linking constraints* in the sense that they hinder the decomposition of (P) even if x_0 has fixed value. Variables in x_0 are called *linking variables*. Sets X and X_0 are defined by linear constraints, and they are assumed to be nonempty and bounded throughout the chapter for convenience of discussion.

Many engineering optimization problems can be formulated in form of (P). A typical example is optimization under uncertainty through two-stage stochastic programming, which has been adopted in many areas of process systems engineering, such as supply chain optimization [116], natural gas network design and operation [117], refinery planning [118], expansion of chemical processes [119], etc. In two-stage stochastic programming, the first-stage decisions are to be implemented before the realization of uncertainty, and the second-stage (or recourse) decisions are made to satisfy the second-stage (or recourse) problem for all scenarios addressed by the problem formulation. In the context of two-stage stochastic programming (rigorously

speaking, its deterministic equivalent program [21]), Problem (P) can be written as:

$$
\begin{aligned}
&\min_{x_0,x_1,...,x_s} \sum_{\omega=1}^{s} \left(c_{0,\omega}^{\mathrm{T}} x_0 + c_{\omega}^{\mathrm{T}} x_\omega\right) \\
&\text{s.t. } B_1 x_0 + \sum_{\omega=1}^{s} A_{1,\omega} x_\omega \le b_1, \qquad \text{①} \\
&\quad B_{2,\omega} x_0 + A_{2,\omega} x_\omega \le b_{2,\omega}, \quad \omega \in \{1,...,s\}, \quad \text{②} \\
&\quad x_0 \in X_0, \\
&\quad x_\omega \in X_\omega, \quad \omega \in \{1,...,s\},
\end{aligned}
\tag{SP}
$$

where x_0 includes the first-stage variables, x_ω includes the second-stage variables for scenario ω, and there are totally s scenarios addressed. ① represents strong linking constraints that include x_ω for all scenarios (and therefore hinder the decomposition even when x_0 is fixed). ② represents weak linking constraints, each linking first-stage variables and second-stage variables for one scenario. Traditional two-stage stochastic programming formulation (which minimizes an expected cost) includes weak linking constraints but not strong linking constraints. Recently, two variants of two-stage stochastic programming formulations have attracted more attention and both of them include strong linking constraints. One variant is risk-averse two-stochastic programming [120] [121] [27] [122], which not only minimizes an expected cost, but also minimizes or limits some risk measure of loss, such as conditional value-at-risk (CVaR) [123]. When the risk measure is bounded in the formulation, the loss in different scenarios need to appear in a same constraint, which is a strong linking constraint. The other variant is chance-constrained two-stage stochastic programming [29] [124] that integrates the notion of chance constraint programming [28] into the two-stage formulation. In this formulation, the first-stage decisions are made such that the

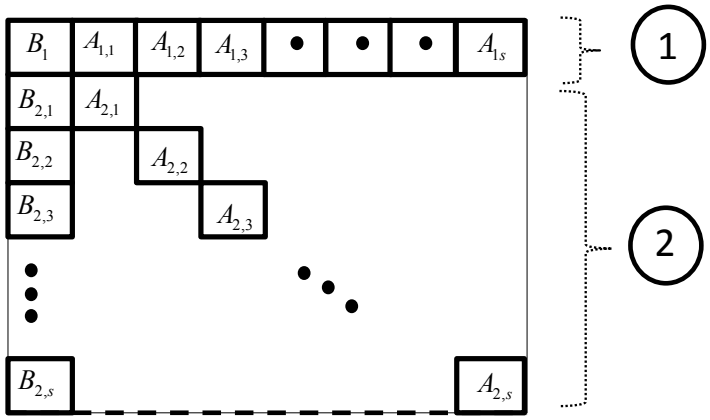


Figure 4.1: Block structure of constraint ① and ② in Problem (SP)

probability of the recourse problem being feasible is larger than a predefined level. The probability requirement is modeled with a constraint that includes all scenarios, so this constraint is a strong linking constraint. Figure 4.1 illustrates the structure of Problem (SP).

When Problem (P) does not include strong linking constraints, it can be efficiently solved by Benders decomposition (BD) [34] (or called L-shaped method [43] when applied to Problem (SP)) BD is efficient because, rather than solving the original problem directly, it solves a sequence of upper bounding problems with fixed x_0 values and a sequences of lower bounding problems that do not include x. Both upper bounding and lower bounding problems are much easier to solve than the original problem. However, when strong linking constraints are present, the upper bounding problems are not decomposable and BD does not have computational advantage. On the one hand, Dantzig-Wolfe decomposition (DWD) [37] [49] or Lagrangian decomposition [53] [38] can be applied to exploit the structure of Problem (P) by dualizing the weak and/or strong linking constraints. However, these methods are rigorous

only if Problem (P) has zero dual gap, which is very unlikely considering the problem includes integer variables. While new decomposition methods are needed for solving Problem (P) with both strong linking and weak linking constraints, the relevant research is rarely seen in the literature. The existing ideas for solving Problem (P) include the modification of Benders decomposition method that yields decomposable upper bounding problems [125] [29] and the combination of BD and Lagrangian decomposition [125].

The main contribution of this chapter is the development of a new decomposition method that is able to efficiently solve Problem (P), based on a novel cross decomposition (CD) method recently developed [126]. The idea of CD was first proposed by Van Roy [82]; it combines BD with Lagrangian decomposition in order to achieve improved efficiency. Several variants of CD method has then been developed (e.g., [84] [54]), and recently, we have developed a novel variant that combines BD and DWD [126]. There are two major advantages of our CD method. One is that the upper bound of the problem can be updated by not only the BD upper bounding problems, but also the DWD upper bounding problems (while CD methods using Lagrangian decomposition update the upper bound only through BD upper bounding problems). The other is that problem infeasibility is handled in a systematic way. We extend our CD method in this chapter via adding an additional DWD, so that the structure of Problem (P) can be readily exploited for efficient optimization.

The remaining part of the chapter is structured as follows. In section 4.2, we propose a novel bilevel decomposition strategy for solving Problem (P) and prove its validity. The discussion of this strategy motivates the extension of CD. In section 4.3, we present extended CD method, which includes a Phase I procedure that

prevents infeasible DWD subproblems. In section 4.4, we discuss in detail how to apply the proposed extended CD method to CVaR constrained two-stage stochastic programming. In section 4.5, the advantages of CVaR constrained two-stage stochastic programming and the proposed solution method are demonstrated via the case study of a bioenergy and bioproduct supply chain optimization problem. Relevant conclusions are duly drawn in section 4.6 with further discussions on future work.

4.2 A bilevel decomposition strategy for (P)

The extended cross decomposition is motivated by a bilevel decomposition strategy. In this strategy, Problem (P) is solved by BD via viewing x_0 as complicating variables, and the BD upper bounding problems (or called primal problems), which include strong linking constraints and cannot be directly decomposed, are solved by DWD. Figure 4.2 illustrates the bilevel decomposition. Here we index the upper level iterations by k and the lower level iterations for each upper level iteration by l.

4.2.1 The upper level decomposition

In the upper level, BD is used to solve Problem (P). At each iteration k, x_0 is fixed to a constant x_0^k, and the following Benders feasibility problem is solved:

$$\begin{aligned} obj_{BFP^k} = \min_{x, z_1 \geq 0, z_2 \geq 0} \ & \|z_1\| + \|z_2\| \\ \text{s.t. } & A_1 x \leq b_1 - B_1 x_0^k + z_1, \\ & A_2 x \leq b_2 - B_2 x_0^k + z_2, \\ & x \in X. \end{aligned} \tag{BFP^k}$$

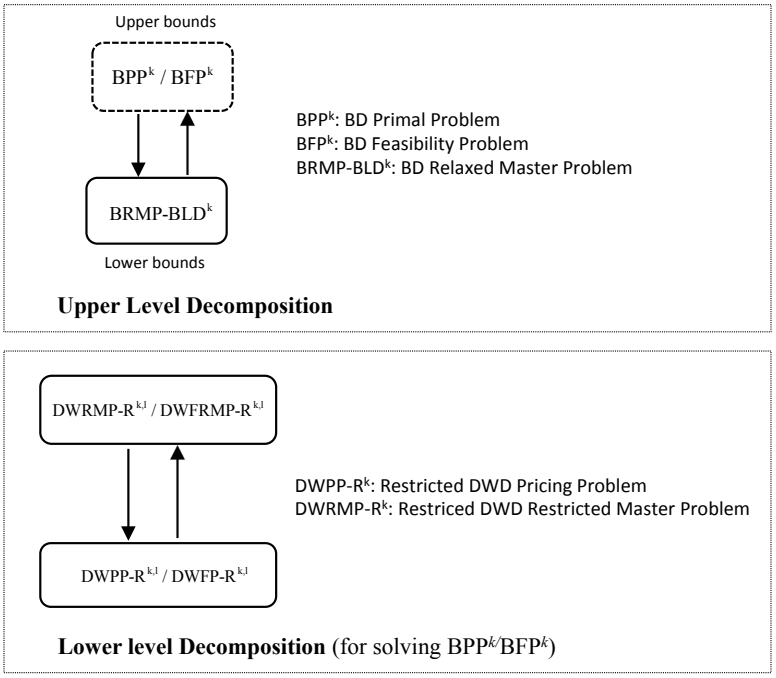


Figure 4.2: A bilevel decomposition strategy combing BD and DWD

If $obj_{BFP^k} > 0$, then Problem (P) is infeasible for $x_0 = x_0^k$, and a BD feasibility cut will be generated. Otherwise, the following Benders primal problem will be solved and a BD optimality cut will be generated:

$$\begin{aligned} obj_{BPP^k} = \min_{x} \ & c^T x + c_0^T x_0^k \\ \text{s.t.} \ & A_1 x \leq b_1 - B_1 x_0^k, \\ & A_2 x \leq b_2 - B_2 x_0^k, \\ & x \in X. \end{aligned} \tag{BPP^k}$$

Note that obj_{BPP^k} is a valid upper bound for Problem (P). Then the following lower bounding problem (called BD relaxed master problem) is solved at iteration k:

$$\begin{aligned}
\min_{x_0,\,\eta}\ & \eta \\
\text{s.t.}\ & \eta \geq obj_{BPP^i} + \left(c_0^T + \left(\lambda_1^i\right)^T B_1\right)\left(x_0 - x_0^k\right) + \left(\lambda_2^i\right)^T B_2\left(x_0 - x_0^k\right), \quad \forall i \in S_{opt}^k, \\
& 0 \geq obj_{BFP^i} + \left(\mu_1^i\right)^T B_1\left(x_0 - x_0^k\right) + \left(\mu_2^i\right)^T B_2\left(x_0 - x_0^k\right), \quad \forall i \in S_{feas}^k, \\
& x_0 \in X_0.
\end{aligned} \tag{BRMP-Std^k}$$

Here the first group of constraints are optimality cuts generated in previous iterations i at which Problem (BPP^i) is feasible, and λ_1^i, λ_2^i are Lagrange multipliers for the strong and weak linking constraints in (BPP^i), respectively. The second group of constraints are feasibility cuts generated from previous iterations i at which Problem (BPP^i) is infeasible, and μ_1^i, μ_2^i are Lagrange multipliers for the strong and weak linking constraints in (BFP^i), respectively. Readers are referred to [126] for more discussion on (BRMP-Std^k).

4.2.2 The lower level decomposition

Problem (BPP^k) or (BFP^k) is not naturally decomposable because of the strong linking constraints, but it can be solved by a DWD procedure for efficient optimization.

Specifically, (BPP^k) can be solved by iteratively solving the following two subproblems:

$$\begin{aligned} obj_{DWRMP-R^{k,l}} = \min_{\theta^0,\cdots,\theta^{l-1}\geq 0} \; & c^T\left(\sum_{i=0}^{l-1}\theta^i x^{k,i}\right) + c_0^{\mathrm{T}} x_0^k \\ \text{s.t.} \; & A_1\left(\sum_{i=0}^{l-1}\theta^i x^{k,i}\right) \leq b_1 - B_1 x_0^k, \\ & A_2\left(\sum_{i=0}^{l-1}\theta^i x^{k,i}\right) \leq b_2 - B_2 x_0^k, \\ & \sum_{i=0}^{l-1}\theta^i = 1. \end{aligned} \tag{DWRMP-R^{k,l}}$$

$$\begin{aligned} obj_{DWPP-R^{k,l}} = \min_{x} \; & c^{\mathrm{T}}x + \left(\lambda_1^{k,l}\right)^{\mathrm{T}} A_1 x \\ \text{s.t.} \; & A_2 x \leq b_2 - B_2 x_0^k, \\ & x \in X. \end{aligned} \tag{DWPP-R^{k,l}}$$

Problem (DWRMP-R^{k,l}) represents the Dantzig-Wolfe restricted master problem and Problem (DWPP-R^{k,l}) represents the Dantzig-Wolfe pricing problem. l indexes the DWD iteration. In (DWRMP-R^{k,l}), $x^{k,i}$ represents the solution of (DWPP-R^{k,i}) that is obtained in a previous DWD iteration $i(< l)$, and it is often called a column. When $l = 1$, an initial column $x^{k,0}$ is needed to construct (DWRMP-R^{k,l}). This initial column can be obtained at the beginning of the algorithm, via solving the following initial pricing problem:

$$\begin{aligned} \min_{x} \; & c^T x \\ \text{s.t.} \; & x \in X. \end{aligned} \tag{IPP}$$

Let the solution of the above problem be x^0. In the first DWD iteration for solving a (BFP^k) or (BPP^k), always set $x^{k,0} = x^0$. Lagrange multipliers of the strong linking constraints of (DWRMP-R^{k,l}) are represented by $\lambda_1^{k,l}$, and they are used to construct (DWPP-R^{k,l}). $obj_{DWRMP-R^{k,l}}$ is a valid upper bound of Problem (BPP^k), and $obj_{DWPP-R^{k,l}} + c_0^T x_0^k + (\lambda_1^{k,l})^T(B_1 x_0^k - b_1)$ is a valid lower bound of Problem (BPP^k). The DWD procedure reaches an optimal solution of Problem (BPP^k) when the upper and lower bounds converge. Readers are referred to [126] for more explanation on DWD.

Problem (BFP^k) can also be solved by a similar DWD procedure. The DWD restricted master problem is:

$$
\begin{aligned}
obj_{DWFRMP-R^{k,l}} = \min_{\theta^0,\cdots,\theta^{l-1},z_1,z_2\geq 0} \ & \|z_1\| + \|z_2\| \\
\text{s.t. } & A_1\left(\sum_{i=0}^{l-1}\theta^i x^{k,i}\right) \leq b_1 - B_1 x_0^k + z_1, \\
& A_2\left(\sum_{i=0}^{l-1}\theta^i x^{k,i}\right) \leq b_2 - B_2 x_0^k + z_2, \\
& \sum_{i=0}^{l-1}\theta^i = 1.
\end{aligned}
\qquad \text{(DWFRMP-R}^{k,l}\text{)}
$$

Let $\mu_1^{k,l}$, $\mu_2^{k,l}$ be Lagrange multipliers for the strong and weak linking constraints of (DWFRMP-R^{k,l}), respectively, then the DWD pricing problem can be expressed as:

$$
\begin{aligned}
obj_{DWFP-R^{k,l}} = \min_{x,z_2\geq 0} \ & \left(\mu_1^{k,l}\right)^{\mathrm{T}} A_1 x + \|z_2\| \\
\text{s.t. } & A_2 x \leq b_2 - B_2 x_0^k + z_2, \\
& x \in X.
\end{aligned}
\qquad \text{(DWFP-R}^{k,l}\text{)}
$$

The proposed DWD procedure for solving Problem (BFP^k) is similar to but slightly different from the Phase I of a standard DWD procedure, so we prove the validity of the procedure via the next two propositions. The first proposition states that the solution of (DWFP-R^{k,l}) yields a valid lower bound of Problem (BFP^k), and the second proposition states the procedure terminates in a finite number of iterations with an optimal solution to Problem (BFP^k).

Proposition 4.1. *$obj_{DWFP-R^{k,l}} + \left(\mu_1^{k,l}\right)^T \left(B_1 x_0^k - b_1\right)$ is a valid lower bound of Problem* (BFP^k).

Proof. Since (DWFRMP-R^{k,l}) cannot be unbounded or infeasible, it must have an optimal solution. We can express the optimal objective value of (DWFRMP-R^{k,l}) as:

$$
\begin{aligned}
&obj_{DWFRMP-R^{k,l}} \\
&= \min_{x \in X^{k,l},\, z_1, z_2 \geq 0} (\mu_1^{k,l})^T (A_1 x + B_1 x_0^k - b_1 - z_1) + (\mu_2^{k,l})^T (A_2 x + B_2 x_0^k - b_2 - z_2) + \|z_1\| + \|z_2\| \\
&= \left\{ \min_{x \in X^{k,l}} (\mu_1^{k,l})^T (A_1 x + B_1 x_0^k - b_1) + (\mu_2^{k,l})^T (A_2 x + B_2 x_0^k - b_2) \right\} \\
&\quad + \left\{ \min_{z_1 \geq 0} \|z_1\| - (\mu_1^{k,l})^T z_1 \right\} + \left\{ \min_{z_2 \geq 0} \|z_2\| - (\mu_2^{k,l})^T z_2 \right\},
\end{aligned}
$$

where set $X^{k,l} = \{x \in \mathbb{R}^{n_x} : x = \sum_{i=0}^{l-1} \theta^i x^i,\ \sum_{i=1}^{l-1} \theta^i = 1,\ \theta^i \geq 0, \forall i = 0, \cdots, l-1\}$.

Suppose $\exists \hat{z}_1 \geq 0$ such that $\|\hat{z}_1\| - (\mu_1^{k,l})^T \hat{z}_1 \leq -\epsilon$ ($\epsilon > 0$), then $\forall \alpha > 0$, $\|\alpha \hat{z}_1\| - (\mu_1^{k,l})^T \alpha \hat{z}_1 \leq -\alpha\epsilon$, which implies that $\min_{z_1 \geq 0} \|z_1\| - (\mu_1^{k,l})^T z_1 = -\infty$. This contradicts the fact that $obj_{DWFRMP-R^{k,l}}$ is finite. Therefore $\|z_1\| - (\mu_1^{k,l})^T z_1 \geq 0$, $\forall z_1 \geq 0$, which results in

$$\min_{z_1 \geq 0} \|z_1\| - (\mu_1^{k,l})^T z_1 = 0,$$

(where the minimum value is attained at $z_1 = 0$). Similarly,

$$\min_{z_2 \geq 0} \|z_2\| - (\mu_2^{k,l})^T z_2 = 0.$$

According to weak duality of Problem (BFP^k), the following value is a lower bound of (BFP^k):

$$\begin{aligned}
&\min_{\substack{x \in X,\, z_1, z_2 \geq 0, \\ A_2 x \leq b_2 - B_2 x_0^k + z_2}} (\mu_1^{k,l})^T (A_1 x + B_1 x_0^k - b_1 - z_1) + \|z_1\| + \|z_2\| \\
=&\left\{ \min_{\substack{x \in X, z_2 \geq 0 \\ A_2 x \leq b_2 - B_2 x_0^k + z_2}} (\mu_1^{k,l})^T A_1 x + \|z_2\| + (\mu_1^{k,l})^T (B_1 x_0^k - b_1) \right\} \\
&+ \left\{ \min_{z_1 \geq 0} \|z_1\| - (\mu_1^{k,l})^T z_1 \right\}. \\
=&obj_{DWFP-R^{k,l}} + \left(\mu_1^{k,l}\right)^T \left(B_1 x_0^k - b_1\right).
\end{aligned}$$

□

In order to establish the finite convergence of decomposition, we make the following assumption on the linear programming (LP) solver used to solve the LP subproblems.

Assumption 4.1. *The primal and dual optimal solutions of a LP problem returned by the LP solver are extreme points of the LP and its dual problems, respectively.*

Proposition 4.2. *The DWD procedure for solving Problem* (BFP^k)*, i.e., iteratively solving* (DWFRMP-R^{k,l}) *and* (DWFP-R^{k,l})*, is finite.*

Proof. At the DWD iteration l, suppose that the solution of (DWFP-R^{k,l}), $x^{k,l}$, is

also the solution of a previously solved pricing problem. On the one hand,

$$obj_{DWFP-R^{k,l}} = \min_{\substack{z_2 \geq 0, \\ A_2 x^{k,l} \leq b_2 - B_2 x_0^k + z_2}} ||z_2|| + (\mu_1^{k,l})^T A_1 x^{k,l}.$$

On the other hand, the optimal value of (DWFRMP-R^{k,l}) can be expressed as:

$$\begin{aligned} obj_{DWFRMP-R^{k,l}} &= \min_{\substack{x \in X^{k,l}, z_1, z_2 \geq 0 \\ A_2 x \leq b_2 - B_2 x_0^k + z_2}} ||z_1|| + ||z_2|| + \left(\mu_1^{k,l}\right)^T \left(B_1 x_0^k + A_1 x - b_1 - z_1\right) \\ &= \left\{ \min_{\substack{x \in X^{k,l}, z_2 \geq 0 \\ A_2 x \leq b_2 - B_2 x_0^k + z_2}} ||z_2|| + (\mu_1^{k,l})^T A_1 x \right\} + (\mu_1^{k,l})^T \left(B_1 x_0^k - b_1\right) \\ &\quad + \left\{ \min_{z_1 \geq 0} ||z_1|| - (\mu_1^{k,l})^T z_1 \right\} \\ &= \left\{ \min_{\substack{x \in X^{k,l}, z_2 \geq 0 \\ A_2 x \leq b_2 - B_2 x_0^k + z_2}} ||z_2|| + (\mu_1^{k,l})^T A_1 x \right\} + (\mu_1^{k,l})^T \left(B_1 x_0^k - b_1\right) \end{aligned}$$

where $X^{k,l} = \{x \in \mathbb{R}^{n_x} : x = \sum_{i=0}^{l-1} \theta^i x^i,\ \sum_{i=0}^{l-1} \theta^i = 1,\ \theta^i \geq 0, \forall i = 0, \cdots, l-1\}$. Since $x^{k,l}$ has been generated before, it is a point in $X^{k,l}$, so

$$\min_{\substack{x \in X^{k,l}, z_2 \geq 0 \\ A_2 x \leq b_2 - B_2 x_0^k + z_2}} ||z_2|| + (\mu_1^{k,l})^T A_1 x \quad \leq \min_{\substack{z_2 \geq 0, \\ A_2 x^{k,l} \leq b_2 - B_2 x_0^k + z_2}} ||z_2|| + (\mu_1^{k,l})^T A_1 x^{k,l},$$

and therefore

$$obj_{DWFRMP-R^{k,l}} \leq obj_{DWFP-R^{k,l}} + (\mu_1^{k,l})^T \left(B_1 x_0^k - b_1\right).$$

This means that, the upper and lower bounds of Problem (BFP^k) converge once an extreme point of X is generated at the second time. Since polyhedral set X has only

a finite number of extreme points, so the DWD procedure always terminates in a finite number of iterations.

□

4.2.3 Integration of the two levels

At the termination of the DWD procedure for solving Problem (BPP^k),

$$obj_{BPP^k} = obj_{DWPP-R^{k,l}} + c_0^T x_0^k + (\lambda_1^{k,l})^T (B_1 x_0^k - b_1), \tag{4.1}$$

where l indexes the last DWD iteration. $(x_0^k, x^{k,l})$ is a feasible solution for the original problem (P), and it can be used to update the upper bound of (P). The feasible solution that gives the best upper bound of the current BD iteration is called the incumbent solution, denoted by (x_0^*, x^*).

Let $\lambda_2^{k,l}$ be Lagrange multipliers for the weak linking constraints in (DWPP-R^{k,l}), according to strong duality of (DWPP-R^{k,l}),

$$obj_{DWPP-R^{k,l}} = \min_{x \in X} c^T x + \left(\lambda_1^{k,l}\right)^T A_1 x + \left(\lambda_2^{k,l}\right)^T \left(A_2 x - b_2 + B_2 x_0^k\right). \tag{4.2}$$

From equations (4.1) and (4.2),

$$obj_{BPP^k} = \min_{x \in X} c^T x + c_0^T x_0^k + \left(\lambda_1^{k,l}\right)^T \left(A_1 x - b_1 + B_1 x_0^k\right) + \left(\lambda_2^{k,l}\right)^T \left(A_2 x - b_2 + B_2 x_0^k\right),$$

which implies that $\lambda_1^{k,l}$ and $\lambda_2^{k,l}$ are (optimal) Lagrange multipliers of (BPP^k). Therefore, the optimality cuts in the standard BD relaxed master problem (BRMP-Std^k)

can be written as (via substituting (4.1)):

$$\eta \geq obj_{DWPP-R^{i,j}} + c_0^T x_0 + \left(\lambda_1^{i,j}\right)^T (B_1 x_0 - b_1) + \left(\lambda_2^{i,j}\right)^T B_2 \left(x_0 - x_0^k\right), \quad \forall (i,j) \in T_{opt}^k,$$

where T_{opt}^k includes index pairs that index the BD iteration at which an optimality cut is generated and the last DWD iteration for this BD iteration. We can also rewrite the feasibility cuts in the similar way. As a consequence, Problem (BRMP-Std^k) can be rewritten as:

$$\begin{aligned}
\min_{x_0,\,\eta} \quad & \eta \\
\text{s.t.} \quad & \eta \geq obj_{DWPP-R^{i,j}} + c_0^T x_0 + \left(\lambda_1^{i,j}\right)^T (B_1 x_0 - b_1) + \left(\lambda_2^{i,j}\right)^T B_2 \left(x_0 - x_0^k\right), \\
& \forall (i,j) \in T_{opt}^k, \\
& 0 \geq obj_{DWFP-R^{i,j}} + \left(\mu_1^{i,j}\right)^T (B_1 x_0 - b_1) + \left(\mu_2^{i,j}\right)^T B_2 \left(x_0 - x_0^k\right), \quad \forall (i,j) \in T_{feas}^k, \\
& x_0 \in X_0.
\end{aligned} \tag{BRMP-BLD^k}$$

where T_{feas}^k includes index pairs that index the BD iteration at which a feasibility cut is generated and the last DWD iteration for this BD iteration. If Problem (BRMP-BLD^k) is infeasible, then Problem (P) is also infeasible.

Figure 4.3 provides the flowchart of the bilevel decomposition algorithm. The algorithm considers two tolerances, ϵ and σ. ϵ is the tolerance for the solution of Problem (P), and it is also the tolerance for the upper level decomposition. σ is the tolerance for each lower level DWD procedure. $\epsilon >> \sigma$ is required to ensure the final solution is ϵ-optimal. According to Assumption 4.1, both the upper level BD procedure and the lower level DWD procedures can terminate in a finite number of

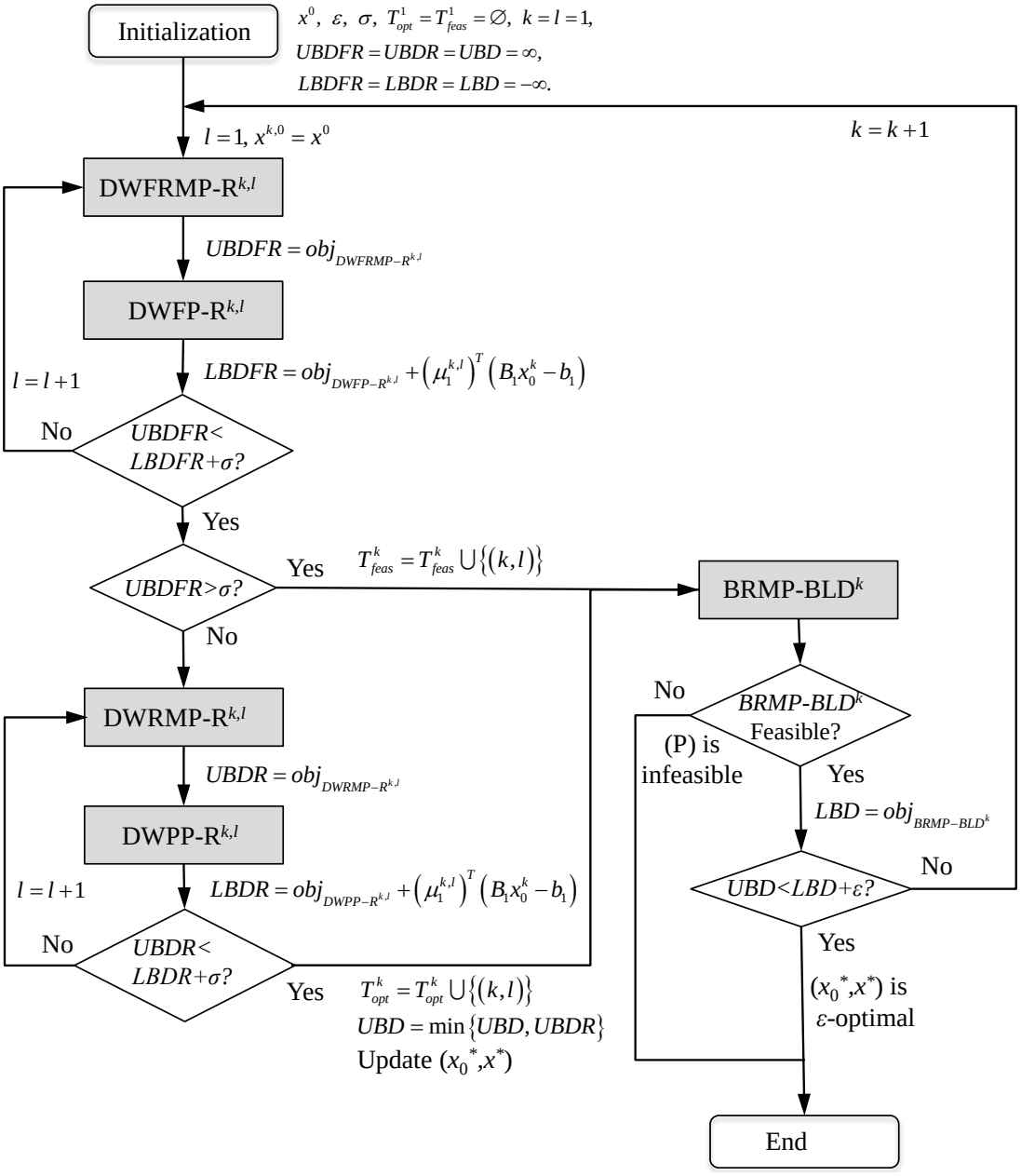


Figure 4.3: The flowchart of the bilevel decomposition method

iterations, so the bilevel decomposition method has the following finite termination property.

Theorem 4.1. *The bilevel decomposition algorithm shown in Figure 4.3 terminates in finite time with an ϵ-optimal solution or a certification that Problem* (P) *is infeasible, if Assumption 4.1 holds and all subproblems can be solved in finite time.*

We have not seen in the literature the bilevel decomposition method was ever developed for solving Problem (P), but methods using similar ideas exist, such as the one developed by Bruno and Sagastizábal [125]. In Bruno and Sagastizábal's bilevel decomposition method, the upper level is Lagrangian decomposition and the lower level is BD, but the method is rigorous for (P) only when no integer variables are present, because it requires strong duality of (P).

4.3 The extended cross decomposition method

4.3.1 The basic ECD framework and subproblems

In this section, we develop a decomposition framework motivated by ideas from bilevel decomposition presented in section 4.2 and cross decomposition [126]. In this framework, which is shown in Figure 4.4, an upper and a lower level decomposition strategy are integrated to efficiently solve Problem (P). At the upper level, a cross decomposition approach for (P) where the following problems; DW restricted master problem, DW pricing problem, Benders primal and feasibility problems (BPP^k and BFP^k) and Benders relaxed master problems, are iteratively solved is presented. Problems (BPP^k) and (BFP^k), that are not decomposable, are then solved via a finitely convergent DWD procedure at the lower level, as in the bilevel decomposition. We refer to this approach as the *extended cross decomposition* (ECD) approach. The ECD lower level subproblems are same to the lower level subproblems in the bilevel decomposition method proposed in the last section, so we only describe the upper level

subproblems here.

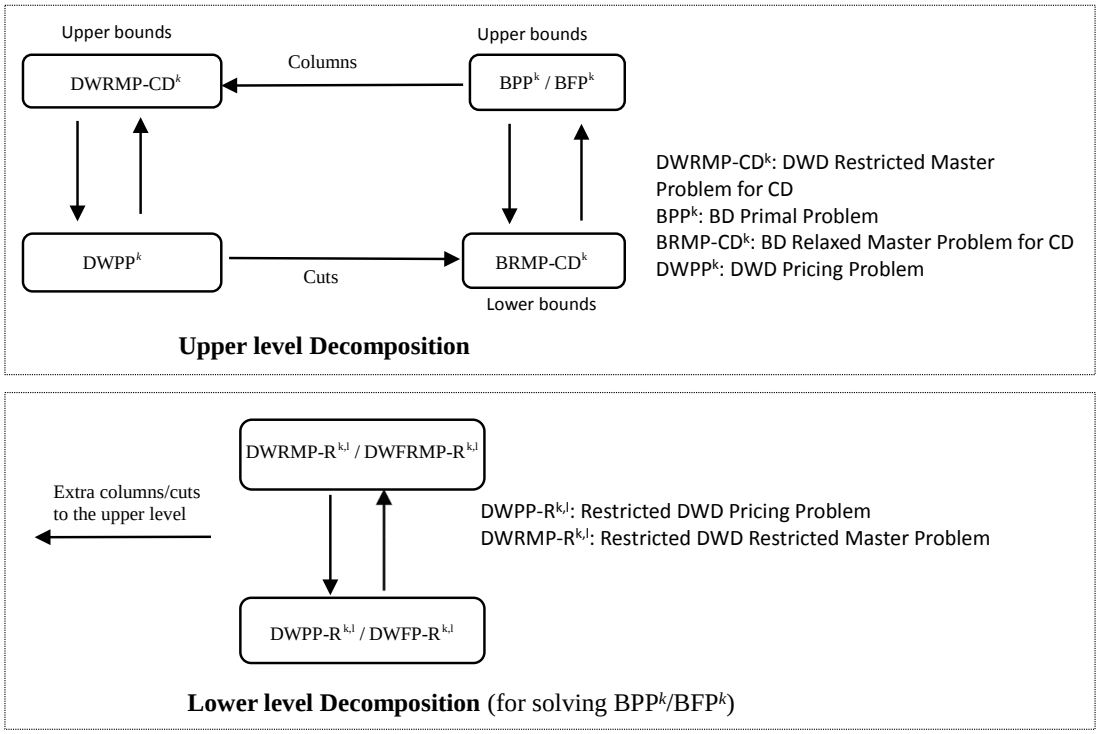


Figure 4.4: Extended cross decomposition method

At a particular upper level iteration k of ECD, the following DW restricted master

problem is solved first:

$$
\begin{aligned}
obj_{DWRMP-CD^k} = \min_{x_0,\theta^i} \quad & c_0^T x_0 + \sum_{i=0}^{k-1} \theta^i x^i \\
s.t. \quad & B_1 x_0 + A_1 \sum_{i=0}^{k-1} \theta^i x^i \leq b_1, \\
& B_2 x_0 + A_2 \sum_{i=0}^{k-1} \theta^i x^i \leq b_2, \\
& \sum_{i=0}^{k-1} \theta^i = 1, \\
& \theta^i \geq 0,
\end{aligned}
\tag{DWRMP-CD^k}
$$

where x^i is a column generated at the upper level and $i = 0, 1, ..., k-1$ are indices of the generated columns. Problem (DWRMP-CD^k) provides valid upper bounds to Problem (P). Just like in the bilevel decomposition, an initial feasible column, $x^0 \in X$, is needed to solve Problem (DWRMP-CD^k). It can be obtained by solving the initial pricing problem (IPP). Here we assume that Problem (DWRMP-CD^k) is always feasible for convenience, and later we will discuss how to guarantee the feasibility via a Phase I procedure.

Let λ_1^k and λ_2^k be the Lagrange multiplier associated with strong and weak linking constraints in Problem (DWRMP-CD^k) respectively, then we can construct and solve the following DW pricing problem:

$$
\begin{aligned}
obj_{DWPP^k} = \min_{x} \quad & c^T x + \left(\lambda_1^k\right)^T A_1 x + \left(\lambda_2^k\right)^T A_2 x \\
s.t. \quad & x \in X.
\end{aligned}
\tag{DWPP^k}
$$

The solution of Problem (DWPP^k), denoted by x^k, provides a new column for Problem

(DWRMP-CD^k).

If a full DWD iteration is completed and the algorithm is to enter a BD iteration, the following Benders relaxed master problem is solved:

$$
\begin{aligned}
&\min_{x_0,\eta} \eta \\
&\text{s.t. } \eta \geq obj_{DWPP^i} + c_0^T x_0 + \left(\lambda_1^i\right)^{\mathrm{T}} (B_1 x_0 - b_1) + \left(\lambda_2^i\right)^{\mathrm{T}} (B_2 x_0 - b_2), \quad \forall i \in U_{opt}^k, \\
&\quad \eta \geq obj_{DWPP-R^{i,j}} + c_0^T x_0 + \left(\lambda_1^{i,j}\right)^{\mathrm{T}} (B_1 x_0 - b_1) + \left(\lambda_2^{i,j}\right)^{\mathrm{T}} B_2 \left(x_0 - x_0^j\right), \quad \forall (i,j) \in T_{opt}^k, \\
&\quad 0 \geq obj_{DWFP-R^{i,j}} + \left(\mu_1^{i,j}\right)^{\mathrm{T}} (B_1 x_0 - b_1) + \left(\mu_2^{i,j}\right)^{\mathrm{T}} B_2 \left(x_0 - x_0^j\right), \quad \forall (i,j) \in T_{feas}^k, \\
&\quad x_0 \in X_0,
\end{aligned}
\tag{BRMP-CD^k}
$$

where U_{opt}^k includes the indices of all previous iterations in which Problem (DWPP^k) is solved, while T_{opt}^k and T_{feas}^k have been defined in section 4.2. The optimality and feasibility cuts in (BRMP-CD^k) come from previous BD iterations and DW iterations, and readers are referred to [126] for more discussions on (BRMP-CD^k) .

Note that in ECD, DW restricted master problems and DW pricing problems are included in both the upper level and the lower level. However, in the upper level both strong and weak linking constraints complicate the problem and therefore they are dualized in the DW pricing problems, while in the lower level only strong linking constraints complicate the problem (because the weak linking constraints no longer complicate the problem when x_0 is fixed) and therefore weak linking constraints are not dualized in the DW pricing problem.

4.3.2 Synergizing the upper and the lower level

The two levels in ECD can be synergized to yield stronger bounds and accelerate convergence, like the synergy of DWD and BD iterations within the CD method. Specifically, columns and cuts can be generated from the solutions of the DW pricing problems at the lower level, and they can be added to Problem (DWRMP-CD^k) and (BRMP-CD^k) at the upper level in order to yield better upper and lower bounds for the original problem (P). As a result, the DW restricted master problem can be enhanced as:

$$
\begin{aligned}
\min_{x_0,\theta^i} \quad & c_0^T x_0 + \sum_{i=0}^{t-1} \theta^i \hat{x}^i \\
s.t. \quad & B_1 x_0 + A_1 \sum_{i=0}^{t-1} \theta^i \hat{x}^i \leq b_1, \\
& B_2 x_0 + A_2 \sum_{i=0}^{t-1} \theta^i \hat{x}^i \leq b_2, \\
& \sum_{i=0}^{t-1} \theta^i = 1, \\
& \theta^i \geq 0,
\end{aligned}
\qquad \text{(DWRMP-ECD}^k\text{)}
$$

where $\hat{x}^i$ represent a column generated from either the upper level or the lower level, and t indexes all the columns. If $\hat{x}^i$ is generated at the kth upper level iteration, then $\hat{x}^i = x^k$. If $\hat{x}^i$ is generated at the lth lower level iteration for the kth upper level iteration, then $\hat{x}^i = x^{k,l}$.

The enhanced Benders relaxed master problem can be written as:

$$
\begin{aligned}
&\min_{x_0,\eta} \eta \\
&\text{s.t.} \quad \eta \geq obj_{DWPP^i} + c_0^T x_0 + \left(\lambda_1^i\right)^{\mathrm{T}} \left(B_1 x_0 - b_1\right) + \left(\lambda_2^i\right)^{\mathrm{T}} \left(B_2 x_0 - b_2\right), \quad \forall i \in U_{opt}^k, \\
&\qquad \eta \geq obj_{DWPP-R^{i,j}} + c_0^T x_0 + \left(\lambda_1^{i,j}\right)^{\mathrm{T}} \left(B_1 x_0 - b_1\right) + \left(\lambda_2^{i,j}\right)^{\mathrm{T}} B_2 \left(x_0 - x_0^j\right), \quad \forall (i,j) \in \hat{T}_{opt}^k, \\
&\qquad 0 \geq obj_{DWFP-R^{i,j}} + \left(\mu_1^{i,j}\right)^{\mathrm{T}} \left(B_1 x_0 - b_1\right) + \left(\mu_2^{i,j}\right)^{\mathrm{T}} B_2 \left(x_0 - x_0^j\right), \quad \forall (i,j) \in \hat{T}_{feas}^k, \\
&\qquad x_0 \in X_0,
\end{aligned}
$$

(BRMP-ECD^k)

where $\hat{T}_{opt}^k$ includes index pairs that index the BD iteration at which an optimality cut is generated and all DWD iterations for this BD iteration, while $\hat{T}_{feas}^k$ includes index pairs that index the BD iteration at which a feasibility cut is generated and all DWD iterations for this BD iteration.

Figure 4.5 shows the algorithmic flowchart of ECD. The left part of the flowchart depicts the ECD upper level, where Problems (DWRMP-ECD^k), (DWPP^k) and (BRMP-ECD^k) are solved, and the right part of the flowchart depicts the ECD lower level, where Problems (DWFRMP-R^{k,l}) and (DWFP-R^{k,l}) are solved iteratively. The finite convergence property of the lower level procedure is proved in the last section, and the finite convergence property of the upper level CD procedures is proved in [126], so we have the following finite convergence property of ECD.

Theorem 4.2. *If Assumption 4.1 holds and all subproblems can be solved in finite time, then the extended cross decomposition method in Figure 4.5 terminates in a finite number of steps with an ϵ-optimal solution of Problem* (P) *or a certification that Problem* (P) *is infeasible.*

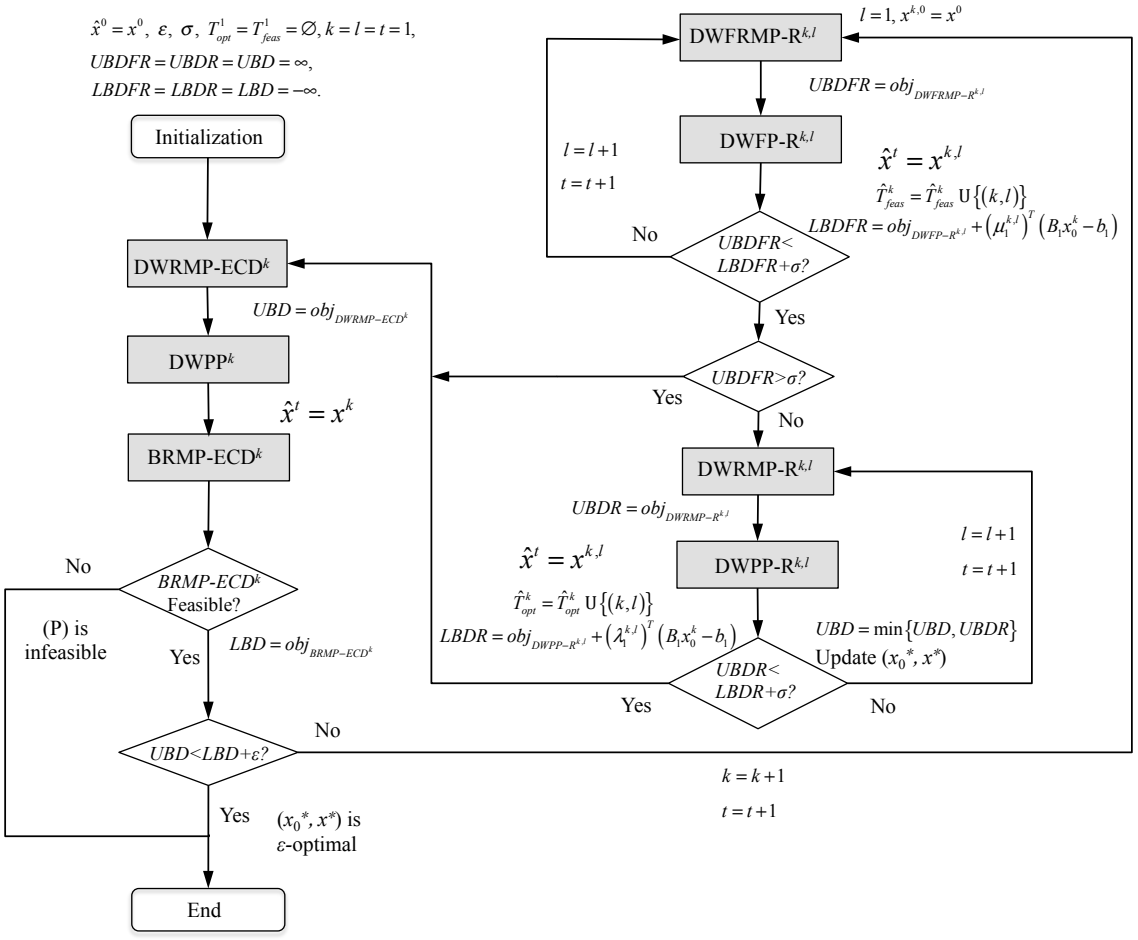


Figure 4.5: The flowchart of the extended cross decomposition method

4.3.3 Further discussions

Phase 1 procedure and subproblems

In this section, we present a Phase I procedure to prevent the infeasibility of Problem (DWRMP-ECD^k). The Phase I procedure essentially solves the following feasibility

problem:

$$\begin{aligned} obj_{FP} = \min_{x_0,x,z_1\geq 0,z_2\geq 0} \ & \|z_1\| + \|z_2\| \\ \text{s.t. } & A_1x \leq b_1 - B_1x_0 + z_1, \\ & A_2x \leq b_2 - B_2x_0 + z_2, \\ & x \in X. \end{aligned} \tag{FP}$$

where z_1 and z_2 are slack variables and $||\cdot||$ denotes any norm function. We use the 1-norm in the case study in this chapter.

Problem (FP) is always feasible with the assumption that X and X_0 are nonempty. The Phase I procedure actually employs ECD to solve Problem (FP), and the subproblems need to be solved in this procedure is illustrated in Figure 4.6. At iteration k at the upper level, the following DW restricted master problem is solved:

$$\begin{aligned} \min_{x_0,\theta^0,\ldots,\theta^{k-1}\geq 0,z_1\geq 0,z_1\geq 0} \ & \|z_1\| + \|z_2\| \\ \text{s.t. } & B_1x_0 + A_1\sum_{i=0}^{t-1}\theta^i x^i \leq b_1 + z_1, \\ & B_2x_0 + A_2\sum_{i=0}^{t-1}\theta^i x^i \leq b_2 + z_2, \\ & \sum_{i=0}^{t-1}\theta^i = 1, \\ & x_0 \in X_0. \end{aligned} \tag{DWFRMP-ECD^k}$$

Let μ_1^k and μ_2^k be the Lagrange multipliers for the strong and weak linking constraints of the Problem (DWFRMP-ECD^k), then the following DW pricing problem can be

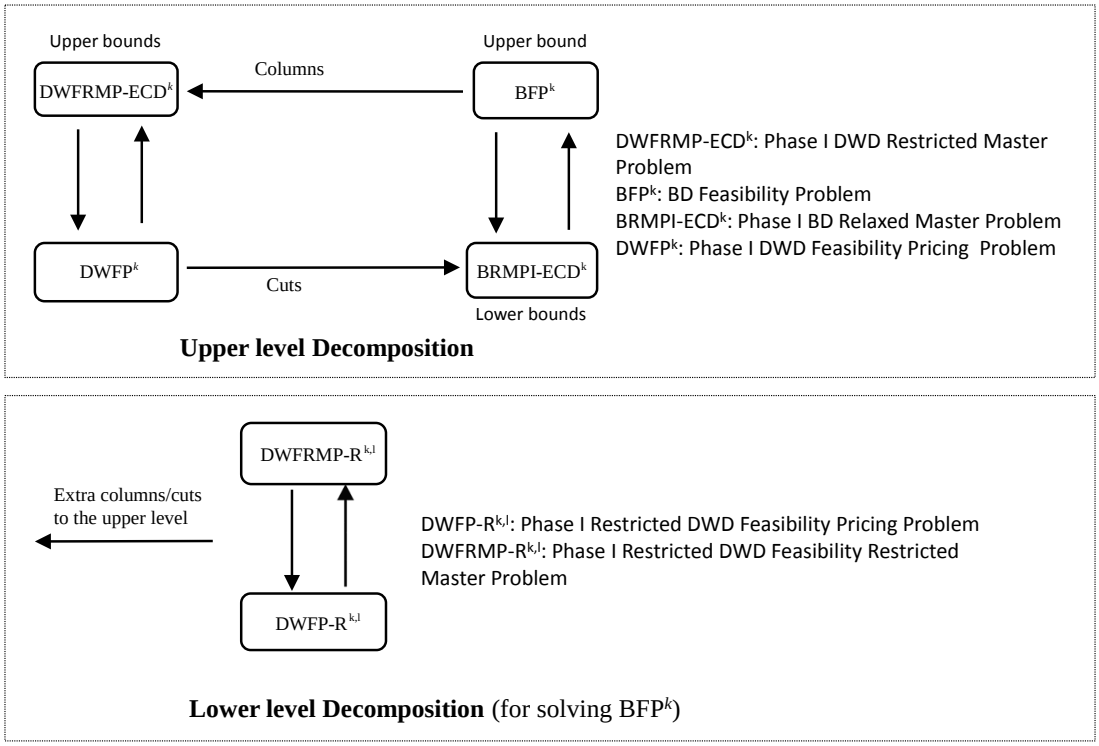


Figure 4.6: Diagram of the Phase I procedure for extended cross decomposition method

constructed and solved:

$$\begin{aligned} \min_{x} \ & \left(\mu_1^k\right)^T A_1 x + \left(\mu_2^k\right)^T A_2 x \\ s.t. \ & x \in X. \end{aligned} \tag{DWFP^k}$$

The BD relaxed master problem solved in Phase I can be written as:

$$
\begin{aligned}
&\min_{x_0,\eta} \eta \\
&\text{s.t.} \quad \eta \geq obj_{DWFP^i} + \left(\mu_1^i\right)^{\mathrm{T}} \left(B_1 x_0 - b_1\right) + \left(\mu_2^i\right)^{\mathrm{T}} \left(B_2 x_0 - b_2\right), \quad \forall i \in U_{feas}^k, \\
&\qquad \eta \geq obj_{DWFP-R^{i,j}} + \left(\mu_1^{i,j}\right) \left(B_1 x_0 - b_1\right) + \left(\mu_2^{i,j}\right)^{\mathrm{T}} B_2 \left(x_0 - x_0^i\right), \quad \forall (i,j) \in \hat{T}_{feas}^k, \\
&\qquad x_0 \in X_0.
\end{aligned}
\tag{BRMPI-ECD^k}
$$

where U_{feas}^k includes the indices of all previous iterations in which Problem (DWFP^k) is solved. Note that there are no feasibility cuts generated in Phase I, because Problem (BFP^k) is always feasible (provided X and X_0 are nonempty).

Remark 4.1. *The Phase I procedure illustrated in Figure 4.6 terminates finitely with an δ-optimal solution of Problem* (FP)*. If $obj_{FP} > \delta$, then Problem* (P) *is infeasible; otherwise, Problem* (P) *is feasible and the optimal solution of Problem* (FP) *is a feasible solution of Problem* (P)*.*

In order for the optimality of Problem (FP) to precisely imply the feasibility of Problem (P), we need to select a sufficiently small optimality tolerance δ for Phase I (as least no larger than the feasibility tolerance of Problem (P)). The algorithmic flowchart of Phase I is similar to the one given in Figure 4.5, so it is omitted.

The optimal solution of Problem (FP) and all the other columns generated in Phase I will be added in Problem (DWRMP-ECD^k) in Phase II, so Problem (DWRMP-ECD^k) is always feasible. The cuts generated in Phase I can also be added to the BD relaxed master problem in Phase II, but these cuts will all appear like feasibility cuts, as

explained in [126]. The Phase II BD relaxed master problem can be written as:

$$\min_{x_0,\eta} \eta$$

$$\begin{aligned}
\text{s.t.} \quad & \eta \geq obj_{DWPP^j} + c_0^T x_0 + \left(\lambda_1^j\right)^{\mathrm{T}} (B_1 x_0 - b_1) + \left(\lambda_2^j\right)^{\mathrm{T}} (B_2 x_0 - b_2) \quad \forall j \in U_{opt}^k, \\
& \eta \geq obj_{DWPP-R^{i,j}} + c_0^T x_0 + \left(\lambda_1^{i,j}\right)^{\mathrm{T}} (B_1 x_0 - b_1) + \left(\lambda_2^{i,j}\right)^{\mathrm{T}} B_2 \left(x_0 - x_0^j\right), \quad \forall (i,j) \in \hat{T}_{opt}^k, \\
& 0 \geq obj_{DWFP^i} + \left(\mu_1^i\right)^{\mathrm{T}} (B_1 x_0 - b_1) + \left(\mu_2^i\right)^{\mathrm{T}} (B_2 x_0 - b_2), \quad \forall i \in U_{feas}^k, \\
& 0 \geq obj_{DWFP-R^{i,j}} + \left(\mu_1^{i,j}\right)^{\mathrm{T}} (B_1 x_0 - b_1) + \left(\mu_2^{i,j}\right)^{\mathrm{T}} B_2 \left(x_0 - x_0^i\right), \quad \forall (i,j) \in \hat{T}_{feas}^k, \\
& x_0 \in X_0.
\end{aligned}$$

(BRMPII-ECD^k)

Adaptive switching between upper and lower level subproblems

In [126], it is demonstrated that the CD method can be improved by switching between DWD and BD iterations in an adaptive way. The switching rules proposed in [126] can effectively avoid ineffective iterations at which columns and cuts generated do not help to close the optimality gap, so it can be adapted here to improve the convergence rate. Specifically, the ECD switching rules determine the alternation between the lower and the upper levels, following two criteria:

1. After an upper level DWD iteration, solve Problem (BRMPI-ECD^k) (for Phase I) or (BRMPII-ECD^k) (for Phase II) to update the lower bound. If the decrease of the upper bound in the DWD iteration is more than the increase of the lower bound, then the algorithm will go to another DWD iteration. Otherwise, the algorithm will go to the lower level in order to solve (BFP^k) (for Phase I) or (BPP^k) (for Phase II).

2. After the convergence of a lower level procedure, solve Problem (DWFRMP-ECD^k)

(for Phase I) or (DWRMP-ECD^k) (for Phase II). If the optimal objective value of the problem is better than the current upper bound, then the algorithm will go to an upper level DWD iteration. Otherwise, the algorithm will go to another BD iteration, which requires solving (BFP^k) (for Phase I) or (BPP^k) (for Phase II) via another lower level procedure.

We tag the algorithm following the above rules as "ECD2" and the standard algorithm without the rules as "ECD1". The advantage of ECD2 will be seen through the case study.

4.4 Application of ECD: Risk-averse two-stage stochastic programming

4.4.1 Background

Different risk measurement metrics are adopted in the literature for risk-averse two-stage stochastic programming, such as variance, variability index, probabilistic financial risk, downside risk, value-at-risk (VaR), etc [127]). In this chapter, the conditional value-at-risk, CVaR, developed by Rockafellar and Uryasev [123] is used to account for risks because of its ability to account for worse case, computational tractability and its close relationship to VaR [123] [128] [129] [130].

CVaR can be included two-stage stochastic programming in two ways. One is to penalize CVaR in the objective function; in this case Problem (SP) does not have strong linking constraints, and it can be solved by classical decomposition methods. For example, Ahmed [27] used a cutting plane approach to solve a risk-averse stochastic linear program; the risk measured by a *dispersion* metric. Schultz and Tiedemann [131] applied Lagrangian decomposition to solve a risk averse stochastic mixed-integer programming problem. Noyan [121] developed two decomposition techniques based

on generic BD to solve a stochastic LP minimizing CVaR. Qi et al. used multicut BD to solve a risk-averse problem [132].

The other way is to bound CVaR in constraints. Some applications with CVaR included in constraints include portfolio optimization [123], oil and energy optimization [125], oil supply chain [133], large-scale industrial batch plants [134] [135], traveling saleman's problem [130]. Modeling CVaR as objective and as constraints have been shown by Krokhmal et al. to provide the same efficient frontier [128]. However, CVaR constrained approach is preferred because decision makers interpret and quantify right hand side easier than penalty parameters in the objective function, as argued by Fabian and Veszpremi [136]. The research on decomposition based strategy to solve CVaR constrained two-stage stochastic programming is very limited in the literature. Huang and Zheng [130] proposed a BD-type approach for the traveling saleman's problem with risk constraints. In their strategy, they seperated the risk constraints from scenario coupling constraints. Feasibility cuts were then developed to exclude solutions already generated. Bruno and Sagastizábal [125] developed two decomposition approaches to solve a two-stage stochastic linear program with CVaR constraints. The first approach is a Benders-like procedure, and the second approach is a bilevel decomposition procedure with Lagrangian decomposition in the upper level and BD in the lower level. The second approach is rigorous only for linear programming problems.

4.4.2 CVaR-constrained two-stage stochastic programming

We consider the following CVaR constraints for scenario-based risk-averse two-stage stochastic programming derived in Appendix B based on [128] [134] [135]:

$$\zeta + \frac{1}{(1-\beta)} \sum_{\omega \in S} p_\omega \psi_\omega \leq b_0, \tag{4.3}$$

$$\psi_\omega \geq f_\omega(x_\omega) - \zeta, \quad \psi_\omega \geq 0, \quad \forall \omega \in \{1, ..., s\}, \tag{4.4}$$

where $f_\omega(x_\omega)$ denotes a scenario dependent loss function of the second-stage variables at scenario ω, $p_\omega \in \{0, 1\}$ denotes the probability of scenario ω ($\sum_{\omega=1}^{s} p_\omega = 1$), $\beta \in \{0, 1\}$ is a user specified probability, b_0 is a user specified risk threshold for the loss function over all scenarios. When the CVaR constraints are included, the two-stage stochastic programming problem can be written as:

$$\begin{aligned}
&\min_{\substack{x_0, x_1, ..., x_s \\ \zeta, \psi_1, ..., \psi_s}} \sum_{\omega=1}^{s} \left(c_{0,\omega}^{\mathrm{T}} x_0 + c_\omega^{\mathrm{T}} x_\omega \right) \\
&\text{s.t. } A_{0,\omega} x_0 + A_\omega x_\omega \leq b_{0,\omega}, \quad \omega \in \{1, ..., s\}, \quad (A) \\
&\qquad \psi_\omega \geq f_\omega(x_\omega) - \zeta, \quad \omega \in \{1, ..., s\}, \qquad (B) \\
&\qquad \zeta + \sum_{\omega=1}^{s} \hat{p}_\omega \psi_\omega \leq b_0, \qquad (C) \\
&\qquad x_\omega \in X_\omega, \psi_\omega \geq 0, \quad \forall \omega \in \{1, ..., s\}, \\
&\qquad x_0 \in X_0, \zeta \in \mathbb{R},
\end{aligned} \tag{CVaR-SP}$$

where we define $\hat{p}_\omega = \frac{1}{(1-\beta)} p_\omega$ for convenience. In this formulation, x_0 denotes the first-stage decisions and x_ω denotes second-stage decisions for scenario ω. Constraints (A) and (B) are weak linking constraints, because when x_0 is fixed, they are decomposable

over the scenarios. Constraints (C) are strong linking constraints, because they cannot be decomposed over the scenarios no matter whether x_0 is fixed.

ECD is developed under the assumption that all variables are explicitly bounded, because otherwise some ECD subproblems may be unbounded. In (CVaR-SP), ψ_ω and ζ are not explicitly bounded although their boundedness is already implied by the formulation, so next we are to show how to estimate the bounds for ψ_ω and ζ in order to prevent unbounded ECD subproblems.

First, from equation (4.4), a valid lower bound of ψ_ω is $\psi_\omega^{lo} = 0$, $\forall \omega \in \{1, \cdots, s\}$.

Second, from equations (4.3) and (4.4),

$$\zeta \leq b_0 - \frac{1}{1-\beta} \sum_{\omega=1}^{s} p_\omega \psi_\omega b_0 \leq b_0,$$

so a valid upper bound on ζ is $\zeta^{up} = b_0$.

Next, we show that a valid lower bound on ζ is

$$\zeta^{lo} = \min \left\{ -\frac{(1-\beta)}{\beta} b_0 + \frac{1}{\beta} f^{lo}, f^{lo} \right\},$$

where f^{lo} is a valid lower bound for the loss function for any scenario, i.e.,

$$f^{lo} \leq \min_{\omega \in \{1, \cdots, s\}} \left\{ \min_{x_\omega \in X_\omega} f_\omega(x_\omega) \right\}.$$

We show this by showing that any ζ that is less than ζ^{lo} does not satisfy (1). This is

because $\forall \zeta < \zeta^{lo}$,

$$\begin{aligned}
\zeta + \frac{1}{1-\beta}\sum_{\omega=1}^{s} p_\omega \psi_\omega &\geq \zeta + \frac{1}{1-\beta}\sum_{\omega=1}^{s} p_\omega (f_\omega(x_\omega) - \zeta) \\
&\geq \zeta + \frac{1}{1-\beta}\sum_{\omega=1}^{s} p_\omega (f^{lo} - \zeta) \\
&= \zeta + \frac{(f^{lo} - \zeta)}{1-\beta}\sum_{\omega=1}^{s} p_\omega \\
&= \zeta + \frac{(f^{lo} - \zeta)}{1-\beta} \\
&= \frac{-\beta}{1-\beta}\zeta + \frac{1}{1-\beta} f^{lo} \\
&> \frac{-\beta}{1-\beta}\left\{-\frac{(1-\beta)}{\beta} b_0 + \frac{1}{\beta} f^{lo}\right\} + \frac{1}{1-\beta} f^{lo} \\
&= b_0.
\end{aligned}$$

Finally, we show that $\psi_\omega^{up} = f_\omega^{up} - \zeta^{lo}$ is a valid upper bound for ψ_ω, where f_ω^{up} is a valid lower bound for the loss function for scenario ω, i.e.,

$$f_\omega^{up} \geq \max_{x_\omega \in X_\omega} f_\omega(x_\omega).$$

According to equation (4.4), this can be shown by showing that $f_\omega^{up} - \zeta^{lo} \geq 0$, and $f_\omega^{up} - \zeta^{lo} \geq f_\omega(x_\omega) - \zeta$, and these two relations result directly from the definitions of f_{up}^{ω} and ζ^{lo}.

In summary, the bounds ζ and ψ_ω can be computed as follows:

$$\begin{aligned}
&\psi_\omega^{lo} = 0, \quad \forall \omega \in \{1, \cdots, s\}, \\
&\zeta^{up} = b_0, \\
&\zeta^{lo} = \min\{-\frac{(1-\beta)}{\beta} + \frac{1}{\beta} f^{lo}, f^{lo}\}, \\
&\psi_\omega^{up} = f_\omega^{up} - \zeta^{lo}, \quad \forall \omega \in \{1, \cdots, s\}.
\end{aligned}$$

The ECD subproblems, including those for Phase I and Phase II, are explained in Appendix B.

4.5 Case Study

4.5.1 Case Study Problem

The case study considers an energy and bio-product supply chain optimization (SCO) problem originally studied in [10]. The supply chain considered has four layers involving material collecting, material preprocessing, energy and bio-product production and product distribution. The goal of the strategic SCO is to determine the optimal configuration of the supply chain network and the technologies used in the production plants, such that the total profit is maximized and the customer demands at the demand locations are satisfied. McLean and Li [109] considered uncertainties in the supply chain and formulate a risk-neutral two-stage optimization problem in the form of problem (SP), where the first-stage decisions are binary variables determining whether or not specific units are to be developed and specific technologies are to be adopted for energy/bio-product production, and the second-stage decisions are continuous variables determining material and product flows in the operation of the

supply chain. We consider two uncertain parameters in the problem, the yield of corn stover and the minimum demand for electricity at each of the three demand locations. The yield and demand uncertainties are assumed to be uniformly distributed in ranges [790 890] $t/km^2 \cdot year$ and [1 3] $10^6 \cdot MWh/year$, respectively. Other details of the case study is provided in Appendix B.

We consider two formulations for the case study problem. One is a risk-neutral stochastic programming formulation without CVaR constraints, where the uncertainty in the minimum electricity demands are not considered. The other a risk-averse stochastic programming formulation with CVaR constraints, where the loss function $f_\omega(x_\omega)$ is total unsatisfied minimum electricity demands (i.e., the total minimum electricity demand minus the total electricity supply). The parameters for the CVaR constraints are explained in Appendix B.

4.5.2 Solution methods and Implementation

The risk-neutral formulation does not have strong linking constraints. We compare four solution methods for this formulation, which are monolith (i.e., solving the full problem directly using an existing commercial optimization solver), BD (i.e., classical Benders decomposition), CD1 (the CD method proposed in [126] without the adaptive alternation between BD and DWD iterations), and CD2 (the CD methods proposed in [126] with the adaptive alternation). The risk-averse formulation does have strong linking constraints (which come form the CVaR constraints), and we also compare four solution methods for it, which are monolith, BLD (i.e., the bilevel decomposition method proposed in section 4.2), ECD1 (the proposed ECD method without adaptive switching between the two levels), and ECD2 (the proposed ECD

method with adaptive switching between the two levels).

The simulation was run on a virtual machine setup running Ubuntu 16.04 on a computer allocated with a 2.4 GHz CPU and 4 GB of memory. The two formulations and the solution methods were programmed on GAMS 24.6.1 [111] with CPLEX 12.6.3 [108] being the LP/MILP solver. GUSS [112], a GAMS extension, was utilized in all decomposition methods (with default GUSS options), in order to achieve efficient model generation and solution for the decomposed scenario problems. The relative termination tolerance used for all solution methods was $\epsilon = 10^{-3}$.

4.5.3 Results and Discussion

The results for the risk-neutral formulation are shown on Tables 4.1, 4.2, 4.3 and 4.4 while those for the risk-averse formulation are given on Tables 4.5, 4.6, 4.7 and 4.8. The objective values shown in the tables present negative profits. The results show that both decomposition and monolith approaches attain the same optimal profit (within the tolerance) for every problem instance. We can also see that the optimal profit attained by the risk-averse formulation is lower than that of the risk-neutral formulation by 8%. This results from the restriction of the CVaR constraints, and indicates that the optimal profit predicted by the risk-neutral formulation is actually not realistic if we want to satisfy most part of the minimum electricity demand.

The computational results from the risk-neutral formulation reemphasizes the computational superiority of cross decomposition for large-scale MILPs. It can be seen from the results in Tables 4.1, 4.2, 4.3 and 4.4 that for the 9 scenario instance (a relatively small-scale problem), the monolith approach is better than the decomposition methods. This is understandable because the problem is not large enough

for the benefit of exploiting problem structure being significant. However, for all scenario instances greater than 9, the monolith approach suffers from solving large-scale problems. We see a drastic rise in solution time for monolith (from 9 scenarios to 25 scenarios, and from 25 scenarios to 49 scenarios) because of the large number of simplex iterations required for solution, eventhough the number of explored nodes for branch-and-bound does not consistently increase. This behavior is as a result of the change in problem nature for different realization of uncertainty, which in turn give rise to increasing difficulty of removing initial dual infeasibilities when the monolith solver solves the dual LP problem. On the other hand, we see no such behavior when decomposition is applied. Overall, decomposition is better than the monolith approach for scenarios greater than 9. Cross decomposition is superior to BD; CD1, even though takes more CD iterations for all instances, is superior to BD, because of fewer BD primal problem are solved. Apparently, BD primal problem is more expensive to solve compared to the DW pricing problem as seen from the tables. For the largest case, CD1 is able to save 18 % of solver time compared to BD. The monolith for this instance is not able to find the optimal solution within 24 hours. CD2 has superior computational performance compared to all the other approaches considered, CD1, BD and monolith; for all scenarios compared to CD1 and BD, and for scenario instances greater than 9 for the monolith. The overall iteration by CD2 is consistently better than CD1 and BD; with an average solver time decrease of 40 % and 32 % compared to BD and CD1 respectively. The computational efficiency of CD can be attributed to the better initial upper bounds and cuts generated from the DW restricted and pricing problems respective. This was discussed extensively in Ogbe and Li [126].

For the risk-averse formulation, we can see similar results from Tables 4.5, 4.6, 4.7 and 4.8. For small-scale problems, the monolith approach is very efficient. For large problem instances however, we see similar drastic rise in solution time for monolith (from 9 scenarios to 25 scenarios, and from 49 scenarios to 81 scenarios) as large number of simplex iterations are required for solution, eventhough the number of explored nodes for branch-and-bound does not change significantly. Decomposition is necessary for efficient solution; solving the pricing problem dominates the wall time as no primal problems are solved. The number of pricing problems solved is therefore indicative of the level of difficulty in solving the overall problem. For BLD, the total number of pricing problems solved corresponds to the number of lower level DWD iterations shown on Table 4.6. The number of upper level iterations and total number of pricing problems solved for ECD1 are less than that for BLD except for the 121 scenario instance. This translates to better performance of ECD1 compared to BLD. Even more impressive performance is achieved by the ECD2. ECD2 is superior to ECD1 for all instances except the 81 scenario case, and better than BLD for all scenario instances. For example, for the largest scenario case considered, ECD2 can achieve as much as 10 times reduction in wall time compared to BD, and over an order of magnitude compared to the monolith. The efficiency of ECD2 can be due to the fact that it is able to avoid solving inefficient upper DWD iterations (especially solving inefficient upper level pricing problems) compared to ECD1, and also ECD2 can reduce the number of lower level problems solved compared to BLD.

The upper and lower bound progression shown on Figures 4.7 and 4.8 further demonstrates the effectiveness of the proposed extended decomposition strategy. For the risk-neutral formulation, the upper bounds provided by CD2 are consistently

better than that from CD1 and BD. Initially, the lower bounds are not as good as those from BD, but eventually becomes better than BD and closes the optimality gap quickly. CD1, on the other hand, has more iterations compared to BD, but the computational time is less than BD for most instances as mentioned earlier. We see for the risk-averse formulation, that the upper bounds of ECD2 are not necessarily as good as that from BLD for all scenario instances. However, as the algorithm progresses, both upper and lower bounds from ECD (all instances for ECD2 and particular instances of 9, 49, 81 and 169 for ECD1) quickly approach each other faster than those from BLD, and ultimately converges to the optimal solution faster than BLD. It is important to note that for this case, the number of pricing problems solved gives a better indication of the computational efficiency than the number of upper BD iterations. Hence, the bound evolution plots should be used in conjunction with the computational results to fully assess the performance of the ECD.

4.6 Conclusions and Future work

An extended cross decomposition method was developed for problems with strong and weak linking constraints, as in Problem (P). We show that the algorithm converges to the optimal solution within the set tolerance. A Phase 1 procedure for the extended cross decomposition was also developed to handle infeasible subproblems.

The problem with CVaR constraints is Problem (CVaR-SP). Problem (CVaR-SP) reduces to a risk neutral two-stage stochastic problem which can be solved by ordinary cross decomposition or Benders decomposition. CD2 has superior performance over CD1, BD and monolith for large scenario instances. This is seen from the computational times and bound evolution plots. Compared to BD, about 37 % of solver time

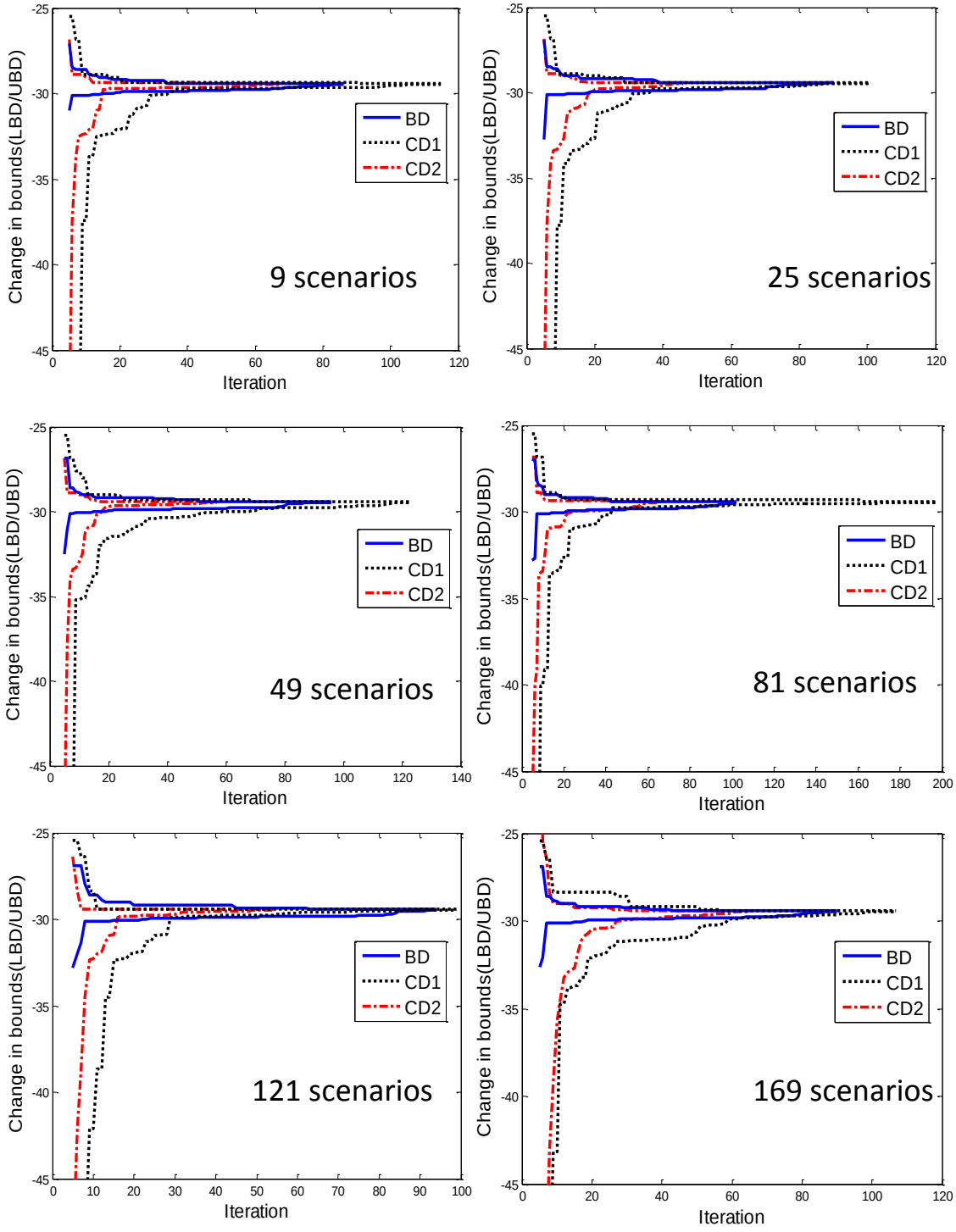


Figure 4.7: Comparison of bound evolution in different decomposition methods (Risk neutral stochastic program)

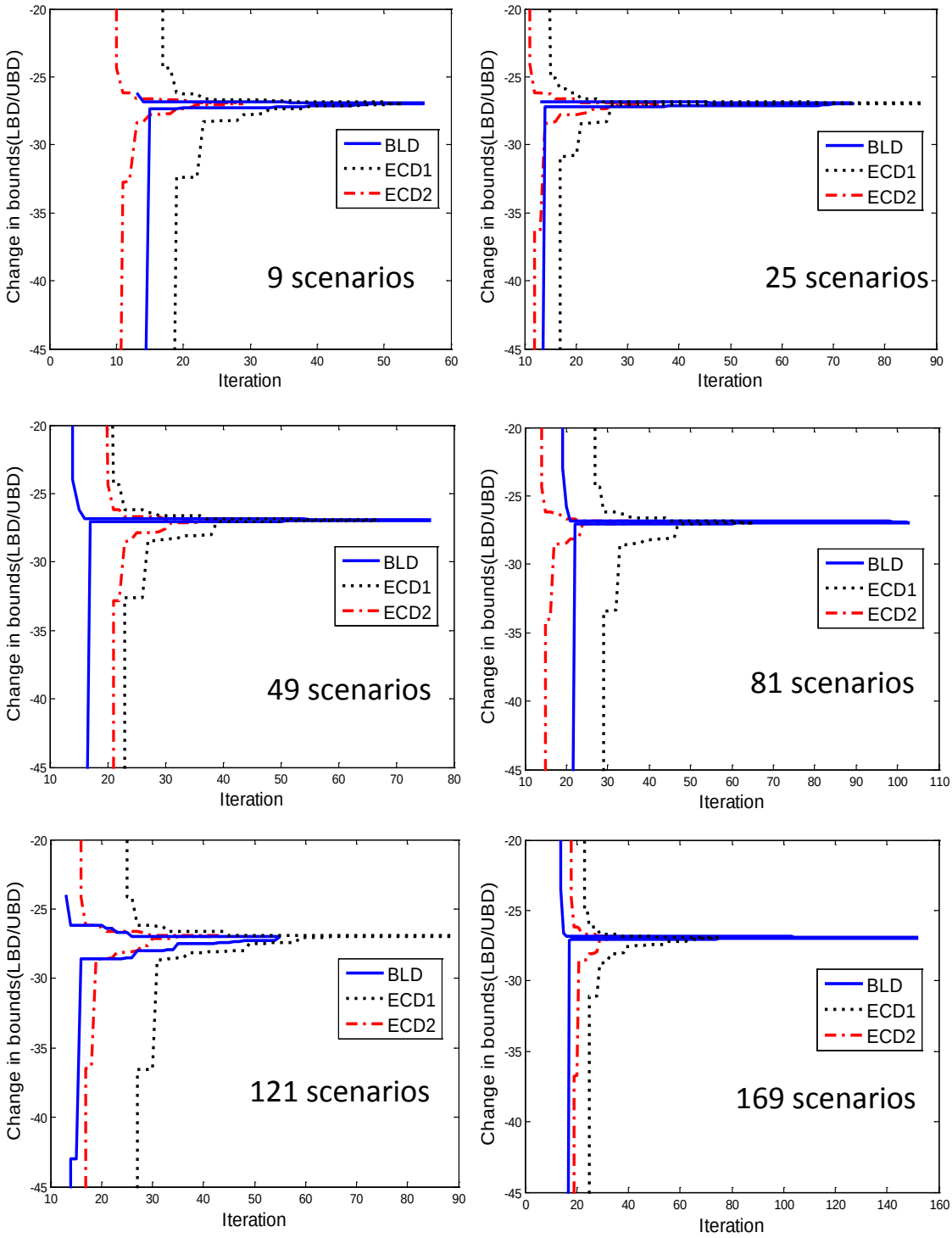


Figure 4.8: Comparison of bound evolution in different decomposition methods (Risk averse stochastic program)

can be saved using CD2. Additionally, CD2 performs significantly better than the monolith for the large scenario case, where the monolith could not return a solution within a day.

The extended cross decomposition showed good performance over bilevel decomposition and the monolith, for Problem (CVaR-SP). As we have seen from the results, ECD2 can achieve 10 times reduction in wall time compared to BD, and over an order of magnitude reduction in wall time compared to the monolith. In the future, we would like to extend the current paradigm to handle risk averse nonlinear programs. In this case, generalized Benders decomposition [35], rather than Benders decomposition, is applicable. Another future direction will be to extend the approach to multiperiod and multistage programming problems.

Table 4.1: Results for risk neutral stochastic program - Monolith (time in sec)

Number of scenarios	9	25	49	81	121	169
Number of iterations	71183	246978	637861	955705	1335655	-
Number of nodes explored	49	122	74	61	51	-
Optimal obj. (Million $)	-29.42	-29.43	-29.43	-29.43	-29.43	-
Wall time	144	2410	9892	32465	67264	86421‡

Table 4.2: Results for risk neutral stochastic program - BD (time in sec)

Number of scenarios	9	25	49	81	121	169
Number of iterations	86	90	96	102	94	90
Optimal obj. (Million $)	-29.43	-29.42	-29.42	-29.43	-29.42	-29.42
Time for primal problem	247	814	1770	3130	4425	6153
Time for BD relaxed MP	4.6	4.52	3.5	2.97	3.3	2.8
Total solver time	251	819	1773	3133	4428	6156
Wall time	352	968	2124	3929	5664	8245

Table 4.3: Results for risk neutral stochastic program - CD1 (time in sec)

Number of scenarios	9	25	49	81	121	169
Num. of iterations	115	101	123	197	99	107
Num. of DWD iterations	58	51	62	99	50	54
Num. of BD iterations.	57	50	61	98	49	53
Optimal obj. (Million $)	-29.43	-29.42	-29.42	-29.42	-29.43	-29.43
Time for pricing problems	65.5	183	417	1223	1073	1572
Time for DW restricted MP	4.7	9.7	33.8	139	53.6	83.8
Time for primal problems	160.6	429	1056	2914	2343	3348
Time for BD relaxed MP	3.4	2.2	4.7	3.8	2.2	2.2
Total solver time	234	829	1526	4297	3472	5007
Wall time	342	795	1968	6697	4994	7752

Table 4.4: Results for risk neutral stochastic program - CD2 (time in sec)

Number of scenarios	9	25	49	81	121	169
Num. of CD iterations	67	46	57	68	54	65
Num. of DWD iterations	3	3	3	3	3	4
Num. of BD iterations	64	43	54	65	51	61
Optimal obj. (Million $)	-29.43	-29.43	-29.42	-29.42	-29.43	-29.42
Time for pricing problem	2.8	9.6	15.5	32.5	46.7	70.9
Time for DW restricted MP	2.4	2.5	4.8	10.6	9.6	17.7
Time for primal problem	185	600	994	1962	2466	3740
Time for BD relaxed MP	1.87	1.2	1.4	1.6	1.3	1.8
Total solver time	192	408	897	2006	2523	3831
Wall time	282	522	1328	2897	3916	7022

‡: Resource limit of 24 hours for MIP solution reached

Table 4.5: Results for risk averse stochastic program - Monolith (time in sec)

Number of scenarios	9	25	49	81	121	169
Number of iterations	37930	187070	279729	581145	780983	908994
Number of nodes explored	15	20	20	21	13	22
Optimal obj. (Million $)	-26.96	-26.96	-26.96	-26.96	-26.96	-26.96
Wall time	118	2563	4971	37851	55117	111677‡

Table 4.6: Results for risk averse stochastic program - BLD (time in sec)

Num. of scenarios	9	25	49	81	121	169
Num. of upper BD iterations	56	74	76	103	55	152
Num. of lower DWD iterations	245	310	364	491	236	741
Optimal obj. (Million $)	-26.96	-26.94	-26.96	-26.96	-26.96	-26.96
Time for pricing problems	232	986	2156	5132	2689	18006
Time for DW restricted MP	3.4	8.21	18.9	43.5	37.9	154
Time for BD relaxed MP	1.8	2.41	1.45	1.83	0.63	3.4
Total solver time	238	996	2176	5177	3728	18164
Wall time	392	1320	2991	8224	5012	40184

Table 4.7: Results for risk averse stochastic program - ECD1 (time in sec)

Number of scenarios	9	25	49	81	121	169
Num. of upper BD iterations	53	87	67	65	89	75
Num. of upper DWD iterations	27	44	34	33	45	38
Num. of lower DWD iterations	109	186	160	140	189	154
Num. of pricing prob. solved #	136	230	194	173	234	192
Optimal obj. (Million $)	-26.94	-26.95	-26.96	-26.96	-26.96	-26.93
Time for pricing probs.	123	707	1114	1738	3924	4501
Time for DW restricted MP	6.1	49.7	73.4	83.3	335.6	271
Time for BD relaxed MP	0.5	1.6	0.57	0.45	0.79	0.6
Total solver time	130	759	1188	1840	4260	4774
Wall time	205	1026	1543	2368	5790	6336

Table 4.8: Results for risk averse stochastic program - ECD2 (time in sec)

Number of scenarios	9	25	49	81	121	169
Num. of upper BD iterations	29	36	42	52	43	36
Num. of upper DWD iterations	4	4	4	4	4	4
Num. of lower DWD iterations	108	137	157	216	166	133
Num. of pricing probs. solved #	112	141	161	220	170	137
Optimal obj. (Million $)	-26.96	-26.94	-26.96	-26.94	-26.94	-26.96
Time for pricing probs.	103	425	813.1	2103.3	2611	2669
Time for DW restricted MP	6.7	29.3	55.5	181.4	159	116
Time for BD relaxed MP	0.3	0.48	0.5	1.2	0.58	0.4
Total solver time	111	455	869	2302	2771	2786
Wall time	187	632	1171	3115	3783	3924

#: Total number of IPP, DWPP, DWFP, DWPP-R and DWFP-R solved
‡: CPLEX found an MIP solution after 18 hours but could not solve final LP within 24 hours

Chapter 5

A Joint Decomposition Method for Global Optimization of Multiscenario Mixed-integer Nonlinear Nonconvex Programs *

5.1 Introduction

Global optimization is a field of mathematical programming devoted to obtaining global optimal solutions; and it has over the years found enormous applications in Process Systems Engineering (PSE). Mixed-integer nonlinear programs are global optimization problems where some decision variables are integer while others are continuous. Discrete decisions and nonconvex nonlinearities introduce combinatorial behavior for such problems [137] [73]. Various applications of mixed-integer nonlinear programming for PSE systems include natural gas network design and operation [117], gasoline blending and scheduling problems [79], expansion of chemical processes

*This chapter has been submitted for publication as Ogbe E, Li X, Joint decomposition method for multiscenario mixed-integer nonlinear nonconvex programs, Journal of Global Optimization. The equations, assumptions, propositions, theorems, symbols and notations defined in this chapter are self-contained.

[119], reliable design of software [138] [139], pump network problem [140] [17], chemical process design synthesis [141], planning of facility investments for electric power generation [142], etc.

In most PSE problems, there are factors or parameters in the model that are usually not known with certainty. One way to explicitly characterize uncertainty is using the two-stage stochastic programming approach [21] [18]. Using the classical scenario based two-stage stochastic programming formulation to model uncertainty, we obtain the following multiscenario problem [43] [21]:

$$\begin{aligned} \min_{\substack{x_0 \\ z_1,\ldots,z_s}} \ & \sum_{\omega=1}^{s} [f_{0,\omega}(x_0) + f_\omega(z_\omega)] \\ \text{s.t.} \ & g_{0,\omega}(x_0) + g_\omega(z_\omega) \leq 0, \quad \forall \omega \in \{1, ..., s\}, \\ & z_\omega \in Z_\omega, \quad \forall \omega \in \{1, ..., s\}, \\ & x_0 \in X_0, \end{aligned} \tag{P0}$$

where functions $f_{0,\omega} : X_0 \to \mathbb{R}$, $f_\omega : Z_\omega \to \mathbb{R}$, $g_{0,\omega} : X_0 \to \mathbb{R}^m$, $g_\omega : Z_\omega \to \mathbb{R}^m$. Elements in sets X_0 and Z_ω can include continuous and/or integer variables. The first stage decision variable is x_0 and is made before the uncertainty is realized. The second stage decision $z_1, ..., z_s$ is made after the uncertainty is revealed. The associated cost for x_0 is $f_{0,\omega}$ and that for z_ω is f_ω for every scenario ω. When at least one set or function in the problem is nonconvex, Problem (P0) is a nonconvex mixed-integer nonlinear program (MINLP), or a nonconvex nonlinear program (NLP) if no integer variables are involved. To fully capture the uncertainty effect in Problem (P0), usually a large number of scenarios need to be considered. This implies that Problem (P0)

is potentially a large-scale nonconvex MINLP, which is typically very challenging to solve.

Based on ideas from branch-and-bound for mixed-integer programing, branch-and-bound method has been employed to globally solve nonconvex MINLPs [143] [144] [73]. The method entails systematically generating lower and upper bounds of the optimal objective function value over subdomain of the search space. To generate lower bounds, McCommick developed tighter convex relaxations called *McCommick relaxations* [70]. Other ways to generate lower bounds is through the use of *Lagrangian relaxation* (or *decomposition*) [38][56][88]. In Lagrangian decomposition, the linking constraints in Problem (P0) are dualized and the resultant subproblem, which provides a lower bound to Problem (P0), is solved. Several ways of generating multipliers for the Lagrangian subproblem exist; including subgradient methods [106], cutting plane methods [38], and the Dantzig-Wolfe master problem (also known the restricted Lagrangian master problem) [82] [126]. Branch-and-bound alone have been successful mostly for small to medium sized problems.

One idea to improve branch-and-bound based strategies is incorporation of domain reduction techniques. Domain reduction entails eliminating portions of the feasible domain using feasibility and optimality information. Bound tightening or contraction [145], range reduction [146] and generation of cutting planes [147] are different domain reduction strategies that have been successful in solving nonconvex problems [139]. In bound contraction, the variable bounds are shrunk at every iteration by solving bound contraction subproblems [145]. In range reduction, the bounds on the variables are shrunk based on simple calculations using Lagrange multiplier information [146]. For cutting planes generation, Lagrangian relaxation information provides

cuts that is used to cut-off portion of the feasible domain that does not contain the global optimum [78]. Current state-of-the-art commercial deterministic global optimization solvers embody branch-and-bound and enhancements such as tighter convex relaxations and domain reduction techniques, such as the Branch-And-Reduce Optimization Navigator (BARON) [73] and Algorithms for coNTinuous/Integer Global Optimization of Nonlinear Equations (ANTIGONE) [148]. They generally provide rigorous frameworks for global solutions to Problem (P0). However, because these methods are based on branch-and-bound search, the search tree can become prohibitively large when the size of the problem becomes large.

Multiscenario problems, such as Problem (P0), have special structure; they can be solved efficiently using decomposition methods. *Benders decomposition* (BD) [34] (known as L-shaped method in the stochastic programming literature [43] [21]) is one class of decomposition methods applied for mixed-integer linear programs. Geoffrion [35] generalized BD into *Generalized Benders Decomposition* (GBD), for solving convex MINLPs. Li et al. developed a further extension, called *Nonconvex Generalized Benders Decomposition* [79], for solving nonconvex MINLPs, but this method can guarantee finite termination with an optimal solution only if the linking variable is fully integer. *Outer Approximation* (OA) is another decomposition technique for solving nonconvex MINLP, with nonlinearities in the second stage [36] [149]. This method solves a sequence of upper bounding continuous NLP subproblems and a sequence of lower bounding MILP subproblems until the bounds converge to an optimum. Karuppiah and Grossmann applied Lagrangian decomposition based scheme to solve Problem (P0) [25]. In order to guarantee convergence to a global optimum, explicit branch-and-bound of linking variables are needed. They also presented bound

contraction as an optional scheme in their Lagrangian-based branch-and-bound strategy. A more recent algorithm combining NGBD and Lagrangian decomposition has been proposed by Kannan and Barton [81], and this algorithm also requires explicit branch-and-bound for convergence.

Efforts have been taken to achieve better computational efficiency by combining classical decomposition methods. In 1983, Van Roy proposed a *cross decomposition* method that combines Lagrangian decomposition and Benders decomposition [82] to solve MILP problems which do not have second stage integer variables. Since then, a number of extensions and variants of cross decomposition has been developed [83] [84] [102] [150] [54] [126]. All of these methods require that no nonconvexity comes from second stage variables as otherwise finite convergence cannot be guaranteed.

In this chapter, we develop a new decomposition method for global optimization of Problem (P0), without the need for explicit branch-and-bound. The user of the method does not have to select a heuristic that is normally required by a branch-and-bound search. This is a practical advantage, especially considering that a branch-and-bound heuristic cannot fully exploit the problem structure. The method was inspired by cross decomposition, and it follows a similar algorithm design philosophy, combining primarily generalized Benders decomposition and Lagrangian decomposition. However, its decomposition procedure is rather different in many details due to the nonconvexity it has to deal with, so we do not call it cross decomposition, but a new name *joint decomposition*.

The remaining part of the article is organized as follows. In section 5.2, we give a brief introduction to generalized Benders decomposition and Lagrangian decomposition, using a reformulation of Problem (P0). Then in section 5.3, we present the

basic joint decomposition algorithm and the convergence proof. Section 5.4 discusses enhancements to the basic joint decomposition algorithm, including domain reduction and use of extra convex relaxation subproblems. The joint decomposition methods are tested for two case study problems adapted from the literature, and the simulation results shown in section 5.5 demonstrate the effectiveness and the computational advantages of the methods. The article ends with concluding remarks in section 5.6.

5.2 Problem reformulation and classical decomposition methods

In order to bring up the joint decomposition idea, we reformulate Problem (P0) and briefly discuss how the reformulated problem can be solved via classical GBD and LD methods. We first separate the convex part and the nonconvex part of the problem. Specifically, let $z_\omega = (z_{c,\omega}, z_{nc,\omega})$, where $z_{c,\omega}$ includes *convex variables* that are only involved in convex functions and sets and $z_{nc,\omega}$ includes *nonconvex variables* that are involved in at least one nonconvex function or nonconvex set. In addition, we introduce duplicate variables $x_1, ..., x_s$ for variable x_0, to express the relation among all scenarios using *nonanticipativity constraints* (NACs) [88][89] [78]. We then rewrite Problem (P0) as the following:

$$
\begin{aligned}
&\min_{\substack{x_0,z_{0,1},\ldots,z_{0,s}\\ z_{c,1},\ldots,z_{c,s}\\ z_{nc,1},\ldots,z_{nc,s}}} \sum_{\omega=1}^{s}[f_{0,\omega}(z_{0,\omega}) + f_{c,\omega}(z_{c,\omega}) + f_{nc,\omega}(z_{nc,\omega})] \\
&\text{s.t.} \quad x_0 = z_{0,\omega}, \quad \forall \omega \in \{1,...,s\}, \\
&\qquad g_{0,\omega}(z_{0,\omega}) + g_{c,\omega}(z_{c,\omega}) + g_{nc,\omega}(z_{nc,\omega}) \leq 0, \quad \forall \omega \in \{1,...,s\}, \\
&\qquad x_0 \in X_0, \\
&\qquad z_{0,\omega} \in \hat{X}_0,\ z_{c,\omega} \in Z_{c,\omega},\ z_{nc,\omega} \in Z_{nc,\omega}, \quad \forall \omega \in \{1,...,s\},
\end{aligned}
\tag{P1}
$$

where x_0 is called *linking variable* as it links different scenarios. Set $X_0 \subset \mathbb{R}^{n_0}$ can be either convex or nonconvex, set $Z_{c,\omega} \subset \mathbb{R}^{n_c}$ is convex, set $Z_{nc,\omega} \subset \mathbb{R}^{n_{nc}}$ is nonconvex. Functions $f_{c,\omega} : Z_{c,\omega} \to \mathbb{R}$, $g_{c,\omega} : Z_{c,\omega} \to \mathbb{R}^{m_c}$ are convex, functions $f_{nc,\omega} : Z_{nc,\omega} \to \mathbb{R}$, and $g_{nc,\omega} : Z_{nc,\omega} \to \mathbb{R}^{m_{nc}}$ include only nonconvex variables (but they themselves may be either convex or nonconvex). Set $\hat{X}_0 \in \mathbb{R}^{n_0}$ is a convex relaxation of X_0. The restriction $z_{0,\omega} \in \hat{X}_0$ is actually redundant with the presence of NAC; however, it tightens the problem when the NACs are dualized.

Note that in order to generate a convex relaxation of X_0, extra variables may be introduced [75], so the dimension of the relaxation may be larger than that of X_0. Here $\hat{X}_0$ can be understood as the projection of the relaxation set on the $\mathbb{R}^{n_0}$ space. For simplicity of notation, in this chapter we always express a convex relaxation (of a set or a function) on the original variable space and do not explicitly show the extra variables needed for constructing the relaxation.

For convenience of subsequent discussion, we further rewrite Problem (P1) in the following form:

$$
\begin{aligned}
\min_{\substack{x_0,x_1,\ldots,x_s \\ y_1,\ldots,y_s}} \quad & \sum_{\omega=1}^{s} c_\omega^T x_\omega \\
\text{s.t.} \quad & x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1,...,s\}, \\
& A_\omega x_\omega + B_\omega y_\omega \leq 0, \quad \forall \omega \in \{1,...,s\}, \\
& x_0 \in X_0, \\
& x_\omega \in X_\omega,\ y_\omega \in Y_\omega, \quad \forall \omega \in \{1,...,s\},
\end{aligned} \tag{P}
$$

where $x_\omega \in \mathbb{R}^{n_x}$, $y_\omega \in \mathbb{R}^{n_y}$, X_ω is convex, Y_ω is nonconvex, and $H_\omega \in \mathbb{R}^{n_0} \times \mathbb{R}^{n_x}$ selects from x_ω the duplicated x_0 for scenario ω. The derivation for reformulation (P) is shown in Appendix C.1.

Problem (P) is difficult to solve because of the presence of linking variables x_0 and nonconvex variables y_ω. The classical GBD method can be used to solve Problem (P) by treating x_0 and y_ω as *complicating variables*, while the LD method can be used to solve Problem (P) by dualizing NACs so that x_0 no longer links different scenarios. In the next two subsections we briefly introduce GBD and LD for Problem (P), and we make the following assumptions for Problem (P) for convenience of discussion.

Assumption 5.1. *X_0, X_ω and Y_ω for all $\omega \in \{1,...,s\}$ are non-empty and compact.*

Assumption 5.2. *After fixing $(x_0, y_1, \cdots, y_s)$ to any point in $X_0 \times Y_1 \times \cdots \times Y_s$, if Problem (P) is feasible, it satisfies Slater condition.*

Assumption 5.1 is a mild assumption, as for most real-world applications, the variables are naturally bounded and the functions involved are continuous. If a discontinuous function is involved, it can usually be expressed by continuous functions

and extra integer variables. Assumption 5.2 ensures strong duality of convex subproblems that is required for GBD. If this assumption is not satisfied for a problem, we can treat the convex variables that fail the Slater condition to be complicating variables, so that after fixing all complicating variables the Slater condition is satisfied.

5.2.1 Generalized Benders decomposition

At each GBD iteration l, fixing the complicating variables $x_0 = x_0^{(l)}$, $y_\omega = y_\omega^{(l)}$ ($\forall \omega \in \{1, ..., s\}$) results in an upper bounding problem that can be decomposed into the following Benders primal subproblem for each scenario ω:

$$\begin{aligned} obj_{\mathrm{BPP}_\omega^{(l)}} = \min_{x_\omega} \quad & c_\omega^T x_\omega \\ \text{s.t.} \quad & x_0^{(l)} = H_\omega x_\omega, \\ & A_\omega x_\omega + B_\omega y_\omega^{(l)} \leq 0, \\ & x_\omega \in X_\omega, \end{aligned} \tag{$\mathrm{BPP}_\omega^{(l)}$}$$

$obj_{\mathrm{BPP}_\omega^l}$ is the optimal objective value of ($\mathrm{BPP}_\omega^{(l)}$). For convenience, we indicate the optimal objective value of a problem in the above way for all subproblems discussed in this chapter. Obviously, $\sum_{\omega=1}^{s} obj_{\mathrm{BPP}_\omega^{(l)}}$ represents an upper bound for Problem (P). If ($\mathrm{BPP}_\omega^{(l)}$) is infeasible for one scenario, then solve the following Benders feasibility

subproblem for each scenario ω:

$$
\begin{aligned}
obj_{\mathrm{BFP}_\omega^{(l)}} = \min_{x_\omega, z_{1,\omega}^+, z_{1,\omega}^-, z_{2,\omega}} \quad & ||z_{1,\omega}^+|| + ||z_{1,\omega}^-|| + ||z_{2,\omega}|| \\
\text{s.t.} \quad & x_0^{(l)} = H_\omega x_\omega + z_{1,\omega}^+ - z_{1,\omega}^-, \\
& A_\omega x_\omega + B_\omega y_\omega^{(l)} \leq z_{2,\omega}, \\
& x_\omega \in X_\omega, \quad z_{1,\omega}^+, z_{1,\omega}^-, z_{2,\omega} \geq 0,
\end{aligned}
\tag{$\mathrm{BFP}_\omega^{(l)}$}
$$

where $z_{1,\omega}^+$, $z_{1,\omega}^-$, and $z_{2,\omega}$ are slack variables.

At the same iteration, the following Benders relaxed master problem is solved to yield a lower bound for Problem (P):

$$
\begin{aligned}
\min_{\substack{x_0, \eta_0, \eta_1, ..., \eta_s \\ y_1, ..., y_s}} \quad & \eta_0 \\
\text{s.t.} \quad & \eta_0 \geq \sum_{\omega=1}^{s} \eta_\omega \\
& \eta_\omega \geq obj_{\mathrm{BPP}_\omega^{(j)}} + (\lambda_\omega{}^{(j)})^{\mathrm{T}} B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^{\mathrm{T}} \left(x_0 - x_0^{(j)} \right), \\
& \qquad\qquad \forall \omega \in \{1, ..., s\}, \quad \forall j \in T^{(l)}, \\
& 0 \geq obj_{\mathrm{BFP}_\omega^{(j)}} + (\lambda_\omega{}^{(j)})^{\mathrm{T}} B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^{\mathrm{T}} \left(x_0 - x_0^{(j)} \right), \\
& \qquad\qquad \forall \omega \in \{1, ..., s\}, \quad \forall j \in S^{(l)}, \\
& x_0 \in X_0, \\
& y_\omega \in Y_\omega, \quad \forall \omega \in \{1, ..., s\},
\end{aligned}
\tag{$\mathrm{BRMP}^{(l)}$}
$$

where $\mu_\omega^{(l)}$ includes Lagrange multipliers for the first group of constraints in Problem ($\mathrm{BPP}_\omega^{(l)}$) or ($\mathrm{BFP}_\omega^{(l)}$), and $\lambda_\omega^{(l)}$ includes Lagrange multipliers for the second group of constraints in Problem ($\mathrm{BPP}_\omega^{(l)}$) or ($\mathrm{BFP}_\omega^{(l)}$). Set $T^{(l)}$ includes indices of Benders

iterations at which only ($\text{BPP}_\omega^{(l)}$) is solved, and set $S^{(l)}$ includes indices of Benders iterations at which ($\text{BFP}_\omega^{(l)}$) is solved. Note that Problem ($\text{BRMP}^{(l)}$) is used in the multicut BD or GBD, which is different from the one used in the classical single cut BD or GBD. The multicut version of the Benders master problem is known to be tighter than the single cut version [46] [151], so it is considered in this chapter.

Remark 5.1. *The finite convergence property of GBD is stated and proved in [35]. In Section 3, we will provide more details in the context of our new decomposition method.*

Remark 5.2. *For* (P)*, the relaxed master problem* ($\text{BRMP}^{(l)}$) *can still be very difficult as its size grows with the number of scenarios. However, if most variables in* (P) *are convex variables, the size of* ($\text{BRMP}^{(l)}$) *is much smaller than that of* (P)*, and therefore* ($\text{BRMP}^{(l)}$) *is much easier to solve than* (P)*.*

5.2.2 Lagrangian decomposition

We start discussing LD from the Lagrangian dual of Problem (P) that is constructed by dualizing the NACs of the problem:

$$obj_{\text{DP}} = \max_{\pi_1,\cdots,\pi_s \geq 0} obj_{\text{LS}}(\pi_1, \cdots, \pi_s), \tag{DP}$$

where $obj_{\mathrm{LS}}(\pi_1, \cdots, \pi_s)$ is the optimal objective value of the following Lagrangian subproblem with given $(\pi_1, \cdots, \pi_s)$:

$$\begin{aligned} \min_{\substack{x_0,x_1,\ldots,x_s \\ y_1,\ldots,y_s}} \ & \sum_{\omega=1}^{s} [c_\omega^T x_\omega + \pi_\omega^T (x_0 - H_\omega x_\omega)] \\ \text{s.t.} \quad & A_\omega x_\omega + B_\omega y_\omega \le 0, \quad \forall \omega \in \{1, ..., s\}, \\ & x_0 \in X_0, \\ & x_\omega \in X_\omega, y_\omega \in Y_\omega, \quad \forall \omega \in \{1, ..., s\}. \end{aligned} \tag{LS($\pi_1, \cdots, \pi_s$)}$$

Due to weak duality, Problem (DP) or any Lagrangian subproblem is a lower bounding problem for Problem (P). Typically, the LD method is incorporated in a branch-and-bound framework that only needs to branch on linking variables x_0 to guarantee convergence to an ϵ-optimal solution. At each branch-and-bound node or LD iteration k, a set of multipliers $(\pi_1^k, \cdots, \pi_s^k)$ are selected to construct a Lagrangian subproblem for (DP), and this subproblem can be naturally decomposed into $s+1$ subproblems, i.e.,

$$\begin{aligned} obj_{\mathrm{LS}_0^k} = \min_{x_0} \ & \sum_{\omega=1}^{s} (\pi_\omega^k)^T x_0 \\ \text{s.t} \quad & x_0 \in X_0, \end{aligned} \tag{LS$_0^k$}$$

and

$$\begin{aligned} \min_{x_\omega, y_\omega} \ & c_\omega^T x_\omega - (\pi_\omega^k)^T H_\omega x_\omega \\ \text{s.t.} \quad & A_\omega x_\omega + B_\omega y_\omega \le 0, \\ & x_\omega \in X_\omega, \quad y_\omega \in Y_\omega, \end{aligned} \tag{LS$_\omega^k$}$$

for all $\omega \in \{1, \cdots, s\}$. Let obj_{LS^k} be the optimal objective value of the Lagrangian subproblem, then $obj_{\mathrm{LS}^k} = \sum_{\omega=1}^{s} obj_{LS_\omega^k} + obj_{LS_0{}^k}$. Clearly, $obj_{\mathrm{LS}^k} \leq obj_{\mathrm{DP}}$ always holds. If $(\pi_1^k, \cdots, \pi_s^k)$ happens to be an optimal solution of (DP), then $obj_{\mathrm{LS}^k} = obj_{\mathrm{DP}}$.

In all LD methods, the lower bounds are generated by solving the Lagrangian subproblems. However, the upper bounds and the multipliers for constructing Lagrangian subproblems may be generated in different ways. In this subsection, we introduce one approach to generate the upper bounds and the multipliers. In this approach, at each iteration k, a primal problem is constructed via fixing $x_0 = x_0^k$, and this problem can be separated into s primal subproblem in the following form:

$$\begin{aligned} obj_{\mathrm{PP}_\omega^k} = \min_{x_\omega, y_\omega} \quad & c_\omega^T x_\omega \\ \text{s.t.} \quad & x_0^k = H_\omega x_\omega, \\ & A_\omega x_\omega + B_\omega y_\omega \leq 0, \\ & x_\omega \in X_\omega, \quad y_\omega \in Y_\omega, \end{aligned} \tag{PP_ω^k}$$

Let obj_{PP^k} be the optimal objective value of the primal problem, then $obj_{PP^k} = \sum_{\omega=1}^{s} obj_{\mathrm{PP}_\omega^k}$.

For generation of multipliers, we take the idea from Dantzig-Wolfe decomposition, which is essentially a special LD method. Consider the convex hull of nonconvex set Y_ω:

$$\tilde{Y}_\omega = \{y_\omega \in \mathbb{R}^{n_y} : y_\omega = \sum_{i \in I} \theta_\omega^{[i]} y_\omega^{[i]}, \ \sum_{i \in I} \theta_\omega^{[i]} = 1, \ \theta_\omega^{[i]} \geq 0, \forall i \in I\},$$

where $y_\omega^{[i]}$ denotes a point in Y_ω that is indexed by i. The index set I may need to be an infinite set for $\tilde{Y}_\omega$ being the convex hull. Replace Y_ω with its convex hull for all ω in (P), then we get the following Dantzig-wolfe master problem, or called primal

master problem in this chapter:

$$
\begin{aligned}
\min_{\substack{x_0,\theta_1^{[i]},\ldots,\theta_s^{[i]} \\ x_1,\ldots,x_s}} \quad & \sum_{\omega=1}^{s} c_\omega^T x_\omega \\
\text{s.t.} \quad & x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1,...,s\}, \\
& A_\omega x_\omega + B_\omega \sum_{i\in I} \theta_\omega^{[i]} y_\omega^{[i]} \leq 0, \quad \forall \omega \in \{1,...,s\}, \\
& \sum_{i\in I} \theta_\omega^{[i]} = 1, \quad \theta_\omega^{[i]} \geq 0, \quad \forall i \in I, \quad \forall \omega \in \{1,...,s\}, \\
& x_0 \in X_0, \\
& x_\omega \in X_\omega, \quad \forall \omega \in \{1,...,s\}
\end{aligned}
\tag{PMP}
$$

Clearly, Problem (PMP) is a relaxation of Problem (P), and it is either fully convex or partially convex (as set X_0 can still be nonconvex). At LD iteration k, the following restriction of (PMP) can be solved:

$$
\begin{aligned}
\min_{\substack{x_0,\theta_1^{[i]},\ldots,\theta_s^{[i]} \\ x_1,\ldots,x_s}} \quad & \sum_{\omega=1}^{s} c_\omega^T x_\omega \\
\text{s.t.} \quad & x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1,...,s\}, \\
& A_\omega x_\omega + B_\omega \sum_{i\in I^k} \theta_\omega^{[i]} y_\omega^{[i]} \leq 0, \quad \forall \omega \in \{1,...,s\}, \\
& \sum_{i\in I^k} \theta_\omega^{[i]} = 1, \quad \theta_\omega^{[i]} \geq 0, \quad \forall i \in I^k, \quad \forall \omega \in \{1,...,s\}, \\
& x_0 \in X_0, \\
& x_\omega \in X_\omega, \quad \forall \omega \in \{1,...,s\},
\end{aligned}
\tag{RPMP^k}
$$

where index set $I^k \subset I$ is finite. I^k may consist of indices of y_ω that are generated

in the previously solved primal problems and Lagrangian subproblems. Problem (RPMP^k) can be solved by a state-of-the-art optimization solver directly or by GBD. The multipliers for the NACs obtained at the solution of (RPMP^k) can be used to construct a Lagrangian subproblem for iteration k.

Actually, we can construct a different Lagrangian dual of Problem (P) by dualizing both the NACs and the second group of constraints in the problem, as what we do for GBD in the last subsection. However, this Lagrangian dual is not as tight as Problem (DP) (as stated by the following proposition), so it is not preferred for a LD method. The following proposition follows from Theorem 3.1 of [88] and its proof is omitted here.

Proposition 5.1. *Consider the following Lagrangian dual of Problem* (P)*:*

$$obj_{\text{DP2}} = \max_{\substack{\mu_1,\cdots,\mu_s\geq 0\\ \lambda_1,\cdots,\lambda_s\geq 0}} obj_{\text{LS2}}(\mu_1,\cdots,\mu_s,\lambda_1,\cdots,\lambda_s), \tag{DP2}$$

where

$$\begin{aligned} obj_{\text{LS2}} = \min_{\substack{x_0,x_1,\ldots,x_s\\ y_1,\ldots,y_s}} & \sum_{\omega=1}^{s}[c_\omega^T x_\omega + \mu_\omega^T(x_0 - H_\omega x_\omega) + \lambda_\omega^T(A_\omega x_\omega + B_\omega y_\omega)] \\ s.t. \quad & x_0 \in X_0, \\ & x_\omega \in X_\omega, y_\omega \in Y_\omega, \quad \forall \omega \in \{1,...,s\}. \end{aligned}$$

The dual gap of (DP) *is no larger than the dual gap of* (DP2)*.*

5.3 The joint decomposition method

5.3.1 Synergizing LD and GBD

In the LD method described in the last section, at each iteration the subproblems to be solved are much easier than the original problem (P), as either the size of the subproblem is independent of number of scenarios, such as (PP_ω^k), (LS_0^k), and (LS_ω^k), or the subproblem is a MILP or convex MINLP that can be solved by existing optimization solvers or by GBD relatively easily, such as (RPMP^k). However, without branching on the linking variables x_0, LD cannot guarantee finding a global solution, and we do not always know how to exploit the problem structure to efficiently branch on x_0 and whether the branching can be efficient enough.

On the other hand, GBD can find a global solution, but it requires solving the non-convex relaxed master problem ($\text{BRMP}^{(l)}$) at each iteration. The size of ($\text{BRMP}^{(l)}$) may be much smaller than the size of (P) if most variables in (P) are convex variables, but ($\text{BRMP}^{(l)}$) can still be difficult to solve, especially considering that it needs to be solved at each iteration and its size grows with the number of iterations.

Therefore, there may be a way to combine LD and GBD, such that we solve as many as possible LD subproblems and Benders primal subproblems ($\text{BPP}_\omega^{(l)}$) (as they are relatively easy to solve), but avoid solving many difficult Benders relaxed master problems ($\text{BRMP}^{(l)}$). This idea is similar to the one that motivates cross decomposition [82], but it leads to very different subproblems and a very different algorithmic procedure. The subproblems are very different, because for problem (P), we prefer dualizing only NACs in LD in order to achieve the smallest possible dual gap (according to Proposition 5.1), but we have to dualize both the NACs and the second group of constraints in GBD. In addition, due to the different nature of the subproblems,

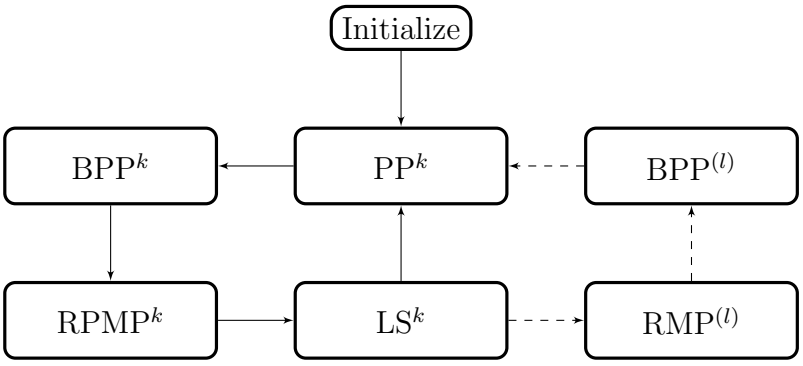
the order in which the subproblems are solved and how often the problems are solved are different. Therefore, we do not name the proposed method cross decomposition, but call it *joint decomposition* (JD).

Figure 5.1 shows the basic framework of JD. Each JD iteration includes one LD iteration part, as indicated by the solid lines, and possibly one GBD iteration, as indicated by the dashed lines. In a JD iteration, the GBD iteration is performed only when the LD iteration improves over the previous LD iteration substantially. The GBD iteration is same to the one described in the last section, except that the relaxed master problem (BRMP$^{(l)}$) includes more valid cuts (which will be described later). The LD iteration is slightly different from the one described in the last section. One difference is that, after solving (PP$^k_\omega$) at LD iteration k, a Benders primal problem (BPP^k) is constructed using x^k_0 (which is used for constructing (PP$^k_\omega$)) and $(y_1, \cdots, y_s)$ (which is from the optimal solution of (PP$^k_\omega$)). The (BPP^k) is solved to generate a Benders cut that can be added to (BRMP$^{(l)}$). The other difference is that (RPMP^k), (LS$^k_0$), (LS$^k_\omega$) (decomposed from (LS^k)) slightly differ from the ones described in the last section, and they will be described later.

Remark 5.3. *The JD method requires that all subproblems can be solved using an existing optimization solver within reasonable time. If this requirement is not met, then JD does not work, or we have to further decompose the difficult subproblems into smaller, solvable subproblems.*

5.3.2 Feasibility issues

According to Assumption 5.1, a subproblem in JD either has a solution or is infeasible. Here we explain how JD handles infeasibility of a subproblem.



RPMP^k: Restricted Primal Master Problem
LS^k: Lagrangian subproblem, decomposed into (LS$_0^k$) and (LS$_\omega^k$) ($\omega = 1, \cdots, s$). RMP$^{(l)}$: Relaxed Master Problem, with extra cuts from LS^k and BPP^k.
BPP$^{(l)}$: Benders Primal Problem, decomposed into (BPP$_\omega^{(l)}$) ($\omega = 1, \cdots, s$).
PP^k: Primal Problem, decomposed into (PP$_\omega^k$) ($\omega = 1, \cdots, s$).
BPP^k: Benders Primal Problem, solved after PP^k is solved.

Figure 5.1: The basic joint decomposition framework

First, if a lower bounding problem (LS^k) or (BRMP$^{(l)}$) is infeasible, then the original problem (P) is infeasible and JD can terminate.

Second, if (BPP^k) or (BPP$^{(l)}$) is infeasible, then JD will solve the corresponding Benders feasibility problem (BFP^k) or (BFP$^{(l)}$) to yield a feasibility cut. If (BFP^k) or (BFP$^{(l)}$) is infeasible, then (P) is infeasible and JD can terminate.

Third, if (PP$_\omega^k$) is infeasible, then JD will solve a feasibility problem that "softens" the second group of constraints: and this problem can be separated into s subproblems

as follows:

$$\begin{aligned} \min_{x_\omega, y_\omega, z_\omega} \quad & ||z_\omega|| \\ \text{s.t.} \quad & x_0^k = H_\omega x_\omega, \\ & A_\omega x_\omega + B_\omega y_\omega \leq z_\omega, \\ & x_\omega \in X_\omega, \quad y_\omega \in Y_\omega, \quad z_\omega \geq 0. \end{aligned} \tag{FP$_\omega^k$}$$

If (FP$_\omega^k$) is infeasible for one scenario ω, then (P) is infeasible and JD can terminate. If (FP$_\omega^k$) is feasible for all scenarios, then JD can construct and solve a feasible Benders feasibility problem (BFP^k) to yield a Benders feasibility cut for (BRMP$^{(l)}$).

Finally, problem (RPMP^k) can actually be infeasible if none of the $(y_1^{[i]}, \cdots, y_s^{[i]})$ in the problem is feasible for the original problem (P). To prevent this infeasibility, we can generate a point $(\hat{y}_1, \cdots, \hat{y}_s)$ that is feasible for (P), by solving the following initial feasibility problem:

$$\begin{aligned} \min_{\substack{x_0, x_1, \cdots, x_s \\ y_1, \cdots, y_s \\ z_1, \cdots, z_\omega}} \quad & \sum_{\omega=1}^{s} ||z_\omega|| \\ \text{s.t.} \quad & x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1, ..., s\}, \\ & A_\omega x_\omega + B_\omega y_\omega \leq z_\omega, \quad \forall \omega \in \{1, ..., s\}, \\ & x_0 \in X_0, \\ & x_\omega \in X_\omega,\ y_\omega \in Y_\omega,\ z_\omega \geq 0, \quad \forall \omega \in \{1, ..., s\}. \end{aligned} \tag{IFP}$$

Problem (IFP) is not naturally decomposable over the scenarios, but it can be solved by JD. When solving (IFP) using JD, the restricted primal master problem (RPMP^k) must have a solution (according to Assumption 5.1).

5.3.3 The tightened subproblems

The relaxed master problem described in section 5.2 can be tightened with the solutions of previously solved subproblems in JD. The tightened problem, called joint decomposition relaxed master problem, can be written as:

$$
\begin{aligned}
\min_{\substack{x_0,\eta_0,\eta_1,\ldots,\eta_s \\ y_1,\ldots,y_s}} \quad & \eta_0 \\
\text{s.t.} \quad & \eta_0 \geq \sum_{\omega=1}^{s} \eta_\omega, \\
& \eta_\omega \geq obj_{\mathrm{BPP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^{\mathrm{T}} B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^{\mathrm{T}} \left(x_0 - x_0^{(j)} \right), \\
& \qquad\qquad \forall \omega \in \{1,...,s\}, \quad \forall j \in T^{(l)}, \\
& 0 \geq obj_{\mathrm{BFP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^{\mathrm{T}} B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^{\mathrm{T}} \left(x_0 - x_0^{(j)} \right), \\
& \qquad\qquad \forall \omega \in \{1,...,s\}, \quad \forall j \in S^{(l)}, \\
& \eta_\omega \geq obj_{\mathrm{BPP}_\omega^{j}} + (\lambda_\omega^{j})^{\mathrm{T}} B_\omega (y_\omega - y_\omega^{j}) + (\mu_\omega^{j})^{\mathrm{T}} \left(x_0 - x_0^{j} \right), \qquad (\mathrm{JRMP}^{(l)}) \\
& \qquad\qquad \forall \omega \in \{1,...,s\}, \quad \forall j \in T^{k}, \\
& 0 \geq obj_{\mathrm{BFP}_\omega^{j}} + (\lambda_\omega^{j})^{\mathrm{T}} B_\omega (y_\omega - y_\omega^{j}) + (\mu_\omega^{j})^{\mathrm{T}} \left(x_0 - x_0^{j} \right), \\
& \qquad\qquad \forall \omega \in \{1,...,s\}, \quad \forall j \in S^{k}, \\
& \eta_0 \leq UBD, \\
& \eta_0 \geq LBD, \\
& \eta_\omega \geq obj_{\mathrm{LS}_\omega^{i}} + (\pi_\omega^{i})^{\mathrm{T}} x_0, \quad \forall \omega \in \{1,...,s\}, \quad \forall i \in R^{k}, \\
& x_0 \in X_0, \quad y_\omega \in Y_\omega, \quad \forall \omega \in \{1,...,s\},
\end{aligned}
$$

where the index set $R^k = \{1, \cdots, k\}$, UBD is the current best upper bound for (P), and LBD is the current best lower bound for (P).

Proposition 5.2. *Problem* (JRMP$^{(l)}$) *is a valid lower bounding problem for Problem* (P).

Proof. Since it is already known that Problem (BRMP$^{(l)}$) is a valid lower bounding problem and UBD and LBD are valid upper and lower bounds, we only need to prove that the cuts from Lagrangian subproblems together with the Benders optimality cuts do not exclude an optimal solution. Let obj_{P} be the optimal objective value of (P), then

$$obj_{\mathrm{P}} = \sum_{\omega=1}^{s} obj_{\mathrm{PP}_\omega}(x_0),$$

where

$$obj_{\mathrm{PP}_\omega}(x_0) = \min\{c_\omega^T x_\omega : x_0 = H_\omega x_\omega,\ A_\omega x_\omega + B_\omega y_\omega \le 0,\ x_\omega \in X_\omega,\ y_\omega \in Y_\omega\}.$$

On the one hand, $\forall \pi_\omega^i, i \in R^k$,

$$\begin{aligned} & obj_{\mathrm{PP}_\omega}(x_0) \\ \ge & \min\{c_\omega^T x_\omega + (\pi_\omega^i)^T(x_0 - H_\omega x_\omega) : A_\omega x_\omega + B_\omega y_\omega \le z_\omega,\ x_\omega \in X_\omega,\ y_\omega \in Y_\omega\} \\ = & obj_{\mathrm{LS}_\omega^i} + (\pi_\omega^i)^{\mathrm{T}} x_0. \end{aligned} \tag{5.1}$$

On the other hand,

$$obj_{\mathrm{PP}_\omega}(x_0) = \min_{y_\omega \in Y_\omega} v_\omega(x_0, y_\omega),$$

where $v_\omega(x_0, y_\omega) = \min\{c_\omega^T x_\omega : x_0 = H_\omega x_\omega,\ A_\omega x_\omega + B_\omega y_\omega \le 0\}$. From weak duality,

$\forall j \in T^{(l)}$,

$$\begin{aligned}
&v_\omega(x_0, y_\omega) \\
\geq &\min\{c_\omega^T x_\omega + (\lambda_\omega^{(j)})^{\mathrm{T}}(A_\omega x_\omega + B_\omega y_\omega) + (\mu_\omega^{(j)})^{\mathrm{T}}(x_0 - H_\omega x_\omega) : x_\omega \in X_\omega\} \\
= &obj_{\mathrm{BPP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^{\mathrm{T}} B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^{\mathrm{T}} \left(x_0 - x_0^{(j)}\right).
\end{aligned}$$

Thefore, $\forall y_\omega \in Y_\omega$,

$$obj_{\mathrm{PP}_\omega}(x_0) \geq obj_{\mathrm{BPP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^{\mathrm{T}} B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^{\mathrm{T}} \left(x_0 - x_0^{(j)}\right). \tag{5.2}$$

Equations (5.1)-(5.2) indicate that the cuts from Lagrangian subproblems together with the Benders optimality cuts do not exclude an optimal solution of (P).

□

For convenience, we call the cuts from the Lagrangian subproblems, Lagrangian cuts. The Benders cuts and the Lagrangian cuts in (JRMP$^{(l)}$) imply that, $\forall i \in R^k$,

$$UBD \geq \eta_0 \geq \sum_{\omega=1}^{s} \eta_\omega \geq \sum_{\omega=1}^{s} obj_{LS_\omega^i} + \sum_{\omega=1}^{s} (\pi_\omega^i)^{\mathrm{T}} x_0.$$

Now we get new constraints

$$UBD \geq \sum_{\omega=1}^{s} obj_{LS_\omega^i} + \sum_{\omega=1}^{s} (\pi_\omega^i)^{\mathrm{T}} x_0, \quad \forall i \in R^k, \tag{*}$$

which only include variable x_0 and do not link different scenarios. This constraint can be used to enhance any subproblems that involves x_0 as variables. Specifically,

problems (LS_0^k), (LS_ω^k), (RPMP^k) can be enhanced as:

$$\begin{aligned}
&\min_{x_\omega, y_\omega} \ c_\omega^T x_\omega - (\pi_\omega^k)^T H_\omega x_\omega \\
&\text{s.t.} \quad A_\omega x_\omega + B_\omega y_\omega \le 0, \\
&\qquad UBD \ge \sum_{\omega=1}^{s} obj_{LS_\omega^i} + \sum_{\omega=1}^{s} (\pi_\omega^i)^{\mathrm{T}} x_0, \quad \forall i \in R^k, \\
&\qquad x_\omega \in X_\omega, y_\omega \in Y_\omega.
\end{aligned} \tag{LS_ω^k}$$

$$\begin{aligned}
&\min_{x_0} \ \sum_{\omega=1}^{s} (\pi_\omega^k)^T x_0 \\
&\text{s.t.} \quad UBD \ge \sum_{\omega=1}^{s} obj_{LS_\omega^i} + \sum_{\omega=1}^{s} (\pi_\omega^i)^{\mathrm{T}} x_0, \quad \forall i \in R^k, \\
&\qquad x_0 \in X_0.
\end{aligned} \tag{LS_0^k}$$

$$\begin{aligned}
&\min_{\substack{x_0, \theta_1^{[i]}, \ldots, \theta_s^{[i]} \\ x_1, \ldots, x_s}} \quad \sum_{\omega=1}^{s} c_\omega^T x_\omega \\
&\text{s.t.} \quad x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1, ..., s\}, \\
&\qquad A_\omega x_\omega + B_\omega \sum_{i \in I^k} \theta_\omega^{[i]} y_\omega^{[i]} \le 0, \quad \forall \omega \in \{1, ..., s\}, \\
&\qquad \sum_{i \in I^k} \theta_\omega^{[i]} = 1, \quad \theta_\omega^{[i]} \ge 0, \quad \forall i \in I^k, \quad \forall \omega \in \{1, ..., s\}, \\
&\qquad UBD \ge \sum_{\omega=1}^{s} obj_{LS_\omega^i} + \sum_{\omega=1}^{s} (\pi_\omega^i)^{\mathrm{T}} x_0, \quad \forall i \in R^k, \\
&\qquad x_0 \in X_0, \quad x_\omega \in X_\omega, \quad \forall \omega \in \{1, ..., s\},
\end{aligned} \tag{RPMP^k}$$

Note that the index set I^k includes indices for all constant points $y_\omega^{[i]}$ in Problem

(RPMP^k), and the constant points $y_\omega^{[i]}$ come from all previously solved PP, FP, LS and JRMP.

5.3.4 The basic joint decomposition algorithm

Table 5.1 shows the basic JD algorithm. As described in section 5.3.1, a JD iteration always include a LD iteration and sometimes a GBD iteration as well. Whether the GBD iteration is performed at JD iteration k depends on whether LD iteration k improves over LD iteration $k-1$ substantially, i.e., whether $obj_{LS^k} \geq obj_{LS^{k-1}} + \epsilon$. This strategy implies the following result.

Proposition 5.3. *The JD algorithm shown in Table 5.1 cannot perform an infinite number of LD iterations between two GBD iterations.*

Proof. The initial point $(x_0^1, y_1^{[1]}, \cdots, y_s^{[1]})$ that are feasible for Problem (P) can lead to a finite upper bound UBD. According to Assumption 5.1, all Lagrangian subproblems are bounded, so between two GBD iterations, the first LD iteration leads to a finite obj_{LS}, and the subsequent LD iterations increase obj_{LS} by at least $\epsilon > 0$ (because otherwise a GBD iteration has to be performed). Therefore, in a finite number LD iterations either obj_{LS} exceeds $UBD - \epsilon$ and the algorithm terminates with an ϵ-optimal solution, or a GBD iteration is performed. This completes the proof. □

Remark 5.4. *If an initial feasible point for Problem* (P) *is not known, the initial feasibility problem* (IFP) *can be solved to get a feasible point for* (P) *or verify that Problem* (P) *is infeasible (when the optimal objective value of Problem* (IFP) *is positive). Note that it is easy to find a feasible point of Problem* (IFP).

In the JD algorithm, we use k to index both a JD iteration and a LD iteration, as every JD iteration includes one LD iteration. We use l (together with '()') to

Table 5.1: The basic joint decomposition algorithm

Initialization

(I.a) Select $x_0^1, y_1^{[1]}, \cdots, y_s^{[1]}$ that are feasible for Problem (P).

(I.b) Give termination tolerance $\epsilon > 0$. Let index sets $T^1 = S^1 = R^1 = \emptyset$, $I^1 = \{1\}$, iteration counter $k = 1$, $i = 1$, $l = 1$, bounds $UBD = +\infty$, $LBD = -\infty$.

LD Iteration

(1.a) Solve Problem (PP_ω^k). If Problem (PP_ω^k) is infeasible, solve Problem (FP_ω^k). Let the solution obtained be (x_ω^k, y_ω^k), and update $i = i+1$, $I^k = I^k \cup \{i\}$, $(y_1^{[i]}, \cdots, y_s^{[i]}) = (y_1^k, \cdots, y_s^k)$.

(1.b) Solve Problem (BPP_ω^k) by fixing $(x_0, y_1, ..., y_s) = (x_0^k, y_1^k, ..., y_s^k)$. If ($\text{BPP}_\omega^k$) is feasible for all ω, generate Benders optimality cuts with the obtained dual solution μ_ω^k and λ_ω^k, and update $T^{k+1} = T^k \cup \{k\}$. If $\sum_{\omega=1}^s obj_{PP_\omega^k} < UBD$, update $UBD = \sum_{\omega=1}^s obj_{PP_\omega^k}$, and incumbent solution $(x_0^*, x_1^*, \cdots, x_s^*, y_1^*, \cdots, y_s^*) = (x_0^k, x_1^k, \cdots, x_s^k, y_1^k, \cdots, y_s^k)$. If Problem ($\text{BPP}_\omega^k$) is infeasible for at least one ω, solve Problem (BFP_ω^k). Generate Benders feasibility cuts with the obtained dual solution μ_ω^k and λ_ω^k, and update $S^{k+1} = S^k \cup \{k\}$.

(1.c) Solve Problem (RPMP^k). Let x_0^k, $\{\theta_\omega^{[i,k]}\}_{i \in I^k, \omega \in \{1,...,s\}}$ be the optimal solution obtained, and $\pi_1^k, ..., \pi_s^k$ be Lagrange multipliers for the NACs.

(1.d) Solve Problems (LS_ω^k) and (LS_0^k), and let the obtained solution be (x_ω^k, y_ω^k), x_0^k. If $obj_{LS^k} = \sum_{\omega=1}^s obj_{LS1_\omega^k} + obj_{LS_0{}^k} > LBD$, update $LBD = obj_{LS^k}$. Generate a Lagrangian cut and update $R^{k+1} = R^k \cup \{k\}$. Update $i = i+1$, $I^{k+1} = I^k \cup \{i\}$, $(y_1^{[i]}, \cdots, y_s^{[i]}) = (y_1^k, \cdots, y_s^k)$.

(1.e) If $UBD \leq LBD + \epsilon$, terminate and return the incumbent solution as an ϵ-optimal solution. If $obj_{LS^k} \geq obj_{LS^{k-1}} + \epsilon$, $k = k+1$, go to step (1.a); otherwise $k = k+1$ and go to step (2.a);

GBD Iteration

(2.a) Solve Problem ($\text{JRMP}^{(l)}$), and let the obtained solution be $(x_0^{(l)}, y_1^{(l)}, ..., y_s^{(1)})$. Update $i = i+1$, $I^{k+1} = I^k \cup \{i\}$, $(y_1^{[i]}, \cdots, y_s^{[i]}) = (y_1^{(l)}, \cdots, y_s^{(l)})$. If $obj_{RMP^{(l)}} > LBD$, update $LBD = obj_{JRMP^{(l)}}$.

(2.b) Solve Problem ($\text{BPP}_\omega^{(l)}$) by fixing $(x_0, y_1, \cdots, y_s) = (x_0^{(l)}, y_1^{(l)}, \cdots, y_s^{(l)})$. If ($\text{BPP}_\omega^{(l)}$) is feasible for all ω, generate Benders optimality cuts with the dual solution μ_ω^k and λ_ω^k, and update $T^{(l+1)} = T^{(l)} \cup \{l\}$. If $\sum_{\omega=1}^s obj_{BPP_\omega^{(l)}} < UBD$, update $UBD = obj_{BPP^{(l)}}$ and the incumbent solution $(x_0^*, x_1^*, \cdots, x_s^*, y_1^*, \cdots, y_s^*) = (x_0^{(l)}, x_1^{(l)}, \cdots, x_s^{(l)}, y_1^{(l)}), \cdots, y_s^{(l)})$. If Problem ($\text{BPP}_\omega^{(l)}$) is infeasible for at least one ω, solve Problem ($\text{BFP}_\omega^{(l)}$). Generate Benders feasibility cuts with the obtained dual solution μ_ω^l and λ_ω^l, and update $S^{(l+1)} = S^{(l)} \cup \{l\}$.

(2.c) If $UBD \leq LBD + \epsilon$, terminate and return the incumbent solution as an ϵ-optimal solution; otherwise $l = l+1$, go to step (1.a).

index a GBD iteration, and usually $l < k$ because not every JD iteration includes one GBD iteration. We use i (together with '[]') to index the columns generated for constructing Problem (RPMP^k). Next, we establish the finite convergence property of the JD algorithm.

Proposition 5.4. *If set X_ω is polyhedral $\forall \omega \in \{1, \cdots, s\}$, the JD algorithm shown in Table 5.1 cannot perform an infinite number of GBD iterations.*

Proof. In this case, the GBD part of the algorithm reduces to BD, and BD is known to have finite termination property [34] [30]. The finite termination property results from:

(a) The Benders master problem (BRMP$^{(l)}$) (and therefore JRMP$^{(l)}$ as well) requires only a finite number of Benders cuts to equal Problem (P), due to linear duality theory;

(b) A same Benders cut cannot be generated twice before the optimality gap is closed.

□

Proposition 5.5. *If $X_0 \times Y_1 \times \cdots \times Y_s$ is a finite discrete set, the JD algorithm shown in Table 5.1 cannot perform an infinite number of GBD iterations.*

Proof. This result comes from the fact that a point in $X_0 \times Y_1 \times \cdots \times Y_s$ cannot be generated twice before the optimality gap is closed. For more details readers can see Theorem 2.4 of [35]. □

Proposition 5.6. *The JD algorithm shown in Table 5.1 cannot include an infinite number of GBD iterations at which the Benders primal problem BPP is feasible.*

Proof. A similar proposition has been proved in the context of GBD in [35] (as Theorem 2.5). The central idea of the proof can be used here for JD.

Suppose the JD algorithm includes an infinite number of GBD iterations at which the Benders primal problem BPP is feasible. Let superscript (n) index these GBD iterations, $\{(\eta_0^{(n)}, x_0^{(n)}, y_1^{(n)}, ..., y_s^{(n)})\}$ be the sequence of optimal solutions of JRMP and $\{(\mu_\omega^{(n)}, \lambda_\omega^{(n)})\}$ be the sequence of dual solutions of BPP. Since $\{\eta_0^{(n)}\}$ is nondecreasing and is bounded from above, so a subsequence of it converges to a finite value, say η_0^*. Due to the compactness of X_0, $Y_1, \cdots, Y_s$, a subsequence of $\{(x_0^{(n)}, y_1^{(n)}, ..., y_s^{(n)})\}$, say, $\{(x_0^{(n_i)}, y_1^{(n_i)}, ..., y_s^{(n_i)})\}$, converges to $(x_0^*, y_1^*, ..., y_s^*) \in X_0 \times Y_1 \times \cdots \times Y_s$. Solving BPP in this subsequence of GBD iterations can be viewed as point-to-set mappings from points in $X_0 \times Y_1 \times \cdots \times Y_s$ to the relevant Lagrange multiplier sets. From Lemma 2.1 of [35] and Assumption 5.2, such a mapping is uniformly bounded in some open neighborhood of the point it maps from. Let such open neighborhood of $(x_0^*, y_1^*, ..., y_s^*)$ be $N(x_0^*, y_1^*, ..., y_s^*)$, then $\exists t$ such that $\forall n_i > t$, $(x_0^{(n_i)}, y_1^{(n_i)}, ..., y_s^{(n_i)}) \in N(x_0^*, y_1^*, ..., y_s^*)$, and then the relevant subsequence of Lagrange multipliers is bounded, which must contain a subsequence converging to $\{\mu_\omega^\star, \lambda_\omega^\star\}$. Therefore, there exists a subsequence of $\{(\eta_0^{(n)}, x_0^{(n)}, y_1^{(n)}, ..., y_s^{(n)}, \mu_\omega^{(n)}, \lambda_\omega^{(n)})\}$, say, $\{(\eta_0^{(m)}, x_0^{(m)}, y_1^{(m)}, ..., y_s^{(m)}, \mu_\omega^{(m)}, \lambda_\omega^{(m)})\}$, which converges to $\{(\eta_0^*, x_0^*, y_1^*, ..., y_s^*, \mu_\omega^*, \lambda_\omega^*)\}$.

Consider any GBD iteration $m > 1$ in this convergent subsequence. Let UBD and LBD be the upper and lower bounds after this GBD iteration, then

$$obj_{BPP^{(m-1)}} \geq UBD,$$

$$LBD \geq \eta^{(m)},$$

and that the JD algorithm does not terminate after GBD iteration m implies

$$UBD > LBD + \epsilon,$$

therefore

$$obj_{BPP^{(m-1)}} > \eta^{(m)} + \epsilon. \tag{5.3}$$

According to how JRMP is constructed,

$$\begin{aligned}\eta^{(m)} \geq & obj_{BPP^{(m-1)}} + \\ & \sum_{\omega=1}^{s} \left[(\lambda_\omega^{(m-1)})^{\mathrm{T}} B_\omega (y_\omega^{(m)} - y_\omega^{(m-1)}) + (\mu_\omega^{(m-1)})^{\mathrm{T}} \left(x_0^{(m)} - x_0^{(m-1)} \right) \right].\end{aligned} \tag{5.4}$$

Equations (5.3) and (5.4) imply that

$$0 > \sum_{\omega=1}^{s} \left[(\lambda_\omega^{(m-1)})^{\mathrm{T}} B_\omega (y_\omega^{(m)} - y_\omega^{(m-1)}) + (\mu_\omega^{(m-1)})^{\mathrm{T}} \left(x_0^{(m)} - x_0^{(m-1)} \right) \right] + \epsilon. \tag{5.5}$$

However, when m is sufficiently large, $y_\omega^{(m)} - y_\omega^{(m-1)}$ and $x_0^{(m)} - x_0^{(m-1)}$ are sufficiently close to 0 while $\mu_\omega^{(m-1)}$ and $\lambda_\omega^{(m-1)}$ are sufficiently close to limit points μ_ω^* and λ_ω^*, so the right-hand-side of Equation (5.5) is a positive value (as $\epsilon > 0$). This contradiction implies that the JD algorithm cannot include an infinite number of GBD iterations at which BPP is feasible.

□

Lemma 2.1 in [35] can be rewritten as the following, in the context of Problem (P).

Lemma 5.1. *Assume that $X_1, ..., X_s$ are compact convex sets and Slater condition is*

satisfied for (P) *for any fixed* $(\bar{x}_0, \bar{y}_1, , ..., \bar{y}_s) \in X_0 \times Y_1 \times ... \times Y_s$. *Then the point-to-set mapping* $\Theta : X_0 \times Y_1 \times ... \times Y_s \to U$ *is uniformly bounded in an open neighborhood of any* $(\bar{x}_0, \bar{y}_1, , ..., \bar{y}_s) \in X_0 \times Y_1 \times ... \times Y_s$. *Here,* U *denotes a set of optimal multipliers sets, and an element in* U *is a set of optimal multipliers of a BPP with a fixed* $(\bar{x}_0, \bar{y}_1, ..., \bar{y}_s)$.

Theorem 5.1. *With an initial feasible point, the JD algorithm shown in Table 5.1 terminates in a finite number of iterations with an* ϵ*-optimal solution, if one the following three conditions is satisfied:*

(a) Set X_ω *is polyhedral* $\forall \omega \in \{1, \cdots, s\}$.

(b) Set $X_0 \times Y_1 \times \cdots \times Y_s$ *is finite discrete.*

(c) There are only a finite number of GBD iterations at which the Benders primal problem BPP is infeasible.

Proof. From Proposition 5.3, the JD algorithm can only include a finite number of LD iterations. From Propositions 5.4 and 5.5, when condition (a) or (b) is satisfied, the JD algorithm can only include a finite number of BD iterations. From Proposition 5.6, the JD algorithm can only have a finite number of GBD iterations at which the Benders primal problem BPP is feasible, and together with condition (c), it implies that the JD algorithm can only include a finite number of BD iterations. Therefore, if one of the three conditions is satisfied, the JD algorithm can only include a finite number LD and BD iterations before termination.

On the other hand, according to Proposition 5.2, the JD algorithm never excludes an optimal solution. This together with the termination criterion ensures that the solution returned is ϵ-optimal. □

Remark 5.5. *Condition (c) in Theorem 5.1 is actually not a very restrictive condition, because we can always "soften" the complicating constraints in Problem* (P) *(i.e., penalize the violation of these constraints in the objective function) so that Problem* ($\text{BPP}_{\omega}^{(l)}$) *is always feasible.*

5.4 Enhancements to joint decomposition

The solution of Problem ($\text{JRMP}^{(l)}$) is the bottle neck of the JD algorithm, even considering that the problem is solved only when necessary. Problem ($\text{JRMP}^{(l)}$) is challenging due to two major reasons. One is that the number of nonconvex variables in Problem ($\text{JRMP}^{(l)}$) is dependent on the number of scenarios, so the size of Problem ($\text{JRMP}^{(l)}$) is large (although smaller than the original problem). The other is that the number of constraints in the problem grows with the JD iteration; in other words, Problem ($\text{JRMP}^{(l)}$) becomes more and more challenging as JD progresses. In this section, we introduce two ways to mitigate the difficulty in solving Problem ($\text{JRMP}^{(l)}$):

1. To solve a convex relaxation of Problem ($\text{JRMP}^{(l)}$) before solving Problem ($\text{JRMP}^{(l)}$). If the solution of the convex relaxation can improve the lower bound, then skip solving Problem ($\text{JRMP}^{(l)}$).

2. To perform domain reduction iteratively in JD in order to keep reducing the ranges of nonconvex and linking variables. This way, the convex relaxation of Problem ($\text{JRMP}^{(l)}$) is progressively tightened and Problem ($\text{JRMP}^{(l)}$) itself does not become much harder as the algorithm progresses.

In addition, domain reduction for the linking and the nonconvex variables can make other nonconvex JD subproblems easier, including Problems (LS_{ω}^{k}) and (PP_{ω}^{k}).

Domain reduction for the linking variables can also tighten the Lagrangian relaxation gap [23]; in extreme cases, the Lagrangian relaxation gap can diminish and there is no need to solve Problem (JRMP$^{(l)}$) in JD to close the optimality gap. Note that we do not perform domain reduction for convex variables, because normally reducing ranges on convex variables do not help much to tighten convex relaxations and ease the solution of nonconvex subproblems.

5.4.1 Convex relaxation and domain reduction

The convex relaxation of Problem (JRMP$^{(l)}$) is a valid lower bounding problem for Problem (JRMP$^{(l)}$) and consequently for Problem (P) as well. It can be written as:

$$
\begin{aligned}
\min_{\substack{x_0,\eta_0,\eta_1,...,\eta_s \\ y_1,...,y_s}} \quad & \eta_0 \\
\text{s.t.} \quad & \eta_0 \geq \sum_{\omega=1}^{s} \eta_\omega, \\
& \eta_\omega \geq obj_{\mathrm{BPP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^{\mathrm{T}} B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^{\mathrm{T}} \left(x_0 - x_0^{(j)} \right), \\
& \qquad\qquad \forall \omega \in \{1,...,s\}, \quad \forall j \in T^{(l)}, \\
& 0 \geq obj_{\mathrm{BFP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^{\mathrm{T}} B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^{\mathrm{T}} \left(x_0 - x_0^{(j)} \right), \\
& \qquad\qquad \forall \omega \in \{1,...,s\}, \quad \forall j \in S^{(l)}, \\
& \eta_\omega \geq obj_{\mathrm{BPP}_\omega^{j}} + (\lambda_\omega^{j})^{\mathrm{T}} B_\omega (y_\omega - y_\omega^{j}) + (\mu_\omega^{j})^{\mathrm{T}} \left(x_0 - x_0^{j} \right), \qquad \text{(JRMPR}^{(l)}\text{)} \\
& \qquad\qquad \forall \omega \in \{1,...,s\}, \quad \forall j \in T^{k}, \\
& 0 \geq obj_{\mathrm{BFP}_\omega^{j}} + (\lambda_\omega^{j})^{\mathrm{T}} B_\omega (y_\omega - y_\omega^{j}) + (\mu_\omega^{j})^{\mathrm{T}} \left(x_0 - x_0^{j} \right), \\
& \qquad\qquad \forall \omega \in \{1,...,s\}, \quad \forall j \in S^{k}, \\
& \eta_0 \leq UBD, \\
& \eta_0 \geq LBD, \\
& \eta_\omega \geq obj_{\mathrm{LS}_\omega^{i}} + (\pi_\omega^{i})^{\mathrm{T}} x_0, \quad \forall \omega \in \{1,...,s\}, \quad \forall i \in R^{k}, \\
& x_0 \in \hat{X}_0, \quad y_\omega \in \hat{Y}_\omega, \quad \forall \omega \in \{1,...,s\}.
\end{aligned}
$$

Here $\hat{X}_0$ and $\hat{Y}_\omega$ denote the convex relaxations of X_0 and Y_ω. Let $obj_{JRMPR^{(l)}}$ be the optimal objective of Problem (JRMPR$^{(l)}$).

Since Problem (JRMPR$^{(l)}$) is also a valid convex relaxation of Problem (P), the

solution of Problem (JRMPR$^{(l)}$) can be exploited to eliminate the parts of variable ranges that cannot include an optimal solution of Problem (P), using marginal based domain reduction method. This method was first proposed in [146] (and it was called range reduction therein). The following proposition lays the foundation of marginal based domain reduction for noconvex variables y_ω in JD, which results directly from Theorem 2 in [146].

Proposition 5.7. *Consider the following bounds on* $y_{\omega,j}$ $(\forall \omega \in \{1, \cdots, s\}, \quad \forall j \in \{1, \cdots, n_y\})$*:*

$$y_{\omega,j} - y^{up}_{\omega,j} \leq 0,$$
$$y^{lo}_{\omega,j} - y_{\omega,j} \leq 0,$$

whose Lagrange multipliers obtained at the solution of Problem (JRMPR$^{(l)}$) *are* $u_{\omega,j}$, $v_{\omega,j}$*. Let* $\mathbb{J}^{(l)}_{1,\omega}$ *include indices of upper bounds whose* $u_{\omega,j}$ *are nonzero, and* $\mathbb{J}^{(2)}_{1,\omega}$ *include indices of lower bounds whose* $v_{\omega,j}$ *are nonzero, then the following constraints do not exclude an optimal solution of* (P)*:*

$$y_{\omega,j} \geq y^{up}_{\omega,j} - \frac{(UBD - obj_{JRMPR^{(l)}})}{u_{\omega,j}}, \quad \forall j \in \mathbb{J}^{(l)}_{1,\omega}, \ \ \forall \omega \in \{1, ..., s\},$$
$$y_{\omega,j} \leq y^{lo}_{\omega,j} + \frac{(UBD - obj_{JRMPR^{(l)}})}{v_{\omega,j}}, \quad \forall j \in \mathbb{J}^{(l)}_{2,\omega}, \ \forall \omega \in \{1, ..., s\}.$$

The following proposition states similar result for the linking variables x_0:

Proposition 5.8. *Consider the following bounds on* $x_{0,j}$ *(*$\forall j \in \{1, \cdots, n_0\}$*):*

$$x_{0,j} - x_{0,j}^{up} \leq 0,$$
$$x_{0,j}^{lo} - x_{0,j} \leq 0,$$

whose Lagrange multipliers obtained at the solution of Problem (JRMPR$^{(l)}$) *are* $u_{0,j}$, $v_{0,j}$. *Let* $\mathbb{J}_{1,0}^{(l)}$ *include indices of upper bounds whose* $u_{0,i}$ *are nonzero, and* $\mathbb{J}_{2,0}^{(l)}$ *include indices of lower bounds whose* $v_{0,i}$ *are nonzero, then the following constraints do not exclude an optimal solution of* (P)*:*

$$x_{0,j} \geq x_{0,j}^{up} - \frac{(UBD - obj_{JRMPR})}{u_{0,j}}, \quad \forall j \in \mathbb{J}_{1,0}^{(l)}$$
$$x_{0,j} \leq x_{0,j}^{lo} + \frac{(UBD - obj_{JRMPR^{(l)}})}{v_{0,j}}, \quad \forall j \in \mathbb{J}_{2,0}^{(l)}$$

According to Propositions 5.7 and 5.8, the bounds of nonconvex and linking variables can be updated via the following range reduction calculation:

$$\begin{aligned}
y_{\omega,j}^{up} &= \min\left\{y_{\omega,j}^{up},\ \ y_{\omega,j}^{lo} + \frac{G^{(l)}}{u_{\omega,j}}\right\}, \quad \forall j \in \mathbb{J}_{1,\omega}^{(l)},\ \ \forall \omega \in \{1, ..., s\},\\
y_{\omega,j}^{lo} &= \max\left\{y_{\omega,j}^{lo},\ \ y_{\omega,j}^{up} - \frac{G^{(l)}}{v_{\omega,j}}\right\}, \quad \forall j \in \mathbb{J}_{2,\omega}^{(l)},\ \ \forall \omega \in \{1, ..., s\},\\
x_{0,j}^{up} &= \min\left\{x_{0,j}^{up},\ \ x_{0,j}^{lo} + \frac{G^{(l)}}{u_{0,j}}\right\}, \quad \forall j \in \mathbb{J}_{1,0}^{(l)},\\
x_{0,j}^{lo} &= \max\left\{x_{0,j}^{lo},\ \ x_{0,j}^{up} - \frac{G^{(l)}}{v_{0,j}}\right\}, \quad \forall j \in \mathbb{J}_{2,0}^{(l)},
\end{aligned} \tag{MDR$^{(l)}$}$$

where $G^{(l)} = UBD - obj_{RMPCR^{(l)}}$.

The effectiveness of marginal based domain reduction relies on how many bounds are active, the magnitude of Lagrange multipliers of active bounds at the solution of

JRMPR$^{(l)}$, and how often JRMPR$^{(l)}$ is solved. In order to ahieve effective domain reduction more consistently, we also introduce optimization based domain reduction in JD. Optimization based domain reduction, or called bound contraction or bound tighening [145] [152], is to maximize or minimize a single variable over a convex relaxation of the feasible set of the original problem. For example, if we are to estimate the upper bound of a linking variable $x_{0,j}$ at JD iteration k, we can solve the following optimization problem:

$$
\begin{aligned}
&\max_{\substack{x_0,x_1,\ldots,x_s \\ y_1,,\ldots,y_s}} \; x_{0,i} \\
&\text{s.t. } \; x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1,...,s\}, \\
&\qquad A_\omega x_\omega + B_\omega y_\omega \leq 0, \quad \forall \omega \in \{1,...,s\}, \\
&\qquad \sum_{\omega=1}^{s} c_\omega^{\mathrm{T}} x_\omega \leq UBD, \\
&\qquad x_0 \in X_0^k, \\
&\qquad x_\omega \in X_\omega, \; y_\omega \in \hat{Y}_\omega^k, \quad \forall \omega \in \{1,...,s\}.
\end{aligned}
\tag{ODRStd$_i^k$}
$$

The third group of constraints in Problem (ODRStd$_i^k$) utilizes the known upper bound of (P) to tighten the convex relaxation, but it cannot be included in Problem (ODRStd$_i^k$) when UBD is not available (e.g., before a feasible solution of (P) is known). We now index sets X_0, $\hat{Y}_\omega$ with the JD iteration number k, as these sets may change after the domain reduction calculations.

Problem (ODRStd$_i^k$) represents the standard optimization based domain reduction formulation, but it can be further enhanced in the JD algorithm, via the incorporation of valid cuts derived from other JD subproblems. First, we can add the following

constraint:

$$\sum_{\omega=1}^{s} c_\omega^{\mathrm{T}} x_\omega \geq LBD.$$

This constraint is redundant in the classical branch-and-bound based global optimization, as LBD is obtained via convex relaxation as well. In JD, LBD is obtained via Lagrangian subproblems and JD relaxed master problems, which may be tigher than convex relaxations of the original problem, so this constraint may enhance Problem (ODRStd$_i^k$). Second, we can include constraints (*) (that are drived from Problem (JRMP$^{(l)}$)). Therefore, we can write the enhanced optimization based domain reduction formulation as:

$$\begin{aligned}
&\min_{\substack{x_0,x_1,\ldots,x_s\\ y_1,\ldots,y_s}} / \max_{\substack{x_0,x_1,\ldots,x_s\\ y_1,,\ldots,y_s}} \; x_{0,i} \\
\text{s.t. } & x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1,...,s\}, \\
& A_\omega x_\omega + B_\omega y_\omega \leq 0, \quad \forall \omega \in \{1,...,s\}, \\
& \sum_{\omega=1}^{s} c_\omega^{\mathrm{T}} x_\omega \leq UBD, \\
& \sum_{\omega=1}^{s} c_\omega^{\mathrm{T}} x_\omega \geq LBD, \\
& UBD \geq \sum_{\omega=1}^{s} obj_{LS_\omega^i} + \sum_{\omega=1}^{s} (\pi_\omega^i)^{\mathrm{T}} x_0, \quad \forall i \in R^k, \\
& x_0 \in X_0^k, \\
& x_\omega \in X_\omega,\ y_\omega \in \hat{Y}_\omega^k, \quad \forall \omega \in \{1,...,s\}.
\end{aligned} \tag{ODR$_i^k$}$$

If we are to estimate an upper bound, then Problem (ODR$_i^k$) is a maximization problem; otherwise, Problem (ODR$_i^k$) is a minimization problem.

Although Problem (ODR$_i^k$) is convex, it can have a very large size because its

size grows with the number of scenarios. Therefore, we proposed to solve Problem (ODR_i^k) for x_0 but not for y_ω. Actually, we can see in the case study section that optimization based domain reduction is time consuming even when we only solve Problem (ODR_i^k) for x_0.

5.4.2 The enhanced joint decomposition method

Figure 5.2 shows the framework of the JD method that includes solving convex relaxation, Problem ($\text{JRMPR}^{(l)}$), bound tightening for x_0 and the domain reduction calculations. In this framework, optimization based domain reduction is performed at the beginning of the algorithm and in every LD iteration (right before the solution of nonconvex Lagrangian subproblems). Convex relaxation, Problem ($\text{JRMPR}^{(l)}$) is solved before solving Problem ($\text{JRMP}^{(l)}$), and after solving Problem ($\text{JRMPR}^{(l)}$), marginal based domain reduction is performed. Problem ($\text{JRMP}^{(l)}$) is not solved if Problem ($\text{JRMPR}^{(l)}$) can improve the lower bound significantly; this strategy can postpone solving Problem ($\text{JRMP}^{(l)}$) to a later time, so that the ranges of x_0 can be reduced as much as possible when a Problem ($\text{JRMP}^{(l)}$) has to be solved. The detailed algorithm for the enhanced JD is shown in Table 5.2.

Theorem 5.2. *The decomposition algorithm described in Table 5.2 terminates in a finite number of steps with an ϵ-optimal solution of Problem* (P)*, if one the following three conditions is satisfied:*

(a) Set X_ω is polyhedral $\forall \omega \in \{1, \cdots, s\}$.

(b) Set $X_0 \times Y_1 \times \cdots \times Y_s$ is finite discrete.

(c) There are only a finite number of GBD iterations at which the Benders primal problem BPP is infeasible.

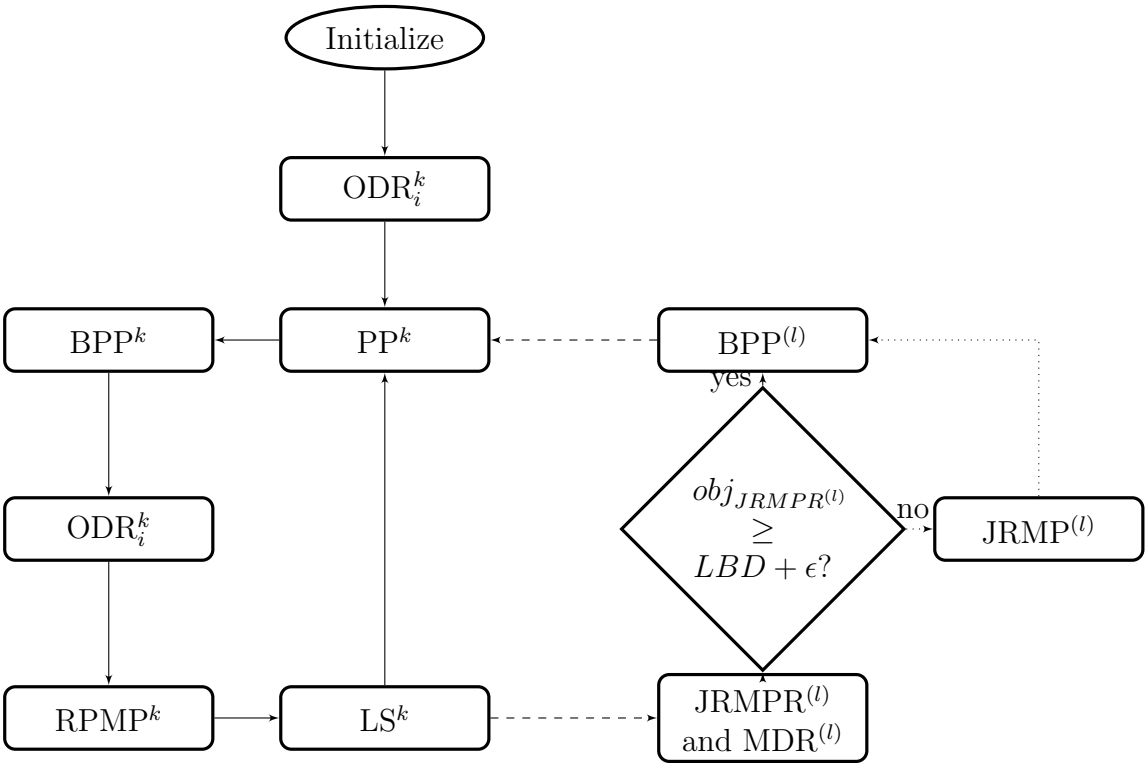


Figure 5.2: The enhanced joint decomposition framework

Proof. This can be proved by showing that, solving Problem (JRMPR$^{(l)}$) in every GBD iteration in JD and including domain reduction calculations do not invalidate the finite termination to an ϵ-optimal solution.

First, we can show that there cannot be an infinite number of GBD iterations at which Problem (JRMPR$^{(l)}$) is solved but Problem (JRMP$^{(l)}$) is not solved. Consider a GBD iteration at which Problem (JRMPR$^{(l)}$) is solved but Problem (JRMP$^{(l)}$) is not solved, then Problem (JRMPR$^{(l)}$) is not unbounded (because otherwise Problem (JRMP$^{(l)}$) needs to be solved) and the lower bound LBD is finite. The upper bound UBD is also finite (because an initial feasible solution exists). Therefore, it is not possible that LBD can be improved by $\epsilon > 0$ for an infinite number of GBD iterations,

so there cannot be an infinite number of GBD iterations at which Problem (JRMPR$^{(l)}$) is solved but Problem (JRMP$^{(l)}$) is not solved. According to the proof of Theorem 5.1, JD can only include a finite number of LD iterations, and a finite number of GBD iterations at which Problem (JRMP$^{(l)}$) is solved, if one of the three listed conditions are satisfied.

Second, domain reduction reduces the ranges of x_0 and $y_1, ..., y_s$ but does not exclude any optimal solution from the reduced ranges. So the Lagrangian relaxation problems and JD relaxation master problems are still valid lower bounding problems and they cannot cut off any optimal solution.

□

5.5 Case Studies

5.5.1 Case study problems

Case Study A - This problem is variant of the stochastic Haverly pooling problem [117], which was originally developed based on the classical Haverly pooling problem [153] [154]. Figure 5.3 shows the superstructure of the pooling system to be developed. The circles denote four sources that supply intermediate gasoline products with different sulfur percentages and costs, the ellipse denotes a blender (or called a pool) at which some intermediate products can be blended, and the rectangles denote product sinks at which the final products are blended. The goal of optimization is to minimize the negative profit of the system by determining: (1) Whether the pool and the two product sinks are to be developed in the system; (2) The capacities of the sources and the pipelines. The stochastic pooling model of the problem can be found in Appendix C.2. Two uncertain parameters, percentage of sulfur in source 4

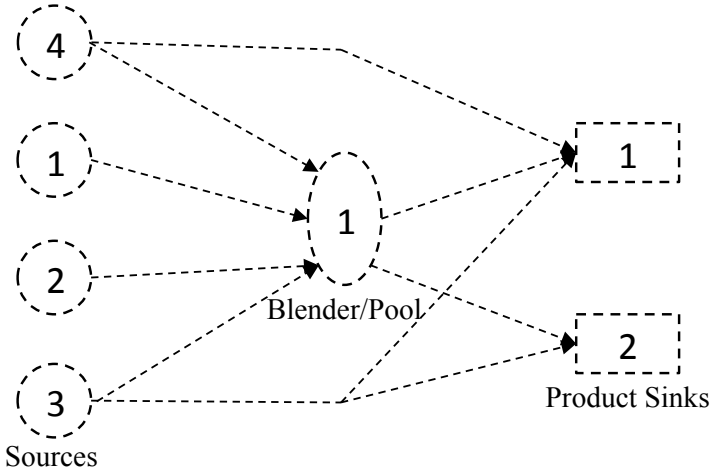


Figure 5.3: Superstructure of case study A problem

and upper limit on the demand at sink 1, were considered. They were assumed to following independent normal distributions, with means of 2.5 and 180, and standard deviations of 0.08 and 10. Other parameters used in the problem can be found in [117]. For this problem, x_0 contains 3 binary variables and 13 continuous variables, x_ω contains $7s$ continuous variables and y_ω contains $14s$ continuous variables, where s stands for the total number of scenarios. In the case study, each uncertain parameter was sampled for 5, 6, 7, 8, 9 and 10 scenario values, via the sampling rule described in [117], and this led to problem instances with 25, 36, 49, 64, 81 and 100 scenarios.

Case Study B - This problem is a variant of the Sarawak Gas Production System (SGPS) design problem [155], and the original form of the design problem appeared in [117]. Figure 5.4 shows the superstructure of the SGPS system under consideration, where the circles represent gas fields (sources), ellipses represent offshore gas platforms (pools) at which gas flows from different gas fields are mixed and split, rectangles represent onshore liquefied natural gas (LNG) plants (product terminals). Symbols

with solid lines represent the part of the system that is already developed, and symbols with dashed lines represent the superstructure of the part of the system that needs to be designed in the problem. The goal of optimization is to maximize expected net present value while satisfying specifications for gas qualities at the LNG plants in the presence of uncertainty. There are two uncertain parameters, i.e., the quality of CO_2 at gas field M1 and upper limit on the demand at LNG plant 2. They were assumed to follow independent normal distributions with means of 3.34% and 2155 Mmol/day, and standard deviations of 1% and 172.5 Mmol/day. In the case study, each uncertain parameter was sampled for 5, 6, 7, 8, 9 and 10 scenario values, via the same sampling rule described in [117], which led to problem instances with 25, 36, 49, 64, 81 and 100 scenarios. The problem was also formulated following the new stochastic pooling model provided in Appendix C.2. In the resulting formulation, x_0 contains 5 binary variables and 29 continuous variables. The 5 binary variables are to determine whether gas fields HL, SE, M3, M1 and JN are to be developed, and the 29 continuous variables are the capacities of other units to be developed. x_ω contains $8s$ variables and y_ω contains $85s$ variables, where s stands for the total number of scenarios.

5.5.2 Solution approaches and implementation

The case studies was run on a virtual machine allocated with a 3.2 GHz CPU. The virtual machine ran Linux operating system (Ubuntu 13.0) with 6 GB of memory. Three solution approaches were compared in the case studies: Monolith, JD1, JD2. The monolith approach solved the problems using a commercial optimization solver, ANTIGONE 1.1 [148], which adopted CONOPT 3 [156] as its NLP solver and CPLEX

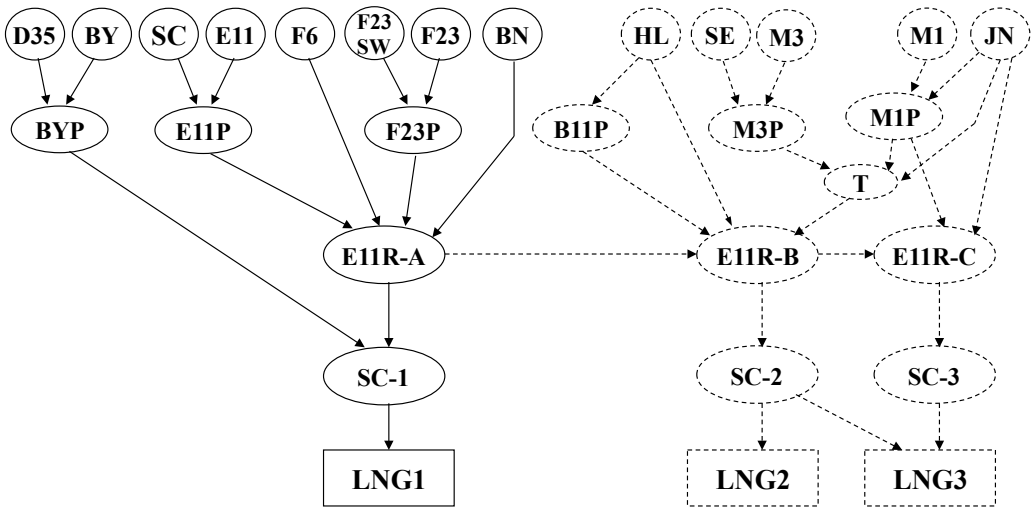


Figure 5.4: Superstructure of case study B problem

12.6 [108] as its LP/MILP solver. JD1 refers to the basic JD algorithm, and JD2 refers to the enhanced JD algorithm. The case study problems and the subproblems required in JD1 and JD2 were all modeled on GAMS 24.6.1 [111], but JD1 and JD2 algorithms were programmed on MATLAB 8.1.0 [157]. Data exchange between MATLAB and GAMS was realized via GAMS GDXMRW facility [158]. Nonconvex NLP/MINLP subproblems in JD1 and JD2 were solved by ANTIGONE 1.1, and LP/MILP subproblems in JD1 and JD2 were solved by CPLEX 12.6.

In JD2, the construction of Problems (ODR_i^k) and ($\text{JRMPR}^{(l)}$) require the convex relaxation of nonconvex sets X_0 and Y_ω. In the case studies, X_0 was a mixed integer set defined by linear constraints, and it was relaxed into a polyhedral set via continuous relaxation. Y_ω was a nonconvex continuous set defined with bilinear functions, and it was relaxed into a polyhedral set via standard McCormick relaxation [70]. The relative and absolute termination tolerances for Case Study A were set to 10^{-3}, and for Case study B were set to 10^{-2}. JD1 and JD2 started with all design decisions

being 0 (which are feasible for the case study problems).

During the execution of JD1 and JD2, large computing overhead may be incurred due to frequent model generation in GAMS and data exchange between GAMS and MATLAB. So both "Total solver time" and "Total run time" are recorded for the simulation studies, which refer to the total time for the subproblem solvers to solve each individual subproblem and the wall time for solving the overall problem. The computing overhead can be significantly reduced if JD1 and JD2 are implemented using general purpose programming languages, such as C++.

5.5.3 Results and discussion

Summary of the results for case study A is presented on Tables 5.3, 5.4, 5.5 below. Table 5.3 shows the results for the monolith implementation. As seen from the results, the monolith approach (with ANTIGONE) solves different scenario instances of the problem for a maximum of 81 scenarios within 10 minutes. However, a larger problem with 100 scenarios does not return a solution after an hour. This typifies the behavior of branch-and-bound based solvers for large scale MINLP. Table 5.4 shows the result for the basic joint decomposition method, JD1, proposed. JD1 solves the problem within 10 minutes except for the 81 scenario case that requires 17 minutes of solver time. Also, for a large scale problem with 100 scenarios, JD1 solves it easily (within 10 minutes). This is a big improvement over the monolith. Furthermore, even for relatively small problem with 64 scenarios, solver time for JD1 was about half that of the monolith run time. The results on Table 5.5 shows the performance of enhanced joint decomposition, JD2. JD2 does not do very well compared to the monolith, for small scale problems with few scenario, but it solves large scale scenario instance

better than the monolith. Also, JD2 has comparable performance to JD1. On the other hand, JD2 generally solves fewer number of JRMP$^{(l)}$ compared to JD1 (except for 64 scenarios where JD2 solves 5 JRMP and JD1 solves 4 JRMP$^{(l)}$) because of the embedded enhancements. This means that JD2 can potentially solve Problem (P) better if JRMP$^{(l)}$ is more difficult to solve. We see this evidently in the instance with 49 and 81 scenarios, where more than 35 % and 70 % of solver time is saved respectively. However, because of the time spent on solving Problem (ODR$_i^k$) for domain reduction, total solver time for JD2 can still be worse than that of JD1 (as in scenario 25 and 36) despite the reduction in the number of JRMP$^{(l)}$ solved.

Tables 5.6, 5.7 presents the results for case study B. Monolith implementation terminates for 25 and 36 scenarios but with an optimality gap greater than the specified tolerance. For large scenario cases (above 49 scenarios), the monolith does not terminate after the two-day run time limit. The results for JD1 is not shown, because JD1 does not terminate for the case study B within the run time limit either; the JRMP$^{(l)}$ solved does not converge within the time limit of 1 day set for decomposition, after a sufficient number of cuts are added. On the other hand, JD2, as shown on Table 5.7 solves the problem for all scenario instances within a day. Also, because of the enhancements to the algorithm, JD2 can sometimes solve a few JRMP$^{(l)}$, as in the 36 scenarios instance, or as many as 35 JRMP$^{(l)}$ subproblems in the 49 scenario case, and still converge to the solution.

Note that Tables 5.3, 5.4, 5.5, 5.6 and 5.7 does not include the times to solve easy LP and MILP subproblems like Problem (BPP$_\omega^{(l)}$), Problem (BFP$_\omega^{(l)}$), Problem (LS$_0^k$) and Problem (JRMPR$^{(l)}$), because those times are very small, within 1 % of the overall solver time. It is interesting to also note that the number of nonconvex

variables, y_ω, is far more than the number of convex variables, x_ω in both case studies; the number of y_ω is twice the number of x_ω for case study A and about 10 times the number of x_ω for case study B. However, JD2 still performs significantly well. In Remark 5.2 we imply that the JD framework may not be practical for problems for which the size of (BRMP$^{(l)}$) is much smaller than that of (P). However, the case results indicate that JD1 and JD2 are better than what we initially thought, and this may be due to the special structure of (BRMP$^{(l)}$). It remains an interesting question for future work. We believe that the integration of domain reduction also has helped to accelerate the solution of (BRMP$^{(l)}$).

5.6 Concluding Remarks

Two joint decomposition methods, JD1, and JD2, are developed in this chapter for global optimization of Problem (P). It has been proved that the algorithms can terminate in finite number of iterations with an ϵ-optimal solution if some mild conditions are satisfied. As seen from the simulation results, for a relative simple problem (case study A), JD1 and JD2 perform better than the monolith approach for large scenario instances, and JD2 outperforms JD1 for some scenario instances. For a more difficult problem (case study B), the superiority of JD2 is more clear.

It is also seen from the simulation studies that Problem (JRMP$^{(l)}$) can be much easier to solve than the original problem, even when their sizes are close. We will investigate in the future why this happens. In addition, we will consider nonconvex relaxations/restrictions of (JRMP$^{(l)}$) that are decomposable over the scenarios and can be used to yield tighter bounds.

In this chapter, we consider optimization based domain reduction for only the

linking variables x_0. In the future, we can also consider efficient optimization based domain reduction for some key nonconvex variables y_ω, which can tighten the convex relaxation of Problem (JRMP$^{(l)}$), and therefore reduce the number of JRMP$^{(l)}$ to be solved and increase the efficiency of solving each JRMP$^{(l)}$.

Table 5.2: Enhanced joint decomposition method - Enhancement is in bold font

Initialization

(I.a) Select $x_0^1, y_1^{[1]}, \cdots, y_s^{[1]}$ that are feasible for Problem (P).

(I.b) Give termination tolerance $\epsilon > 0$. Let index sets $T^1 = S^1 = R^1 = \emptyset$, $I^1 = \{1\}$, iteration counter $k = 1$, $i = 1$, $l = 1$, bounds $UBD = +\infty$, $LBD = -\infty$.

(I.c) **Solve Problem** (ODR_i^k) **to update bounds of all** $x_{0,i}$**.**

LD Iteration

(1.a) Solve Problem (PP_ω^k). If Problem (PP_ω^k) is infeasible, solve Problem (FP_ω^k). Let the solution obtained be (x_ω^k, y_ω^k), and update $i = i + 1$, I^k=$I^k \cup \{i\}$, $(y_1^{[i]}, \cdots, y_s^{[i]}) = (y_1^k, \cdots, y_s^k)$.

(1.b) Solve Problem (BPP_ω^k) by fixing $(x_0, y_1, ..., y_s) = (x_0^k, y_1^k, ..., y_s^k)$. If ($\text{BPP}_\omega^k$) is feasible for all ω, generate Benders optimality cuts with the obtained dual solution μ_ω^k and λ_ω^k, and update $T^{k+1} = T^k \cup \{k\}$. If $\sum_{\omega=1}^s obj_{PP_\omega^k} < UBD$, update $UBD = \sum_{\omega=1}^s obj_{PP_\omega^k}$, and incumbent solution $(x_0^*, x_1^*, \cdots, x_s^*, y_1^*, \cdots, y_s^*) = (x_0^k, x_1^k, \cdots, x_s^k, y_1^k, \cdots, y_s^k)$. If Problem ($\text{BPP}_\omega^k$) is infeasible for at least one ω, solve Problem (BFP_ω^k). Generate Benders feasibility cuts with the obtained dual solution μ_ω^k and λ_ω^k, and update $S^{k+1} = S^k \cup \{k\}$.

(1.c) **Solve Problem** (ODR_i^k) **to update bounds of all** $x_{0,i}$**.**

(1.d) Solve Problem (RPMP^k). Let x_0^k, $\{\theta_\omega^{[i,k]}\}_{i \in I^k, \omega \in \{1,...,s\}}$ be the optimal solution obtained, and $\pi_1^k, ..., \pi_s^k$ be Lagrange multipliers for the NACs.

(1.e) Solve Problems (LS_ω^k) and (LS_0^k), and let the obtained solution be (x_ω^k, y_ω^k), x_0^k. If $obj_{LS^k} = \sum_{\omega=1}^s obj_{LS_\omega^k} + obj_{LS_0{}^k} > LBD$, update $LBD = obj_{LS^k}$. Generate a Lagrangian cut and update $R^{k+1} = R^k \cup \{k\}$. Update $i = i + 1$, $I^{k+1} = I^k \cup \{i\}$, $(y_1^{[i]}, \cdots, y_s^{[i]}) = (y_1^k, \cdots, y_s^k)$.

(1.f) If $UBD \leq LBD + \epsilon$, terminate and return the incumbent solution as an ϵ-optimal solution. If $obj_{LS^k} \geq obj_{LS^{k-1}} + \epsilon$, $k = k + 1$, go to step (1.a); otherwise $k = k + 1$ and go to step (2.a);

GBD Iteration

(2.a) **Solve Problem** ($\text{JRMPR}^{(l)}$)**, and then perform marginal based domain reduction** ($\text{MDR}^{(l)}$)**. If** $obj_{JRMPR^{(l)}} \geq LBD + \epsilon$**, let the obtained solution be** $(x_0^{(l)}, y_1^{(l)}, ..., y_s^{(1)})$**, update** $LBD = obj_{JRMPR^{(l)}}$**,** $i = i + 1$**,** $I^{k+1} = I^k \cup \{i\}$**,** $(y_1^{[i]}, \cdots, y_s^{[i]}) = (y_1^{(l)}, \cdots, y_s^{(l)})$**, go to step (2.c). Otherwise, go to set (2.b).**

(2.b) Solve Problem ($\text{JRMP}^{(l)}$), and let the obtained solution be $(x_0^{(l)}, y_1^{(l)}, ..., y_s^{(1)})$. Update $i = i + 1$, $I^{k+1} = I^k \cup \{i\}$, $(y_1^{[i]}, \cdots, y_s^{[i]}) = (y_1^{(l)}, \cdots, y_s^{(l)})$. If $obj_{RMP^{(l)}} > LBD$, update $LBD = obj_{JRMP^{(l)}}$.

(2.c) Solve Problem ($\text{BPP}_\omega^{(l)}$) by fixing $(x_0, y_1, \cdots, y_s) = (x_0^{(l)}, y_1^{(l)}, \cdots, y_s^{(l)})$. If ($\text{BPP}_\omega^{(l)}$) is feasible for all ω, generate Benders optimality cuts with the dual solution μ_ω^k and λ_ω^k, and update $T^{(l+1)} = T^{(l)} \cup \{l\}$. If $\sum_{\omega=1}^s obj_{BPP_\omega^{(l)}} < UBD$, update $UBD = obj_{BPP^{(l)}}$ and the incumbent solution $(x_0^*, x_1^*, \cdots, x_s^*, y_1^*, \cdots, y_s^*) = (x_0^{(l)}, x_1^{(l)}, \cdots, x_s^{(l)}, y_1^{(l)}), \cdots, y_s^{(l)})$. If Problem ($\text{BPP}_\omega^{(l)}$) is infeasible for at least one ω, solve Problem ($\text{BFP}_\omega^{(l)}$). Generate Benders feasibility cuts with the obtained dual solution μ_ω^l and λ_ω^l, and update $S^{(l+1)} = S^{(l)} \cup \{l\}$.

(2.d) If $UBD \leq LBD + \epsilon$, terminate and return the incumbent solution as an ϵ-optimal solution; otherwise $l = l + 1$, go to step (1.a).

Table 5.3: Results for case study A - Monolith (Unit for time: sec)

Number of scenarios	25	36	49	64	81	100
Optimal obj. ($)	-532.1	-530.6	-531.2	-531.5	-531.1	-531.1
Total run time	18	44	138	395	651	– †

† No global solution was returned within 3600 seconds.

Table 5.4: Results for case study A - JD1 (Unit for time: sec)

Number of scenarios	25	36	49	64	81	100
Num. of iterations	9	12	9	12	11	12
Num. of JRMP$^{(l)}$ solved	4	5	4	4	6	5
Optimal obj. ($)	-531.7	-530.3	-530.8	-531.5	-531.1	-530.9
Time for LS$^k_\omega$	21	46	40	70	76	95
Time for PP^k	6	12	11	22	27	32
Time for JRMP$^{(l)}$	2	7	134	97	884	23
Total solver time	30	69	187	194	994	158
Total run time	107	241	343	446	1322	541

Table 5.5: Results for case study A - JD2 (Unit for time: sec)

Number of scenarios	25	36	49	64	81	100
Num. of iterations	8	10	10	13	11	12
Num. of JRMPR$^{(l)}$ solved	3	5	5	9	6	7
Num. of JRMP$^{(l)}$ solved	0	1	1	5	3	3
Optimal obj. ($)	-532.1	-530.3	-531.0	-531.5	-531.1	-531.1
Time for LS$^k_\omega$	16	29	42	69	73	98
Time for PP^k	4	6	11	20	22	27
Time for JRMP$^{(l)}$	0	2	9	117	34	43
Time for ODR^k	25	42	56	101	113	160
Total solver time	46	81	120	313	249	335
Total run time	115	209	311	673	586	778

Table 5.6: Results for case study B - Monolith (Unit for time: sec)

Number of scenarios	25	36	49	64	81	100
Optimal obj. (Billion $)	-33.87	-33.67	-33.81	-33.76	-33.78	-33.79
Total run time	140431 †	154517 ‡	– #	–	–	–

† ANTIGONE terminated with an optimality gap of 1.4 %.
‡ ANTIGONE terminated with an optimality gap of 2.1 %.
# No global solution was returned within 172800 seconds (2 days).

Table 5.7: Results for case study B - JD2 (Unit for time: sec)

Number of scenarios	25	36	49	64	81	100
Num. of iterations	28	21	46	27	27	26
Num. of JRMPR$^{(l)}$ solved	21	14	41	18	17	18
Num. of JRMP$^{(l)}$ solved	13	6	35	14	10	11
Optimal obj. (Billion $)	-33.74	-33.52	-33.65	-33.59	-33.62	-33.52
Time for LS$^k_\omega$	4342	970	17697	3004	4610	6452
Time for PP^k	945	821	1799	659	1584	1746
Time for JRMP$^{(l)}$	2930	909	7258	2936	7481	30542
Time for ODR^k	7124	6495	28989	23141	24056	34089
Total solver time	15387	9240	56166	29929	37938	73249
Total run time	18605	11899	70767	33854	43104	82772

Chapter 6

Conclusions and Future Work

This thesis presents novel decomposition based approaches to solve large-scale mixed-integer linear and nonlinear optimization problems with angular, dual-angular, and hybrid-angular structures. The summary of the results in this thesis are discussed in section 6.1. Section 6.2 discusses future research direction.

6.1 Conclusions

The following four subsections summarize the conclusions drawn from the thesis.

6.1.1 A new cross decomposition for stochastic mixed-integer linear programming

In chapter 2, we proposed a new cross decomposition framework to solve two-stage stochastic MILPs. This decomposition approach exploits the synergy between BD and DWD to efficiently solve the original problem. The two key contributions in this CD framework is the use of DWD restricted master problem for generating upper bounds, and the ability to deal with infeasible problems or problems for which initial feasible solutions are difficult to find using a phase 1 procedure. From this decomposition

framework, we develop two cross decomposition algorithms, CD1 and CD2. CD1 alternates between a BD iteration and a DWD iteration, while CD2 determines when to switch from BD to DWD and back, adaptively.

We demonstrate the performance of the new CD approaches by comparing with the monolith and BD approaches, using case study of a bio-product SCO problem and an industrial chemical SCO problem. Both case studies show that the three decomposition methods outperform the monolith approach significantly when the number of scenarios is large. In the first case study, we see that both CD1 and CD2 require much fewer iterations and therefore less total solver times; using CD2, we achieve a reduction in solution time by more than 80% over BD (and 90% over the monolith approach) when the number of scenarios is 361. BD has better convergence in the second case study compared to CD1, but CD2 is still a better alternative because of its superior performance.

6.1.2 Multicolumn-multicut cross decomposition for stochastic mixed-integer linear programming

A MCMC CD algorithm is developed in chapter 3 to solve stochastic MILPs. We derive tighter upper and lower bounds for the MCMC CD through the multicolumn and multicut formulations of the DWD restricted master and Benders relaxed master problems respectively, and the new formulation does not perturb the finite convergence of the algorithm. Using case study of a bio-product supply chain optimization problem, we show that that MCMC CD is computational faster than the classical BD method by an order of magnitude when the number of scenarios is large, and it is also significantly faster than CD and MC BD for all scenario instances considered.

6.1.3 Extended cross decomposition for stochastic mixed-integer programs with strong and weak linking constraints

In chapter 4, an extension of the cross decomposition method was developed for two-stage stochastic MILPs with strong and weak linking constraints. A CVaR constrained two-stage stochastic programming problem was taken as an example in the above problem class. Case study with risk neutral and risk averse constraints was considered.

The risk neutral form of the case study was solved using monolith, ordinary cross decomposition and Benders decomposition. CD2 applied to this case has superior performance over CD1, BD and monolith for large scenario instances. About 37 % of solver time can be saved using CD2 compared to BD. Furthermore, for the largest scenario case where the monolith could not return a solution within a day, CD2 returns under less than two hour. For the risk averse case study, the proposed ECD showed good performance over the naive BLD and the monolith. ECD2 can achieve as much as 10 times reduction in wall time compared to BLD, and over an order of magnitude reduction in wall time compared to the monolith.

6.1.4 Joint decomposition for multiscenario mixed-integer nonlinear nonconvex programming

We developed two joint decomposition methods, JD1, and JD2, in chapter 5 to globally solve two-stage stochastic nonconvex MINLPs. The two algorithms have been shown to converge in finite number of iterations with an ϵ-optimal global solution. The results from the simulations suggests that for a relative simple problem (case study A), JD1 and JD2 perform better than the monolith approach for large scenario

instances, and JD2 outperforms JD1 for some scenario instances. For a larger and more realistic problem (case study B), the superiority of JD2 is more glaring, as JD2 returns the solution to the problem where the monolith and JD1 fails. Finally, it can be observed that the relaxed master problem, which has similar size to the original problem, can be much easier to solve than the original problem because of the domain reduction strategy embedded, as demonstrated in the case studies.

6.2 Future Work

The following areas will be the focus of future work; (i) dealing with multiple primal and dual solutions from subproblems, (ii) developing more efficient domain reduction techniques and (iii) more applications of the proposed decomposition strategies. These three aspects are discussed below.

6.2.1 Multiple primal and dual solutions

The decomposition techniques developed in this thesis rely on optimal solutions from primal and dual subproblems. However, in a lot of application, multiple primal and dual solution exists [48] [159]. Applications where this problem is prevalent is network design and operation. This is because in such applications, for any fixed network design, there exist multiple production schemes to achieve the same objective. Multiple primal and dual solutions increases the number of iterations for decomposition methods, deteriorating convergence. Strategies to overcome this problem can include:

(a) Cut strengthening: The seminal paper by Magnanti and Wong [48] for BD and a paper by Van Roy for CD [83] describe a way to strengthen cuts in the relaxed master problem in BD/CD application. They showed that by solving extra

subproblems at each BD iteration, the best Lagrangian multiplier for each cut can be obtained, significantly reducing the overall BD iterations. This approach can be applied in the new CD in the future.

(b) Column strenthening: The pricing problem can have multiple solutions which can affect the tightness of the DW restricted master problem, thus affecting the performance of CD/ECD/JD. Therefore, similar to cut strengthening, column strengthening in the DW restricted master problem could lead to improvement in decomposition performance.

6.2.2 Novel domain reduction techniques

The bottleneck in the performance of joint decomposition technique in chapter 5 is the fact that a nonconvex Benders relaxed master problem needs to be solved. However, we have seen that domain reduction; either through marginal based domain reduction, or through bound tightening, can significantly reduce the computational complexity of subproblems, especially the Benders relaxed master problem. Going forward, the following new efficient domain reduction techniques can be investigated to further improve the performance of joint decomposition.

(a) New marginal based domain reduction schemes: The current marginal based domain reduction by Ryoo and Sahinidis [146] solve probing subproblems (if the marginal values of constraints are zeros) that do not tighten the convex relaxation. A potential future direction is to implement a probing procedure, where solving a probing problem for a particular variable can enable the bounds on the other variables in the problem to be improved, thereby tightening the convex relaxation.

(b) New bound tightening schemes: In the joint decomposition method, we consider optimization based domain reduction for only the linking variables x_0. In the future, efficient optimization based domain reduction for some of the important nonconvex variables y_ω can be considered. This can tighten the convex relaxation of Problem (JRMP$^{(l)}$) and therefore increase the efficiency of solving each JRMP$^{(l)}$. Additionally, developing a new improved bound tightening technique is a potential future direction.

6.2.3 More applications of the proposed decomposition approaches

The focus of this thesis was to develop and apply decomposition based techniques to solve scenario based stochastic programs. In the future, application of decomposition methods can be extended to areas in large-scale mathematical programming such as:

(a) Multiperiod programming: Here, the different periods are considered as independent blocks, and the linking variables or constraints comes from links between these blocks. With proper problem reformulation, the decomposition approaches presented in this thesis can be applied to such problems.

(b) Large scale physical systems with loosely connected parts: One application of these kinds of problems, i.e., dynamically decoupled subsystems with linking constraints, was described in chapter 2. Another application in process systems engineering is plantwide optimization, which require the optimization of a system consisting of distillation columns, heat exchangers, seperators, reactors, connected together. Because these problems have loosely connected structures, applying decomposition methods can explore this special structure for efficient solution.

Bibliography

[1] L. T. Biegler, I. E. Grossmann, Retrospective on optimization, Computers & Chemical Engineering 28 (8) (2004) 1169 – 1192.

[2] I. E. Grossmann, L. T. Biegler, Part II. Future perspective on optimization, Computers & Chemical Engineering 28 (8) (2004) 1193 – 1218.

[3] L. Biegler, Nonlinear Programming: Concepts, Algorithms and Applications to Chemical Processes, SIAM, 2010.

[4] I. E. Grossmann, J. A. Caballero, H. Yeomans, Mathematical programming approaches to the synthesis of chemical process systems, Korean Journal of Chemical Engineering 16 (4) (1999) 407–426.

[5] W. Rudin, Principles of Mathematical Analysis, 3rd Edition, McGraw-Hill, Newyork, U.S.A, 1976.

[6] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, UK, 2004.

[7] C. A. Floudas, C. E. Gounaris, A review of recent advances in global optimization, Journal of Global Optimization 45 (1) (2008) 3–38.

[8] I. E. Grossmann, Review of nonlinear mixed-integer and disjunctive programming techniques., Optimization and Engineering 3 (2002) 227–252.

[9] I. E. Grossmann, Advances in mathematical programming models for enterprise-wide optimization, Computers & Chemical Engineering 47 (2012) 2 – 18, {FOCAPO} 2012.

[10] L. Čuček, H. Lam, J. Klemeš, P. Varbanov, Z. Kravanja, Synthesis of regional networks for the supply of energy and bioproducts, Clean. Technol. Environ. Policy 12 (2010) 635–645.

[11] S. A. Papoulias, I. E. Grossmann, A structural optimization approach in process synthesis-II Heat recovery networks, Computers & Chemical Engineering 7 (6) (1983) 707 – 721.

[12] A. W. Westerberg, M. J. Andrecovich, Utility bounds for nonconstant QΔT for heat-integrated distillation sequence synthesis, AIChE Journal 31 (9) (1985) 1475–1479.

[13] V. T. Voudouris, I. E. Grossmann, Mixed-integer linear programming reformulations for batch process design with discrete equipment sizes, Industrial & Engineering Chemistry Research 31 (5) (1992) 1315–1325.

[14] G. Paules, C. Floudas, Synthesis of flexible distillation sequences for multiperiod operation, Computers & Chemical Engineering 12 (4) (1988) 267 – 280.

[15] A. C. Kokossis, C. Floudas, Optimization of complex reactor networks-II. non-isothermal operation, Chemical Engineering Science 49 (7) (1994) 1037 – 1051.

[16] M. C. Wellons, G. V. Reklaitis, Scheduling of multipurpose batch chemical plants. 2. multiple-product campaign formation and production planning, Industrial & Engineering Chemistry Research 30 (4) (1991) 688–705.

[17] T. Westerlund, F. Pettersson, I. E. Grossmann, Optimization of pump configurations as a MINLP problem, Computers and Chemical Engineering 18 (9) (1994) 845–858.

[18] N. Sahinidis, Optimization under uncertainty: state-of-the-art and opportunities., Computers and Chemical Engineering 28 (2004) 971–983.

[19] A. Shapiro, Stochastic programming approach to optimization under uncertainty, Mathematical Programming 112 (1) (2008) 183–220.

[20] A. Shapiro, D. Dentcheva, A. Ruszczynski, Lectures on Stochastic Programming: Modeling and Theory, SIAM, Philadelphia, 2009.

[21] J. Birge, F. Louveaux, Introduction to Stochastic Programming, Springer, New York, 2010.

[22] M. E. Bruni, P. Beraldi., D. Conforti, A solution approach for two-stage stochastic nonlinear mixed integer programs, Algorithmic Operations Research 4 (1).

[23] C. C. Carøe, R. Schultz, Dual decomposition in stochastic integer programming, Operations Research Letters 24 (12) (1999) 37 – 45.

[24] F. Oliveira, V. Gupta, S. Hamacher, I. Grossmann, A Lagrangean decomposition approach for oil supply chain investment planning under uncertainty with risk considerations, Computers & Chemical Engineering 50 (2013) 184 – 195.

[25] R. Karuppiah, I. Grossmann, A Lagrangean based branch-and-cut algorithm for global optimization of nonconvex mixed-integer nonlinear programs with decomposable structures, Journal of Global Optimization 41 (2008) 163–186.

[26] P. Kall, S. Wallace, Stochastic Programming, 2nd Edition, Springer, New York, 1994.

[27] S. Ahmed, Convexity and decomposition of mean-risk stochastic programs, Math. Program. 106 (2006) 433–446.

[28] A. Charnes, W. W. Cooper, Chance-constrained programming, Management Science 6 (1) (1959) 73–79.

[29] X. Liu, S. Küçükyavuz, J. Luedtke, Decomposition algorithms for two-stage chance-constrained programs, Mathematical Programming 157 (1) (2016) 219–243.

[30] L. Lasdon, Optimization Theory for Large Systems, 1st Edition, Macmillian, Toronto, Ontario, 1970.

[31] J. Tebboth, A computational study of dantzig-wolfe decomposition, Ph.D. thesis, University of Buckingham, Buckingham, UK (2001).

[32] N. Sahinidis, I. Grossman, Reformulation of the multiperiod MILP model for capacity expansion of chemical processes, Operations Research 40 (1992) S127–S144.

[33] L. Sokoler, L. Standardi, K. Edlund, N. Poulsen, H. Madsen, J. Jørgensen, A DantzigWolfe decomposition algorithm for linear economic model predictive

control of dynamically decoupled subsystems, Journal of Process Control 24 (8) (2014) 1225 – 1236.

[34] J. Benders, Partitioning procedures for solving mixed-variables programming problems, Numerische Mathematik 4 (1962) 238–252.

[35] A. M. Geoffrion, Generalized Benders decomposition, Journal of Optimization Theory and Applications 10 (4) (1972) 237–260.

[36] M. Duran, I. Grossmann, An outer-approximation algorithm for a class of mixed-integer nonlinear programming, Math. Prog. 66 (1986) 327–349.

[37] G. Dantzig, P. Wolfe, The decomposition principle for linear programs, Operations Research 8 (1960) 101–111.

[38] M. Fisher, Lagrangian relaxation methods for solving integer programming problems, INFORMS 27 (1981) 1–18.

[39] D. Bertsekas, Nonlinear Programming, 2nd Edition, Athena Scientific, Cambridge, MA, 1999.

[40] R. J.-B. W. R. T. Rockafellar, Scenarios and policy aggregation in optimization under uncertainty, Mathematics of Operations Research 16 (1) (1991) 119–147.

[41] D. Gabay, B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite element approximation, Computers & Mathematics with Applications 2 (1) (1976) 17 – 40.

[42] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via alternating method of multipliers, Foundations and Trends in Machine Learning 3 (2010) 1–122.

[43] R. M. Van Slyke, R. Wets, L-shaped linear programs with applications to optimal control and stochastic programming, SIAM Journal on Applied Mathematics 17 (1969) 638–663.

[44] R. Rahmaniani, T. Crainic, M. Gendreau, W. Rei, The Benders decomposition method: A literature review, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) (2016) 1–35.

[45] D. Topkis, Cutting-plane algorithms with linear and geometric rates of convergence, Journal of Optimization Theory and Applications 36 (1982) 1–19.

[46] J. Birge, F. V. Louveaux, A multicut algorithm for two-stage stochastic linear programs, European Journal of Operational Research 34 (3) (1988) 384 – 392.

[47] F. You, S. Grossmann, Multicut Benders decomposition algorithm for process supply chain planning under uncertainty, Annals of Oper. Res. 210 (2013) 191–210.

[48] R. T. Magnanti, T. L. Wong, Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria, Operation Research.

[49] L. H. Applegren, A column generation algorithm for a ship scheduling problem, INFORMS (1969) 53–68.

[50] P. A. Vance, C. Barnhart, E. L. Johnson, G. L. Nemhauser, Solving binary cutting stock problems by column generation and branch-and-bound, Computational Optimization and Applications 3 (1994) 111–130.

[51] J. Desrosiers, L. Marco E, A Primer in Column Generation, Springer-Verlag, 2005.

[52] R. M. Geoffrion, A. Bride, Lagrangean relaxation applied to capacitated facility location problems, A I I E Transactions 10 (1978) 40–47.

[53] A. M. Geoffrion, Lagrangean relaxation for integer programming, Mathematical Programming Study 2 (1974) 82–114.

[54] S. Mitra, P. Garcia-Herreros, I. E. Grossmann, A cross-decomposition scheme with integrated primal–dual multi-cuts for two-stage stochastic programming investment planning problems, Mathematical Programming 157 (1) (2016) 95–119.

[55] I. E. Grossmann, R. M. Apap, B. A. Calfa, P. Garca-Herreros, Q. Zhang, Recent advances in mathematical programming techniques for the optimization of process systems under uncertainty, Computers & Chemical Engineering 91 (2016) 3 – 14.

[56] C. Barnhart, E. Johnson, Branch and price: Column generation for solving huge integer programs, Operations Research 46 (1998) 316–329.

[57] A. Frangioni, Generalized bundle methods, SIAM Journal on Optimization 13 (1) (2002) 117–156.

[58] M. Dempster, R. Merkovsky, A practical geometrically convergent cutting plane algorithm, SIAM. J. Num. Anal 32 (1995) 631–644.

[59] J. Kelly, The cutting-plane method for solving convex programs, J. Soc. Indust. Appl. Math. 8 (4) (1960) 703–712.

[60] R. Gomory, Outline of an algorithm for integer solutions to linear programs, Bull. Amer. Math. Soc. 9 (1958) 275–278.

[61] A. J. Rubiales, P. A. Lotito, L. A. Parente, Stabilization of the generalized benders decomposition applied to short-term hydrothermal coordination problem, IEEE Latin America Transactions 11 (5) (2013) 1212–1224.

[62] F. Oliveira, I. Grossmann, S. Hamacher, Accelerating benders stochastic decomposition for the optimization under uncertainty of the petroleum product supply chain, Computers & Operations Research 49 (2014) 47 – 58.

[63] L. Liberti, Reformulation and convex relaxation techniques for global optimization, Quarterly Journal of the Belgian, French and Italian Operations Research Societies 2 (3) (2004) 255–258.

[64] L. A. Wolsey, G. L. Nemhauser, Integer and Combinatorial Optimization, John Wiley and Sons Inc, NY, USA, 1988.

[65] D. Dung-Zhu, K. Ker-I, Theory of Computational Complexity, Wiley, Hoboken, New Jersey, 2014.

[66] R. Horst, H. Tuy, Global Optimization: Deterministic Approaches, Spri, 1996.

[67] S. A. Vavasis, Complexity Issues in Global Optimization: A Survey, Springer US, Boston, MA, 1995, pp. 27–41.

[68] A. H. Land, A. G. Doig, An automatic method of solving discrete programming problems, Econometrica 28 (3) (1960) 497–520.

[69] D. Bertsimas, J. Tsitsiklis, Introduction to Linear Programming, 2nd Edition, Athena Scientific, Cambridge, MA, 1997.

[70] G. P. McCormick, Computability of global solutions to factorable nonconvex programs: Part I - Convex underestimating problems, Mathematical Programming 10 (1976) 147–175.

[71] J. B. Rosen, P. M. Pardalos, Global minimization of large-scale constrained concave quadratic problems by separable programming., Mathematical Programming 34 (1986) 163–174.

[72] X. Li, Y. Chen, P. I. Barton, Nonconvex generalized benders decomposition with piecewise convex relaxations for global optimization of integrated process design and operation problems, Industrial and Engineering Chemistry Research 51 (2012) 7287–7299.

[73] M. Tawarmalani, N. V. Sahinidis, Global optimization of mixed-integer nonlinear programs: A theoretical and computational study, Mathematical Programming 99 (2004) 563–591.

[74] I. P. Androulakis, C. D. Maranas, C. A. Floudas, αBB: A global optimization method for general constrained nonconvex problems, Journal of Global Optimization 7 (4) (1995) 337–363.

[75] E. P. Gatzke, J. E. Tolsma, P. I. Barton, Construction of convex relaxations using automated code generation technique, Optimization and Engineering 3 (2002) 305–326.

[76] A. M. Gleixner, T. Berthold, B. Müller, S. Weltge, Three enhancements for optimization-based bound tightening, Journal of Global Optimization (2016) 1–27.

[77] N. Sahinidis, BARON: A general purpose global optimization software package., Journal of Global Optimization 8 (1996) 201–205.

[78] R. Karuppiah, I. E. Grossmann, A Lagrangean based branch-and-cut algorithm for global optimization of nonconvex mixed-integer nonlinear programs with decomposable structures, Journal of Global Optimization 41 (2008) 163–186.

[79] X. Li, A. Tomasgard, P. I. Barton, Nonconvex generalized Benders decomposition for Stochastic Separable Mixed-Integer Nonlinear Programs, Journal of Optimization Theory and Applications 151 (2011) 425–454.

[80] E. Balas, R. Jeroslow, Canonical cuts on the unit hypercube, SIAM Journal on Applied Mathematics 23 (1) (1972) 61–69.

[81] R. Kannan, P. Barton, A software framework for the global optimization of nonconvex two-stage stochastic programs, AIChE Proceedings.

[82] T. J. Van Roy, Cross decomposition for mixed integer programming, Mathematical programming 25 (1) (1983) 46–63.

[83] T. J. Van Roy, A cross decomposition algorithm for capacitated facility location, Operations Research 34 (1986) 145–163.

[84] K. Holmberg, On the convergence of cross decomposition, Mathematical Programming 47 (1990) 269 – 296.

[85] L. Papageorgiou, Supply chain optimization for the process industries: Advances and opportunities, Computers and Chemical Engineering 33 (12) (2009) 1931 – 1938.

[86] R. Iyer, I. Grossmann, Optimal planning and scheduling of offshore oil field infrastructure investment and operations., Industrial and Engineering Chemistry Research 37 (1998) 1380–1397.

[87] J. Li, I. Karimi, R. Scrinivasan, Recipe determination and scheduling of gasoline blending operations, AICHE 56 (2010) 441–465.

[88] M. Guignard, S. Kim, Lagrangean decomposition:a model yielding stronger lagrangean bounds, Mathematical Programming 39 (1987) 215–228.

[89] C. Carøe, R. Schultz, Dual decomposition in stochastic integer programming, Operation Research Letters 24 (1999) 37–45.

[90] C. Barnhart, E. Johnson, Branch and price: Column generation for solving huge integer programs, Operations Research 46 (1998) 316–329.

[91] A. Frangioni, About Lagrangian methods in integer optimization, Annals of Operations Research 139 (2005) 163–193.

[92] J. Desrosiers, M. Lubbecke, A column generation approach to the urban transit crew scheduling problem, Transportation Science 23 (1989) 1–13.

[93] F. Vanderbeck, On Dantzig-Wolfe decomposition for integer programming and ways to perform branching in the branch-and-price algorithm, Operations Research 48 (2000) 111–128.

[94] M. E. Lubbecke, J. Desrosiers, Selected topics in column generation, Operation Research 53 (2005) 1007–1023.

[95] F. Vanderbeck, A generic view of Dantzig-Wolfe decomposition for mixed integer programming, Operation Research Letters 23 (2006) 296–306.

[96] A. Oukil, H. Amor, J. Desrosiers, H. Gueddari, Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems, Computers and Operations Research 34 (2007) 817–834.

[97] F. Vanderbeck, Branching in branch-and-price: a generic scheme, Mathematical Programming 130 (2011) 249–294.

[98] E. Coughlan, M. Lubbecke, J. Schulz, A branch-price-and-cut algorithm for multi-mode resource leveling, European Journal of Operational Research 245 (2015) 70–80.

[99] I. Contreras, J. Cordeau, G. Laporte, Benders decomposition for large-scale uncapacited hub location, Operation Research 59 (2011) 1477–1490.

[100] M. Florian, G. Guerin, G. Bushel, The engine scheduling problem in a railway network, INFORMS 14 (1976) 121–138.

[101] E. Balas, C. Bergthaller, Benders method revisited, Journal of Applied and Computational Mathematics 9 (1983) 3–12.

[102] K. Holmberg, Mean value cross decomposition applied to integer programming problems, European Journal of Operational Research 97 (1997) 124–138.

[103] S. Mitra, P. Garcia-Herreros, I. Grossmann, A novel cross-decomposition multi-cut scheme for two-stage stochastic programming, Computer Aided Chemical Engineering 22 (2014) 241–246.

[104] L. Snyder, M. Daskin, Reliable models for facility:the expected failure cost case, Transportation Science 39 (2005) 400–416.

[105] P. Garcia-Herreros, J. Wassick, I. Grossmann, Design of resilient supply chains with risk of facility disruptions, Industrial and Engineering Chemistry Research 53 (2014) 17240–17251.

[106] M. Held, P. Wolfe, H. Crowder, Validation of subgradient optimization, Mathematical Programming 6 (1974) 62–88.

[107] M. Fisher, An applications oriented guide to Lagrangian relaxation, Interfaces 12 (1985) 10–21.

[108] IBM, IBM ILOG CPLEX OPTIMIZER: High-performance mathematical programming engine., Available at https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/ (2014).

[109] K. McLean, X. Li, Robust scenario formulations for strategic supply chain optimization under uncertainty, Industrial and Engineering Chemistry Research 52 (2013) 5721–5734.

[110] K. McClean, E. Ogbe, X. Li, Novel formulation and efficient solution strategy for strategic optimization of an industrial chemical supply chain under demand uncertainty, The Canadian Journal of Chemical Engineering 93 (2015) 971–985.

[111] A. Brook, D. Kendrick, A. Meeraus, GAMS, a user's guide, SIGNUM Newsletter 23 (3-4) (1988) 10–11.

[112] M. R. Bussieck, M. C. Ferris, T. Lohmann, GUSS: Solving Collections of Data Related Models Within GAMS, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 35–56.

[113] C. A. Holloway, A generalized approach to Dantzig-Wolfe decomposition for concave programs, Operations Research 21 (1) (1973) 210–220.

[114] S. Mitra, P. Garcia-Herreros, I. Grossmann, A cross-decomposition scheme with integrated primal-dual multi-cuts for two-stage stochastic programming investment planning problems, Math. Prog. 157 (1) (2016) 95–119.

[115] E. Ogbe, X. Li, A new cross decomposition method for stochastic mixed-integer linear programming, European Journal of Operational Research.

[116] F. You, I. E. Grossmann, Multicut Benders decomposition algorithm for process supply chain planning under uncertainty, Annals of Operations research 210 (1) (2013) 199 – 211.

[117] X. Li, E. Armagan, A. Tomasgard, P. I. Barton, Stochastic pooling problem for natural gas production network design and operation under uncertainty, AIChE Journal 57 (2011) 2120–2135.

[118] W. Li, C.-W. Hui, P. Li, A.-X. Li, Refinery planning under uncertainty, Industrial & Engineering Chemistry Research 43 (21) (2004) 6742–6755.

[119] N. Sahinidis, I. Grossmann, Convergence properties of generalized Benders decomposition, Computers and Chemical Engineering 15 (7) (1991) 481 – 491.

[120] N. Miller, A. Ruszczynski, Risk-averse two-stage stochastic linear programming: Modeling and decomposition, Operations Research 59 (2011) 125–132.

[121] N. Noyan, Risk-averse two-stage stochastic programming with an application to disaster management, Computers and Operations Research 39 (2012) 541–559.

[122] A. Barbaro, M. J. Bagajewicz, Managing financial risk in planning under uncertainty, AIChE Journal 50 (5) (2004) 963–989.

[123] R. Rockafellar, S. Uryasev, Optimization of conditional value-at-risk, Journal of Risk 2 (2000) 21–41.

[124] Q. Wang, Y. Guan, J. Wang, A chance-constrained two-stage stochastic program for unit commitment with uncertain wind power output, IEEE Transactions on Power Systems 27 (1) (2012) 206–215.

[125] S. Bruno, C. Sagastizabal, Optimization of real asset portfolio using a coherent risk measure:application to oil and energy industries, Optim. Eng. 12 (2011) 257–275.

[126] E. Ogbe, X. Li, A new cross decomposition method for stochastic mixed-integer linear programming, European Journal of Operational Research 256 (2) (2017) 487 – 499.

[127] S. R. Cardoso, A. P. Barbosa-Póvoa, S. Relvas, Integrating financial risk measures into the design and planning of closed-loop supply chain, Computers and Chemical Engineering 85 (2016) 105 – 123.

[128] R. Rockafellar, S. Uryasev, Portfolio optimization with conditional value-at-risk objective and constraints, Journal of Risk 4 (2002) 43–68.

[129] R. Rockafellar, S. Uryasev, Conditional value-at-risk for general loss distribution, Journal of Banking and Finance 26 (2002) 1443–1471.

[130] Z. Huang, Q. P. Zheng, Decomposition-based exact algorithms for risk-constrained traveling salesman problems with discrete random arc costs, Optimization Letters 9 (8) (2015) 1553–1568.

[131] R. Schultz, S. Tiedemann, Conditional value-at-risk in stochastic programs with mixed-integer recourse, Mathematical Programming 105 (2006) 365–386.

[132] Q. Zhang, J. L. Cremer, I. E. Grossmann, A. Sundaramoorthy, J. M. Pinto, Risk-based integrated production scheduling and electricity procurement for continuous power-intensive processes, Computers and Chemical Engineering 86 (2016) 90 – 105.

[133] M. Carneiro, G. Ribas, , S. Hamacher, Risk management in oil and gas industries: A cvar approach, Industrial & Engineering Chemistry Research 49 (2010) 3286–3294.

[134] P. Verderame, C. Floudas, Multisite planning under demand and transportation time uncertainty: Robust optimization and conditional value-at-risk frameworks, Industrial and Engineering Chemistry Research 50 (2010) 4959–4982.

[135] P. Verderame, C. Floudas, Operational planning of large-scale industrial batch plants under demand and amount uncertainty: II. conditional value-at-risk framework, Industrial and Engineering Chemistry Research 49 (2010) 260–275.

[136] C. Fabian, A. Veszpremi, Algorithms for handling cvar constraints in dynamic stochastic programming models with applications to finance, Journal of Risk 10 (2008) 111–131.

[137] C. A. Floudas, Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications, Oxford University Press, 1995.

[138] O. Berman, N. Ashrafi, Optimization models for reliability of modular software systems, IEEE Trans. Softw. Eng. 19 (1993) 1119–1123.

[139] C. A. Floudas, P. M. Pardalos, C. S. Adjiman, W. R. Esposito, Z. Gumus, S. T. Harding, J. L. Klepeis, C. A. Meyer, C. A. Schweiger, Handbook of Test Problems for Local and Global Optimization, Kluwer Academic Publishers, 1999.

[140] C. S. Adjiman, I. P. Androulakis, C. A. Floudas, Global optimization of mixed-integer nonlinear problems, AIChE Journal 46 (9) (2000) 1769–1797.

[141] M. Duran, I. Grossmann, A mixed-integer nonlinear programming algorithm for process systems synthesis, AIChE Journal 32 (1986) 592–606.

[142] J. Bloom, Solving an electricity generating capacity expansion problem by generalized benders' decomposition, Operations Research 31 (1983) 84–100.

[143] J. Falk, R. Soland, An algorithm for seperable nonconvex programming problems, Management Science 15 (1969) 550–569.

[144] R. Soland, An algorithm for seperable nonconvex programming problems II: Nonconvex constraints, Management Science 17 (1971) 759–772.

[145] J. Zamora, I. E. Grossmann, A branch and contract algorithm for problems with concave univariate, binear and linear fractional terms, Journal of Global Optimization 14 (1999) 217 – 249.

[146] H. S. Ryoo, N. V. Sahinidis, A branch-and-reduce approach to global optimization, Journal of Global Optimization 8 (1996) 107–138.

[147] R. Misener, C. A. Floudas, A framework for globally optimizing mixed-integer signomial programs, Journal of Optimization Theory and Applications 161 (2014) 905–932.

[148] R. Misener, C. A. Floudas, ANTIGONE: Algorithms for coNTinuous / Integer Global Optimization of Nonlinear Equations, Journal of Global Optimization 59 (2014) 503–526.

[149] P. Kesavan, R. J. Allgor, E. P. Gatzke, P. I. Barton, Outer approximation algorithms for separable nonconvex mixed-integer nonlinear programs., Mathematical Programming, Series A 100 (2004) 517–535.

[150] N. Deep, S. Shahidehpour, Cross decomposition for multi-area optimal reactive power planning, IEEE transactions on power systems 8 (1993) 1539–1544.

[151] E. Ogbe, X. Li, Multicolumn-multicut cross decomposition for stochastic mixed-integer linear programming, Computer Aided Chemical Engineering 37 (2015) 737–742.

[152] C. Maranas, C. A. Floudas, Global optimization in generalized geometric programming, Computers and Chemical Engineering 21 (4) (1997) 351–369.

[153] C. Haverly, Studies of the behaviour of recursion for the pooling problem., ACM SIGMAP Bulletin 25 (1978) 29–32.

[154] C. Haverly, Behaviour of recursion model: More studies., ACM SIGMAP Bulletin 26 (1979) 22–28.

[155] A. Selot, L. K. Kuok, M. Robinson, T. L. Mason, P. I. Barton, A short-term operational planning model for natural gas production systems., AIChE Journal 54 (2) (2008) 495–515.

[156] A. S. Drud, CONOPT A large-scale GRG code, ORSA Journal on Computing 6 (1994) 207–216.

[157] MATLAB, version 8.1.0 (R2013a), The MathWorks Inc., Natick, Massachusetts, 2013.

[158] S. Dirse, M. Ferris, J. Ramakrishnan, GDXMRW: Interfacing GAMS and MATLAB, Available at http://www.gams.com/dd/docs/tools/gdxmrw.pdf (2014).

[159] P. Wentges, Accelerating Benders' decomposition for the capacitated facility location problem, Mathematical Methods of Operations Research 44 (2) (1996) 267–290.

Appendix A

Proof of Propositions from chapter 2

Proof of Proposition 2.1. The equivalence holds when $\Lambda_{opt} = \Lambda_{feas} = \{\lambda \in \mathbb{R}^m : \lambda \geq 0\}$ due to the strong Lagrangian duality [35]. Furthermore, from the LP duality, Λ_{opt} or Λ_{feas} only needs to include all extreme dual multipliers of Problem (BPP^k) or Problem (BFP^k) for the equivalence of Problem (BMP) and Problem (P) [30]. □

Proof of Proposition 2.2. Since X is a bounded polyhedral set, $conv(E(X)) = X$, so

$$conv(\{x^1, \cdots, x^{n_F}\}) \supset conv(E(X)) = X.$$

On the other hand, $\{x^1, \cdots, x^{n_F}\} \subset X$ implies

$$conv(\{x^1, \cdots, x^{n_F}\}) \subset X.$$

Therefore, $conv(\{x^1, \cdots, x^{n_F}\}) = X$ and Problem (DWMP) is equivalent to Problem (P). □

Proof of Proposition 2.3. According to strong duality of Problem (BPP^k),

$$\begin{aligned} obj_{BPP^k} &= \inf_{x\in X} \left(c_0^T x_0^k + c^T x + (\lambda^k)^T (A_0 x_0^k + Ax - b_0)\right) \\ &= \inf_{x\in X} \left(c^T x + (\lambda^k)^T Ax\right) + (c_0^T + (\lambda^k)^T A_0) x_0^k - (\lambda^k)^T b_0, \end{aligned}$$

so

$$\begin{aligned} & obj_{BPP^k} + \left(c_0^T + (\lambda^k)^T A_0\right)\left(x_0 - x_0^k\right) \\ = & \inf_{x\in X} \left(c^T x + (\lambda^k)^T Ax\right) + (c_0^T + (\lambda^k)^T A_0) x_0 - (\lambda^k)^T b_0, \end{aligned}$$

Similarly, according to strong duality of Problem (DWPP^k),

$$obj_{DWPP^k} = \inf_{x\in X} \left(c^T x + (\lambda^k)^T Ax\right),$$

so

$$\begin{aligned} & obj_{DWPP^k} + \left(c_0^T + (\lambda^k)^T A_0\right) x_0 - (\lambda^k)^T b_0 \\ = & \inf_{x\in X} \left(c^T x + \left(\lambda^k\right)^T Ax\right) + \left(c_0^T + (\lambda^k)^T A_0\right) x_0 - (\lambda^k)^T b_0. \end{aligned}$$

□

Proof of Proposition 2.4. According to strong duality of Problem (BFP^k),

$$\begin{aligned} obj_{BFP^k} &= \inf_{x\in X, z\geq 0} \left(||z|| + (\lambda^k)^T (A_0 x_0 + Ax - b_0 - z)\right) \\ &= \inf_{z\geq 0} (||z|| - (\lambda^k)^T z) + \inf_{x\in X} (\lambda^k)^T Ax + (\lambda^k)^T (A_0 x_0^k - b_0) \end{aligned}$$

Since obj_{BFP^k} is finite and $\inf_{x\in X} (\lambda^k)^T Ax$ is finite (as set X is compact), $\inf_{z\geq 0}(||z|| - (\lambda^k)^T z)$ is finite. When $z = 0$, $||z|| - (\lambda^k)^T z = 0$, so $\inf_{z\geq 0}(||z|| - (\lambda^k)^T z) \leq 0$. Next

we prove that $\inf_{z\geq 0}(||z|| - (\lambda^k)^T z) = 0$ by contradiction. Suppose $\exists \hat{z} > 0$ such that $\exists \epsilon > 0$, $||\hat{z}|| - (\lambda^k)^T \hat{z} \leq -\epsilon$, then $\forall \alpha > 0$, $||\alpha\hat{z}|| - (\lambda^k)^T \alpha\hat{z} \leq -\alpha\epsilon$, which implies that $\inf_{z\geq 0}(||z|| - (\lambda^k)^T z) = -\infty$, which contradicts that $\inf_{z\geq 0}(||z|| - (\lambda^k)^T z)$ is finite. Therefore, $\forall z > 0$, $||z|| - (\lambda^k)^T z) \geq 0$, so $\inf_{z\geq 0}(||z|| - (\lambda^k)^T z) = 0$. As a result, the above expression can be simplified as:

$$obj_{BFP^k} = \inf_{x\in X} (\lambda^k)^T Ax + (\lambda^k)^T (A_0 x_0^k - b_0),$$

so

$$\begin{aligned} & obj_{BFP^k} + (\lambda^k)^T A_0 \left(x_0 - x_0^k\right) \\ = & \inf_{x\in X} (\lambda^k)^T Ax + (\lambda^k)^T (A_0 x_0^k - b_0) + (\lambda^k)^T A_0 \left(x_0 - x_0^k\right) \\ = & \inf_{x\in X} (\lambda^k)^T Ax + (\lambda^k)^T (A_0 x_0 - b_0). \end{aligned}$$

□

Proof of Proposition 2.5. According to Proposition 2.3, the first and the third group of constraints in Problem (BRMP$_r^k$) are valid Benders optimality cuts, and according to Proposition 2.4, the second group of constraints in Problem (BRMP$_r^k$) are valid Benders feasibility cuts. So according to Proposition 2.1, Problem (BRMP$_r^k$) is a relaxation of Problem (BMP) and therefore a valid lower bounding problem for Problem (P). □

Proof of Proposition 2.6. Since the columns x^i involved in Problem (DWRMP^k) come from the solutions of Problems (BPP^k), (BFP^k), and (DWPP^k), so they are all points in set X. Therefore, the feasible set of Problem (DWRMP^k) is a subset of that of the master problem (DWMP), and so according to Proposition 2.2, Problem (DWRMP^k)

is a valid upper bounding problem for Problem (P). □

Proof of Proposition 2.7. According to Propositions 2.3 and 2.4, the first three groups of constraints are valid cuts for the BD relaxed master problem. So it is left to prove that the fourth group of constraints are also valid cuts. $\forall i \in U_{feas}^k$, according to strong duality of Problem (DWFP^i),

$$obj_{DWFP^i} = \inf_{x \in X} (\lambda^i)^T Ax,$$

so

$$obj_{DWFP^i} + (\lambda^i)^T (A_0 x_0 - b_0) = \inf_{x \in X} (\lambda^i)^T Ax + (\lambda^i)^T (A_0 x_0 - b_0).$$

Thus the fourth group of constraints in Problem (BRMP^k) are valid cuts, according to Proposition 2.1. Therefore, Problem (BRMP^k) is a valid lower bounding problem for Problem (P). □

Proof of Proposition 2.8. This can be proved by showing that the optimal value of Problem DWRMP or DWFRMP cannot keep decreasing for an infinite number of iterations. Let's consider Problem DWRMP first. After each DWD iteration, Problem DWRMP is updated with a column that is the solution of Problem DWPP. The solution of DWPP returned by a solver is one of the extreme points of set X (considering Assumption 2.2), so the total number of new columns can be generated from solving Problem DWPP is finite. As a result, Problem (DWRMP^k) will remain the same after a finite number of steps and its optimal objective value will not change thereafter. Similarly, the optimal value of Problem (DWFRMP^k) will not change after a finite number of steps. □

Appendix B

From chapter 4

B.1 Derivation of CVaR constraints

B.1.1 Background of CVaR

Suppose $u \in \mathbb{R}^p$ is a random vector representing uncertainty and is governed by a probability measure P on a set Y that is independent of the decision vector x. To define the CVaR, we first define a loss function $f : \mathbb{R}^n \times \mathbb{R}^p \mapsto \mathbb{R}$. For a general distribution, we define the distribution function $\chi : \mathbb{R}^n \times \mathbb{R}_{>0} \mapsto \mathbb{R}$ as:

$$\chi(x, \zeta) = P\{u : f(x, u) \leq \zeta\}$$

The value-at-risk, VaR, for a loss random variable associated with x within a specified probability interval $(0, 1)$ is given by:

$$\zeta_\beta(x) = \min\{\zeta \in \mathbb{R} : \chi(x, \zeta) \geq \beta\}$$

The conditional expectation of the loss function above a certain value $\zeta_\beta(x)$, $CVaR_\beta : \mathbb{R}_{>0} \mapsto \mathbb{R}$ introduced by [123] [128] is defined as the following:

$CVaR_\beta(x)$ =mean of the $\beta - tail$ distribution of $f(x,u)$

According to Rockfellar and Uryasev, we can define a function $F : \mathbb{R}^n \times \mathbb{R}_{>0} \mapsto \mathbb{R}$ for general distributions as:

$$F_\beta(x,\zeta) = \zeta + \frac{1}{(1-\beta)}\mathbb{E}\{[f(x,u)-\zeta]^+\}$$

where $[f(x,u)-\zeta]^+ = \max\{0, f(x,u)-\zeta\}$

The following results show the relationship between $CVaR_\beta(x)$ and $F_\beta(x,\zeta)$.

Rockefellar and Uryesav [123] [128] $F_\beta(x,\zeta)$ is convex and continuously differentiable as a function of ζ. The $CVaR_\beta$ of the loss associated with any $x \in X \subset \mathbb{R}^n$ is given by:

$$CVaR_\beta(x) = \min_{\zeta\in\mathbb{R}} F_\beta(x,\zeta)$$

if $f(x,u)$ is convex w.r.t x, then $F_\beta(x,\zeta)$ is convex on (x,ζ).

Rockefellar and Uryesav [123] [128] The problem of minimizing the $CVaR_\beta(x)$, $\forall x \in X \subset \mathbb{R}^n, F_\beta(x,\zeta)$ is equivalent to the problem of minimizing $F_\beta(x,\zeta)$ over all $(x,\zeta) \in X \times \mathbb{R}$ implying that:

$$\min_{x\in X} CVaR_\beta(x) = \min_{(x,\zeta)\in X\times\mathbb{R}} F_\beta(x,\zeta)$$

The proof of the above results are in the reference [123, 128].

B.1.2 Discretization and Linearization

$F_\beta(x, \zeta)$ can be approximated by a summation by sampling the probability distribution over many scenario realization $\omega \in \{1, ..., s\}$. The loss function is then given by $f(x, u_\omega)$ and can be determined in part by sampling the underlying distribution of the stochastic vector u as well.

$$\tilde{F}_\beta(x, \zeta) = \zeta + \frac{1}{(1-\beta)} \sum_{\omega \in \{1,...,s\}} p_\omega [f(x, u_\omega) - \zeta]^+$$

where ω is the scenario realization and p_ω is the probability that a given scenario ω, will occur. From the above results, $F_\beta(x, \zeta)$, after discretization, can been approximated by the function $\tilde{F}_\beta(x, \zeta)$. Furthermore, using dummy variables ψ_ω, $\omega \in \{1, ..., s\}$, the function $\tilde{F}_\beta(x, \zeta)$ can be replaced by the affine function

$$\tilde{F}_\beta(x, \zeta) = \zeta + \frac{1}{(1-\beta)} \sum_{\omega \in \{1,...,s\}} p_\omega \psi_\omega$$

where,

$$\psi_\omega \geq f(x, u_\omega) - \zeta, \psi_\omega \geq 0, \forall \omega \in \{1, ..., s\}, \zeta \in \mathbb{R}$$

The above equation is the scenario based formulation of CVaR. If the loss function $f(x, u_\omega)$ is affine with respect to x, then the function $\tilde{F}_\beta$ is convex and piecewise linear. $\tilde{F}_\beta$ is convex if $f(x, u_\omega)$ is convex with respect to x.

B.2 ECD subproblems for CVaR-constrained two-stage stochastic programming

The extended cross decomposition (ECD) was presented earlier for a generic Problem (P). In this section, we apply the ECD to Problem (CVaR-SP). In the ECD, we treat the variables that are independent of scenarios, (x_0, ζ) different from the variables (x_ω, ψ_ω), $\forall \omega \in \{1, ..., s\}$, associated with scenarios. The ECD subproblems for two-stage stochastic programming with CVaR constraints are described in the subsequent subsections. They are presented under the Phase I and Phase II subproblems subsections.

First, the following subproblem is solved to provide initial point or column for the algorithm to commence.

$$\begin{aligned} \min_{x_\omega, \psi_\omega} \ & c_\omega^{\mathrm{T}} x_\omega \\ & x_\omega \in X_\omega. \end{aligned} \tag{CVaR-IPP$_\omega$}$$

Let $x_\omega^\star$ be the optimal solution of Problem (CVaR-IPP$_\omega$). The initial feasible column needed to start the algorithm is $(x_\omega^0, \psi_\omega^0)$, where $x_\omega^0 = x_\omega^\star$ and $\psi_\omega^0 = 0$.

B.2.1 Phase I subproblems

The DW restricted master problem to solve at the upper level for Phase I is given as:

$$
\begin{aligned}
\min_{\substack{x_0,\zeta \\ \theta^0,\ldots,\theta^{t-1}\geq 0 \\ z_{1,1},\ldots,z_{2,s} \\ z_{2,1},\ldots,z_{2,s} \\ z_3}} \quad & || \left(z_{1,1},...,z_{1,s}\right) || + || \left(z_{2,1},...,z_{2,s}\right) || + ||z_3|| \\
\text{s.t.} \quad & A_\omega x_0 + A_\omega \sum_{i=0}^{t-1} \theta^i x_\omega^i \leq b_{0,\omega} + z_{1,\omega}, \quad \omega \in \{1,...,s\}, \\
& -\sum_{i=0}^{t-1} \theta^i {\psi_\omega}^i + f_\omega(\sum_{i=0}^{t-1} \theta^i x_\omega^i) - \zeta \leq z_{2,\omega}, \quad \omega \in \{1,...,s\}, \\
& \zeta + \sum_{\omega=1}^{s} \hat{p}_\omega (\sum_{i=0}^{t-1} \theta^i \psi_\omega^i) \leq b_0 + z_3, \\
& \sum_{i=0}^{t-1} \theta^i = 1, \\
& x_0 \in X_0, \zeta \in \mathbb{R}, z_{1,\omega}, z_{2,\omega} \geq 0, \forall \omega \in \{1,...,s\}, z_3 \geq 0.
\end{aligned}
\tag{CVaR-DWFRMP^k}
$$

The 1-norm is used for the case studies. Let $\mu_{1,\omega}^k$, $\mu_{2,\omega}^k$ be the Lagrange multiplier associated with the weak linking constraints (first and second group of constraints), and, μ_3^k be associated with the strong linking constraints (third group of constraints).

The DW pricing problem for Phase I is given below:

$$
\begin{aligned}
\min_{x_\omega,\psi_\omega} \quad & \left(\mu_{1,\omega}^k\right)^{\mathrm{T}} A_\omega x_\omega + \mu_{2,\omega}^k(-\psi_\omega + f_\omega(x_\omega)) + \left(\mu_3^l\right) \hat{p}_\omega \psi_\omega \\
& x_\omega \in X_\omega, \psi_\omega \geq 0.
\end{aligned}
\tag{CVaR-DWFP$_\omega^k$}
$$

Let the solution be given by $(x_\omega^k, \psi_\omega^k)$ and the optimal objective $obj_{CVaR-DWFP^k} = \sum_{\omega=1}^{s} obj_{CVaR-DWFP_\omega^k}$. The solution to Problem (CVaR-DWFP$_\omega^k$) provides a column for Problem (CVaR-DWFRMP^k).

The DW restricted master problem at the lower level can be written as:

$$
\begin{aligned}
\min_{\substack{\theta^0,\ldots,\theta^{l-1}\geq 0\\ z_{1,1},\ldots,z_{2,s}\\ z_{2,1},\ldots,z_{2,s}\\ z_3}} \; & || \left(z_{1,1},...,z_{1,s}\right) || + || \left(z_{2,1},...,z_{2,s}\right) || + ||z_3|| \\
\text{s.t. } & A_\omega \sum_{i=0}^{l-1} \theta^i x_\omega^{k,i} \leq b_{0,\omega} - A_\omega x_0^k + z_{1,\omega}, \quad \omega \in \{1,...,s\}, \\
& -\sum_{i=0}^{l-1} \theta^i {\psi_\omega}^{k,i} + f_\omega(\sum_{i=0}^{l-1} \theta^i x_\omega^{k,i}) \leq \zeta^k + z_{2,\omega}, \quad \omega \in \{1,...,s\}, \\
& \sum_{\omega=1}^{s} \hat{p}_\omega (\sum_{i=0}^{l-1} \theta^i \psi_\omega^{k,i}) \leq -\zeta^k + b_0 + z_3, \\
& \sum_{i=0}^{l-1} \theta^i = 1, \\
& z_{1,\omega}, z_{2,\omega} \geq 0, \forall \omega \in \{1,...,s\}, z_3 \geq 0.
\end{aligned}
\tag{CVaR-DWFRMPR^{k,l}}
$$

Let $\mu_3^{k,l}$ be the Lagrange multiplier associated with the strong linking constraints for (CVaR-DWFRMPR^{k,l}).

The DW pricing problem at the lower level can be written as:

$$
\begin{aligned}
\min_{x_\omega,\psi_\omega,z_{1,\omega},z_{2,\omega}} \; & \left(\mu_3^{k,l}\right) \hat{p}_\omega \psi_\omega + ||z_{1,\omega}|| + ||z_{2,\omega}|| \\
\text{s.t.} \quad & A_\omega x_\omega \leq -A_{0,\omega} x_0^k + b_{0,\omega} + z_{1,\omega}, \\
& -\psi + f_\omega(x_\omega) \leq \zeta^k + z_{2,\omega}, \\
& x_\omega \in X_\omega, \psi_\omega \geq 0, \\
& z_{1,\omega}, z_{2,\omega} \geq 0,
\end{aligned}
\tag{CVaR-DWFPR$_\omega^{k,l}$}
$$

Again, let the solution of (CVaR-DWFP$^k_\omega$) be given by $(x^{k,l}_\omega, \psi^{k,l}_\omega)$ and the optimal objective $obj_{CVaR-DWFPR^{k,l}} = \sum_{\omega=1}^{s} obj_{CVaR-DWFPR^{k,l}_\omega}$. The solution provides a column for Problem (CVaR-DWFRMPR^{k,l}). An initial pricing problem is solved to generate a starting column for the DWD procedure and was given previously.

The relaxed master problem to solve at Phase I is now given as the following:

$$
\begin{aligned}
&\min_{x_0,\zeta,\eta} \ \eta \\
&\text{s.t.} \quad \eta \geq obj_{CVaR-DWFP^i} + \sum_{\omega=1}^{s} (\mu_{1,\omega}{}^i)^{\mathrm{T}} A_{0,\omega} x_0 - \sum_{\omega=1}^{s} \left(\mu_{1,\omega}{}^i\right) b_{0,\omega} - \sum_{\omega=1}^{s} \left(\mu_{2,\omega}{}^i\right) \zeta \\
&\qquad + \mu_3^i \zeta - \mu_3^i b_0, \quad \forall i \in U^k_{feas}, \\
&\qquad \eta \geq obj_{CVaR-DWFPR^{i,j}} + \sum_{\omega=1}^{s} (\mu_{1,\omega}{}^{i,j})^{\mathrm{T}} A_{0,\omega} (x_0 - x_0^i) - \sum_{\omega=1}^{s} \left(\mu_{2,\omega}{}^{i,j}\right) (\zeta - \zeta^i) \\
&\qquad + (\mu_3^{i,j}) \zeta - (\mu_3^{i,j}) b_0, \quad \forall (i,j) \in T^k_{feas}, \\
&\qquad x_0 \in X_0, \zeta \in \mathbb{R}, \eta \in \mathbb{R},
\end{aligned}
$$

(CVaR-BRMPI^k)

B.2.2 Phase II subproblems

Based on the columns obtained from the subproblems above, the following DW restricted master problem is solved at the upper level.

$$
\begin{aligned}
\min_{\substack{x_0,\zeta \\ \theta^0,\ldots,\theta^{t-1}\geq 0}} \; & \sum_{\omega=1}^{s}\left(c_{0,\omega}^{\mathrm{T}}x_0 + c_{\omega}^{\mathrm{T}}(\sum_{i=0}^{t-1}\theta^i x_{\omega}^i)\right) \\
\text{s.t. } & A_\omega x_0 + A_\omega \sum_{i=0}^{t-1}\theta^i x_\omega^i \leq b_{0,\omega}, \quad \omega \in \{1,...,s\}, \\
& -\sum_{i=0}^{t-1}\theta^i {\psi_\omega}^i + f_\omega(\sum_{i=0}^{t-1}\theta^i x_\omega^i) - \zeta \leq 0, \quad \omega \in \{1,...,s\}, \\
& \zeta + \sum_{\omega=1}^{s}\hat{p}_\omega(\sum_{i=0}^{t-1}\theta^i\psi_\omega^i) \leq b_0, \\
& \sum_{i=0}^{t-1}\theta^i = 1, \\
& x_0 \in X_0, \zeta \in \mathbb{R}.
\end{aligned}
\tag{CVaR-DWRMP^k}
$$

Let $\lambda_{1,\omega}^k$, $\lambda_{2,\omega}^k$, and λ_3^k, be the Lagrange multiplier associated with the first, second, and third group of constraints of (CVaR-DWRMP^k). The pricing problem is the following decomposable problem:

$$
\begin{aligned}
\min_{x_\omega,\psi_\omega} \; & \left(c_\omega^{\mathrm{T}} + \left(\lambda_{1,\omega}^k\right)^{\mathrm{T}} A_\omega\right) x_\omega + \lambda_{2,\omega}^k(-\psi_\omega + f_\omega(x_\omega)) + \left(\lambda_3^k\right)\hat{p}_\omega\psi_\omega \\
& x_\omega \in X_\omega, \psi_\omega \geq 0.
\end{aligned}
\tag{CVaR-DWPP$_\omega^k$}
$$

Let the solution of (CVaR-DWPP$_\omega^k$) be given by $(x_\omega^k, \psi_\omega^k)$ and the optimal objective $obj_{CVaR-DWPP^k} = \sum_{\omega=1}^{s} obj_{CVaR-DWPP_\omega^k}$. The solution provides a column for Problem (CVaR-DWRMP^k). The Phase II DW restricted master problem solved at the lower level is given as:

$$
\begin{aligned}
\min_{\theta^0,\ldots,\theta^{l-1}\geq 0} \; & \sum_{\omega=1}^{s}\left(c_{0,\omega}^{\mathrm{T}}x_0^k + c_\omega^{\mathrm{T}}(\sum_{i=0}^{l-1}\theta^i x_\omega^i)\right) \\
\text{s.t.} \; & A_\omega \sum_{i=0}^{l-1}\theta^i x_\omega^i \leq b_{0,\omega} - A_\omega x_0^k, \quad \omega\in\{1,\ldots,s\}, \\
& -\sum_{i=0}^{l-1}\theta^i {\psi_\omega}^i + f_\omega(\sum_{i=0}^{l-1}\theta^i x_\omega^i) \leq \zeta^k, \quad \omega\in\{1,\ldots,s\}, \\
& \sum_{\omega=1}^{s}\hat{p}_\omega(\sum_{i=0}^{l-1}\theta^i\psi_\omega^i) \leq -\zeta^k + b_0, \\
& \sum_{i\in J^i}\theta^i = 1,
\end{aligned}
\tag{CVaR-DWRMPR^{k,l}}
$$

Let $\lambda_3^{k,l}$ be the Lagrange multiplier associated with the strong linking constraints. Similarly, the Phase II DW pricing problem solved at the lower level is the following:

$$
\begin{aligned}
\min_{x_\omega,\psi_\omega} \; & \left(c_\omega^{\mathrm{T}} + \left(\lambda_{1,\omega}^{k,l}\right)^{\mathrm{T}} A_\omega\right) x_\omega + \lambda_3^{k,l}\left(\hat{p}_\omega\psi_\omega\right) \\
\text{s.t.} \; & A_\omega x_\omega = -A_{0,\omega}x_0^k + b_{0,\omega}, \quad \omega\in\{1,\ldots,s\}, \\
& -\psi_\omega + f_\omega(x_\omega) \leq \zeta^k, \quad \omega\in\{1,\ldots,s\}, \\
& x_\omega \in x_\omega, \psi_\omega \geq 0.
\end{aligned}
\tag{CVaR-DWPPR$^{k,l}_\omega$}
$$

Again let the solution of Problem (CVaR-DWPPR$^{k,l}_\omega$) be given by $(x_\omega^{k,l}, \psi_\omega^{k,l})$ with the optimal objective $obj_{CVaR-DWPPR^{k,l}} = \sum_{\omega=1}^{s} obj_{CVaR-DWPPR_\omega^{k,l}}$. The solution provides a column for Problem (CVaR-DWRMPR^{k,l}). The Phase II relaxed master problem, a lower bound to Problem (CVaR-SP) is the following:

$$\min_{x_0,\zeta,\eta} \eta$$

$$\begin{aligned}
\text{s.t.} \quad & \eta \geq obj_{CVaR-DWPP^i} + \sum_{\omega=1}^{s} \left(c_{0,\omega}^{\mathrm{T}} + (\lambda_{1,\omega}{}^i)^{\mathrm{T}} A_{0,\omega}\right) x_0 - \sum_{\omega=1}^{s} \left(\lambda_{1,\omega}{}^i\right)^{\mathrm{T}} b_{0,\omega} \\
& - \sum_{\omega=1}^{s} \lambda_{2,\omega}{}^i \zeta + \lambda_3^i(\zeta - b_0), \quad \forall i \in U_{opt}^k, \\
& 0 \geq obj_{CVaR-DWFP^i} + \sum_{\omega=1}^{s} (\mu_{1,\omega}{}^i)^{\mathrm{T}} A_{0,\omega} x_0 - \sum_{\omega=1}^{s} \left(\mu_{1,\omega}{}^i\right)^{\mathrm{T}} b_{0,\omega} - \sum_{\omega=1}^{s} \mu_{2,\omega}{}^i \zeta \\
& + \mu_3^i(\zeta - b_0), \forall i \in U_{feas}^k, \\
& \eta \geq obj_{CVaR-DWPPR^{i,j}} + \sum_{\omega=1}^{s} c_{0,\omega}^{\mathrm{T}} x_0 + \sum_{\omega=1}^{s} (\lambda_{1,\omega}{}^{i,j})^{\mathrm{T}} A_{0,\omega}(x_0 - x_0^i) \\
& - \sum_{\omega=1}^{s} \lambda_{2,\omega}{}^{i,j}(\zeta - \zeta^i) + \lambda_3^{i,j}(\zeta - b_0), \quad \forall (i,j) \in T_{opt}^k, \\
& 0 \geq obj_{CVaR-DWFPR^{i,j}} + \sum_{\omega=1}^{s} (\mu_{1,\omega}{}^{i,j})^{\mathrm{T}} A_{0,\omega}(x_0 - x_0^i) - \sum_{\omega=1}^{s} \mu_{2,\omega}{}^{i,j}(\zeta - \zeta^i) \\
& \quad + \mu_3^{i,j}(\zeta - b_0), \forall (i,j) \in T_{feas}^k, \\
& x_0 \in X_0, \zeta \in \mathbb{R}, \eta \in \mathbb{R}.
\end{aligned}$$

(CVaR-BRMPII^k)

B.3 BLD subproblems for CVaR-constrained two-stage stochastic programming

The lower level subproblems for bilevel decomposition are same to that for ECD. At the upper level however, the following Benders relaxed master problem is solved.

$$\min_{x_0,\zeta,\eta} \eta$$

$$\begin{aligned}
\text{s.t.} \quad & \eta \geq obj_{CVaR-DWPPR^{i,j}} + \sum_{\omega=1}^{s} c_{0,\omega}^{\mathrm{T}} x_0 + \sum_{\omega=1}^{s} (\lambda_{1,\omega}{}^{i,j})^{\mathrm{T}} A_{0,\omega}(x_0 - x_0^i) \\
& - \sum_{\omega=1}^{s} \lambda_{2,\omega}{}^{i,j}(\zeta - \zeta^i) + \lambda_3^{i,j}(\zeta - b_0), \quad \forall (i,j) \in T_{opt}^k, \\
& 0 \geq obj_{CVaR-DWFPR^{i,j}} + \sum_{\omega=1}^{s} (\mu_{1,\omega}{}^{i,j})^{\mathrm{T}} A_{0,\omega}(x_0 - x_0^i) - \sum_{\omega=1}^{s} \mu_{2,\omega}{}^{i,j}(\zeta - \zeta^i) \\
& + \mu_3^{i,j}(\zeta - b_0), \forall (i,j) \in T_{feas}^k, \\
& x_0 \in X_0, \zeta \in \mathbb{R}, \eta \in \mathbb{R}.
\end{aligned}$$

(CVaR-BRMP-BLD^k)

B.4 Some details of the case study problem

The case study problem is a modification of Uncertain Case A in [109]. The changes include (a) the different uncertain parameter settings, (b) the additional CVaR constraints, and (c) some parameter values. The first two changes are already described in section 4.4. The new parameter values are given below:

- The fixed investment cost of incineration: 3 Million \$.
- The variable investment cost of incineration: 90.32 \$/t.
- The capacity of a technology facility: dry-grind 100,000 t/year, digestion 16,000 t/year, incineration 200,000 t/year.
- The maximum demand for heat at demand location j_3: 6.26×10^8 MJ/year.
- The maximum demand for electricity at each demand location: 6×10^6 MWh/year.

The upper threshold for the CVaR measure $b_0 = 0$, and the probability $\beta = 90\%$. Note that the upper and lower bounds of the loss function are needed to compute bounds of ζ and ψ_ω in the CVaR constraints. The upper bound of the loss function is estimated as the sum of largest possible minimum electricity demands at all demand locations. The lower bound of the loss function is estimated as the sum of smallest possible minimum electricity demands at all demand locations minus the total maximum demands at all demand locations.

Appendix C

From chapter 5

C.1 Reformulation from (P1) to (P)

Define new variables t_ω, $\alpha_{c,\omega}$, $\alpha_{nc,\omega}$, $\beta_{c,\omega}$, $\beta_{nc,\omega}$, such that Problem (P1) can be written as:

$$
\begin{aligned}
\min \ & \sum_{\omega=1}^{s} t_\omega \\
\text{s.t.} \ & x_0 = z_{0,\omega}, \quad \forall \omega \in \{1, ..., s\}, \\
& \beta_{c,\omega} + \beta_{nc,\omega} \leq 0, \quad \forall \omega \in \{1, ..., s\}, \\
& t_\omega \geq \alpha_{c,\omega} + \alpha_{nc,\omega}, \quad \forall \omega \in \{1, ..., s\}, \\
& \alpha_{c,\omega} \geq f_{0,\omega}(z_{0,\omega}) + f_{c,\omega}(z_{c,\omega}), \quad \forall \omega \in \{1, ..., s\}, \\
& \alpha_{nc,\omega} \geq f_{nc,\omega}(z_{nc,\omega}), \quad \forall \omega \in \{1, ..., s\}, \\
& \beta_{c,\omega} \geq g_{0,\omega}(z_{0,\omega}) + g_{c,\omega}(z_{c,\omega}), \quad \forall \omega \in \{1, ..., s\}, \\
& \beta_{nc,\omega} \geq g_{nc,\omega}(z_{nc,\omega}), \quad \forall \omega \in \{1, ..., s\}, \\
& x_0 \in X_0, \\
& z_{0,\omega} \in \hat{X}_0, \ z_{c,\omega} \in Z_{c,\omega}, \ z_{nc,\omega} \in Z_{nc,\omega}, \quad \forall \omega \in \{1, ..., s\}.
\end{aligned}
\tag{C.1}
$$

Define $x_\omega = (z_{0,\omega}, z_{c,\omega}, t_\omega, \alpha_{c,\omega}, \beta_{c,\omega})$, $y_\omega = (z_{nc,\omega}, \alpha_{nc,\omega}, \beta_{nc,\omega})$, then the above formulation can be written as the following Problem (P):

$$\begin{aligned} \min \ & \sum_{\omega=1}^{s} c_\omega^T x_\omega \\ \text{s.t.} \quad & x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1, ..., s\}, \\ & A_\omega x_\omega + B_\omega y_\omega \leq 0, \quad \forall \omega \in \{1, ..., s\}, \\ & x_0 \in X_0, \\ & x_\omega \in X_\omega, \ y_\omega \in Y_\omega, \quad \forall \omega \in \{1, ..., s\}, \end{aligned} \tag{C.2}$$

where the matrices

$$c_\omega = \begin{bmatrix} 0 \\ 0 \\ I \\ 0 \\ 0 \end{bmatrix}, \quad H_\omega = [I \ \ 0 \ \ 0 \ \ 0 \ \ 0], \quad A_\omega = \begin{bmatrix} 0 & 0 & 0 & 0 & I \\ 0 & 0 & -I & I & 0 \end{bmatrix}, \quad B_\omega = \begin{bmatrix} 0 & 0 & I \\ 0 & I & 0 \end{bmatrix},$$

and the sets

$$\begin{aligned} X_\omega = & \{(z_{0,\omega}, z_{c,\omega}, t_\omega, \alpha_{c,\omega}, \beta_{c,\omega}) : z_{0,\omega} \in \hat{X}_0, \ z_{c,\omega} \in Z_{c,\omega}, \\ & \alpha_{c,\omega} \geq f_{0,\omega}(z_{0,\omega}) + f_{c,\omega}(z_{c,\omega}), \ \ \beta_{c,\omega} \geq g_{0,\omega}(z_{0,\omega}) + g_{c,\omega}(z_{c,\omega})\}, \\ Y_\omega = & \{(z_{nc,\omega}, \alpha_{nc,\omega}, \beta_{nc,\omega}) : z_{nc,\omega} \in Z_{nc,\omega}, \ \alpha_{nc,\omega} \geq f_{nc,\omega}(z_{nc,\omega}), \beta_{nc,\omega} \geq g_{nc,\omega}(z_{nc,\omega})\}. \end{aligned}$$

The "0" and "I" in the matrices represent zero and identity matrices, and their dimensions are conformable to the relevant variables.

C.2 The stochastic pooling problem with mixed-integer first-stage decisions

The two-stage stochastic pooling problem from Li et al. [117] is modified here to address continuous design (first-stage) decisions. The nomenclature used in [117] is adopted to describe the model, in which the scenarios are indexed by h (rather than ω).

In the modified model, the design decisions on sources, pools, product terminals, denoted by y_i^{S}, y_j^{P}, y_k^{T}, can be continuous, integer, or mixed integer. If $y_i^S \in \{0,1\}$, then the design decision is to determine whether source i is to be developed, and the related parameter Z_i^{UB} represents the fixed capacity of the source. If y_i^S is continuous and $y_i^{\mathrm{S}} \in [0,1]$, then it is a capacity design decision, specifically it represents the ratio of source i capacity to the maximum allowed capacity of the source (denoted by Z_i^{UB}). The design decisions on the pipelines among sources, pools, and terminals are all continuous, denoted by $y_{i,j}^{\mathrm{SP}}$, $y_{i,k}^{\mathrm{ST}}$, y_{j,j^-}^{PP}, $y_{j,k}^{\mathrm{PT}} \in [0,1]$. They represent the ratios of the pipeline capacities to the maximum allowed capacities (denoted by $F_{i,j}^{\mathrm{SP,UB}}$, $F_{i,k}^{\mathrm{ST,UB}}$, $F_{j,j^-}^{\mathrm{PP,UB}}$, $F_{j,k}^{\mathrm{PT,UB}}$).

All design and operational decision variables are nonnegative, and we do not impose other lower bounds on these variables in order to simplify discussion. The new stochastic pooling model consists primarily of three submodels, for the sources, pools, and product terminals, respectively.

C.2.1 Model for the sources

The following group of constraints (C.3) represents the submodel for the sources. Eq. (C.1a-C.1c) are same to Eq. (12-14) in [117], except that the lower flow bounds are

not imposed. Eq. (C.1d-C.1f) are developed in place of the topology constraints Eq. (15-16) (which are invalid for continuous design decisions). Eq. (C.1d-C.1e) limit the capacity of a pipeline by the capacity of the source it connects. If $y_i^{\mathrm{S}} = 0$, then there cannot exist a pipeline connecting it, in other words, the capacity of a pipeline connecting it has to be zero. Eq. (C.1f) requires that the total capacity of all pipelines connecting to a source should be no less than the capacity of the source. This is to ensure enough pipeline capacity to move all materials generated in the source to other parts of the system in real-time.

$$\sum_{j\in\Theta_i^{\mathrm{SP}}} f_{i,j,h}^{\mathrm{SP}} + \sum_{k\in\Theta_i^{\mathrm{ST}}} f_{i,k,h}^{\mathrm{ST}} \leq y_i^{\mathrm{S}} Z_i^{\mathrm{UB}}, \tag{C.3a}$$

$$f_{i,j,h}^{\mathrm{SP}} \leq y_{i,j}^{\mathrm{SP}} F_{i,j}^{\mathrm{SP,UB}}, \tag{C.3b}$$

$$f_{i,k,h}^{\mathrm{ST}} \leq y_{i,k}^{\mathrm{ST}} F_{i,k}^{\mathrm{ST,UB}}, \tag{C.3c}$$

$$y_{i,j}^{\mathrm{SP}} F_{i,j}^{\mathrm{SP,UB}} \leq y_i^{\mathrm{S}} Z_i^{\mathrm{UB}}, \tag{C.3d}$$

$$y_{i,k}^{\mathrm{ST}} F_{i,k}^{\mathrm{ST,UB}} \leq y_i^{\mathrm{S}} Z_i^{\mathrm{UB}}, \tag{C.3e}$$

$$y_i^{\mathrm{S}} Z_i^{\mathrm{UB}} \leq \sum_{j\in\Theta_i^{\mathrm{SP}}} y_{i,j}^{\mathrm{SP}} F_{i,j}^{\mathrm{SP,UB}} + \sum_{k\in\Theta_i^{\mathrm{ST}}} y_{i,k}^{\mathrm{ST}} F_{i,k}^{\mathrm{ST,UB}}, \tag{C.3f}$$

$$\forall i \in \{1,...,n\},\ \forall j \in \Theta_i^{\mathrm{SP}},\ \forall k \in \Theta_i^{\mathrm{ST}},\ \forall h \in \{1,...,b\}.$$

C.2.2 Model for the pools

The following group of constraints (C.4) represents the submodel for the pools. Eq. (C.2a-C.1e) are same to Eq. (17-21) in [117], except that the lower flow bounds are not imposed. Eq. (C.2f-C.2k) are developed in place of the topology constraints (23-26) in [117]. The interpretation of Eq. (C.2f-C.2k) is similar to that of Eq. (C.1d-C.1f)

and therefore omitted.

$$f^{\mathrm{PT}}_{j,k,w,h} = s^{\mathrm{PT}}_{j,k,h}\left(\sum_{i\in\Omega^{\mathrm{SP}}_j} f^{\mathrm{SP}}_{i,j,h}U_{i,w,h} + \sum_{j^+\in\Omega^{\mathrm{PP}+}_j} f^{\mathrm{PP}}_{j^+,j,w,h}\right), \tag{C.4a}$$

$$f^{\mathrm{PP}}_{j,j^-,w,h} = s^{\mathrm{PP}}_{j,j^-,h}\left(\sum_{i\in\Omega^{\mathrm{SP}}_j} f^{\mathrm{SP}}_{i,j,h}U_{i,w,h} + \sum_{j^+\in\Omega^{\mathrm{PP}+}_j} f^{\mathrm{PP}}_{j^+,j,w,h}\right), \tag{C.4b}$$

$$\sum_{j^-\in\Omega^{\mathrm{PP}-}_j} s^{\mathrm{PP}}_{j,j^-,h} + \sum_{k\in\Omega^{\mathrm{PT}}_j} s^{\mathrm{PT}}_{j,k,h} = 1, \quad s^{\mathrm{PP}}_{j,j^-,h}, s^{\mathrm{PT}}_{j,k,h} \geq 0, \tag{C.4c}$$

$$y^{\mathrm{PP}}_{j,j^-}F^{\mathrm{PP,LB}}_{j,j^-} \leq \sum_{w\in\{1,\ldots,l\}} f^{\mathrm{PP}}_{j,j^-,w,h} \leq y^{\mathrm{PP}}_{j,j^-}F^{\mathrm{PP,UB}}_{j,j^-}, \tag{C.4d}$$

$$y^{\mathrm{PT}}_{j,k}F^{\mathrm{PT,LB}}_{j,k} \leq \sum_{w\in\{1,\ldots,l\}} f^{\mathrm{PT}}_{j,k,w,h} \leq y^{\mathrm{PT}}_{j,k}F^{\mathrm{PT,UB}}_{j,k}, \tag{C.4e}$$

$$y^{\mathrm{P}}_j Z^{\mathrm{P,UB}}_j \geq y^{\mathrm{SP}}_{i,j}F^{\mathrm{SP,UB}}_{i,j}, \tag{C.4f}$$

$$y^{\mathrm{P}}_j Z^{\mathrm{P,UB}}_j \geq y^{\mathrm{PP}}_{j^+,j}F^{\mathrm{PP,UB}}_{j^+,j}, \tag{C.4g}$$

$$y^{\mathrm{P}}_j Z^{\mathrm{P,UB}}_j \geq y^{\mathrm{PP}}_{j,j^-}F^{\mathrm{PP,UB}}_{j,j^-}, \tag{C.4h}$$

$$y^{\mathrm{P}}_j Z^{\mathrm{P,UB}}_j \geq y^{\mathrm{PT}}_{j,k}F^{\mathrm{PT,UB}}_{j,k}, \tag{C.4i}$$

$$y^{\mathrm{P}}_j Z^{\mathrm{P,UB}}_j \leq \sum_{j^+\in\Omega^{\mathrm{PP}+}_j} y^{\mathrm{PP}}_{j^+,j}F^{\mathrm{PP,UB}}_{j^+,j} + \sum_{i\in\Omega^{\mathrm{SP}}_j} y^{\mathrm{SP}}_{i,j}F^{\mathrm{SP,UB}}_{i,j}, \tag{C.4j}$$

$$y^{\mathrm{P}}_j Z^{\mathrm{P,UB}}_j \leq \sum_{j^-\in\Omega^{\mathrm{PP}\text{-}}_j} y^{\mathrm{PP}}_{j,j^-}F^{\mathrm{PP,UB}}_{j,j^-} + \sum_{k\in\Omega^{\mathrm{PT}}_j} y^{\mathrm{PT}}_{j,k}F^{\mathrm{PT,UB}}_{j,k}, \tag{C.4k}$$

$\forall j \in \{1,...,r\},\ \forall j^- \in \Omega^{\mathrm{PP}\text{-}}_j,\ \forall k \in \Omega^{\mathrm{PT}}_j,\ \forall w \in \{1,...,l\},\ \forall h \in \{1,...b\}.$

C.2.3 Model for the product terminals

The following group of constraints (C.5) represents the submodel for the terminals. Eq. (C.3a-C.3b) are same to Eq. (27-28) in [117], except that the lower flow bounds and content bounds are not imposed. Again, Eq. (C.3c-C.3e) are developed in place of the old topology constraints that are invalid for continuous design decisions (i.e., Eq. (23-26) in [117]).

$$\sum_{j\in\Pi_k^{\mathrm{PT}}}\sum_{w\in\{1,\dots,l\}} f^{\mathrm{PT}}_{j,k,w,h} + \sum_{i\in\Pi_k^{\mathrm{ST}}} f^{\mathrm{ST}}_{i,k,h} \le y_k^{\mathrm{T}} D^{\mathrm{UB}}_{k,h}, \tag{C.5a}$$

$$\sum_{j\in\Pi_k^{\mathrm{PT}}} f^{\mathrm{PT}}_{j,k,w,h} + \sum_{i\in\Pi_k^{\mathrm{ST}}} f^{\mathrm{ST}}_{i,k,h} U_{i,w,h} \le \left(\sum_{j\in\Pi_k^{\mathrm{PT}}}\sum_{w\in\{1,\dots,l\}} f^{\mathrm{PT}}_{j,k,w,h} + \sum_{i\in\Pi_k^{\mathrm{ST}}} f^{\mathrm{ST}}_{i,k,h}\right) V^{\mathrm{UB}}_{k,w} \tag{C.5b}$$

$$y_k^{\mathrm{T}} D_k^{\mathrm{UB}} \ge y^{\mathrm{ST}}_{i,k} F^{\mathrm{ST,UB}}_{i,k} \tag{C.5c}$$

$$y_k^{\mathrm{T}} D_k^{\mathrm{UB}} \ge y^{\mathrm{PT}}_{j,k} F^{\mathrm{PT,UB}}_{j,k}, \tag{C.5d}$$

$$y_k^{\mathrm{T}} D_k^{\mathrm{UB}} \le \sum_{i\in\Pi_k^{\mathrm{ST}}} y^{\mathrm{ST}}_{i,k} F^{\mathrm{ST,UB}}_{i,k} + \sum_{k\in\Pi_k^{\mathrm{PT}}} y^{\mathrm{PT}}_{j,k} F^{\mathrm{PT,UB}}_{j,k} \tag{C.5e}$$

$$\forall k \in \{1,\dots,m\},\ \forall w \in \{1,\dots,l\},\ \forall h \in \{1,\dots,b\}.$$

The modified stochastic pooling model can be stated as:

minimize objective

s.t. Eq. (C.1a-C.1f), Eq. (C.2a-C.2k), Eq. (C.3a-C.3e),

$y_i^{\mathrm{S}}, y_j^{\mathrm{P}}, y_k^{\mathrm{T}} \in \{0,1\}$ or $[0,1]$,

$y_{i,j}^{\mathrm{SP}}, y_{i,k}^{\mathrm{ST}}, y_{j,j^-}^{\mathrm{PP}}, y_{j,k}^{\mathrm{PT}} \in [0,1]$,

all flow rates are nonnegative,

redudant constraints for accelerating global optimizaiton (Eq. (38-39) in [117]).

The objective can be negative net present value, or negative annualized profit, as specified in [117].