THE PRODUCTION-ASSEMBLY-DISTRIBUTION SYSTEM DESIGN PROBLEM:

MODELING AND SOLUTION APPROACHES

A Dissertation

by

DONG LIANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2007

Major Subject: Industrial Engineering

THE PRODUCTION-ASSEMBLY-DISTRIBUTION SYSTEM DESIGN PROBLEM:

MODELING AND SOLUTION APPROACHES

A Dissertation

by

DONG LIANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Wilbert E. Wilhelm
Committee Members,	Sergiy Butenko
	Illya V. Hicks
	David R. Larson
Head of Department,	Brett A. Peters

December 2007

Major Subject: Industrial Engineering

ABSTRACT

The Production-Assembly-Distribution System Design Problem: Modeling and Solution

Approaches. (December 2007)

Dong Liang, B.S.; M.S. Shanghai Jiao Tong University

Chair of Advisory Committee: Dr. Wilbert E. Wilhelm

This dissertation, which consists of four parts, is to (i) present a mixed integer programming model for the strategic design of an assembly system in the international business environment established by the North American Free Trade Agreement (NAFTA) with the focus on modeling the material flow network with assembly operations, (ii) compare different decomposition schemes and acceleration techniques to devise an effective branch-and-price solution approach, (iii) introduce a generalization of Dantzig-Wolf Decomposition (DWD), and (iv) propose a combination of dual-ascent and primal drop heuristics.

The model deals with a broad set of design issues (bill-of-materials restrictions, international financial considerations, and material flows through the entire supply chain) using effective modeling devices. The first part especially focuses on modeling material flows in such an assembly system.

The second part is to study several schemes for applying DWD to the production-assembly- distribution system design problem (PADSDP). Each scheme exploits selected embedded structures. The research objective is to enhance the rate of DWD

convergence in application to PADSDP through formulating a rationale for decomposition by analyzing potential schemes, adopting acceleration techniques, and assessing the impacts of schemes and techniques computationally. Test results provide insights that may be relevant to other applications of DWD.

The third part proposes a generalization of column generation, reformulating the master problem with fewer variables at the expense of adding more constraints; the sub-problem structure does not change. It shows both analytically and computationally that the reformulation promotes faster convergence to an optimal solution in application to a linear program and to the relaxation of an integer program at each node in the branch-and-bound tree. Further, it shows that this reformulation subsumes and generalizes prior approaches that have been shown to improve the rate of convergence in special cases.

The last part proposes two dual-ascent algorithms and uses each in combination with a primal drop heuristic to solve the uncapacitated PADSDP, which is formulated as a mixed integer program. Computational results indicate that one combined heuristic finds solutions within 0.15% of optimality in most cases and within reasonable time, an efficacy suiting it well for actual large-scale applications.

DEDICATION

To my family

ACKNOWLEDGEMENTS

NOMENCLATURE

AVUB	Aggregated Variable Upper Bound
BOM	Bill Of Materials
B&B	Branch-and-Bound
B&P	Branch-and-Price
CG	Column Generation
DAA1	Dual-Ascent Algorithm one
DAA2	Dual-Ascent Algorithm two
DH	Drop Heuristic
DRMP	Linear programming Dual of RMP
DWD	Dantzig-Wolfe Decomposition
DWDG	Generalization of DWD
NAFTA	North American Free Trade Agreement
PADSDPN	Production-Assembly-Distribution System Design Problem under NAFTA
PADSDP	Production-Assembly-Distribution System Design Problem
RMP	Restricted Master Problem
UPADSDP	Uncapacitated Production-Assembly-Distribution System Design Problem

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Problem Statement

The trend of globalization has encouraged companies to locate operations in countries that offer comparative advantages. This trend affects the design of supply chain systems by requiring that domestic designs be extended to international ones. The North American Free Trade Agreement (NAFTA) forms the second largest free-trade zone in the world. Lower tariffs and shorter transportation distances make it possible for U.S.-based companies to locate assembly operations in Mexico.

The proposed research addresses the need of providing decision support aids for the strategic design of a production-assembly-distribution system in the international environment established by NAFTA. The problem we study is the production-assembly-distribution system design problem under NAFTA (PADSDPN). It is important to study PADSDPN because optimizing a production-assembly-distribution system and its supply chain enhances the competitiveness of a company in the global economy and because the PADSDPN structure and those of a number of related problems have proven especially challenging to branch-and-price (B&P). To facilitate description of PADSDPN, we use *facility* to indicate a supplier, a production plant, an assembly plant, or a distribution center; and *component* to indicate a part, sub-assembly or end product.

The domestic version of PADSDPN is the production-assembly-distribution system design problem (PADSDP), which extends the supply chain design problem (also

_____

This dissertation follows the style of *IIE Transactions*.

called the production-distribution network design problem) with the consideration of bill-of-materials (BOM) restrictions on material flows. A BOM identifies the components of an end product and indicates how raw materials and subassemblies form end products. Because of BOM restrictions, components are not independent and material flows through assembly operations are not conserved in the traditional way. The supply chain design problem involves prescribing decisions that (i) select a set of facilities based on their locations, capacities, and technologies; (ii) allocate components to facilities for supply, production, assembly, or distribution; and (iii) plan the flow of materials (e.g. production, assembly, transportation, inventories, and backorders) through the supply chain. The objective of the supply chain design problem is to maximize after-tax profits (or minimize total costs).

The international supply chain design problem considers international business factors such as border-crossing costs, transfer prices, income taxes, local-content rules, safe-harbor income tax rates, and exchange rates. The international PADSDP extends the international supply chain design problem by including BOM restrictions. PADSDPN is the international supply chain design problem with the special considerations associated with BOM and NAFTA.

1.2. Organization of the Dissertation

This dissertation is organized as follows. In section 2, we present a brief literature review on formulations of the international supply chain design problem, solution approaches to it and its related problems, and B&P methods. In section 3, we present the mixed integer programming formulation of PADSDPN. In section 4, we compare a set

of decomposition schemes of DWD for PADSDP and adopt acceleration techniques in an attempt to develop insights into methods for improving the convergence of DWD in application to PADSDP. In section 5, we propose a generalization of column generation, reformulating the master problem with fewer variables at the expense of adding more constraints; the sub-problem structure does not change. This generalization can be used to design an acceleration technique and improve the convergence of DWD. In section 6, we design a combination of dual-ascent and primal heuristic for uncapacitated PADSDP. Finally, in section 7, we summarize conclusions and future research of this dissertation.

2. LITERATURE REVIEW

This section presents brief reviews of the bodies of literature that deal with the international supply chain design problem and the branch-and-price (B&P) approach to justify the proposed research. This section is organized as follows: subsection 2.1 reviews the literature on the international supply chain design problem; subsection 2.2 reviews the literature on classic problems related to our application, like facility location, network design, and multi-commodity network flow problems, and describes solution approaches to these classic problems; subsection 2.3 focuses on the literature related to B&P approach; subsection 2.4 reviews the literature on the dual-ascent methods; and subsection 2.5 presents a brief summary of this section.

2.1. International Supply Chain Design Problem

The international supply chain design problem is a production-distribution network design problem that considers international business issues. Recent reviews (Geunes and Pardalos, 2003; Sarmiento and Nagi, 1999; Vidal and Goetschalckx, 1997, 2002) and books (Tayur et al., 1999) describe the state-of-the-art associated with the international supply chain design problem.

Most production-distribution models in the literature have been mixed integer programs. Although some studies used stochastic programming models to address uncertainty (Hodder and Dincer, 1986; Alonso-Ayuso et al., 2003; Huchzermeier, 1991; Santoso et al., 2004), the state of the art does not allow instances of practical size and scope to be solved.

The production/distribution network design problem can be formulated as a mixed integer program (Schmidt and Wilhelm, 2000). Vidal and Goetschalckx (forthcoming) discussed the limitation of modeling techniques applied to the network design problem and classified factors according to whether they can be modeled accurately or not. Most research on the network design problem considers these factors and assumes that they can be modeled using linear functions.

International business factors, including tariffs, exchange rates, transfer prices, income taxes and regional trade rules (non-tariff trade barriers) are relatively difficult to formulate using linear functions. Cohen et al. (1989) formulated a non-linear model that considered transfer prices, exchange rates, and overhead allocation. Vidal and Goetschalckx (2001) maximized after-tax profits using a non-convex model that considered transfer prices and transportation charges. In spite of that, it is possible to linearize these non-linearities (Wilhelm et al., 2005).

Although many papers on the international supply chain design problem have used mixed integer programming models; each addressed only a subset of relevant factors. Bartmess and Cerny (1993) presented a model with exchange rates, political impacts, taxes, transfer prices, and costs, but without the design of the logistics network. Kouvelis and Rosenblatt (1997) proposed a model for global logistic network design with income taxes and regional trade rules, but without transfer prices. However, in real-life applications, it is important to integrate all decisions (Verter and Dincer, 1992). So, the challenge of modeling the international supply chain design problem is to integrate all relevant factors in one linear model.

In this proposed research, we present a comprehensive mixed integer program focusing on assembly systems with international issues that represent the Texas–Mexico environment under NAFTA. In particular, we emphasize modeling material flows in the assembly system.

2.2. Related Problems

PADSDPN is related to several classical problems: facility location, supply chain design, production-distribution network design, fixed-charge network flow, network design and multi-commodity network flow problems. Figure 1 depicts a hierarchy among these problems, each of which has been studied extensively. Because the facility location problem is embedded in each of these related problems except the multi-commodity network flow problem and is known to be NP-hard (Nemhauser and Wolsey, 1999), each of them except the multi-commodity network flow problem is NP-hard. We present a hierarchical relationship of all related problems in Figure 1.

The kernel of these related problems is the facility location problem, which is to locate a set of facilities in order to minimize the cost of satisfying demands (Hale, 2003). The facility location problem has been well studied and widely used. The simplest version of the facility location problem is the uncapacitated facility location problem, which is NP-hard (Guignard and Jonhnson, 1979; Krarup and Pruzan, 1983). However, some special cases of the uncapacitated facility location problem can be solved in polynomial time, for example, the one on trees (Shaw, 1993, 1999). Krarup and Pruzan (1983) and Cournuejols et al. (1990) presented surveys of the uncapacitated facility location problem. Another, more difficult facility location problem is the capacitated one, which takes

the capacities of alternative facilities into consideration. Sridharan (1995) provided an overview of the capacitated facility location problem. The facility location problem can be formulated as integer programming models (Aikens, 1985).
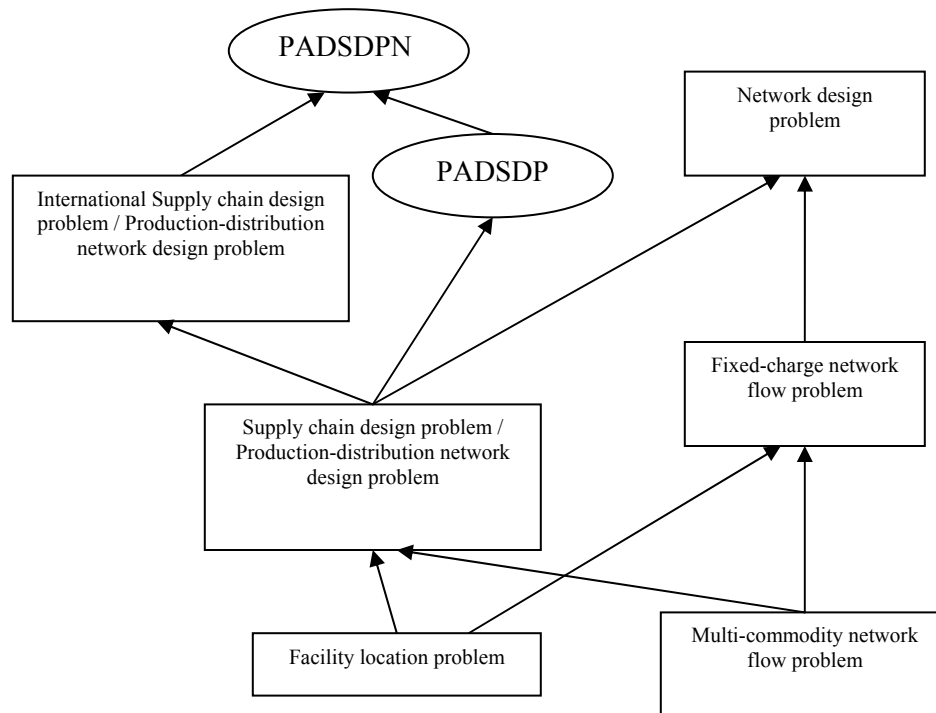
Figure 1. Hierarchical relationship of related problems

Various extensions have been proposed for the facility location problem, adding factors like bill-of-materials (BOM) relationships among components, multi-commodities, multi-echelons, inventories, and choices of facility types (Daskin, 1995; Drezner, 1995). For example, Barahona and Jensen (1998) studied the uncapacitated facility location problem with inventory. Hinojosa (2000) presented a the capacitated facil-

ity location problem with two echelons and multiple time periods. Mazzol and Neebe (1999), Dasci and Verter (2001) and Lee (1991) studied the multi-commodity capacitated facility location problem with a choice of facility types. Melkote and Daskin (2001) proposed a model for the capacitated facility location problem with network designs.

Many studies have developed solution approaches for the facility location problem, including B&B (Akinc and Khumawala, 1977); polyhedral studies (Cho et al., 1983; Aardal, 1998; Aardal et al., 1995); dual decomposition approaches, including dual-ascent methods (Guignard and Spielberg, 1979; Balakrishnan et al., 1989), Lagrangian Relaxation (Christofides and Beasley, 1983; Mazzol and Neebe, 1999; Hinojosa, 2000) and Danzig-Wolfe Decomposition (DWD) (Barahona and Jensen, 1998); primal decomposition approaches, including Benders' decomposition (Geoffrion and Graves, 1974; Cohen and Lee, 1988); the cross decomposition approach (Van Roy, 1983; Lee, 1993); and heuristics (Shi et al., 2004).

One important generalization of the facility location problem is the fixed-charge network flow problem. Nemhauser and Wolsey (1999) provided examples on how to model the facility location problem as the fixed-charge network flow problem, which is to establish arcs to form a network that supports commodity flows to satisfy demands. The fixed-charge network flow problem is widely applied to transportation planning and telecommunication problems (Magnanti and Wong, 1984; Ambrosino and Scutella, 2005). The fixed-charge network flow problem is a special case of the network design problem. It assumes that only one possible capacity is installed on each arc while the network design problem allows the selection from alternative capacities for each arc.

Both the fixed-charge network flow and network design problem can be formulated as mixed integer programs. The supply chain design problem is also called the production-distribution network design problem (Hale, 2003; Klose and Drexl, 2005). It is a special case of the network design problem in which the network is acyclic.

PADSDP extends the supply chain design problem by including BOM relationships in a multi-echelon assembly system. Material flows through nodes representing assembly operations in PADSDP are not conserved as they are in the minimum-cost network flow problem because the flow of components into such a node is not equal to the flow of assemblies that departs the node. A hyper-graph in which each arc can have more than two adjacent nodes can be used to represent the assembly flow network (Gallo and Pallottino, 1992). The international supply chain design problem extends the supply chain design problem considering of international business factors. PADSDPN can be viewed as either the extension of PADSDP or the international supply chain design problem with the special considerations associated with BOM and NAFTA.

Solution approaches for the network design problem comprise three categories: polyhedral studies, such as branch & cut (Magnanti and Raghavan, 2005); decomposition approaches, such as Benders' decomposition (Costa, 2005), B&P (Shaw, 1993; Henningsson et al., 2002; Teo and Shu, 2004) and Lagrangian relaxation (Gendron and Crainic, 1994, 1996; Crainic et al., 2001; Elhedhli and Goffin, 2005); and heuristics, such as tabu search (Craninic et al., 2000), genetic algorithms (Turkay and Artac, 2005), simulated annealing (Jayaraman and Rose, 2003), Greedy Randomized Adaptive Search

Procedures (GRASP) (Alvarez et al., 2005) and primal-dual heuristics (Melkonian and Tardos, 2005).

Costa (2005) presented a survey of research that applied Benders' decomposition to the fixed-charge network flow problem. One milestone in the application of Benders' decomposition is that Geoffrion and Graves (1974) used Benders' decomposition to solve the multi-commodity capacitated facility location problem. Gendron et al. (1999) presented a comprehensive survey of models and dual decompositions for the network design problem and listed several Lagrangian relaxation decomposition schemes for the network design problem. Gendron and Crainic (1994, 1996) and Crainic et al. (2001) discussed the performance of sub-gradient and bundle methods in solving the fixed-charge network flow problem. Clarke and Gong (1995) presented a path-based model for the general capacitated network design problem and used B&P to optimize it. Chabrier et al. (2004) presented a case study of the network design problem comparing different optimization techniques. They emphasized the need for a robust solution approach and provided some benchmarks.

Benders' decomposition can be regarded as the dual version of DWD; i.e., the application of DWD to the dual of the original problem (Nemhauser and Wolsey, 1999). The difficulty of applying Benders' decomposition is that the master problem includes integer constraints and we must solve the master problem iteratively. The performance of Benders' decomposition typically degrades as the size of an instance increases because the master problem becomes more difficult to solve. In comparison with Benders' decomposition, B&P uses the B&B framework to deal with integer constraints. The mas-

ter problem of B&P is relatively easy to solve so that B&P is able to solve large-scale instances.

One important structure embedded in the network design problem is the multi-commodity network flow problem, which is a generalization of the network flow problem. In the multi-commodity network flow problem, several commodities share the same underlying network and compete for arc capacities as the total cost for the flows of all commodities is minimized.

The multi-commodity network flow problem can be formulated as a linear program and is a so-called easy problem. However, many studies have investigated methods to design a more efficient way to solve the multi-commodity network flow problem instead of using the linear programming simplex method. Kennington (1978), Assad (1978), Ahuja et al. (1993) and Ahuja (1997) provided surveys on its formulations and solution approaches.

One important issue related to the multi-commodity network flow problem is the formulation; two have been studied extensively: the arc-based model, whose variables represent flows on arcs, and the path-based model, whose variables represent flows on paths. Actually, the formulation of the multi-commodity network flow problem is closely related to the solution approach. The DWD of the arc-based model is the corresponding path-based model. Other formulations are extensions of the basic two. For example, cycle-based models whose variables represent flows on cycles can be reformulated as path-based models (Barnhart et al., 1995) because one cycle can be represented as a symmetric difference of a set of paths. Jones et al. (1993) presented a very good

analysis of the impact of formulation on the success of DWD in solving the multi-commodity network flow problem. They compared the origin-destination-specific formulations and the destination- specific formulations. A column in the former represents a path-following flow from one single origin to one single destination. A column in the latter represents a tree-following flow from all origins to one single designation. Because one tree-following flow can be decomposed into a set of path-following flows, the authors represented the tree-following column as the aggregation of a set of path-following columns. They observed that the origin-destination-specific formulation typically offers a faster rate of convergence than the destination-specific one and provided computational evaluation to complement their analysis.

Exact solution approaches for the multi-commodity network flow problem comprise two main categories: partitioning and decomposition approaches. Partitioning approaches avoid directly working with a linear programming basis for the entire problem by partitioning the basis into several smaller ones, which correspond to special constraint structures. In the multi-commodity network flow problem these smaller bases correspond to network balance constraints and can be optimized by the more efficient network algorithms, then used to construct a complete basis for the original problem (Rosen 1964; Barnhart et al., 1995; McBride and Mamer, 2004). Decomposition approaches for the multi-commodity network flow problem are traditionally classified as either cost-decomposition (also called price-directive decomposition or dual decomposition) or resource-decomposition (also called resource-directive decomposition or primal decomposition) (Nazareth, 1987; Cappanera and Franaioni, 2003).

The idea underlying resource-decomposition methods is to allocate a portion of the capacity of each arc to each commodity. Given an allocation of arc capacities, the minimum flow cost can be obtained efficiently by solving a set of minimum-cost network flow problems, one for each commodity. So, the difficulty of solving the multi-commodity network flow problem is in finding an optimal allocation of capacities. The sub-gradient method is the most common method for finding an optimal allocation (Shetty and Muthukrishnan, 1990; De Leone et al., 1993).

Compared with direct allocation in resource decomposition, cost decomposition uses prices to indirectly allocate arc capacities, pricing arc capacities and letting commodities find their own minimum cost flow based on these prices. One method for cost decomposition, Lagrangian relaxation, relaxes the capacity constraints into the objective function. Solving the multi-commodity network flow problem is equivalent to finding optimal Lagrangian multipliers. DWD is another cost decomposition method; the equivalence between Lagrangian relaxation and DWD is well known (Lemarechal, 2001). For problems that can be decomposed by DWD, we can define an equivalent Lagrangian relaxation by dualizing constraints corresponding to those in the master problem of DWD (vice versa). In this way, we define corresponding Lagrangian relaxation and DWD formulations, whose sub-problem structures are exactly the same and which give exactly the same bounds. The primary difference between Lagrangian relaxation and DWD is in how prices (i.e., Lagrangian multipliers or dual values of the restricted master problem (RMP)) are updated. Lagrangian relaxation simply moves Lagrangian

multipliers along sub-gradients with pre-defined step sizes, while DWD must find optimal dual values for RMP.

Several solution methods apply to a cost decomposition. Dual-ascent, sub-gradient, and bundle methods are often used in conjunction with Lagrangian relaxation (Ferris and Horn, 1998; Frangioni and Gallo 1999; Villavicencio, 2005). DWD is a form of column generation (CG) (Wilhelm, 2001; Jones et al., 1993; Barnhart et al., 1995; Holmberg and Yuan, 2003). Frangioni (2002) proposed a unified framework for cost decomposition approaches; dual-ascent, sub-gradient, bundle, and CG methods can all be cast into this framework. Another successful method for cost decomposition is the analytic center cutting plane method (ACCPM), which uses the interior point method to update dual values for RMP (Goffin et al., 1996).

2.3. B&P Solution Approaches

For the proposed model, real-world instances will be very large, so that even the linear programming relaxation will be difficult to solve. Decomposition approaches are a good choice for solving such large-scale instances. We propose a B&P approach to optimize the proposed model. Now, we briefly review the B&P method.

Dantzig and Wolfe (1960) and Gilmore and Gomory (1961) initiated two types of CG independently. The former introduced DWD for linear programs and the latter devised CG to solve the cutting stock problem. CG has become one of the techniques that are most successful in dealing with large-scale linear (e.g., Ford and Fulkerson, 1958; Gilmore and Gomory, 1961) and integer programs (e.g., Desaulniers et al., 2001). Embedding CG within a B&B context, B&P is an important development of CG for

solving integer programs. Appelgren (1969) presented the earliest use of B&P, solving a ship-scheduling problem. Desrosiers et al. (1984) applied B&P successfully to the vehicle routing problem; subsequently, it has been used to advantage in many applications, including integer multi-commodity flow (Alvelos and Valerio de Carvalho, 2007; Barnhart et al., 1997), cutting stock (Ben Amor et al., 2005; Valerio de Carvalho, 2005), and crew scheduling (Desaulniers et al., 2001) problems. Recent surveys on CG include those of Wilhelm (2001), Desrosiers and Lubbecke (2002) and Desaulniers et al. (2005).

Implementing CG poses a number of challenges (Vanderbeck, 2002, 2005; Lubbecke et al., 2002). One challenge is to accelerate the convergence performance of CG. Many studies have enhanced the performance of CG. Relative to the challenge of accelerating convergence, stabilization is one of the most important issues. Considering the linear programming dual of the master problem, DWD may converge slowly because dual variable values oscillate, meaning that dual variables may jump from values that generate a *good* column for the primal RMP at one iteration to values that generate a *bad* column at the next iteration. The basic idea is that stabilization dampens oscillation by penalizing it (e.g. Ben Amor et al., 2004). Different types of penalty functions have been used to stabilize dual variable values. Ben Amor et al. (2004) proposed a piecewise linear penalty function. Quadratic functions have also been employed, rendering the master problem non-linear in stabilization methods that are equivalent to bundle methods for non-smooth optimization (Frangioni, 2002).

Another means of accelerating convergence is to add cuts in the dual space to restrict the feasible range of values for each dual variable (Ben Amor et al., 2006). Valerio

de Carvalho (2005) proposed this idea and applied it to accelerate solution of cutting stocking problems. Vanderbeck and Savelsbergh (2006) discussed the use of exchange vectors, each of which is the difference between two feasible sub-problem solutions. Exchange vectors can be used to obtain valid dual cuts. The dual cuts proposed by Valerio de Carvalho (2005) can be also viewed as exchange vectors.

Researchers have also studied the convergence performance of DWD, which is a form of CG, as a function of problem formulation. Jones et al. (1993) studied the effects of column aggregation/disaggregation for the multi-commodity network flow problem by representing a tree-following column as the aggregation of a set of path-following columns, providing a trade-off between the arc-based and the path-based models. More generally, the arc-based model represents the original, compact formulation, which has a few columns and a compact coefficient matrix, and the path-based model represents the extensive formulation of DWD, which has a large number of columns and an extensive coefficient matrix. DWD provides one way to change a compact form to an extensive one and some other transformations do the reverse (Villeneuve et al., 2003).

Barnhart et al. (1995) proposed a cycle-based model with variables representing flows on cycles, each of which can actually be formulated as an exchange vector because one cycle can be viewed as the symmetric difference of two paths (Cornuejols, 2001). The authors implemented DWD on the cycle-based model and obtained significant runtime improvement. Alvelos and Valerio de Carvalho (2007) proposed an extended model on the planar multicommodity flow problem by adding a set of variables and inequalities and used CG to solve the extended model. All these reformulations can be considered to

be variable-redefinition approaches (Martin, 1987), provides one way to investigate compact and extensive formulations.

Another challenge is to adapt CG in the context of B&B. To implement such a B&B framework, which has been used successfully to solve mixed integer programs, the analyst must specify i) branching rules; ii) methods to fix variables; iii) techniques to re-optimize nodes in the B&B tree; and iv) steps to initialize. These issues are closely re-lated to the solution approach used at B&B nodes. For example, one re-optimization technique in a simplex-based B&B algorithm is to simply shares the parent node's basis information with its children. By comparison, an equivalent re-optimization technique for B&P is more complicated because both the basis information and generated columns must be shared between the parent node and its children. Some research has addressed these implementation issues (Desaulniers et al., 2001; Lubbecke et al., 2002; Vander-beck, 2004).

2.4. Dual-ascent Methods

Dual-ascent algorithms work with a dual problem (e.g., linear program dual or Lagran-gian dual), solving it by iteratively updating dual variable values so that they are always feasible and produce successive dual solution values (i.e., bounds) that improve mono-tonically. After finding a good (not necessarily an optimal) dual solution, the algorithm constructs a primal feasible solution by employing complementary slackness (Guignard and Rosenwein, 1989). The pair of primal and dual solutions provides both upper and lower bounds, respectively, on the optimal solution value. Dual-ascent algorithms typi-cally use problem-specific techniques to update dual variable values and can be viewed

as a special implementation of Lagragian relaxation, which is typically solved by the sub-gradient method (Guignard and Rosenwein, 1989; Holmberg, 2001). They have been applied successfully to a variety of problems, including facility location (Erlenkotter, 1978), network design (Balakrishnan et al., 1989), and multi-commodity network flow (Barnhart, 1993; Barnhart and Sheffi, 1993), that are related to PADSDP.

The kernel of PADSDP is the uncapacitated facility location problem, which is to locate a set of facilities in order to minimize the cost of satisfying demands (Nemhauser and Wolsey, 1999). Erlenkotter (1978) and Guignard (1988) proposed dual-ascent algorithms for the uncapacitated facility location problem. A number of studies have applied dual-ascent approaches to adaptations of the uncapacitated facility location problem, including the multi-echelon variation (Van Roy and Erlenkotter, 1982; Gao and Robinson, 1992), the location problem with a balancing requirement (Crainic and Delorme, 1993; Robinson and Gao, 1996), and a multilevel version (Bumb and Kern, 2001). Furthermore, Guignard and Opaswongkarn (1990) proposed a dual-ascent approach to the capacitated facility location problem.

The fixed-charge network flow problem is a generalization of the facility location problem (Nemhauser and Wolsey, 1999). Researchers have designed dual-ascent approaches to both uncapacitated fixed-charge network flow problems (Balakrishnan et al., 1989) as well as the capacitated version (Herrmann et al., 1996; Gendron, 2002). Balakrishnan et al. (1994) devised dual-ascent algorithms for networks with special topological structures (e.g., multi-level networks).

Motivated by these successful results in application to related problems, we develop a dual-ascent approach for UPADSDP. Since material flows through assembly operations in UPADSDP are not conserved in the traditional way, we use a hypergraph in which arcs can have more than two adjacent nodes to represent the assembly flow network (Gallo and Pallottino, 1992).

2.5. Summary

The international supply chain design problem can be viewed as the network design problem with a set of side constraints. This review reveals that dual decomposition approaches work well in application to the network design problem, encouraging us to use this approach in the proposed research. In addition, instances of our proposed model are fairly large in real-life applications. A traditional simplex-based B&B algorithm does not have the capability to solve such large instances, so it is important to devise new solution approaches that can deal with such large-scale instances.

3. MIXED INTEGER PROGRAMMING MODEL*

In this section, we model material flows in the supply chain associated with the assembly operations in a comprehensive mixed integer programming model for the production-assembly-distribution system design problem under NAFTA (PADSDPN) proposed by Wilhelm et al. (2005).

The comprehensive model for PADSDPN is proposed by Wilhelm et al. (2005). It deals with a complex set of issues to integrate relevant decisions and lead to useful results. In this dissertation we discuss on modeling the material network with assembly operations.

3.1. Material Flow Network with Assembly Operations

3.1.1. Traditional networks with balance constraints

We define notation used to formulate the material flow network in Table 1, including indices, sets, parameters, and decision variables. To facilitate presentation, we design logic in the notation. For example, we always use p_1 to denote a component that is used to produce component p, which, in turn, is used to produce component p_2. Thus, these indices can imply the bill-of-materials (BOM) relationship $p_1 \rightarrow p \rightarrow p_2$ in which p_1 (p) is a component in the set of components used to make p (p_2). Similarly, material flows are always transferred from facility f_1 to facility f and then to facility f_2; that is,

the corresponding facility relationship $f_1 \rightarrow f \rightarrow f_2$ in which p_1, p, and p_2 are handled by facilities f_1, f, and f_2, respectively (i.e., p_1 (p) is shipped from f_1 to f (from f to f_2). A special class of components p is end products, which we also denote by the index e (i.e., $p \equiv e$) when it is important to distinguish end products. We define cost parameters G. Relative to the operation that assembles component p, we define $p*$ as the first type of component (i.e. with the smallest index) that comprises component p. p_e^* has the same meaning of $p*$ and emphases that component p is used to assemble end product e. We indicate indices of summation and of constraint enumeration over sets over (e.g., $p \in P$ or $t \in T$) such that this model can be implemented by modeling tools (e.g., AMPL). We present the notation as follows:

Table 1. Notation

Indices

$e \in P^E$	End products
$f_1, f, f_2 \in F$	Facilities: f_1 (f) is immediately upstream of facility f (f_2)
$k \in K$	Transportation modes (here, we consider only one mode)
$l \in L$	Locations, which are specific areas in one country
$p_1, p, p_2 \in P$	Components: p_1 and p_3 (p) are used to produce component p (p_2)
p^*	Predecessor of component p with minimal index, $p^* = \min\{P_p^\uparrow\}$
p_e^*	Predecessor of component p with minimal index in e, $p_e^* = \min\{P_{e,p}^\uparrow\}$
$t \in T$	Time periods in the planning horizon

Table 1. Continued

Parameters: G **parameters represent discounted costs in U.S. dollars**

D_{eft} Demand for end product e at facility f during period t

G_{eft}^{B} Backorder cost for end product e at facility f during period t

G_{f}^{O} Fixed cost of operating facility f

G_{pft}^{I} Inventory holding cost for component p at facility f during period t

G_{pf}^{A} Fixed cost of opening facility f to handle component p

$G_{pff_{2}t}^{S}$ Cost to facility f_{2} of purchasing raw material p from supplier f during period t

$G_{pff_{2}kt}^{T}$ Variable cost of transporting p from f to f_{2} via transportation mode k during period t

G_{pft}^{V} Variable cost of operating facility f to handle component p during period t

M_{p}^{e} Number of each scaled component p in the BOM for end product e

Sets

F Facilities

F_{l} Facility alternative at l

F_{p} Facility alternatives for p

L Locations

P Components

P_{e} Components in the BOM for e

P_{f} Components processed at facility alternative f

T Time periods

Decision variables

b_{ft}^{e} Number of backorders for end product e at facility f during period t

$h_{pp_{2}ft}^{e}$ Sub-amount of h_{pft} for assembly of component p_{2} in end product e

$u_{pp_{2}ff_{2}kt}^{1,e}$ Sub-amount of $u_{pff_{2}kt}^{1}$ for assembly of component p_{2} in end product e

$u_{pp_{2}ft}^{2,e}$ Sub-amount of $u_{pp_{2}ft}^{2}$ used to produce end product e

Table 1. Continued

x_f 1 if facility f is activated, 0 else

y_{pf} 1 if component p is handled by facility f, 0 else

We now present constraints on modeling the material network with assembly operations in the comprehensive PADSDPN model (Wilhelm et al., 2005).

$$u^{2,e}_{p_e^* pft} - u^{2,e}_{p_1 pft} \qquad\qquad\qquad\qquad = 0; \quad \forall\, e\,, p\,, p_1\,, f\,, t \qquad (3\text{-}1)$$

$$\sum_{p_2,f,f_2,k,t} u^{1,e}_{pp_2 ff_2 kt} \qquad\qquad\qquad\qquad = M^e_p \sum_{f,t} D_{eft}\,; \forall\, e\,, p \qquad (3\text{-}2)$$

$$\sum_{f_1,k} u^{1,e}_{pp_2 f_1 fkt} + h^e_{pp_2 f(t-1)} - h^e_{pp_2 ft} - u^{2,e}_{pp_2 ft} \qquad = 0; \quad \forall\, e\,, p\,, p_2\,, f\,, t \qquad (3\text{-}3)$$

$$u^{2,e}_{p_e^* pft} + \sum_{p_2,f_1,k} u^{1,e}_{pp_2 f_1 fkt} + \sum_{p_2} \left(h^e_{pp_2 f(t-1)} - h^e_{pp_2 ft} \right)$$

$$-\sum_{p_2} u^{2,e}_{pp_2 ft} - \sum_{p_2,f_2,k} u^{1,e}_{pp_2 ff_2 kt} \qquad = 0; \quad \forall\, e\,, p\,, f\,, t \qquad (3\text{-}4)$$

$$\sum_{f_1,k} u^{1,e}_{pp_2 f_1 fkt} + h^e_{pp_2 f(t-1)} - h^e_{pp_2 ft} - b^e_{ft} + b^e_{f(t+1)} - \sum_{f_2,k} u^{1,e}_{pp_2 ff_2 kt} = 0; \forall\, e\,, p\,, p_2\,, f\,, t \quad (3\text{-}5)$$

$$\sum_{f_1,k} u^{1,e}_{pp_2 f_1 fkt} \qquad\qquad\qquad\qquad = D_{eft}\,; \; \forall\, e\,, p\,, p_2\,, f\,, t \qquad (3\text{-}6)$$

$$\sum_{f,t} u^{2,e}_{p_1 pft} \qquad\qquad\qquad\qquad = \sum_{f,t} D_{eft}\,; \; \forall\, e\,, p\,, p_1 \qquad (3\text{-}7)$$

$$b^e_{ft}\,, h^e_{pp_2 ft}\,, u^{1,e}_{pp_2 ff_2 kt}\,, u^{2,e}_{pp_2 ft} \qquad\qquad \geq 0 \qquad\qquad\qquad\qquad (3\text{-}8)$$

The above model presents one way to formulate the material flow network with assembly operations. At a node representing an assembly operation, the scaled flow on each incoming arc is the same as the flow on the outgoing arc. For each assembly operation, we redirect each input arc - except the one of lowest index number - to a sink node,

allowing flow to conform to classical network-flow balance at all nodes. We set the demand at each sink node to assure flow of the number of components needed in all end products demanded.

Equalities (3-2) enforce flow balance at source nodes (i.e., suppliers of raw materials), assuring that the sum of all flows out, $u^{1,e}_{pp_2ff_2kt}$, equals the total demand plus the requirement of the sink node, $M_e \sum_{f,t} \overline{D}_{eft}$. The BOM need not be a tree; for example, p could be incorporated in several sub-assemblies p_2 to make one end product e. $A^e_{pp_2}$ defines the number of p used to manufacture p_2 for e and M^e_p indicates the number of different subassemblies, denoted individually using p_2, that incorporate p.

Equalities (3-3) enforce flow balance for component p, which is transported from other facilities f_1 to facility f, considering the sum of shipments $u^{1,e}_{pp_2f_1fkt}$; inventory changes at f, $h^e_{pp_2f,(t-1)} - h^e_{pp_2ft}$; and flow out to manufacture $u^{2,e}_{pp_2ft}$. We invoke equalities (3-3) only relative to the arc of lowest index number, p^*_e. Flow balance equalities (3-1) assure that all flows incoming to the assembly operation, $u^{2,e}_{p^*_epft}$ and $u^{2,e}_{p_1pft}$, are equal with each other (i.e., the flow of components on *each* arc incoming to an assembly operation must equal the flow on the departing arc).

Equalities (3-4) enforce flow balance of component p at facility f considering manufacturing inputs $u^{2,e}_{p_epft}$, the sum of shipment $u^{1,e}_{pp_2f_1fkt}$ from each other f_1 to f, the sum of inventory changes $(h^e_{pp_2f,(t-1)} - h^e_{pp_2ft})$ at f, the sum of manufacturing outputs

$u^{1,e}_{pp_2 f_1 fkt}$ of p for downstream assembly p_2 at f , and the sum of shipments $u^{1,e}_{pp_2 f_1 fkt}$ of

p to each p_2 from f to other f_2 by k . A special case arises if p and p_2 are assigned

to the same f . If these components were produced in f and f_2 , respectively, (3-3) and

(3-4) would allow inventory of p to be held at both f and f_2 (i.e., both before and af-

ter transport between the facilities) as depicted in the material flow network of Figure

2(a). However, if facility f produces both, our model allows only one inventory of f

and f_2 to be held at f so that the material flow network uses only one node to model

this inventory as depicted by Figure 2(b). Furthermore, capacity constraints on quantities

transported within f are not needed, eliminating decision variables $u^{1,e}_{pp_2 f_1 fkt}$ and $u^{1}_{pf_1 fkt}$

for $f = f_2$.

(a): Inventories of component p before
and after transportation in a facility f

(b): Single inventory of component
p held in a facility f

Transporataion   Manufacturing   Inventory
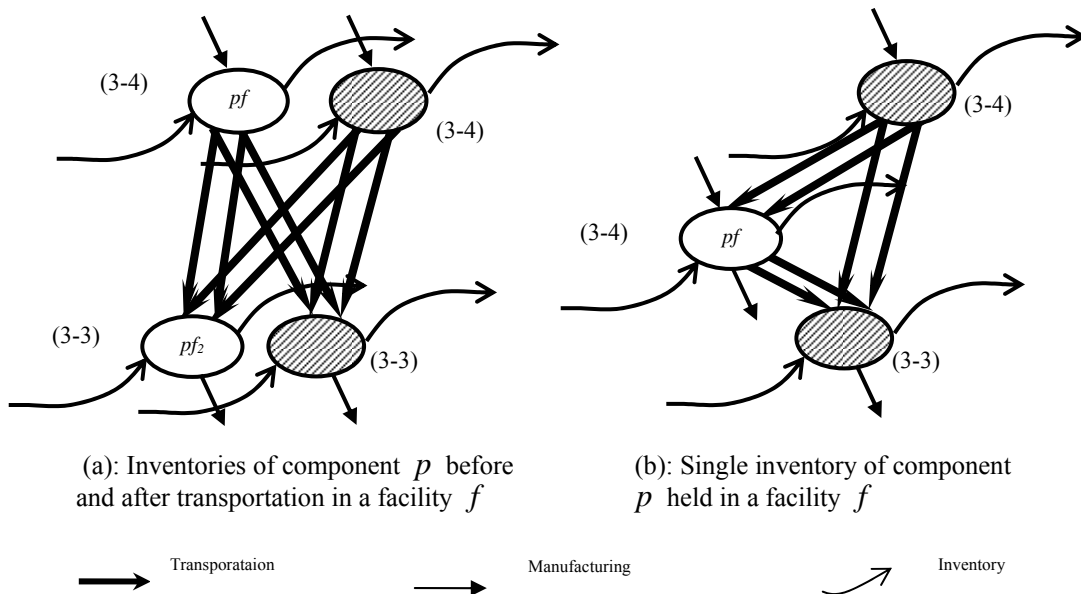
Figure 2. Special-case alternatives for modeling inventory of component p at f

Equalities (3-5) enforce flow balance of end product e at distribution center f in period t, considering the sum of units transported from facility f_1, $u^{1,e}_{pp_2 f_1 fkt}$; change in inventory at facility f, $(h^e_{pp_2 f,(t-1)} - h^e_{pp_2 ft})$; change in the number of outstanding backorders at facility f, $(b^e_{f,(t+1)} - b^e_{ft})$; and the sum of flow out facility f_2, $u^{1,e}_{pp_2 ff_2 kt}$, required to
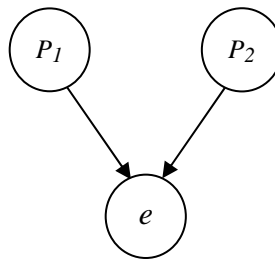
Figure 3. BOM
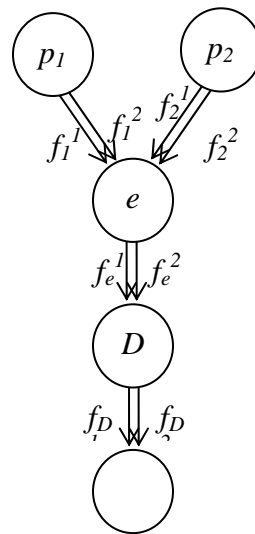
Figure 4. Facility alternatives

satisfy demand. Our model holds end-product inventory at distribution centers rather than at the facility that completes final assembly. Equalities (3-6) ensure the flow balance of end product e handled in distribution center f_1 and transported into at customer zone f by mode k, $u^{1,e}_{pp_2f_1fkt}$, satisfy the demand for e at customer zon f in period t, $\bar{D}_{eft}$. Note that equalities (3-5) and (3-6) hold for end product e. In order to use an index consistently in variable $u^{1,e}_{pp_2f_1fkt}$ to represent the component transported, if $p = e$, we define $p_2 = p = e$. Equalities (3-7) ensure flow balance at the sink node, requiring inflows, $u^{2,e}_{p_1pft}$, to equal the required total, $(M_e - 1)\sum_{f,t}\bar{D}_{eft}$. Equalities (3-2) and (3-6) imply equalities (3-7) because of the properties of the network flow problems. Here we give the balance condition explicitly. Figure 3 depicts a simple BOM, in which components p_1 and p_2 are assembled to form end product e. Figure 4 (Zhu, 2005) describes facility alternatives for producing components p_1 (f_1^1, f_1^2) and p_2 (f_2^1, f_2^2), assembling them to comprise e (f_e^1, f_e^2) and distributing e (f_D^1, f_D^2).

Figure 5 depicts the portion of the material flow network related to time period t; curved arcs represent flows (i.e., inventories or backorders) between consecutive time periods.

3.1.2. Hyper-graph model

Dealing with BOM flows poses special problems in assembly systems. At a node representing an assembly operation, incoming arcs carry flows of different components; and the outgoing arc, flow of the assembly. Such multi-commodity flows do not conform to the classical flow balance associated with minimum-cost network-flow problems. That is,

flow balance equalities (3-1) assure that all flows incoming to the assembly operation are equal with each other, but they destroy the structure of the classic network flow problem. We make the changes to the assembly node such that a hyper-edge includes all involved
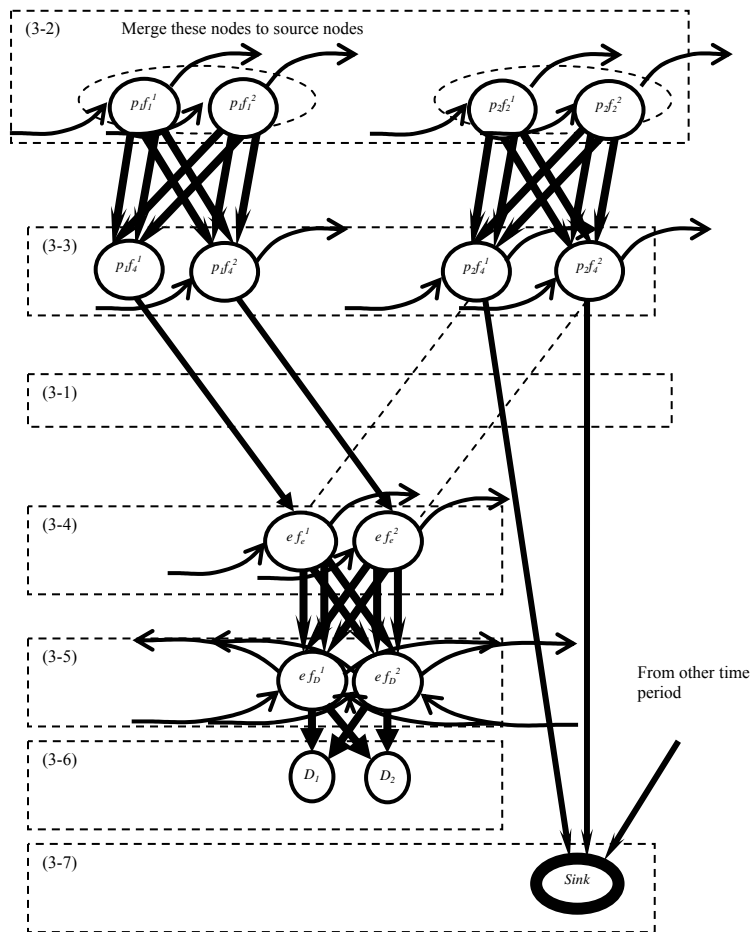
i) Dotted line ellipse represents the source node associated with constraints (9a)
ii) Dotted line rectangles include nodes associated with constraints (9b)-(9f);
iii) Dotted line arcs redirect flow to the sink node, which is constrained by (7d), that is, the flow is equal to the corresponding flow into the assembly node associated with constraints (9b);
iv) Curved arcs within dotted line rectangles, which are associated with constraints (9b) & (9c), represent inventory, within dotted line rectangle to constraints (9d) represent backorder and inventory.

Figure 5. Assembly flow network for end product e in one time period
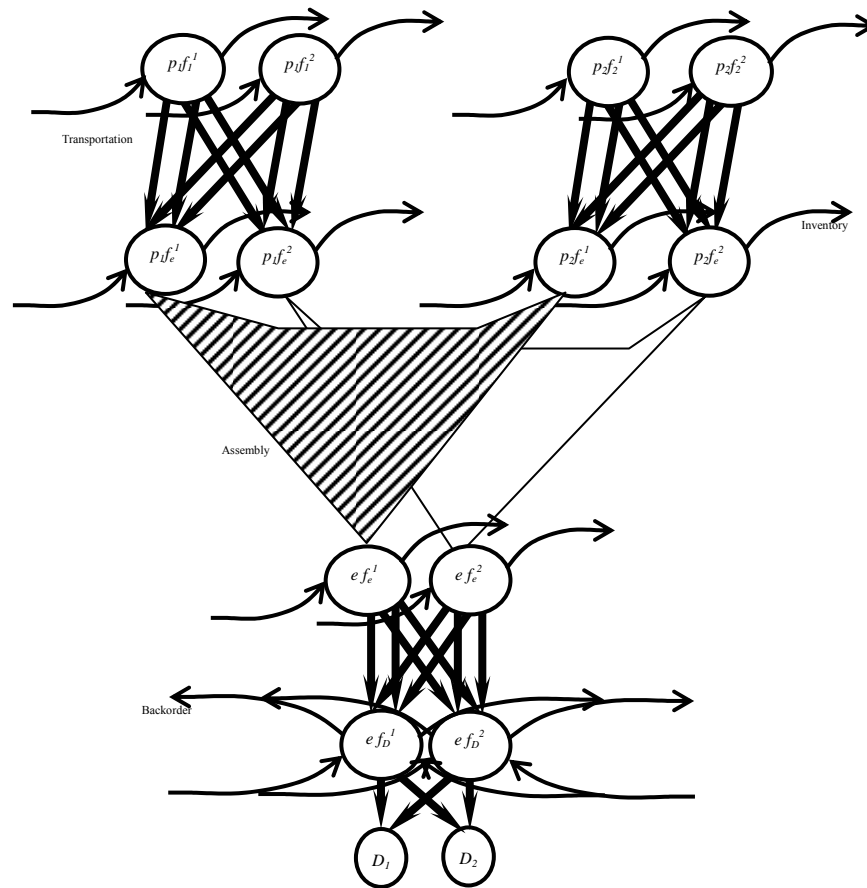
Figure 6. Assembly flow network for end product e in one time period (Hyper-graph)

components and its sub-assembly. The changes can also be viewed as using equalities

(3-1) to replace variables $u^{2,e}_{p_1 Pft}$ by $u^{2,e}_{p_e^* Pft}$. Thus, we obtain the network flow problem over

hyper-graphs, so that the material flow with the assembly operation becomes the net-

work flow problem on hyper-graph. Figure 6 shows the hyper-graph on which we define

the network flow problem. Compared with Figure 5 we have hyper-edges and remove

the sink nodes.

The above hyper-graph of the material flow network with assembly operation can be viewed as an extension of the time-space network. A time-space network is a special application of the state-transfer graph, whose nodes can represent specific locations at specific times and whose edges can represent the changes of states of locations or times, to describe the material flow in our model. To describe the material flow network with assembly operation, we use an extension of the time-space network, the *component-time-space network,* in which each node represents a specific component p at a specific facility alternative f during a specific time period t, and each arc represents a change of state of component p (i.e., production or assembly), facility alternative f (i.e., transportation), or time period t (i.e., inventories or backorders). Because an assembly operation has flows of several parts in and the flow of only the assembly out, we represent it as a hyperedge, which allows multiple tails and heads. The component-time-space network can be also viewed as a special application of the state-transfer graph.

3.2. Summary

Wilhelm et al. (2005) contributed a model that represents the complexities of the international design problem, integrating relevant enterprise-wide decisions in the US-Mexico business environment under NAFTA. This paper was part of a team effort. The contribution of this dissertation is to model the material network with assembly operations.

PADSDPN can be viewed as an extension of PADSDP. The most important piece of PADSDPN is PADSDP. Our research in the rest of the dissertation focuses on the solution approach to PADSDP.

4. COMPARISON OF DECOMPOSITION SCHEMES*

The purpose of this section is to study several decomposition schemes for applying Dantzig-Wolfe decomposition (DWD) to the production-assembly-distribution system design problem (PADSDP). Each scheme exploits selected embedded structures. Specific research objectives are to enhance the rate of DWD convergence in application to PADSDP through formulating a rationale for decomposition by analyzing potential schemes, adopting acceleration techniques, and assessing the impacts of schemes and techniques computationally.

This section is organized in five sections and is self contained. Subsection 4.1 presents the PADSDP model in a compact way to facilitate our discussion. Subsection 4.2 describes certain embedded structures that can be exploited by decomposition schemes. Subsection 4.3 presents a selected set of decomposition schemes and acceleration techniques. Subsection 4.4 evaluates the schemes and acceleration techniques computationally. Finally, subsection 4.5 offers concluding comments.

4.1. Compact Representation of PADSDP Model

Compared with the representation on the material network in section 2, the compact representation of PADSDP model employs z_a to present the flow on a - be it production, assembly, distribution, inventory or backorder, which is scaled in terms of the number of

end products to facilitate the flow conservation at each node. The material flow network for each end product e comprises a layer of nodes and arcs for each time period, $t \in T$; each node represents a pft combination (i.e., component $p \in P_e$ allocated to alternative f during period t). Each directed arc $a \in A_e$ represents a change of state for the material that flows across it. An arc with both ends in one layer (i.e., period) indicates production, assembly, or distribution, while an arc pointing from a node in period t $(t+1)$ to another in period $t+1$ (t) allows inventory (backorder) carry over. Subset $A_{pft} \subseteq A$ denotes directed arcs representing (depending upon the type of facility) the supply, production, assembly, or distribution of p at f during t; and A_{pft}^+ $\left(A_{pft}^-\right)$, directed arcs that start (end) at node pft.

Decision variables x_f and y_{pf}, and set and parameter definition are almost the same as those in section 0. We now present the PADSDP model in a compact way:

Model $\wp$

$$\text{Min} \sum_{f \in F} G_f^O x_f + \sum_{e \in E} \sum_{p \in P_e} \sum_{f \in F_p} G_{pf}^A y_{pf} + \sum_{e \in E} \sum_{a \in A_e} G_a^V z_a \qquad\qquad (4\text{-}1)$$

$$-Q_{ft} x_f + \sum_{e \in E} \sum_{p \in P_e} \sum_{a \in A_{pft}} (q_{pf} N_{pe}) z_a \leq 0 \quad \forall\ f \in F, t \in T \qquad |F| \cdot |T| \qquad (4\text{-}2)$$

$$-x_f + y_{pf} \qquad\qquad\qquad\qquad \leq 0 \quad \forall\ e \in E, p \in P_e, f \in F_p \qquad \sum_{e \in E} \sum_{p \in P_e} |F_p| \qquad (4\text{-}3)$$

$$-\bar{D}_e y_{pf} + \sum_{t \in T} \sum_{a \in A_{pft}} z_a \leq 0 \quad \forall\ e \in E, p \in P_e, f \in F_p \qquad \sum_{e \in E} \sum_{p \in P_e} |F_p| \qquad (4\text{-}4)$$

$$\sum_{f \in F_l} x_f \qquad\qquad\qquad\qquad \leq 1 \quad \forall\ l \in L \qquad |L| \qquad (4\text{-}5)$$

$$\sum_{p \in P_e} \sum_{f \in F_p} G_{pf}^A y_{pf} \qquad \leq B_e \quad \forall\ e \in E \qquad |E| \qquad (4\text{-}6)$$

$$\sum_{a \in A_{pft}^+} z_a - \sum_{a \in A_{pft}^-} z_a \qquad = D_{pft} \quad \forall\ e \in E, p \in P_e, f \in F_p, t \in T \quad \sum_{e \in E} \sum_{p \in P_e} |F_p| \cdot |T| \quad (4\text{-}7)$$

$$x_f \qquad \in \{0,1\} \ \forall \ f \in F \qquad\qquad |F| \qquad (4\text{-}8)$$

$$y_{pf} \qquad \in \{0,1\} \ \forall \ e \in E, p \in P_e, f \in F_p \qquad \sum_{e \in E} \sum_{p \in P_e} |F_p| \quad (4\text{-}9)$$

$$z_a \qquad \geq 0 \qquad \forall \ e \in E, a \in A_e \qquad\qquad |A| \qquad (4\text{-}10).$$

We include the last column in (4-1)-(4-10), which specifies the number of con-straints of each type ($|\bullet|$ denotes the cardinality of set $\bullet$), to facilitate later discussion of our decomposition schemes. We assume that the bill-of-materials (BOM) for each end product e is a tree structure and requires a unique set of components. Sets P and A comprise disjoint subsets P_e and A_e for $e \in E$. Set E is the alias of P^E, the set of end products.

Objective function (4-1)) minimizes total cost, including fixed costs related to opening f, G_f^O, and allocating p to f, G_{pf}^A, and variable costs (sourcing, producing, assembling, transporting, or stocking) for components that flow on arc a, G_a^V. Con-straint (4-2) relates x_f and z_a, requiring the workload for components allocated to f observe capacity Q_{ft} (or 0 if $x_f = 0$) during t. Workload parameter q_{pf} - the time to process each component of type p at f times the number of p that comprise e, N_{pe} - scales flow in terms of the end product. Each constraint (4-3) assures that an alternative is opened if any component is allocated to it. Each constraint (4-4) is an aggregated vari-able upper bound (AVUB) constraint, so that y_{pf} restricts related flows z_a for $a \in \overline{A}_{pf}$ (i.e., $\overline{A}_{pf} \equiv \bigcup_{t \in T} A_{pft}$) to at most $\overline{D}_e$, the total demand for e in all periods (note: z_a is scaled in terms of the number of e). Flow associated with p can go through f only if it

is allocated to f (i.e., $y_{pf} = 1$). Each constraint (4-5) allows at most one alternative to be opened at one location, l. Each constraint (4-6) imposes a budget limitation B_e on the fixed cost associated with all $p \in P_e$ for e. Each constraint (4-7) requires the flow out of node pft minus the flow in to equal the demand for p at f during t, D_{pft}. Constraints (4-8) and (4-9) require variables x_f and y_{pf} to be binary, respectively, and (4-10)) impose non-negativity restrictions on variables z_a.

4.2. Special Structures in the Model

Model $\wp$ embeds three special constraint structures. One structure comprises (4-6) and (4-9), forming an NP-hard, multiple-choice knapsack problem for each e that can be solved by a pseudo-polynomial dynamical programming algorithm (e.g., Dudzinski and Walukiewicz, 1987). This structure, which we call sub-problem type one (SP1), does not have the Integrality Property (Geoffrion, 1974). Set covering constraints (e.g., $\sum_{f \in F_p} y_{pf} \geq 1$, $\forall\ e \in E, p \in P_e$), requiring p to be allocated to at least one alternative, can be also included in SP1. A second structure comprises (4-7) and (4-10), forming a minimum-cost flow problem on a hyper-graph. This structure, sub-problem type two (SP2), is a linear program and can be solved in polynomial time (Cambini et al., 1997). The third structure comprises (4-4), which link variables in SP1 and SP2.

We define an extension of SP1 (SP1E) that comprises SP1 (i.e., (4-6) and (4-9)), (4-4), and (4-10); and an extension of SP2 (SP2E) that comprises SP2 (i.e., (4-7) and (4-10)), (4-4), and the linear relaxation of (4-9) (i.e., $0 \leq y_{pf} \leq 1, e \in E, p \in P_e, f \in F_{pf}$). Proposition 4.1 and Proposition 4.2 show that SP1E polynomially reduces to SP1 and

that SPE2 can be reformulated as SP2. Because (4-4), (4-6), (4-7), (4-9), and (4-10) are

separable relative to e, we introduce notation SP1(e), SP2(e), SP1E(e), and SP2E(e)

to represent SP1, SP2, SP1E, and SP2E, respectively, relative to e.

Proposition 4.1: SP1E(e) polynomially reduces to SP1(e).

Proof: For $e \in E$, SP1E(e) can be formulated as:

$$\text{SP1E}(e) \, Z^*_{SP1E(e)} = \max \sum_{p \in P_e} \sum_{f \in F_p} \sum_{t \in T} \sum_{a \in A_{pft}} c_a z_a + \sum_{p \in P_e} \sum_{f \in F_p} c_{pf} y_{pf}, \; s.t.(4\text{-}4),(4\text{-}6),(4\text{-}9),(4\text{-}10) \text{ for } e.$$

We show that any optimal solution of SP1E(e) can be obtained from an optimal solution

to the corresponding SP1(e). Given an optimal feasible solution sol to SP1E(e), (i) if

$y_{pf} = 0$, values of z_a for $a \in \bar{A}_{pf}$ are zero, and (ii) if $y_{pf} = 1$, values must be prescribed

for z_a, $a \in \bar{A}_{pf}$, to maximize $Z^*_{SP1E(e)}$ while satisfying $\sum_{t \in T, a \in A_{pft}} z_a \leq \bar{D}_e$; that is, values

of z_a for $a \in \bar{A}_{pf}$ are obtained by solving the continuous knapsack problem

CKP(pf) $\qquad g^*_{pf} = \max \sum_{t \in T} \sum_{a \in A_{pft}} c_a z_a, \; s.t. \; (4\text{-}10) \text{ and } \sum_{t \in T} \sum_{a \in A_{pft}} z_a \leq \bar{D}_e \text{ for a given } pf.$

Here, g^*_{pf} is defined as the optimal objective value of a continuous knapsack problem

CKP(pf). Thus, values of z_a in sol satisfy $\sum_{t \in T, a \in A_{pft}} c_a z_a = g^*_{pf} y_{pf}$ for the given pf by

(i) and (ii). Replacing $\sum_{t \in T, a \in A_{pft}} c_a z_a$ with $g^*_{pf} y_{pf}$ in the objective function, we get:

SP1(e) $\qquad \max \sum_{p \in P_e} \sum_{f \in F_p} (g^*_{pf} + c_{pf}) y_{pf}, \quad s.t. \; (4\text{-}6) \text{ and } (4\text{-}9) \text{ for } e.$

The number of continuous knapsack problems of the form CKP(pf) is $\sum_{p \in P_e} |F_p|$. For

each pf, g^*_{pf} can be obtained in polynomial time because CKP(pf) is a linear program

and can be solved in polynomial time (Bazaraa et al., 2005). Thus, SP1E polynomially reduces to SP1. ∎

We define the linear relaxation $\mathcal{LP}$ of model $\wp$, which replaces constraints (4-8) with $0 \le x_f \le 1,\ f \in F$ and (4-9) with $0 \le y_{pf} \le 1, e \in E, p \in P_e, f \in F_{pf}$. Because y_{pf} variables are incorporated only in AVUB constraints (4-4), which force them to be positive, it is clear that (4-4) must be active at optimality in a continuous relaxation, provided cost coefficients are positive (i.e., $G_{pf}^A > 0$).

Proposition 4.2: SP2E(e) can be reformulated as SP2(e).

Proof: AVUB constraints (4-4) hold at equality. Variable y_{pf} can be replaced by $\sum_{t \in T, a \in A_{pft}} z_a / \overline{D}_e$ so that y_{pf} and constraints (4-4) can be eliminated from SP2E(e).∎
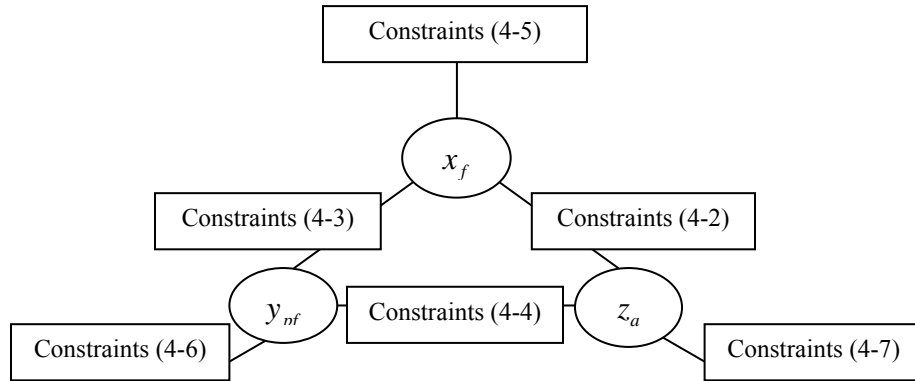
Figure 7. Symmetric structure of model

4.3. Decomposition Schemes and Acceleration Techniques

This subsection discusses the five decomposition schemes that we study and justifies their use. In addition, it describes the techniques that we adopt to accelerate DWD convergence. Here, we point out several characteristics of Model $\wp$ that can be exploited by decomposition schemes.

Figure 7 depicts symmetrical relationships embedded in Model $\wp$: (4-2) links each x_f to several z_a for $a \in \bigcup_{p \in P_f} \overline{A}_{pf}$, (4-3) links each x_f to several y_{pf} for $p \in P_f$, and (4-4) links each y_{pf} to several z_a for $a \in \overline{A}_{pf}$. In addition, Model $\wp$ imposes restrictions on x_f, y_{pf}, and z_a individually.
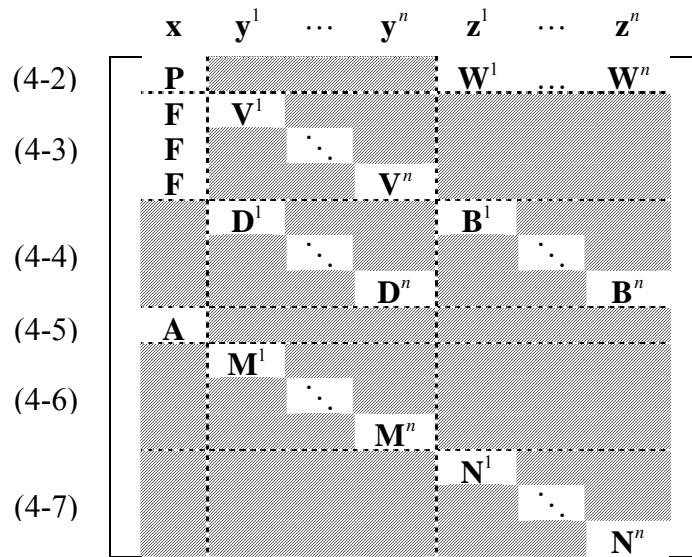
Figure 8. Matrix representation of model (4-2)-(4-7)

Figure 8 depicts (4-2)-(4-7) in matrix notation, highlighting block diagonal forms. $\mathbf{x}$, $\mathbf{y}^1$, $\cdots$, $\mathbf{y}^n$, $\mathbf{z}^1$, $\cdots$, $\mathbf{z}^n$ represent vectors of variables x_f, y_{pf}, z_a, respectively; each superscript relates a vector to one of the $n = |E|$ end products. Matrices $\mathbf{A}$, $\mathbf{B}^1$, $\cdots$, $\mathbf{B}^n$, $\mathbf{D}^1$, $\cdots$, $\mathbf{D}^n$, $\mathbf{F}$, $\mathbf{P}$, $\mathbf{M}^1$, $\cdots$, $\mathbf{M}^n$, $\mathbf{N}^1$, $\cdots$, $\mathbf{N}^n$, $\mathbf{V}^1$, $\cdots$, $\mathbf{V}^n$ and $\mathbf{W}^1$, $\cdots$, $\mathbf{W}^n$ represent constraint coefficients associated with these vectors. Finally, we note that certain constraints (e.g., (4-4), (4-6), and (4-7)) are separable relative to e and we exploit this characteristic in our schemes to facilitate solution by dealing with smaller sub-problems.

4.3.1. Decomposition schemes

Primary concerns in designing a decomposition scheme are to obtain small sub-problems that can be solved effectively, a small restricted master problem (RMP) that is not highly degenerate, a favorable comparison with other schemes, and a form amenable to effective acceleration techniques. One goal is to facilitate solution by obtaining a sub-problem exhibits a block diagonal structure so that it can be separated into several small sub-problems (Bazaraa et al., 2005) to facilitate sub-problem solution, while avoiding the Integrality Property, allowing RMP to give tighter bounds than $\mathscr{LP}$. Another goal is to achieve a favorable rate of convergence as measured by the number of iterations required to solve RMP. This rate depends on the number rows in RMP and restrictions on the values of dual variables in the linear programming dual of RMP (DRMP). Degeneracy of RMP also slows the rate of convergence so that schemes that result in highly degenerate RMPs are not desirable. For example, the polytope formed by (4-3) and bounds on x_f and y_{pf} variables is degenerate. Similarly, (4-4) and bounds on y_{pf} and z_a forms a degenerate polytope.

We focus our study on five decomposition schemes, which are based on the four sub-problem structures described in subsection 4.2 0(SP1, SP1E, SP2, and SP2E): allocation decomposition (A), flow decomposition (F), AVUB-allocation decomposition (VA), AVUB-flow decomposition (VF), and allocation-flow decomposition (AF). Each scheme involves relaxing a subset of binary constraints and comprises a block diagonal structure that can be exploited to facilitate its solution. Table 2 details these schemes.

Table 2. Partitions of constraints and variables for decomposition schemes

<table>
<thead>
<tr><th></th><th></th><th>Scheme (A)</th><th>Scheme (VA)</th><th>Scheme (F)</th><th>Scheme (VF)</th><th>Scheme (AF)</th></tr>
</thead>
<tbody>
<tr><td rowspan="3">Constraints</td><td>MP</td><td>(1)-(4), (6), (7), (9)</td><td>(1), (2), (4), (6), (7)</td><td>(1)-(5), (7), (8)</td><td>(1), (2), (4), (5), (7)</td><td>(1)-(4), (7)</td></tr>
<tr><td>SP</td><td>(5), (8)</td><td>(3), (5), (8), (9)</td><td>(6), (9)</td><td>(3), (6), (8), (9)</td><td>(5), (6), (8), (9)</td></tr>
<tr><td>Relax</td><td>(7): $0 \le x_f \le 1$</td><td>(7): $0 \le x_f \le 1$</td><td>(7): $0 \le x_f \le 1$
(8) $0 \le y_{pf} \le 1$</td><td>(7): $0 \le x_f \le 1$
(8): $0 \le y_{pf} \le 1$</td><td>(7): $0 \le x_f \le 1$</td></tr>
<tr><td rowspan="2">Variables</td><td>MP</td><td>x_f, z_a</td><td>x_f</td><td>x_f, y_{pf}</td><td>x_f</td><td>x_f</td></tr>
<tr><td>SP</td><td>y_{pf}</td><td>y_{pf}, z_a</td><td>z_a</td><td>y_{pf}, z_a</td><td>y_{pf}, z_a</td></tr>
<tr><td colspan="2">SP Structure</td><td>SP1</td><td>SP1E</td><td>SP2</td><td>SP2E</td><td>SP1 & SP2</td></tr>
</tbody>
</table>

While other combinations of constraints could be used to form decomposition schemes, our preliminary analysis excludes them as unworthy of consideration. For example, all of our schemes relegate x_f and related constraints (i.e., (4-2), (4-3), and (4-5)) to master problem because there are few such variables and our preliminary computational experience showed that prescribing them in sub-problem does not provide sufficient leverage to facilitate RMP solution. Further, any scheme with x_f in sub-problem can be viewed as an extension of one of our five schemes. For example, one can view a

scheme that relegates (4-2), (4-3) and (4-4) to master problem as an extension of (AF), but we expect that it would convergence more slowly than (AF) because it requires master problem to coordinate three types of variables, while (AF) deals with only two types. Finally, we avoid schemes for which sub-problem has the Integrality Property.

Considering constraints (4-4), (4-6), and (4-7), a total of $8 = 2^3$ schemes are possible. One of these schemes places (4-4), (4-6), and (4-7) in master problem so that sub-problem is vacuous; another places (4-4), (4-6), and (4-7) in sub-problem, which cannot be solved effectively, and a third relegates (4-6) and (4-7) to master problem and places (4-4) in sub-problem, which poses no special structure to exploit. We study the remaining five possible schemes.

4.3.2. Acceleration techniques

We adopt two acceleration techniques. The first, which we call *column disaggregation* (CD), was proposed in application to the multi-commodity network flow problem by Jones et al. (1993), who analyzed the impact of formulation on the success of DWD. CD decreases the number of columns in master problem. In scheme (F), the solution to SP2(e) prescribes material flow to satisfy the demands for end product e from all customer zones during all periods. We can disaggregate this flow into a set of flows, each of which satisfies the demand for e from one customer zone during one period. We reformulate $\wp$ using disaggregated variable z_a^{eft}, which represents the scaled amount of flow on arc a to satisfy demand D_{eft}, so that $z_a = \sum_{t \in T, \{f \in F_e : D_{eft} > 0\}} z_a^{eft}$ for $e \in E, a \in A_e$. CD increases the number of sub-problems and, thus, the number of convexity constraints in

master problem. Letting CZ_e represent the number of customer zones for e, the number of convexity constraints will increase from $|E|$ in scheme (F) to $\sum_{e \in E} CZ_e \cdot |T|$ for CD, increasing the run time to solve RMP.

The second technique, *extra dual cuts* (EDC), was proposed by Valerio de Carvalho (2005) in application to the cutting stock problem. EDC adds columns to RMP and the corresponding cuts in DRMP. These cuts do not affect the optimal objective value of DRMP but provide additional restrictions in the dual space that can improve the rate at which column generation convergences.

The difficulty in applying EDC is in specifying these dual cuts. In our application, a feasible solution to SP1(e) allocates components to alternatives under budget limitation (4-6). Following Valerio de Carvalho (2005), we obtain EDCs by using the symmetric difference of two feasible solutions to SP1(e). Each EDC is associated with a vector representing the reallocation of a component to a different alternative to reduce cost relative to the budget; that is, i) reallocate p from f to the next cheaper alternative, or ii) forbid p to be allocated to f if f is already the cheapest one. To generate EDCs, we order all alternatives to which p can be allocated according to highest G_{pf}^A value first. The number of EDCs is equal to the number of possible allocations of components to alternatives, $\sum_{e \in E, p \in P_e} |F_p|$. A pre-processing step can generate a batch of these dual cuts and incorporate them in RMP. Reallocating p to a non-adjacent (on the ordered list) cheaper alternative can be achieved by making a sequence of adjacent-alternative reallocations. To forbid the allocation of p to an alternative other than the cheapest one, we

can reallocate it to the cheapest alternative and then forbid the cheapest one. Given a feasible solution $\mathbf{y}^*$ to SP1(e), which represents an allocation of components to alternatives under budget restriction (4-6), each reallocation on $\mathbf{y}^*$ yields another solution to SP1(e). The advantage that EDCs offer is that fewer columns from SP1 need be incorporated in RMP, improving convergence.

Next, we use a small example to illustrate EDCs. Suppose e comprises two components $\{p_1, p_2\}$ and associated sets of alternatives $\{f_1, f_2, f_3\}$ and $\{f_1, f_3\}$, respectively; costs G_{pf}^{A} satisfy $G_{p_1 f_1}^{A} \geq G_{p_1 f_2}^{A} \geq G_{p_1 f_3}^{A}$ and $G_{p_2 f_1}^{A} \geq G_{p_2 f_3}^{A}$. The vector $\mathbf{y} = (y_{p_1 f_1}, y_{p_1 f_2}, y_{p_1 f_3}, y_{p_2 f_1}, y_{p_2 f_3})$. Suppose we have SP1($e$) solution $\bar{\mathbf{y}} = (1,1,0,1,1)^{T}$, which allocates p_1 to f_1 and f_2 and p_2 to f_1 and f_3. We can introduce EDCs representing the reallocation of a component to reduce cost by using a cheaper alternative or by forbidding some alternative. For p_1, we introduce vectors $(-1,1,0,0,0)^{T}$, $(0,-1,1,0,0)^{T}$ and $(0,0,-1,0,0)^{T}$. $(-1,1,0,0,0)^{T}$ reallocates p_1 from f_1 to f_2, $(0,-1,1,0,0)^{T}$ reallocates p_1 from f_2 to f_3, and $(0,0,-1,0,0)^{T}$ forbids the allocation of p_1 to f_3. With these three vectors, we can construct $(-1,0,1,0,0)^{T} = (-1,1,0,0,0)^{T} + (0,-1,1,0,0)^{T}$, indicating the reallocation of p_1 from f_1 to f_3. For p_2, we have $(0,0,0,-1,1)^{T}$ and $(0,0,0,0,-1)^{T}$, which can construct $(0,0,0,-1,0)^{T} = (0,0,0,-1,1)^{T} + (0,0,0,0,-1)^{T}$ to forbid the allocation of p_2 to f_1. The SP1(e) solution $(1,0,1,0,1)^{T}$, which allocates p_1 to f_1 and f_3 and p_2 to f_3, can be constructed from allocation $(1,1,0,1,1)^{T}$ and reallocations $(0,-1,1,0,0)^{T}$, $(0,0,0,-1,1)^{T}$, and $(0,0,0,0,-1)^{T}$.

We use CD and EDC to enhance decomposition schemes (F) and (AF), respectively, giving disaggregation-flow decomposition (DF) and allocation-flow with dual cuts decomposition (AFD).

4.4. Computational Results

We conduct all experiments on a PC with a 2.8 GHz Pentium IV processor and 1 GB of RAM using Visual Studio C++ 6.0 and the CPLEX 9.0 callable library. The purpose of our tests is to assess the influences of decomposition schemes and acceleration techniques on the convergence of DWD so we focus on their performances at the root node of the branch-and-bound (B&B) tree, following studies such as Vanderbeck (2006, 2005), Sherali et al. (2005), Glover et al. (1997) , Hu and Johnson (1999). Reporting the (optimal) integral solution would require use of branch-and-price (B&P) and that is beyond the scope of this subsection.

4.4.1. Test instances

We use real-world data sources to specify attributes of locations (i.e., costs of labor, land and capital), alternatives (i.e., labor, land, and cost/unit of capacity, total capacity, and economies of scale), and components (i.e., setup times; workloads for assembling, stocking, or transporting; and demands for end products). Our experiments are based on six factors (and associated levels): $|T|$ (12 or 24); $|E|$ (1, 4, or 8); the number of components in the BOM for end products, *BOMS* (3, 7, or 15); the type of BOM, *BOMT* (a vertical path or a binary tree); the number of alternatives to which each component can be allocated, *FA* (4 or 8); and the capacity tightness ratio, *CT* (0.1 or 0.2). We define a *scenario* as a unique selection comprising one level of each factor. Each *scenario* is

tested in ten *instances*, each of which has the same factor-level selection but is generated using a unique set of random number seeds. We consider eight scenarios, generating parameters and sets (e.g., G_f^O, P, etc.) used in Model $\wp$. Table 3 details the level of each factor associated with each scenario. Table 4 details the sizes of instances and results of applying CPLEX to $\mathscr{LP}$. Because all instances related to each scenario required about the same run time, we report average values for the ten instances associated with each scenario.

Table 3. Factor levels that determine each scenario

Scenario	$|T|$	$|E|$	BOMS	BOMT	FA	CT
1	12	1	3	Vertical path	4	0.2
2	24	1	3	Vertical path	4	0.2
3	12	4	7	Binary tree	4	0.2
4	24	4	7	Binary tree	4	0.2
5	12	4	15	Binary tree	4	0.2
6	24	4	15	Binary tree	4	0.2
7	12	8	15	Binary tree	8	0.1
8	24	8	15	Binary tree	8	0.1

Table 4. CPLEX results

Scenario	Binary	Continuous	Number Rows	Run Time (secs.)	SITR
1	24	1036	454	0.0111	223.9
2	24	2092	886	0.0326	460.8
3	146	8352	2610	0.1327	1664.5
4	146	16858	4990	0.4235	3300.1
5	294	16997	5396	0.9719	4972.1
6	294	34327	10365	4.6562	9399.8
7	1166	118031	21897	11.9688	12725.4
8	1166	237522	42229	110.3453	26532.4

Measures in Table 4 include the numbers of relaxed binary variables (BIN), continuous variables (CONT), and constraints (ROW), as well as average total run time (RT) in seconds and average number of Simplex iterations (SITR).

Table 5 gives results. The size of a PADSDP instance increases rapidly with $|P| \cdot |F|^2 \cdot |T|$, rendering the linear relaxation of our model challenging to solve and highlighting the importance of studying the performance of DWD at the root node of the B&B search tree. Scheme (VA) cannot solve some instances of scenarios 3 to 8 within 20 minutes, so we designed smaller scenarios 1 and 2 to compare schemes (A) and (VA). We do not report run times for (A) and (VA) on instances 3 to 8, nor run times for other schemes in application to scenarios 1 and 2 because they can solve the larger instances of scenarios 3 to 8. Test results allow the DWD convergence rates of schemes to be compared, providing insights into the impacts of parameters on instance size and the efficacy of each decomposition scheme. We study the performance of each scheme in solving RMP at the root node using several measures of performance: total run time (RT) in seconds, bound prescribed by RMP (BND), improvement in bound (IMPR) (i.e., percent improvement from the value of the optimal $\mathscr{LP}$ solution to BND), number of RMP iterations (RITR), total run time to solve sub-problems (SPT), total run time to solve RMP (RMPT), and number of columns generated (COLS). Note that we add all improving columns from sub-problems at each RMP iteration. We also tested various CPLEX solvers in application to DWD. After comparing primal and dual simplex solvers and finding that the latter is faster in this application, we report results using only the latter.

Our preliminary analysis also showed that CPLEX's barrier algorithm was not competi-

tive so we do not report run times for that option.

Table 5. Results

#	Scenario	Scheme	RT(secs.)	BND	IMPR	RITR	COLS
1	1	A	0.1248	4600628698.5542	0.00%	12.5	10.5
2	1	VA	10.0671	4600628698.5542	0.00%	813.9	811.9
3	2	A	0.4874	8258955987.0453	0.00%	18.6	16.6
4	2	VA	233.184	8258955987.0453	0.00%	2276.8	2274.8
5	3	A	4.0735	55317769707.8944	0.00%	32.6	91.9
6	3	F	0.0374	55476851293.6798	0.00%	10.5	17.9
7	3	VF	0.0391	55476851293.6798	0.00%	13.0	29.4
8	3	DF	0.1002	55476851293.6798	0.00%	7.0	518.7
9	3	AF	0.5530	55317769707.8944	0.00%	58.0	285.7
10	3	AFD	0.0499	55317769707.8944	0.00%	13.5	54.3
11	4	A	15.8937	127445221460.0990	0.02%	40.4	111.4
12	4	F	0.0689	127853210969.3860	0.00%	10.8	21.3
13	4	VF	0.0655	127853210969.3860	0.00%	11.8	28.4
14	4	DF	0.2393	127853210969.3860	0.00%	6.4	1011.6
15	4	AF	2.2126	127445221460.0990	0.02%	66.7	334.4
16	4	AFD	0.0797	127445221460.0990	0.02%	14.4	63.5
17	5	A	56.2719	131513567768.2590	0.09%	89.9	274.2
18	5	F	0.1486	132848989533.7790	0.00%	20.9	51.6
19	5	VF	0.1828	132848989533.7790	0.00%	27.0	70.4
20	5	DF	0.3672	132848989533.7790	0.00%	10.4	1178.9
21	5	AFD	0.4673	131513567768.2590	0.09%	53.1	254.6
22	6	A	384.0343	204840753142.5190	0.03%	119.6	325.7
23	6	F	0.4531	206644712109.8640	0.00%	35.9	83.8
24	6	VF	0.5017	206644712109.8640	0.00%	39.8	99.1
25	6	DF	1.1469	206644712109.8640	0.00%	11.1	2553.5
26	6	AFD	0.9874	204840753142.5190	0.03%	67.9	286.2
27	7	F	1.8843	334941838394.1040	0.00%	36.0	196.3
28	7	VF	3.7233	334941838394.1040	0.00%	63.0	292.5
29	7	DF	35.4953	334941838394.1040	0.00%	24.4	13137.5
30	8	F	10.5938	661818112752.7330	0.00%	106.9	320.2
31	8	VF	10.7579	661818112752.7330	0.00%	70.1	296.2

4.4.2. DWD convergence

In this subsection, we outline a set of measures that relate each scheme to DWD convergence and formulate two conjectures that we assess through computational tests.

DWD represents each sub-problem polytope in master problem as the convex combination of its extreme points (to facilitate discussion, we assume that sub-problem polytopes are bounded); each corresponds to a column in master problem. The number of these columns depends on the sub-problem polytopes, which we describe by three measures: i) the number of decision variables in sub-problem, ii) the number of constraints in sub-problem, and iii) the number of convexity constraints in RMP, which is the same as the number of sub-problems.

The number of extreme points in a sub-problem polytope is likely correlated with (i) and (ii). The standard form of a linear program has n variables and m constraints. The number of basic feasible solutions, which is greater than or equal to the number of feasible extreme points, is less than or equal to C_{m+n}^n (Bazaraa et al., 2005). As m increases, the value of C_{m+n}^n increases. Intuitively, a sub-problem polytope with fewer variables and constraints has fewer extreme points, although this may not be true in all cases. If sub-problem exhibits a block diagonal structure, it can be decomposed into several small sub-problems (Bazaraa et al., 2005), likely reducing the number of columns in master problem and improving the rate of convergence.

To prescribe an optimal solution to RMP, an optimal *basis*, a combination of t columns from the set of s columns in master problem, must be found. An upper bound on the number of possible bases is C_s^t. If s is not large relative to t, C_s^t is not large, so

there are relatively few bases to search over. Thus, we conjecture that a master problem formulation with fewer columns may lead to faster convergence.

Conjecture 4.3: The number of iterations required to prescribe an optimal solution to RMP is positively correlated with the number of columns in master problem.

RMP converges slowly due to the oscillation of dual variable values (Ben Amor et al., 2004). Convergence can be improved by restricting each dual variable value to a small region. We use three measures of RMP to describe restrictions on dual variables values in DRMP: (i) the number of constraints in RMP; (ii) the number of columns in RMP; and (iii) the order of degeneracy of RMP. The number of constraints in RMP determines the number of dual variables in DRMP. More dual variables increase the dimension of the dual space and result in more degrees of freedom so that it may become more difficult to restrict each dual variable value to a small region. A larger number of columns in RMP correspond to more cuts in the associated dual space, which helps to restrict the oscillation of dual variable values. An optimal degenerate solution in RMP corresponds to alternative optimal solutions in DRMP so that RMP degeneracy leads to more degrees of freedom and may allow dual variables to take on a wider range of feasible values.

Conjecture 4.4: Weaker restrictions on values of dual variables in DRMP lead to slower RMP convergence.

4.4.3. Analysis of decomposition schemes

Table 6 summaries measures discussed in subsection 4.4.2 for the five schemes. Measures related to the sub-problem polytope include the number of variables in sub-problem

(SPV), the number of constraints in sub-problem (SPC), the number of extreme points associated with the sub-problem polytope (SPEP), the number of diagonal blocks (i.e., convexity constraints) (CNV). Measures affecting restrictions on dual variable values in DRMP include the number of variables in RMP (RMPV), the number of non-convexity constraints in RMP (RMPC), and constraints that may cause RMP to be degenerate (DEG). Other measures include the expected solvability of sub-problem and RMP, and relationships among bounds.

We use MP(-), RMP(-), DRMP(-), and SP(-) to denote master problem, RMP, DRMP, and sub-problem, respectively, associated with scheme -. Based on the numbers of constraints and variables in master problem and sub-problem, it is easy to obtain SPV, SPC, RMPV, and RMPC in Table 6. SP(A) and SP(VA) employ SP1 and SP1E, respectively. SP(F) and SP(VF) employ SP2 and SP2E, respectively. SP(AF) includes both SP1 and SP2. Suppose that Z_{IP}^* (Z_{LP}^*) is the optimal objective value of $\wp$ ($\mathscr{LP}$) and Z_A^* (Z_{VA}^*, Z_F^*, Z_F^*, Z_{VF}^*, and Z_{AF}^*) is the lower bound provided by scheme (A) ((VA), (F), (VF), and (AF)) (i.e., the optimal solution to RMP at the root node of the B&B tree), $Z_{LP}^* = Z_F^* = Z_{VF}^* \leq Z_A^* = Z_{VA}^* = Z_{AF}^* \leq Z_{IP}^*$ because SP1(e) and SP1E(e) do not have the Integrality Property. The number of convexity constraints in RMP(AF) is equal to $2|E|$, half correspond to SP1; and half, to SP2. Other schemes include $|E|$ convexity constraints.

In scheme (A), the number of extreme points of the SP1(e) polytope is equal to the number of combinations of alternatives to which $p \in P_e$ might be allocated. p must

Table 6. Analyses of decomposition schemes

		Scheme (A)	Scheme (VA)	Scheme (F)	Scheme (VF)	Scheme (AF)
Representations of SP	SPV	$\sum_{p\in P}\lvert F_p\rvert$	$\sum_{t\in T}\sum_{p\in P,\,f\in F_p}\lvert A_{pft}\rvert+\sum_{p\in P}\lvert F_p\rvert$	$\lvert A\rvert$	$\lvert A\rvert+\sum_{p\in P}\lvert F_p\rvert$	$\lvert A\rvert+\sum_{p\in P}\lvert F_p\rvert$
	SPC	$\lvert P\rvert+\lvert E\rvert$	$\sum_{p\in P}\lvert F_p\rvert+\lvert P\rvert+\lvert E\rvert$	$\sum_{p\in P}\lvert F_p\rvert\cdot\lvert T\rvert$	$\sum_{p\in P}\lvert F_p\rvert\cdot(\lvert T\rvert+1)$	$\sum_{p\in P}\lvert F_p\rvert\cdot\lvert T\rvert+\lvert P\rvert+\lvert E\rvert$
	SPEP	$\prod_{p\in P_e}2^{\lvert F_p\rvert}-1$	$(\prod_{p\in P_e}2^{\lvert F_p\rvert}-1)\cdot\prod_{p\in P_e,\,f\in F_p,\,t\in T}(\sum_{t\in T}\lvert A^-_{pft}\rvert)$	SPEP(A) $\ll$ SPEP(F) < SPEP(VA)	Same as (F)	SPEP(AF) = SPEP (A) $\times$ SPEP (F)
	CNV	$\lvert E\rvert$	$\lvert E\rvert$	$\lvert E\rvert$	$\lvert E\rvert$	$2\lvert E\rvert$
	SP Solvability	SP1, pseudo-polynomial	SP1E, polynomially reducible to SP1	SP2, polynomial	SP2E can be reformulated as SP2	SP1 & SP2
Restrictions on duals	RMPV	$\lvert F\rvert+\lvert A\rvert$	$\lvert F\rvert+\lvert A\rvert-\sum_{t\in T}\sum_{p\in P}\sum_{f\in F_p}\lvert A_{pft}\rvert$	$\lvert F\rvert+\sum_{p\in P}\lvert F_p\rvert$	$\lvert F\rvert$	$\lvert F\rvert$
	RMPC	$\lvert L\rvert+\lvert F\rvert\cdot\lvert T\rvert+\sum_{p\in P}\lvert F_p\rvert\cdot(\lvert T\rvert+2)$	$\lvert L\rvert+\lvert F\rvert\cdot\lvert T\rvert+\sum_{p\in P}\lvert F_p\rvert\cdot(\lvert T\rvert+1)$	$\lvert L\rvert+\lvert F\rvert\cdot\lvert T\rvert+2\sum_{p\in P}\lvert F_p\rvert+\lvert E\rvert$	$\lvert L\rvert+\lvert F\rvert\cdot\lvert T\rvert+\sum_{p\in P}\lvert F_p\rvert+\lvert E\rvert$	$\lvert L\rvert+\lvert F\rvert\cdot\lvert T\rvert+2\sum_{p\in P}\lvert F_p\rvert$
	DEG	Constraints (2) with x_f, Constraints (3) with z_a	Constraints (2) with x_f	Constraints (3) with y_{pf}	Constraints (2) with x_f	Constraints (2) with x_f, Constraints (3)
Expected RMP Solvability		Challenged by larger size of RMP(A)	Less challenging than (A) More challenging than (VF)	Promoted by small size of RMP(F)	Less challenging than (F) because of smaller RMP	Less challenging than (F) because of smaller RMP
RMP Bound		$Z_A^*\geq Z_{LP}^*$	$Z_{VA}^*=Z_A^*\geq Z_{LP}^*$	$Z_F^*=Z_{LP}^*$	$Z_{VF}^*=Z_{LP}^*$	$Z_{AF}^*=Z_A^*\geq Z_{LP}^*$

be allocated to at least one alternative in F_p, giving $2^{|F_p|} - 1$ combinations. If $p \in P_e$ can

be allocated independently to alternatives, the number of combinations is

$\prod_{p \in P_e} (2^{|F_p|} - 1)$. In scheme (VA), given a solution to SP1(e), we can obtain a set of so-

lutions to SP1E(e), each using a different solution to one CKP(pf). The number of

CKP(pf) solutions for all pf combinations is $\prod_{p \in P_e, f \in F_p} |\bar{A}_{pf}|$ for a given solution to

SP1(e). The total number of solutions to SP1E(e) is equal to

$\prod_{p \in P_e, f \in F_p} |\bar{A}_{pf}| \cdot \prod_{p \in P_e} (2^{|F_p|} - 1)$. We expect that the SP1E(e) polytope has many more

extreme points than the SP1(e) polytope of scheme (A). Because RMP(A) is huge (only

constraints (4-6) and (4-9) are placed in sub-problem), it can not be expected to be read-

ily solvable. Compared with RMP(A), RMP(VA) includes fewer constraints and vari-

ables. Constraints (4-3) and (4-4) cause RMP(A) to be degenerate while (4-3) cause

RMP(VA) to be degenerate.

In scheme (F), the number of extreme points of the SP2(e) polytope is equal to

the number of spanning hyper-trees in the hyper-graph defining the assembly flow net-

work (Cambini et al., 1997). In scheme (VF), constraints (4-4) hold at equality, so each

solution to SP2(e) corresponds to a solution to SP2E(e). Polytopes associated with

SP2E(e) and SP2(e) have the same number of extreme points. We expect the solvabil-

ity of RMP(F) is better than that of RMP(A) and RMP(VA) because it comprises fewer

constraints and variables. RMP(VF) promotes solvability by comprising fewer con-

straints and variables than RMP(F). (4-3) and (4-4) cause RMP(F) to be degenerate, and

(4-3) cause RMP(VF) to be degenerate.

Table 7. Analyses of enhanced decomposition schemes

		Scheme (DF)	Scheme (AFD)
Representations of SP	SPV	$\lvert A \rvert$	$\lvert A \rvert + \sum_{p\in P}\lvert F_p \rvert$
	SPC	$\sum_{p\in P}\lvert F_p \rvert \cdot \lvert T \rvert$	$\sum_{p\in P}\lvert F_p \rvert \cdot (\lvert T \rvert + 1) + \lvert E \rvert$
	SPEP	SPEP(DF) = SPEP(F)	SPEP(AFD) = SPEP(AF)
	CNV	$\sum_{e\in E} CZ_e \cdot \lvert T \rvert$	$2\lvert E \rvert$
SP Solvability		SP2, polynomial	SP1 & SP2
Restrictions on duals	RMPV	$\lvert F \rvert + \sum_{p\in P}\lvert F_p \rvert$	$\lvert F \rvert + \lvert A \rvert$
	RMPC	$\lvert L \rvert + \lvert F \rvert \cdot \lvert T \rvert + 3\sum_{p\in P}\lvert F_p \rvert + \lvert E \rvert$	$\lvert L \rvert + \lvert F \rvert \cdot \lvert T \rvert + 2\sum_{p\in P}\lvert F_p \rvert$
	DEG	DEG(DF) = DEG(F)	DEG(AFD) = DEG(AF), adding extra dual cuts for constraints (3)
Expected RMP Solvability		More challenging than Scheme (F) because of extra convexity constraints	A little more challenging than Scheme (AF) because of extra dual cuts
RMP Bound		$Z_{DF}^* = Z_{LP}^*$	$Z_{AFD}^* = Z_A^* \geq Z_{LP}^*$

In scheme (AF), sub-problem has a block diagonal structure that is decomposed to form sub-problems with the SP1(e) and SP2(e) structures. The number of columns in MP(AF) is the sum of the number of extreme points associated with both SP1(e) and SP2(e) polytopes. We expect that the solvability of RMP(AF) is as good as RMP(VF) because both have about the same number of constraints and variables. Constraints (4-3) and (4-4) cause RMP(AF) to be degenerate.

Schemes (DF) ((AFD)) is based on (F) ((AF)). CD increases the numbers of sub-problems and convexity constraints in master problem from $\lvert E \rvert$ in (F) to $\sum_{e\in E} CZ_e \cdot \lvert T \rvert$

in (DF), while scheme (AFD) enhances (AF) by adding $\sum_{e \in E, p \in P_e} |F_p|$ EDCs in RMP.

Table 7 summarizes these two enhanced schemes.

4.4.4. Analysis of computational results

Results show the impact of increasing instance size on the efficacy of each scheme. Only (F) and (VF) can solve all instances of the largest scenarios within 20 minutes; they lead the list of schemes ordered according to fastest run time first: (F), (VF), (DF), (AFD), (AF), (A), and (VA).

Because SP1(e) and SP1E(e) do not have the Integrality Property, $Z_{LP}^* = Z_F^* = Z_{VF}^* = Z_{DF}^* \leq Z_A^* = Z_{VA}^* = Z_{AF}^* = Z_{AFD}^* \leq Z_{IP}^*$. IMPR is small because fixed and variable costs are positively correlated in real-world data, for which a location with lower fixed costs may likely have lower variable costs. Thus, budget limitations (4-6) will, in general, not be binding at the optimal solution.

We focus on DWD convergence, exploring the advantages and disadvantages of alternative decomposition schemes, which distribute the challenges of complexity between master problem and sub-problem. We seek to show which factors affect the rate of convergence of DWD. We also use results to evaluate the two conjectures posed in sub-section 4.4.2. Results support our conjectures, demonstrating that the expected number of iterations required to prescribe an optimal solution to RMP is likely to be positively correlated with the number of columns in master problem and that weaker restrictions on the values of dual variables in DRMP lead to slower convergence in prescribing an optimal solution to RMP.

To distinguish the relative performance of each scheme, we compare them in pairs, each of which is chosen relative to measures presented in subsection 4.4.2. Figure 9 depicts these pair wise comparisons, using each vertex to represent a scheme and each edge to represent a comparison. Schemes in the same row embed the same of type of sub-problem. We focus on two performance measures, RITR and RT, for six pairs of schemes compared in the following six subsections.
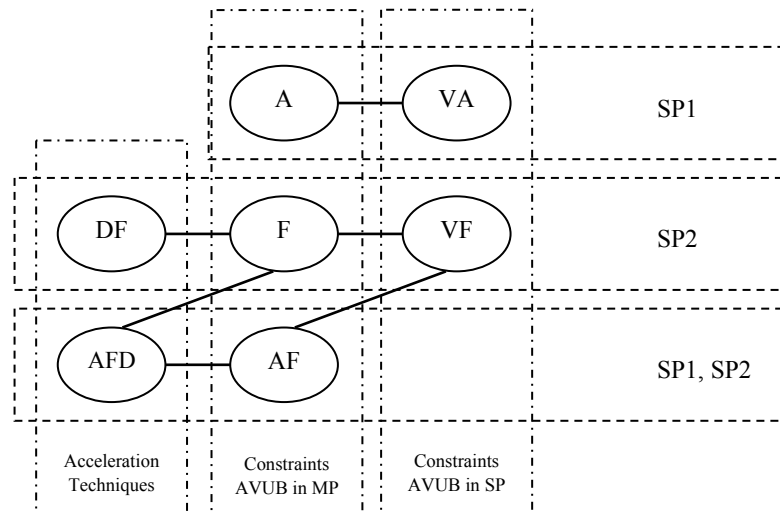
Figure 9. Comparison of decomposition schemes

4.4.4.1. Scheme (F) vs Scheme (DF)

Compared with (F), (DF) employs CD to accelerate convergence. CD decreases the number of columns in master problem so MP(DF) includes fewer columns than MP(F). Table 5 shows that RMP(DF) requires fewer RITR than RMP(F), supporting Conjecture

4.3. However, a disadvantage of CD is that it places more convexity constraints in RMP(DF), making it more challenging. Table 5 shows that RT(DF) exceeds RT(F), reflecting the fact that (DF) takes longer than (F) at each RMP iteration.

4.4.4.2. Scheme (AF) vs Scheme (AFD)

(AFD) enhances (AF) by adding EDCs to RMP(AFD) to accelerate convergence. The solvabilities of RMP and sub-problem in both schemes are about the same. Table 5 shows the effectiveness of EDCs, which can reduce RITR required to solve RMP(AFD). EDCs restrict the values of dual variables in DRMP, tending to accelerate the convergence of RMP. Results support Conjecture 4.4.

4.4.4.3. Scheme (A) vs Scheme (VA)

Table 5 shows that (A) requires fewer RITR and less RT than (VA). Because SP1E in (VA) can be polynomially reduced to SP1 in (A), the solvability of sub-problems in both schemes is about the same. Compared with SP1, SP1E can be expected to have more extreme points, so MP(VA) involves more columns than MP(A). Affirming Conjecture 4.3, RMP(A) converges faster than RMP(VA). The rate of convergence also depends upon restrictions in the dual space. (A) incorporates constraints (4-4) in master problem while (VA) includes them in its sub-problem. According to Conjecture 4.4, restrictions on dual variables in DRMP(A) are apparently weaker than those on DRMP(VA) because DRMP(VA) does not include dual cuts associated with variables z_a (there are more z_a variables than constraints (4-4)); DRMP(VA) does not involve dual variables associated with (4-4) but DRMP(A) does. However, neither (A) nor (VA) promotes solvability because the number of constraints and variables in RMP(A) and RMP(VA) are almost the

same as those in $\mathscr{LP}$. RMP must be solved iteratively, so (A) and (VA) require more RT than CPLEX does to solve $\mathscr{LP}$.

4.4.4.4. Scheme (F) vs Scheme (VF)

SP2E in (VF) can be reformulated as SP2 in (F), so the solvability of sub-problems in both schemes is about the same. SP2 and its corresponding SP2E have the same number of extreme points. Conjecture 4.3 can not be used to estimate the rate of convergence. (F) includes constraints (4-4) in master problem while (VF) includes them in its sub-problem. DRMP(VF) has fewer dual variables that DRMP(F) but RMP(VF) does not involve y_{pf} variables so that associated dual cuts are not incorporated in DRMP(VF). Constraints (4-4) are implied as equalities in (VF), restricting RMP(VF) more than RMP(F), which includes (4-4) as inequalities. Restrictions on the values of variables in DRMP(F) are weaker than those on DRMP(VF). Affirming Conjecture 4.4, (F) requires fewer RITR than (VF). Compared with RMP(A) and RMP(VA), RMP(F) and RMP(VF) incorporate fewer constraints and variables. Both (A) and (VA) promote solvability. Table 4 and Table 5 show that both (A) and (VA) require less RT than CPLEX.

4.4.4.5. Scheme (VF) vs Scheme (AF)

(VF) includes constraints (4-4) in sub-problem and constraints (4-6) in master problem, while (AF) includes (4-6) in sub-problem and (4-4) in master problem. SP(AF) has a block diagonal structure that can be decomposed into independent SP1 and SP2. MP(AF) includes additional columns from SP1 compared with MP(VF). By Conjecture 4.3, we expect that RMP(VF) converges faster than RMP(AF) because it involves fewer columns. In contrast to RMP(VF), RMP(AF) includes constraints (4-4), which induce a

high order of degeneracy but play an important role in coordinating columns from SP1 and SP2. Because (4-4) induce RMP degeneracy, dual variables may assume different values in alternative optimal dual solutions, explaining why RMP(AF) converges more slowly than RMP(VF). Results in Table 5 affirm Conjecture 4.4.

4.4.4.6. Scheme (F) vs Scheme (AFD)

Table 5 shows that RMP(F) converges faster than RMP(AFD). Because (AFD) employs both SP1 and SP2 but (F) employs only SP2, MP(F) involves fewer columns than MP(AFD), apparently promoting the convergence of RMP(F) according to Conjecture 4.3. In particular, as the instance size increases, the number of columns from SP1 in (AFD) increases, allowing (F) to converge much faster than (AFD) on large-scale instances. (F) and (AFD) deal differently with variables y_{pf} and associated constraints (4-6). (F) relegates them to master problem while (AFD) places them in sub-problem. Variables y_{pf} in (F) can be viewed as dual cuts in DRMP(F). Compared with dual cuts in DRMP(AFD), dual cuts related to y_{pf} variables in (F) seem to be more effective based on the results shown in Table 5.

However, RMP(AFD) provides a better bound than RMP(F) because SP1 does not have the Integrality Property. If we use (AFD) in B&P, the tighter bound can be expected to reduce the number of nodes explored in the B&B search tree so that (AFD) would generate fewer columns than (F). If budget limitations (4-6) were tight, we expect that (AFD) would be a good choice in B&P.

4.5. Summary

In this subsection, we study several schemes for applying DWD to PADSDP, and its objectives, enhancing the rate of DWD convergence by formulating a rationale for decomposition by analyzing potential schemes, adopting acceleration techniques, and assessing the impacts of schemes and techniques computationally. It also attains its goal, systematically studying the broad range of factors (e.g., numbers of columns and rows in master problem and its degeneracy, the structure (Integrality Property, separability into small sub-problems) of sub-problem and its solvability) that affect the performance of DWD.

We focus on the methodology, exploring advantages and disadvantages of alternative decomposition schemes, which distribute the challenges of complexity between master problem and sub-problem. We seek to understand what factors affect the performance of DWD, especially its rate of convergence. We summarize a set of measures that describe decomposition schemes and pose conjectures that lend insights into the rate of convergence. Computational tests provide empirical evidence that supports our intuitive conjectures. We also adopt two techniques for accelerating convergence and demonstrate their positive influence.

5. A GENERALIZATION OF DANTZIG-WOLFE DECOMPOSITION*

In this section, we extend the idea of adding extra dual cuts and propose a generalization of Dantzig-Wolfe decomposition (DWD), reformulating the master problem with fewer variables at the expense of adding more constraints; the sub-problem structure does not change. It shows both analytically and computationally that the reformulation promotes faster convergence to an optimal solution in application to a linear program and to the relaxation of an integer program at each node in the branch-and-bound (B&B) tree. Further, it shows that this reformulation subsumes and generalizes three approaches that have been shown to improve the rate of convergence in special cases.

5.1. Introduction

DWD was proposed initially as a technique for solving a large-scale linear program (Dantzig and Wolfe, 1960) by reformulating it in terms of a master problem and one or more sub-problems. The master problem typically comprises a huge number of columns; but, to avoid dealing with all of them explicitly, a restricted master problem (RMP) incorporates a limited number of columns that are generated as needed by solving the associated sub-problem(s) in a well-known column-generation (CG) procedure (Wilhelm, 2001). DWD can also be applied to the linear relaxation of an integer program, embedding the reformulation within branch-and-bound to form a branch-and-price (B&P) approach. Another type of CG (Type II as defined by Wilhelm, 2001) poses a master probl-

---

em – often a set covering model (e.g., Desrochers, Desrosiers and Solomon, 1992) – and formulates one or more sub-problems to generate columns as needed. DWD represents each sub-problem polytope by forming a convex combination of the columns (i.e., extreme-point solutions) it generates, while Type II CG enters improving columns into the RMP basis directly, without forming such convex combinations.

The effectiveness of CG depends upon the number of iterations required to prescribe an optimal solution. All sub-problems are typically solved at each iteration so that, depending upon column-pool management tactics, either the most improving column or all improving columns can be made available to RMP. The average-case performance of CG is quite acceptable in many applications (e.g., multi-commodity network flow and cutting stock problems); although, in the worst case, it may be necessary to generate an exponential number of columns to optimize RMP. A key challenge is to accelerate CG convergence by reducing the number of iterations (equivalently, the number of generated columns) needed to prescribe an optimal solution (Vanderbeck and Savelsbergh, 2006).

The purpose of this section is to present a generalization of DWD (DWDG) with the goal of accelerating convergence. Our primary set of research objectives investigates DWDG in application to linear programs, establishing a theoretical basis, providing geometric insight through a numerical example, and showing how DWDG improves convergence through analytical means, subsuming and generalizing approaches that have been shown to improve convergence in special cases, and reporting computational experience. Our second set of research objectives explores equivalent issues related to DWDG in application to integer programs, for which run time depends on the rate of

convergence as well as the tightness of bounds provided by RMP at each node in the B&B tree.

We have organized the body of this section in three subsections. Subsection 5.2 addresses the primary set of research objectives that deal with applying DWDG to linear programs; similarly, subsection 5.3 addresses the second set of research objectives related to integer programs. Finally, subsection 5.4 summarizes conclusions. This section contributes by generalizing classical DWD, formalizing DWDG, showing that it accelerates convergence, and describing how it subsumes and generalizes approaches that are known to improve convergence in special cases.

5.2. DWDG for Linear Programs

DWD can be viewed as a form of variable redefinition (Martin, 1987), since it represents the sub-problem polytope as a convex combination of its extreme points (Dantzig, 1960). That is, DWD employs an *extreme-point representation* of the sub-problem polytope, while the original problem uses the intersection of a set of half-spaces, the *half-space representation*. Each point in the sub-problem polytope can be represented as a convex combination of a set of vectors (i.e., $\bar{x} = \sum_{k \in K} x_k \bar{\lambda}_k$; $\sum_{k \in k} \bar{\lambda}_k = 1$; $\bar{\lambda}_k \geq 0$, $k \in K$, where x_k denotes an extreme point and K is the index set of extreme points of that polytope). Non-negative values of multipliers $\{\bar{\lambda}_k\}$, $k \in K$ must satisfy the convexity constraint. In addition, each point in the sub-problem polytope can also be represented as a linear combination of a set of vectors (e.g., $\bar{x} = \sum_{j \in J} e_j \bar{x}_j$, where $n = |J|$ is the dimension of vector $\bar{x}$, e_j is the jth unit vector, and $\bar{x}_j$ is the jth element of $\bar{x}$. $\{\bar{x}_j\}$, $j \in J$ must

satisfy all constraints that define facets of the polytope. Compared with the convex combination, this linear combination typically involves fewer variables (i.e., $|K| > n$ where $|K|$ denotes the cardinality of K) but more constraints (i.e., the convex combination entails fewer convexity constraints than the number of constraints required by the linear combination).

An open question is whether or not it is possible to exploit a trade-off between the numbers of variables and constraints in a master problem to improve the rate of convergence of CG by representing the sub-problem polytope as a mixture of both extreme-point and half-space representations (i.e., both convex and linear combinations). We show that such a favorable trade-off exists and can be achieved by DWDG.

This subsection addresses our primary set of research objectives, investigating DWDG in application to linear programs. We begin with a brief review of DWD, then present our reformulation, DWDG. Next, we provide a rich geometrical interpretation of DWDG through a small numerical example. We establish a theoretical foundation for DWDG and analyze how it affects convergence. We show DWDG subsumes and generalizes an approach that has been shown to improve convergence of DWD in application to the linear multi-commodity network flow problem. Finally, we report tests that evaluate convergence numerically. Our investigation shows DWDG can reduce the number of columns that must be generated to prescribe an optimal solution, improving the convergence behavior of CG.

5.2.1. Dantzig-Wolfe decomposition

Consider a linear program (P) of the form

$$(P) \quad max \quad \boldsymbol{c}^T \boldsymbol{x}$$
$$s.t. \quad \boldsymbol{Ax} \leq \boldsymbol{b} \qquad (5\text{-}1)$$
$$\boldsymbol{Dx} = \boldsymbol{d} \qquad (5\text{-}2)$$
$$\boldsymbol{x} \geq \boldsymbol{0} \qquad (5\text{-}3).$$

Here, $\boldsymbol{x} \in \mathbb{R}^n_+$ represents an n-vector of non-negative decision variables and $\boldsymbol{c} \in \mathbb{R}^n$ is the n-vector of objective function coefficients. $\boldsymbol{A} \in \mathbb{R}^{p \times n}$ and $\boldsymbol{D} \in \mathbb{R}^{q \times n}$ are matrices of constraint coefficients. $\boldsymbol{b} \in \mathbb{R}^p$ and $\boldsymbol{d} \in \mathbb{R}^q$ are right-hand-side values.

To facilitate presentation, we deal with a single, bounded sub-problem, noting, however, that, in a typical application, $\boldsymbol{D}$ may have a block diagonal structure so that the sub-problem is separable into a set of independent sub-problems. Multiple and un-bounded sub-problems can easily be addressed but require more intricate notation. Applying DWD, we can decompose (P), placing constraints (5-2) and (5-3) in the sub-problem and relegating constraints (5-1) to the master problem, to obtain the following DWD master problem (MP):

$$(MP) \quad max \quad \sum_{k \in K} \left(\boldsymbol{c}^T \boldsymbol{x}_k \right) \lambda_k$$
$$s.t. \quad \sum_{k \in K} \left(\boldsymbol{Ax}_k \right) \lambda_k \leq \boldsymbol{b} \qquad (5\text{-}4)$$
$$\sum_{k \in K} \lambda_k = 1 \qquad (5\text{-}5)$$
$$\lambda_k \geq 0, k \in K .$$

As before, K represents the index set of extreme points in the sub-problem polytope (i.e., columns in (MP)). Letting $\boldsymbol{\omega} \in \mathbb{R}^p_+$ and $\alpha \in \mathbb{R}$ correspond to dual variable values associated with constraints (5-4) and (5-5), respectively, sub-problem (SP) can be formulated as

$$(SP) \quad min \ (\boldsymbol{\omega}^T \boldsymbol{A} - \boldsymbol{c}^T) \boldsymbol{x} + \alpha$$
$$s.t. \quad (5\text{-}2) \ and \ (5\text{-}3).$$

5.2.2. Reformulation

Consider a second linear program (P') vis-à-vis (P):

$$
\begin{array}{llllll}
(P') & max & \boldsymbol{c}^T \boldsymbol{x}' & + & \boldsymbol{c}^T \boldsymbol{y} & \\
& s.t. & \boldsymbol{Ax}' & + & \boldsymbol{Ay} & \leq \boldsymbol{b} & (5\text{-}6) \\
& & \boldsymbol{x}' & + & \boldsymbol{y} & \geq \boldsymbol{0} & (5\text{-}7) \\
& & \boldsymbol{Dx}' & & & = \boldsymbol{d} & (5\text{-}8) \\
& & & & \boldsymbol{Dy} & = \boldsymbol{0} & (5\text{-}9) \\
& & \boldsymbol{x}' & & & \geq \boldsymbol{0} & (5\text{-}10) \\
& & & & \boldsymbol{y} & \text{unrestricted.} & (5\text{-}11)
\end{array}
$$

$\boldsymbol{x}' \in \mathbb{R}^n_+$, which corresponds to $\boldsymbol{x}$ in (P), is an n-vector of non-negative decision variables and $\boldsymbol{y} \in \mathbb{R}^n$ is an n-vector of decision variables that are unrestricted in sign. All other symbols are the same as in (P).

Definition 5.1: Problems (π_1) and (π_2) are *equivalent* if, for any feasible solution to (π_1), there is a corresponding feasible solution to (π_2) with the same objective function value and vice versa.

Proposition 5.2: Problems (P) and (P') are equivalent.

Proof: $(\Rightarrow)$ Suppose $\boldsymbol{x} = \bar{\boldsymbol{x}}$ is a feasible solution to (P); the corresponding feasible solution to (P'), $(\boldsymbol{x}', \boldsymbol{y}) = (\bar{\boldsymbol{x}}, \boldsymbol{0})$, has the same objective function value, $\boldsymbol{c}^T \bar{\boldsymbol{x}}$. Because $\bar{\boldsymbol{x}}$ satisfies constraints (5-1)-(5-3), it is trivial that $(\bar{\boldsymbol{x}}, \boldsymbol{0})$ satisfies constraints (5-6) -(5-11). Thus, $(\bar{\boldsymbol{x}}, \boldsymbol{0})$ is feasible with respect to (P').

$(\Leftarrow)$ Consider a feasible solution to (P'), $(\boldsymbol{x}', \boldsymbol{y}) = (\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$; the corresponding solution to

(P), $x = (\bar{x} + \bar{y})$, has the same objective function value, $c^T (\bar{x} + \bar{y})$. $(\bar{x}, \bar{y})$ satisfies constraints (5-8) and (5-9) (i.e., $D\bar{x} = d$, $D\bar{y} = 0$), so $(\bar{x} + \bar{y})$ satisfies constraints (5-2) (i.e., $D(\bar{x} + \bar{y}) = D\bar{x} + D\bar{y} = d + 0 = d$). $(\bar{x}, \bar{y})$ satisfies constraints (5-6) and (5-7) (i.e., $A\bar{x} + A\bar{y} \le b$, $\bar{x} + \bar{y} \ge 0$), so $\bar{x} + \bar{y}$ satisfies constraints (5-1) and (5-3) (i.e., $A(\bar{x} + \bar{y}) \le b$, $(\bar{x} + \bar{y}) \ge 0$). Thus, $(\bar{x} + \bar{y})$ is feasible with respect to (P). ∎

Please note the equivalence of solutions $(\bar{x}, \bar{y})$ and $(\bar{x} + \bar{y}, 0)$ in (P').

Corollary 5.3: For feasible solution $(x', y) = (\bar{x}, \bar{y})$ in (P'), $(x', y) = (\bar{x} + \bar{y}, 0)$ is also feasible with respect to (P'). Both $(\bar{x}, \bar{y})$ and $(\bar{x} + \bar{y}, 0)$ have the same objective function value.

5.2.2.1. DWD applied to (P')

We introduce DWDG by applying DWD to problem (P'), forming (MPI'_L), one sub-problem with constraints (5-8) and (5-10), and a second with (5-9) and (5-11):

$$(MPI'_L) \quad max \quad \sum_{k \in K}\left(c^T x_k\right) + \sum_{l \in L}\left(c^T y_l\right)\mu_l \qquad (5\text{-}12)$$

$$s.t. \quad \sum_{k \in K}\left(Ax_k\right)\lambda_k + \sum_{l \in L}\left(Ay_l\right)\mu_l \le b \qquad (5\text{-}13)$$

$$\sum_{k \in K}\left(x_k\right)\lambda_k + \sum_{l \in L}\left(y_l\right)\mu_l \ge 0 \qquad (5\text{-}14)$$

$$\sum_{k \in K}\lambda_k = 1 \qquad (5\text{-}15)$$

$$\lambda_k \ge 0, \; k \in K \qquad (5\text{-}16)$$

$$\mu_l \quad \text{unrestricted}, \; l \in L \qquad (5\text{-}17).$$

Here, K is the same as in (MP). Letting $\omega \in \mathbb{R}^p_+$, $-u \in \mathbb{R}^n_-$ (i.e., $u \in \mathbb{R}^n_+$) and $\alpha \in \mathbb{R}$ correspond to dual variable values associated with constraints (5-13), (5-14), and (5-15), respectively, we define sub-problem (SPI'):

$$(SPI') \quad \min \quad (\boldsymbol{\omega}^T A - \boldsymbol{u}^T - \boldsymbol{c}^T)\boldsymbol{x} + \alpha$$
$$s.t. \qquad \text{(5-8) and (5-10).}$$

where constraints (5-8) and (5-10) are exactly the same as (5-2) and (5-3), respectively, so that sub-problems (SPI') and (SP) have the same set of constraints. A feasible solution to (SPI') generates a vector (i.e., a column) that we denote by $\boldsymbol{x}_k$ for $k \in K$. Sub-problem (SP') is defined as:

$$(SP') \quad \min \quad (\boldsymbol{\omega}^T A - \boldsymbol{u}^T - \boldsymbol{c}^T)\boldsymbol{y}$$
$$s.t. \qquad \text{(5-9) and (5-11).}$$

Constraints (5-9) and (5-11) define a linear subspace of $\mathbb{R}^n$. One important property of a linear subspace is that any vector in it can be represented as a *linear combination* of vectors that *span* it. In particular, if these spanning vectors are linearly independent, they comprise a *basis* of the linear subspace. The size of the basis associated with a linear subspace (i.e, the number of linearly independent vectors spanning the subspace) is equal to the dimension of the subspace (Bazaraa et al., 2005). A feasible solution to (SP') generates a vector (i.e., a column) that we denote by $\boldsymbol{y}_l$ for $l \in L$ (equivalently, $\{\boldsymbol{y}_l \mid L\}$). Letting $\mathfrak{C}$ be the collection of index sets, each of which, L, corresponds to a set of vectors that spans the linear subspace (i.e., $\forall\, L \in \mathfrak{C}$). Note that each selection of L leads to a different (MPI'_L).

Another property of a linear subspace is that if $\boldsymbol{y}_l$ is in the subspace, then $\varepsilon \boldsymbol{y}_l$, for $\forall \varepsilon \in \mathbb{R}$, is also in the subspace (Bazaraa et al., 2005). If $(\boldsymbol{\omega}^T A - \boldsymbol{u}^T - \boldsymbol{c}^T)\boldsymbol{y}_l \neq 0$, the objective function value (SP') can be made unbounded by setting ε to $+\infty$ or $-\infty$. To

avoid this situation, we can restrict the norm of $\boldsymbol{y}$. (SP') is easy to solve by projecting

$-(\boldsymbol{\omega}^T A - \boldsymbol{u}^T - \boldsymbol{c}^T)$ orthogonally onto the subspace defined by (5-9) and (5-11) and nor-

malizing the projected vector.

In addition to μ_l variables for $l \in L$, (MPI'_L) includes n (i.e., the dimension of

vector $\boldsymbol{x}$) additional constraints (5-14) in comparison with (MP). Each selection of L

leads to a different (MPI'_L), and some selections may not require all constraints (5-14)

to be included in (MPI'_L). For example, if we select L such that the rth constraint (5-

14), $\sum_{k \in k} (x_{rk}) \lambda_k + \sum_{l \in L} (y_{rl}) \mu_l \geq 0$, has $y_{rl} = 0$ for each $l \in L$, this constraint is im-

plied by $x_{rk} \geq 0$, $\lambda_k \geq 0$, $k \in K$ and can be eliminated because it is redundant. In par-

ticular, if we select $K = \varnothing$, all constraints (5-14) are redundant and can be eliminated,

and (MPI'_L) is exactly the same as (MP). In the worst case, the number of constraints

(5-14) might be large, but we can control this number by our selection of K. To facili-

tate subsequent discussions, we use $\{\boldsymbol{y}_l \mid L\}$ and $\{\boldsymbol{x}_k \mid K\}$ to abbreviate $\{\boldsymbol{y}_l\}$, $l \in L$ and

$\{\boldsymbol{x}_k\}$, $k \in K$, respectively, so long as ambiguity would not result.

5.2.2.2. Selection of K

Due to the equivalence of (P') and (MPI'_L), $\bar{\boldsymbol{y}} \equiv \sum_{l \in L} (\boldsymbol{y}_l) \bar{\mu}_l$. Further, Corollary 5.3

shows that the optimal solution to (P') may have $\boldsymbol{y} = \boldsymbol{0}$. These relationships raise the

question of whether the optimal solution of (MPI'_L) can be prescribed even if vectors

$\{\boldsymbol{y}_j \mid L\}$ do not span the linear subspace defined by (5-9) and (5-11). The answer to this

question is yes, as we relate below.

If $\mu_l = 0$, the index l corresponding to vector $\boldsymbol{y}_l$ can be removed from L without changing $Z^*(MPI'_L)$, where $Z^*(-)$ represents the optimal objective function value of problem $(-)$. However, columns $L \backslash \{l\}$ may not form a basis. Thus, index set L need not correspond to vectors that span the linear subspace defined by (5-9) and (5-11). Let $\mathfrak{D}$ be the collection of all index sets, including the empty set, and $\mathfrak{C} \subset \mathfrak{D}$.

Proposition 5.4: Problem (MPI'_L) is equivalent to (MP) for each $L \in \mathfrak{D}$.

Proof: ($\Rightarrow$) Let $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}_L$ represent vectors of variables (λ_k), $k \in K$ and (μ_l), $l \in L$; $\overline{\boldsymbol{\lambda}}$, $\overline{\boldsymbol{\mu}}_L$ represent the vector of values $(\overline{\lambda}_k)$, $k \in K$ and $(\overline{\mu}_l)$, $l \in L$, respectively; and $\boldsymbol{0}_L$ represent the $\boldsymbol{0}$ vector of dimension $|L|$.

Suppose $(\boldsymbol{\lambda}, \boldsymbol{\mu}_L) = (\overline{\boldsymbol{\lambda}}, \overline{\boldsymbol{\mu}}_L)$ is a feasible solution to (MPI'_L), $(\boldsymbol{x}', \boldsymbol{y}) = (\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$ (i.e., $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) \equiv \left(\sum_{k \in K} (\boldsymbol{x}_k) \overline{\lambda}_k, \sum_{l \in L} (\boldsymbol{y}_l) \overline{\mu}_l \right)$ is the corresponding feasible solution to (P')). $(\boldsymbol{x}', \boldsymbol{y}) = (\overline{\boldsymbol{x}} + \overline{\boldsymbol{y}}, \boldsymbol{0})$ is also feasible with respect to (P') and has the same objective value by Corollary 5.3. Proposition 5.2 shows that $(\overline{\boldsymbol{x}} + \overline{\boldsymbol{y}})$ is the corresponding solution in (P). We can find a feasible solution to (MP) with the same objective function value that $(\overline{\boldsymbol{x}} + \overline{\boldsymbol{y}})$ gives in (P), which also corresponds to the solution value that $(\overline{\boldsymbol{\lambda}}, \overline{\boldsymbol{\mu}}_L)$ gives in (MPI'_L).

($\Leftarrow$) Trivial. Given any feasible solution to (MP), we only need to add $\boldsymbol{\mu}_L = \boldsymbol{0}_L$ and obtain the corresponding solution to (MPI'_L) with the same objective value. ∎

Actually, the $\{\boldsymbol{y}_l\}$ columns are redundant in (MPI'_L) since the optimal solution

value of (MPI'_L), $Z^*(MPI'_L)$, is the same whether these columns are included or not.

Thus, additional constraints can be imposed on μ_l in (MPI'_L) without changing

$Z^*(MPI'_L)$ and we now propose a sufficient condition to do so. The following proposi-

tion deals with (MPI''_L), which we obtain by including constraints (18) in (MPI'_L):

$$(MPI''_L) \quad max \sum_{k \in K}\left(\boldsymbol{c}^T \boldsymbol{x}_k\right)\lambda_k \quad + \quad \sum_{l \in L}\left(\boldsymbol{c}^T \boldsymbol{y}_l\right)\mu_l$$

$$s.t. \quad \text{(5-13)-(5-17)}$$

$$\sum_{l \in L}\left(\boldsymbol{g}_l \mu_l\right) \leq \boldsymbol{h} \qquad \text{(5-18)}.$$

Proposition 5.5: If $\boldsymbol{\mu}_L = \boldsymbol{0}_L$ satisfy constraints (5-18) (i.e., $\boldsymbol{h} \geq \boldsymbol{0}$), then (MP), (MPI'_L),

and (MPI''_L) are equivalent.

Proof: Given any feasible solution to (MP), we need only add $\boldsymbol{\mu}_L = \boldsymbol{0}_L$ to obtain the cor-

responding solution to (MPI''_L). Given any feasible solution to (MPI''_L), it is also fea-

sible solution with respect to (MPI'_L) since it is a relaxation of (MPI''_L). Given any

feasible solution to (MPI'_L), Proposition 5.4 shows that we can always find a corre-

sponding solution to (MP). So, (MP), (MPI'_L), and (MPI''_L) are equivalent. ∎

In particular, we can specify constraints (5-18) as $\mu_l \leq 0, -\mu_l \leq 0$ (i.e., $\mu_l = 0$) in

(MPI''_L) ; that is, setting any μ_l variable in (MPI'_L) equal to 0 will not change

$Z^*(MPI'_L)$.

Remark 5.6: If $K = \varnothing$, constraints (5-14) are redundant and (MPI'_L) is exactly the same as (MP).

5.2.3. Geometrical interpretation

We now demonstrate that DWDG provides the flexibility of representing the sub-problem polytope by combining both extreme-point and half-space representations. In general, the extreme point representation (e.g. (MP)) uses $|K|$ variables and one con-vexity constraint, while the half-space representation (e.g. (P)) uses n variables and q constraints (5-2) in (P). We illustrate the geometrical interpretation of combining ex-treme-point and half-space representations by using the *project-and-restore* procedure, which includes i) projecting polytope Π along direction y_1 onto hyper-plane Γ to ob-tain polytope $\Pi_\Gamma(y_1)$; ii) formulating the extreme-point presentation for polytope $\Pi_\Gamma(y_1)$; and iii) restoring polytope Π as the linear combination of points in $\Pi_\Gamma(y_1)$ and direction y_1. Figure 10 depicts an example of this procedure. We construct the linear program (LPE):

$$
\begin{array}{lllllll}
(LPE) & max & 14x_1 & +7x_2 & +5x_3 & & \text{(i)} \\
& s.t. & x_1 & +2x_2 & +6x_3 & \leq 54 & \text{(ii)} \\
& & 8x_1 & & +x_3 & \leq 32 & \text{(iii)} \\
F_1 & & 3x_1 & -8x_2 & +9x_3 & \leq 1 & \text{(iv)} \\
F_2 & & 9x_1 & +2x_2 & +x_3 & \leq 55 & \text{(v)} \\
F_3 & & -4x_1 & +2x_2 & +x_3 & \leq 16 & \text{(vi)} \\
F_4 & & -x_1 & +x_2 & -3x_3 & \leq -8 & \text{(vii)} \\
& & x_1 & -3x_2 & +3x_3 & \leq -2 & \text{(viii)} \\
& & -2x_1 & +x_2 & -x_3 & \leq -1 & \text{(ix)} \\
& & 3x_1 & +x_2 & -x_3 & \leq 14 & \text{(x)} \\
& & x_1, & x_2, & x_3 & \geq 0 & \text{(xi)}
\end{array}
$$

Suppose that we apply DWD to (*LPE*) using seven constraints (iv)-(xi) to define

the sub-problem polytope (see Figure 10(a)) and relegating constraints (ii) and (iii) to the

master problem. The seven extreme points of the sub-problem polytope (i.e., x_1, x_2, …,

x_7) are $(3,10,8)$, $(3,11,6)$, $(1,7,6)$, $(4,7,5)$, $(3,10,5)$, $(4,5,3)$, and $(1,5,4)$, respec-

tively. The projection direction is $y_1 = (0,1,1)^T$ and the hyper-plane Γ is $-x_1 + x_2 - 3x_3$

$= -8$, corresponding to constraint (vii). Figure 10(a) shows the projection of Π along

direction y_1 onto hyper-plane Γ to obtain $\Pi_\Gamma(y_1)$. $\Pi_\Gamma(y_1)$ has three extreme points

$(3,10,5)$, $(4,5,3)$, and $(1,5,4)$ (i.e, x_5, x_6, and x_7). Each extreme point of $\Pi_\Gamma(y_1)$ cor-

responds to at least one extreme point of Π (e.g., x_2, x_5). Some extreme points of Π

(e.g., x_1) may be projected to inner points of $\Pi_\Gamma(y_1)$. The following property is associ-

ated with the projection.

Property 5.7: The number of extreme points of $\Pi_\Gamma(y_1)$ is no more than the number of

extreme points of Π. An extreme point in $\Pi_\Gamma(y_1)$ can be represented as a linear combi-

nation of y_1 and a selected extreme point of Π (Damiano and Little, 1988).

Figure 10(b) describes the restoration. A linear combination of points in $\Pi_\Gamma(y_1)$

and direction y_1 generates an affine subspace containing polytope Π. To complete the

restoration of Π, the inequalities that represent facets F_1, F_2, F_3, and F_4 must be in-

corporated to restrict the affine subspace appropriately. The number of such inequalities

that must be incorporated is less than the total number of facets of Π in this example. In

fact, this property holds in general.

(a) project   (b) restore
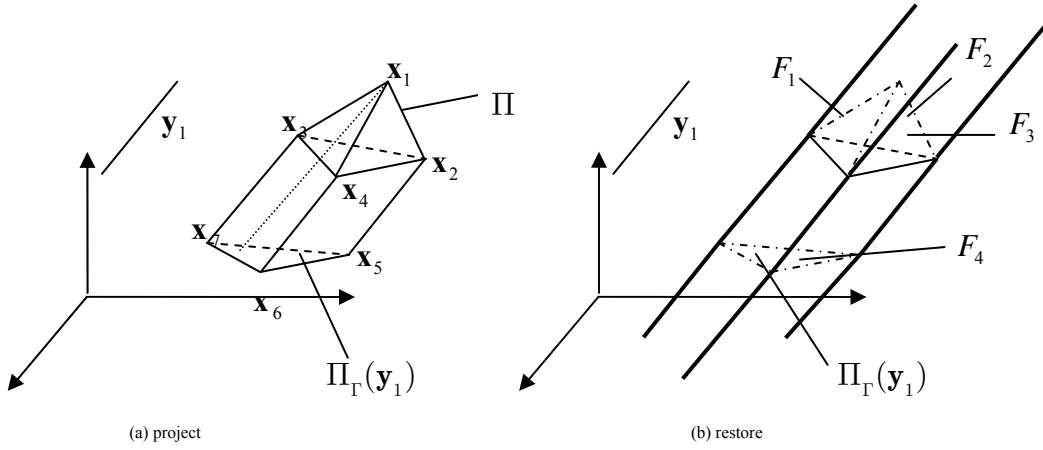
Figure 10. An example sub-problem polytope

Property 5.8: The number of facets invoked to restore Π need be no more than the number of facets of Π (Damiano and Little, 1988).

By Property 5.7, any extreme point of $\Pi_\Gamma(\boldsymbol{y}_1)$, $\boldsymbol{x}_k^\Gamma$, can be represented as a linear combination of $\boldsymbol{y}_1$ and one of the extreme points of Π, $\boldsymbol{x}_k$, (i.e., $\boldsymbol{x}_k^\Gamma = \boldsymbol{x}_k + \overline{\eta}_\kappa \boldsymbol{y}_1$, where $\overline{\eta}_\kappa$ is a constant, $k \in K_\Gamma$; and $K_\Gamma \subseteq K$, is the index set of extreme points of $\Pi_\Gamma(\boldsymbol{y}_1)$).

Any point $\overline{\boldsymbol{x}}^\Gamma$ in $\Pi_\Gamma(\boldsymbol{y}_1)$ can be represented as a convex combination of extreme points (i.e., $\overline{\boldsymbol{x}}^\Gamma = \sum_{k \in K_\Gamma} \boldsymbol{x}_k^\Gamma \lambda_k^\Gamma$, st. $\sum_{k \in K_\Gamma} \lambda_k^\Gamma = 1$ and $\lambda_k^\Gamma \geq 0$, $k \in K_\Gamma$). By using $\boldsymbol{x}_k^\Gamma$ to replace $\boldsymbol{x}_k + \overline{\eta}_\kappa \boldsymbol{y}_1$ (i.e., $\boldsymbol{x}_k^\Gamma = \boldsymbol{x}_k + \overline{\eta}_\kappa \boldsymbol{y}_1$), $k \in K_\Gamma$, we obtain $\overline{\boldsymbol{x}}^\Gamma = \sum_{k \in K_\Gamma}(\boldsymbol{x}_k + \overline{\eta}_\kappa \boldsymbol{y}_1)\lambda_k^\Gamma$ $= \sum_{k \in K_\Gamma} \boldsymbol{x}_k \lambda_k^\Gamma + \boldsymbol{y}_1 \sum_{k \in K_\Gamma} \overline{\eta}_\kappa \lambda_k^\Gamma$, st. $\sum_{k \in K_\Gamma} \lambda_k^\Gamma = 1$ and $\lambda_k^\Gamma \geq 0$, $k \in K_\Gamma$. Any point in Π, $\overline{\boldsymbol{x}}$, can be represented as a linear combination of $\boldsymbol{y}_1$ and a point, $\overline{\boldsymbol{x}}^\Gamma$, in $\Pi_\Gamma(\boldsymbol{y}_1)$ (i.e.,

$$\bar{x} = \bar{x}^\Gamma + y_1\mu_1 = \sum_{k \in K_\Gamma} x_k^\Gamma \lambda_k^\Gamma + y_1 \left(\mu_1 + \sum_{k \in K_\Gamma} \bar{\eta}_\kappa \lambda_k^\Gamma \right), \text{ st. } \sum_{k \in K_\Gamma} \lambda_k^\Gamma = 1; \lambda_k^\Gamma \geq 0, \ k \in K_\Gamma;$$

and μ_1 unrestricted). Property 5.8 shows that $\bar{x}$ must satisfy a subset of constraints that define facets of Π (i.e., $Fx \leq f$). Any point in Π , $\bar{x}$, can be represented as

$$\bar{x} \quad = \quad \bar{x}^\Gamma + y_1\mu_1 = \quad \sum_{k \in K_\Gamma} x_k^\Gamma \lambda_k^\Gamma + \quad y_1 \left(\mu_1 + \sum_{k \in K_\Gamma} \bar{\eta}_\kappa \lambda_k^\Gamma \right) \quad , \quad \text{st.} \quad \sum_{k \in K_\Gamma} \left(Fx_k \right) \lambda_k^\Gamma$$

$$+ \left(Fy_1 \right) \mu_1 + \sum_{k \in K_\Gamma} \left(Fy_1 \bar{\eta}_k \right) \lambda_k^\Gamma \leq f; \sum_{k \in K_\Gamma} \lambda_k^\Gamma = 1; \lambda_k^\Gamma \geq 0, \ k \in K_\Gamma; \text{ and } \mu_1 \text{ unrestricted.}$$

Because $K_\Gamma \subseteq K$, the reformulation employs fewer variables than does the extreme-point representation. Further, because $Fx \leq f$ represents a subset of constraints of Π , the reformulation employs fewer constraints than the half-space representation.

The project-and-restore procedure can be applied recursively. Relevant projection directions are defined by vectors $\{ y_l \mid L \}$ and relevant projection hyper-planes can each be represented as a convex combination of a subset of extreme points of Π . We project polytope Π along direction y_1 onto a hyper-plane Γ to obtain polytope $\Pi_\Gamma(y_1)$, which can be used to restore Π . We then project $\Pi_\Gamma(y_1)$ along y_2 onto another hyper-plane Γ' to obtain $[\Pi_\Gamma(y_1)]_{\Gamma'}(y_2)$, which can be used to restore $\Pi_\Gamma(y_1)$. We can repeat this procedure. Property 5.7 tells us that $[\Pi_\Gamma(y_1)]_{\Gamma'}(y_2)$ has no more extreme points than $\Pi_\Gamma(y_1)$, which, in turn, has no more extreme points than Π . Property 5.8 shows that we can restore Π ($\Pi_\Gamma(y_1)$) from $\Pi_\Gamma(y_1)$ ($[\Pi_\Gamma(y_1)]_{\Gamma'}(y_2)$) by using no more than the number of facets of Π ($\Pi_\Gamma(y_1)$).Property 5.7 and Property 5.8 provide the flexibility to manage the numbers of extreme points and facets employed (i.e., the number of columns and

rows in a master problem) since each extreme point corresponds to a column; and each facet, to a row.

5.2.4. Improving convergence

Different possible selections of L lead to master problems of different sizes, reflecting a range of cases in the trade-off and affecting the number of columns that must be generated to achieve optimality.

Continuing the example depicted in Figure 10, we apply DWD to (LPE). Using the Big-M method, DWD prescribes the optimal solution on the seventh iteration after generating and entering six improving columns:

Iteration	*RMP objective*	*Sub-problem solution*	*Minimum reduced cost*
1	- 500000	(3,10,8)	< 0
2	-119602.7	(4,5,3)	< 0
3	-17324.23	(1,5,4)	< 0
4	129.6205	(3,10,5)	< 0
5	137.8333	(4,7,5)	< 0
6	138.6207	(3,11,6)	< 0
7	138.6667	-	≥ 0; stop

Now, we design DWDG adding column $y_1 = (0,1,1)^T$ and constraints associated with F_1, F_2, F_3 and F_4 (i.e., (iv), (v), (vi), and (vii), respectively) in addition to (i) and (ii) in the master problem. The sub-problem is unchanged. Using the Big-M method, DWDG prescribes the optimal solution on the third iteration after generating and entering two improving columns.

Iteration	*RMP objective*	*Sub-problem solution*	*Minimum reduced cost*
1	-3499988	(3,11,6)	< 0
2	138.5000	(4,7,5)	< 0
3	138.6667	-	≥ 0; stop

This small example shows that DWDG can reduce the number of iterations, accelerating convergence but at the cost of increasing run time per iteration because more constraints must be incorporated in the master problem.

We now provide a theoretical foundation that explains this behavior, dealing with two cases: $L \in \mathfrak{C}$ and $L \in \mathfrak{D}$. First, for the special case in which $L \in \mathfrak{C}$, the following proposition shows that DWDG can converge rapidly.

Proposition 5.9: Given $L \in \mathfrak{C}$, only one column need be generated from set K to represent the feasible solution set of (MPI'_L) and prescribe an optimal solution to (MPI'_L).

Proof: Let x_1 be any column from K and λ_1 be the associated decision variable in (MPI'_L). For any feasible solution $\bar{x}$ to (P), we show that vector $\bar{x}$ can be represented by vectors x_1 and $\{y_l \mid L\}$; that is, $\bar{x} = x_1 \bar{\lambda}_1 + \sum_{l \in L} (y_l) \bar{\mu}_l$ with $\bar{\lambda}_1 = 1$ to satisfy convexity constraint (5-15).

Because x_1 and $\bar{x}$ satisfy constraints (5-8) (i.e., $Dx_1 = d$ and $D\bar{x} = d$), $D(\bar{x} - x_1) = 0$. Thus, $y = (\bar{x} - x_1)$ is in the linear subspace, $Dy = 0$. Vectors $\{y_l \mid L\}$ span this linear subspace, so we can find $\{\bar{\mu}_l \mid L\}$ such that $(\bar{x} - x_1) = \sum_{l \in L} (y_l) \bar{\mu}_l$.

For any feasible solution $\bar{x}$ to (P), we obtain solution $(\bar{\lambda}, \bar{\mu}) = \{\bar{\lambda}_1 = 1; \bar{\lambda}_k = 0, k \in K \setminus \{1\}; \bar{\mu}_l, l \in L\}$ to (MPI'_L) and $\bar{x} = x_1 + \sum_{l \in L} (y_l) \bar{\mu}_l$. Because $\bar{x}$ satisfies constraints (5-1)-(5-3) in (P), $x_1 + \sum_{l \in L} (y_l) \bar{\mu}_l$ satisfies constraints (5-13)- (5-14) in (MPI'_L). Because $x_1 + \sum_{l \in L} (y_l) \bar{\mu}_l$ satisfies constraints (5-15)-(5-16), $(\bar{\lambda}, \bar{\mu})$ is feasible with respect to (MPI'_L).

By Proposition 5.4, (MP) and (MPI'_L) are equivalent, so (P) and (MPI'_L) are equivalent. If $\bar{x} = x_1 + \sum_{l \in L} (y_l) \bar{\mu}_l$ is an optimal solution to (P) with $c^T \bar{x} = c^T \left(x_1 + \sum_{l \in L} (y_l) \bar{\mu}_l \right)$, $(\bar{\lambda}, \bar{\mu})$ is an optimal solution to (MPI'_L) with the same objective function value. Thus, given $\{ y_l \mid L \}$ that span the linear subspace, $Dy = 0$, we need only one column $\{1\}$ (i.e., x_1) from K to obtain an optimal solution to (MPI'_L). $\blacksquare$

Proposition 5.9 shows that, if L represents a set of vectors that span the linear subspace, we can guarantee that a subset of K comprising a single index can be used to represent the set of feasible solutions to (MPI'_L). The number of μ_l variables (i.e., $|L|$) is at most equal to the dimension of the linear subspace defined by (5-9) and (5-11) (i.e., at most n). Employing CG to solve DWDG, we only need to generate at most $n+1$ columns, one from K and at most n vectors that collectively span the subspace. Thus, DWDG can affect the number of columns that need to be generated to prescribe an optimal solution and may facilitate convergence.

However, including μ_l variables in the master problem entails invoking constraints (5-14) and makes the basis of the master problem larger, rendering RMP more challenging to solve at each iteration. DWDG can improve the convergence of DWD by reducing the number of variables in the master problem, albeit at the cost of increasing run time per iteration because more constraints must be incorporated in the master problem.

Proposition 5.9 deals with the special case in which $L \in \mathfrak{C}$. An open question is whether or not Proposition 5.9 can be generalized for $L \in \mathfrak{D}$; that is, whether or not a

subset $\bar{K}_L$ of K can be used to represent the set of feasible solutions to (MPI'_L) if L does not define a basis of the linear subspace. We propose the following proposition for the more general case in which $L \in \mathfrak{D}$.

Proposition 5.10: For each $L \in \mathfrak{D}$, we need only find a subset $\bar{K}_L$ of K to represent the feasible solution set of (MPI'_L).

Proof: If $L = \varnothing$, $\bar{K}_L$ is K and (MPI'_L) is the same as (MP). Otherwise, suppose that we project polytope Π, which is defined by (5-8) and (5-10), along direction $y_1 \in \{y_j \mid L\}$ onto any hyper-plane Γ, which is not parallel to y_1, to obtain polytope $\Pi_\Gamma(y_1)$. Refer to the paragraph following Property 5.8. By Property 5.7, any point in Π, $\bar{x}$, can be represented as $\bar{x} = \bar{x}^\Gamma + y_1 \mu_1 = \sum_{k \in K_\Gamma} x_k \lambda_k^\Gamma + y_1 \left(\mu_1 + \sum_{k \in K_\Gamma} \bar{\eta}_\kappa \lambda_k^\Gamma \right)$; st. $\sum_{k \in K_\Gamma} \lambda_k^\Gamma = 1$; $\lambda_k^\Gamma \geq 0$, $k \in K_\Gamma$; and μ_1 unrestricted). Here, $K_\Gamma \subseteq K$, is the index set of extreme points of $\Pi_\Gamma(y_1)$.

By Proposition 5.4, given any feasible solution to (MPI'_L), we can find a corresponding feasible solution to (MP) (i.e., a point in Π, which is also feasible with respect to (5-2) and (5-3)) with the same objective function value. Thus, any feasible solution to (MPI'_L) can be represented as $\left(\sum_{k \in K_\Gamma} x_k \lambda_k^\Gamma + y_1 \mu_1' \right)$ st. $\sum_{k \in K_\Gamma} \lambda_k^\Gamma = 1$; $\lambda_k^\Gamma \geq 0$, $k \in K_\Gamma$; and μ_1' unrestricted, here we define $\mu_1' = \mu_1 + \sum_{k \in K_\Gamma} \bar{\eta}_\kappa \lambda_k^\Gamma$ and $\bar{K}_L$ can be viewed as K_Γ. This projection procedure can be repeated for each vector in $\{y_l \mid L\}$. ∎

The importance of Proposition 5.10 is that, for any L, we can use a subset $\overline{K}_L$ of K in DWDG. That is, the number of columns in (MPI'_L) may be smaller than the number in (MP). This, we expect would require fewer columns to be generated to prescribe an optimal solution to (MPI'_L).

5.2.5. Case study 1: the linear multi-commodity network flow problem

This section discusses application of DWDG to the multi-commodity network flow problem, describing how it subsumes and generalizes an approach that is known to improve convergence in this special case.

An *arc*-based model in which variables represent flows on *arcs* is a common formulation of the multi-commodity network flow problem. An alternative, *path*-based formulation, in which variables represent flows on *paths*, is also widely used and corresponds to the DWD of the *arc*-based model. Letting (P) (see subsection 0) represent the arc-based model, constraints (5-1) represent capacity limitations on arcs and (5-2) require flow balance at each node based on the network comprising the (index) set of nodes N and arcs ij where $i, j \in N$; $i \neq j$. For each commodity t in the (index) set T, let P^t denote the set of simple paths from the origin of commodity t to its destination; and b^t, the flow requirement for commodity t. If arc ij is on path $p \in P^t$, parameter $a_{ij}^{pt} = 1$; otherwise, $a_{ij}^{pt} = 0$. Letting c^{pt} denote the cost of a unit of flow on path p ; and u_{ij}, the capacity that arc ij provides for the flow of all commodities, the path-based model is

$$min \quad \sum_{t \in T} \sum_{p \in P^t} c^{pt} \lambda^{pt}$$

$$s.t. \quad \sum_{p \in P^t} \lambda^{pt} = b^t \quad t \in T$$

$$\sum_{t \in T} \sum_{p \in P^t} a_{ij}^{pt} \lambda^{pt} \leq u_{ij} \quad i, j \in N; i \neq j$$

$$\lambda^{pt} \geq 0 \quad t \in T, \ p \in P^t.$$

Barnhart et al. (1995) introduced a third formulation, a *cycle*-based model, in which variables represent flows on *cycles*. Alvelos and Valerio de Carvalho (2007) proposed an extended model, which is similar to the *cycle*-based model. This extended model, which is a special case in which DWDG is applied to the arc-based formulation (P), is based on the index set S of circuits. Parameter $\gamma_{ij}^s = 1$ if arc ij is a forward arc of circuit $s \in S$, $= -1$ if ij is a backward arc of s, and 0 if ij does not belong to s. Letting $\hat{c}^s$ denote the cost of one unit of flow on circuit s; and μ^s, the decision variable that prescribes the flow on circuit s, the extended model is

$$min \ \sum_{t \in T} \sum_{p \in P^t} c^{pt} \lambda^{pt} \quad + \quad \sum_{s \in S} \hat{c}^s \mu^s \qquad\qquad (5\text{-}19)$$

$$s.t. \ \sum_{t \in T} \sum_{p \in P^t} a_{ij}^{pt} \lambda^{pt} \quad + \quad \sum_{s \in S} \gamma_{ij}^s \mu^s \quad \leq u_{ij} \ i, j \in N; i \neq j \qquad (5\text{-}20)$$

$$\sum_{t \in T} \sum_{p \in P^t} a_{ij}^{pt} \lambda^{pt} \quad + \quad \sum_{s \in S} \gamma_{ij}^s \mu^s \quad \geq 0 \quad i, j \in N; i \neq j \qquad (5\text{-}21)$$

$$\sum_{p \in P^t} \lambda^{pt} \qquad\qquad\qquad = b^t \ t \in T \qquad\qquad (5\text{-}22)$$

$$\lambda^{pt} \qquad\qquad\qquad \geq 0 \ t \in T, \ p \in P^t \qquad (5\text{-}23)$$

$$\mu^s \quad \geq 0 \ s \in S \qquad\qquad\qquad (5\text{-}24).$$

It is easy to map this extended model to (MPI'_L); constraints (5-19) -(5-24) correspond to (5-12)-(5-17) of (MPI'_L), respectively, and vectors of parameters a_{ij}^{pt} and γ_{ij}^s,

which are generated as sub-problem solutions, correspond to x_k and y_l, respectively.

$\{y_l \mid L\}$ can be viewed as flow on a set of circuits because they satisfy $\boldsymbol{Dy = 0}$ if (5-8)

and (5-10) represent the network flow balance constraints. Constraints (5-21) make sure

that flows on arcs can not be negative.

However, constraints (5-19)-(5-24) and (5-12)-(5-17) are different in two ways.

One is that the right hand side (RHS) of convexity constraint (5-15) is 1 while the RHS

of (5-22) is b^t. We can scale λ^{pt} as $b^t \hat{\lambda}^{pt}$ so that RHS of (5-22) is also equal to 1. The

other is that μ^s is non-negative while μ_l is unrestricted. We can set $\mu_l = \mu_l^+ - \mu_l^-$,

$\mu_l^+, \mu_l^- \geq 0$ and force $\mu_l^- = 0$, which will not change the objective value of (MPI'_L) as

Corollary 5.3 implies.

Alvelos and Valerio de Carvalho (2007) reported computational tests demonstrat-

ing that this extended model prescribes an optimal solution in fewer CG iterations and

(at least 35%) less run time than the path-based model. The reason for this improved

convergence behavior is straightforward. The linear combination of flow on one path,

x_k, with a flow on the circuit, y_l, can represent flow on another path; that is, *redirecting*

a flow from one path to another (Alvelos and Valerio de Carvalho, 2007). And, if we

include $\{y_l \mid L\}$ (i.e., allowing for many flow redirections), it is easy to see that fewer

path flows are needed to achieve optimality. For the planar graph, a simple enumeration

procedure can be used to identify simple circuits $\{y_l \mid L\}$ and the number of such circuits

is polynomial (Alvelos and Valerio de Carvalho, 2007). The authors designed a simple enumeration procedure to find $\{y_l \mid L\}$ and added them to the initial RMP.

The disadvantage of the extended model is that it includes constraints (5-21), which make RMP larger and more challenging to solve. Barnhart et al. (1995) avoided constraints (5-21) by adding extra constraints on μ_l that imply constraint (5-21). Letting (P) (see subsection 5.2.1) represent the arc-based model, constraints (5-1) represent capacity limitations on arcs and (5-2) require flow balance at each node. (MP) (see subsection 5.2.1) represents the path-based model. Barnhart et al. (1995) formulated the cycle-based model (CMP) as follows:

$$
\begin{aligned}
(CMP) \quad max \quad & \sum_{l \in K} c^T(x_l - x_1)\mu_l + \quad c^T x_1 \\
s.t. \quad & \sum_{l \in K} A(x_l - x_1)\mu_l \quad \leq (b - Ax_1) \\
& \sum_{l \in K} \mu_l \quad = 1 \\
& \mu_l \quad \geq 0, \; l \in K .
\end{aligned}
\tag{5-25}
$$

where x_1 corresponds to $\{x_k\}$, $k \in \bar{K}_L = \{1\} \subseteq K$. We define $\{x_l - x_1\}$, $l \in K$ as $\{y_l\}$, $l \in L$, and obtain (MPI''_L), which can be obtained by applying DWDG to the arc-based formulation (P) as we now show:

$$
\begin{aligned}
max \quad & \sum_{k \in \{1\}} \left(c^T x_k\right)\lambda_k \quad + \quad \sum_{l \in K \setminus \{1\}} c^T(x_l - x_1)\mu_l && \tag{5-26} \\
s.t. \quad & \sum_{k \in \{1\}} (Ax_k)\lambda_k \quad + \quad \sum_{l \in K \setminus \{1\}} A(x_l - x_1)\mu_l \quad \leq b && \tag{5-27} \\
& \sum_{k \in \{1\}} (x_k)\lambda_k \quad + \quad \sum_{l \in K \setminus \{1\}} (x_l - x_1)\mu_l \quad \geq 0 && \tag{5-28} \\
& \sum_{k \in \{1\}} \lambda_k \qquad\qquad\qquad\qquad\qquad\qquad = 1 && \tag{5-29} \\
& \lambda_k \qquad\qquad\qquad\qquad\qquad\qquad\qquad \geq 0, \; k \in \{1\} && \tag{5-30}
\end{aligned}
$$

$$\mu_l \quad \text{Unrestricted,} \, l \in K \qquad (5\text{-}31)$$

$$\sum_{l \in K \backslash \{1\}} \mu_l \quad \leq 1 \qquad (5\text{-}32)$$

$$-\mu_l \quad \leq 0 \, , \, l \in K \qquad (5\text{-}33).$$

It is easy to map (*CMP*) to constraints (5-26) -(5-33) to show that the cycle-based model of Barnhart et al. (1995) is a special case of DWDG. Constraints (5-32) and (5-33) correspond to (5-18). Also, (5-32) is exactly the same as (5-25) in (*CMP*) since μ_1 can be viewed as a slack variable. Constraints (5-28) are redundant because $\sum_{k \in \{1\}} (x_k) \lambda_k + \sum_{l \in K \backslash \{1\}} (x_l - x_1) \mu_l = \sum_{l \in K} (x_l) \mu_l \geq 0$. Thus, (*CMP*) is a special case in which DWDG is applied to the arc-based formulation (*P*) of the multi-commodity network flow problem. Barnhart et al. (1995) avoid dealing with constraints (5-28). The authors showed that their method, which is a special application of DWDG obtained by adding restrictions on μ_l, reduced the number of columns generated by an average 29.6% in comparison to DWD. Their test instances were based on a set of message-routing problems in which the underlying network involved 500 nodes, 1300 arcs, and 5850 commodities.

5.2.6. Computational tests on the linear generalized assignment problem

This section describes the computational tests we conduct to study the rate of convergence that DWDG achieves in application to a linear program. For this purpose, we apply DWD to solve the linear relaxation of the generalized assignment problem (GAP) that Savelsbergh (1997) used in his application of B&P to the corresponding integer problems. The master problem includes the (disjoint) assignment constraints $Ax = b$ and

each sub-problem comprises a single (knapsack) constraint, which we solve as a linear program using Dantzig's Rule (Nemhauser and Wolsey, 1999).

The forms of the constraints in GAP differ from the "standard" forms defined in (P) (subsection 0). The GAP constraints relegated to the master problem are "$=$" instead of "$\leq$" as in the corresponding constraint (5-1) of (P). Thus, the associated dual variables $\boldsymbol{\omega}$ in GAP are unrestricted (in (P) the corresponding dual variables associated with (5-1) are nonnegative). GAP sub-problem constraints are "$\leq$" instead of "$=$" as in the corresponding constraint (5-2) of (P), but this can easily be accommodated by changing (P') so that both constraints (5-8) and (5-9) are of the "$\leq$" form, maintaining the equivalence between (P') and (P). Further, the corresponding change to (MPI'_L) is that μ_l in (5-17) must be non-negative to retain the equivalence of (MPI'_L) and (MP): given $\boldsymbol{Nx} \leq \boldsymbol{d}$ and $\boldsymbol{Ny} \leq \boldsymbol{0}$, we can assure $\boldsymbol{N}(\sum_l \boldsymbol{y}_l \mu_l) \leq \boldsymbol{0}$ and $\boldsymbol{N}(\boldsymbol{x} + \sum_l \boldsymbol{y}_l \mu_l) \leq \boldsymbol{d}$ only when $\mu_l \geq 0$ (this cannot be guaranteed if $\mu_l < 0$ were allowed).

We obtain $\boldsymbol{y}$ columns in DWDG in the following way. For each agent i, we generate a set of $\boldsymbol{y}$ columns, each of which represents that reallocation of agent i from job j to job $j+1$. Note that we order jobs according to non-increasing weights associated with agent i in the knapsack constraint. Thus, the reallocation will not violate the knapsack constraint and the number of $\boldsymbol{y}$ columns for each agent is $(n-1)$. We generate a total number of $(n-1) \times m$ $\boldsymbol{y}$ columns in a batch and add them to the initial RMP of DWDG. Also, we need to include constraints corresponding to (5-14) in (MPI'_L) RMP.

Table 8 presents test results. The first column defines the instance using the tuple $(\#,m,n)$, where $\#$ denotes one of the four methods (A, B, C, or D) Savelsbergh (1997) used to generate parameter values; m, the number of agents; and n, the number of jobs. Following Savelsbergh (1997), we use each parameter setting to generate 10 instances and record the average and maximum for two measures of performance: number of iterations, number of degenerate iterations, and run time (in seconds). Six columns give these measures for DWD; and six, for DWDG.

Results reinforce our analysis, clearly showing that DWDG used fewer CG iterations (and, correspondingly, fewer degenerate iterations) to solve each case than DWD. All run times are small; DWDG required less time to solve 7 of the 12 cases and slightly more in the remaining 5 cases. For each of the four $\#$ categories, DWD requires somewhat more time and proportionally slightly more degenerate iterations when m has its largest value. This result can be explained by the fact that the number of extra constraints in RMP of DWDG (i.e., (5-14) in (MPI'_L)) is equal to $(n-1)\times m$. If m is large, the number of rows in RMP of DWDG is fairly large, somewhat increasing the order of degeneracy, and RMP becomes more challenging to solve in comparison with the RMP of DWD.

5.3. DWDG for Integer Programs

This section addresses our second set of research objectives, which relate to the application of DWDG to integer programs. We begin by reviewing use of DWD in B&P, then present our reformulation, DWDG. The third subsection discusses issues relevant to solving integer programs. The following two subsections describe how DWDG sub-

Table 8. Comparison of DWD and DWDG applied to the linear relaxation of GAP

| | DWD | | | | | | DWDG | | | | | |
| | # Iterations | | # Degenerate Iter. | | Run Time (s) | | # Iterations | | # Degenerate Iter. | | Run Time (s) | |
	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
A.3.30	166.2	208	120.4	162	0.1406	0.203	21.9	36	13.4	28	0.0387	0.062
A.5.30	83.2	99	38.5	54	0.0673	0.094	22.6	37	14.2	29	0.0545	0.094
A.10.30	39.0	43	6.9	14	0.0311	0.047	17.6	23	9.1	15	0.0748	0.094
B.3.50	617.6	837	549.2	758	1.7745	2.328	47.5	66	33.7	56	0.1561	0.218
B.5.50	245.4	364	176.7	293	0.6017	0.781	42.1	53	30.5	41	0.2234	0.266
B.20.50	38.3	43	8	14	0.1128	0.125	20.6	27	10.9	18	0.3218	0.406
C.3.30	168.3	244	128.6	204	0.1484	0.203	32.1	43	21.1	33	0.0623	0.078
C.5.30	89.2	126	51.1	90	0.0749	0.094	28.1	37	16.5	25	0.0827	0.109
C.10.30	37.1	43	12	18	0.0372	0.047	21.3	25	11	15	0.1123	0.125
D.3.50	564.2	652	485.7	569	1.7580	2.219	125.6	146	102.6	125	0.4341	0.484
D.5.50	220.3	245	140.4	163	0.6110	0.703	75.1	97	49.5	73	0.4624	0.594
D.20.50	48.5	55	9.8	18	0.1830	0.188	30.5	40	10.2	18	1.1109	1.235

sumes and generalizes approaches that has been shown to improve the rate at which CG converges in application to the cutting stock and production-assembly-distribution system design problems, respectively. Finally, we relate computational tests designed to assess the efficacy of DWDG in application to integer programs.

5.3.1. DWD for integer programs

Consider an integer program (IP) of the form

$$
\begin{aligned}
(IP) \quad max \quad & \boldsymbol{c}^T \boldsymbol{x} \\
s.t. \quad & \boldsymbol{Ax} \leq \boldsymbol{b} \\
& \boldsymbol{Dx} = \boldsymbol{d} && (5\text{-}34) \\
& \boldsymbol{x} \in \mathbb{Z}_+^n && (5\text{-}35).
\end{aligned}
$$

Here, $\boldsymbol{x} \in \mathbb{Z}_+^n$ represents an n-vector of non-negative integer decision variables and $\boldsymbol{c} \in \mathbb{Q}^n$ is an n-vector of rational objective coefficients. $\boldsymbol{A} \in \mathbb{Q}^{p \times n}$ and $\boldsymbol{D} \in \mathbb{Q}^{q \times n}$ are matrices of rational constraint coefficients. $\boldsymbol{b} \in \mathbb{Q}^p$ and $\boldsymbol{d} \in \mathbb{Q}^q$ are rational, right-hand-side coefficients.

As before, we deal with a single (bounded) sub-problem. We decompose (IP), placing constraints (5-34) and (5-35) in the sub-problem, resulting in the master problem (IMP):

$$
\begin{aligned}
(IMP) \quad max \quad & \sum_{k \in K} \left(\boldsymbol{c}^T \boldsymbol{x}_k \right) \lambda_k \\
s.t. \quad & \sum_{k \in K} \left(\boldsymbol{Ax}_k \right) \lambda_k \leq \boldsymbol{b} && (5\text{-}36) \\
& \sum_{k \in K} \lambda_k = 1 && (5\text{-}37) \\
& \lambda_k \geq 0, \quad k \in K && (5\text{-}38) \\
& \sum_{k \in K} \left(\boldsymbol{x}_k \right) \lambda_k \in \mathbb{Z}_+^n && (5\text{-}39).
\end{aligned}
$$

Here, K represents the index set of columns in (IMP), which correspond to extreme points of the sub-problem in the classic *convexification* approach or to feasible integer solutions to the sub-problem in the *discretization* approach (Vanderbeck, 2003). Note that DWD deals with the linear relaxation of (IMP) and does not include constraints (5-39), which are used implicitly in fixing variables in the B&B search tree (i.e., B&P branches on variables x, not λ_k). Letting $\omega \in \mathbb{R}_+^p$ and $\alpha \in \mathbb{R}$ correspond to dual variable values associated with constraints (5-36) and (5-37), respectively, sub-problem (ISP) can be formulated as:

$$(ISP) \quad min \quad (\omega^T A - c^T)x + \alpha$$
$$s.t. \quad \text{(5-34) and (5-35)}.$$

5.3.2. Reformulation

We introduce a second integer program (IP') vis-à-vis (IP):

$$
\begin{array}{llllll}
(IP') & max & c^T x' & + & c^T y & \\
& s.t. & Ax' & + & Ay & \leq b \\
& & x' & + & y & \geq 0 \\
& & Dx' & & = d & \quad (5\text{-}40) \\
& & & Dy & = 0 & \quad (5\text{-}41) \\
& & x' & & \in \mathbb{Z}_+^n & \quad (5\text{-}42) \\
& & & y & \in \mathbb{Z}^n & \quad (5\text{-}43).
\end{array}
$$

Here, $x' \in \mathbb{Z}_+^n$, which corresponds to x in (IP), represents an n-vector of non-negative integer decision variables. $y \in \mathbb{Z}^n$ is an n-vector of integer decision variables that are unrestricted in sign. All other symbols are the same as in (IP).

Proposition 5.11: Problems (IP) and (IP') are equivalent.

Proof: Similar to the proof for Proposition 5.2.∎

We apply DWD to (IP'), placing constraints (5-40) and (5-42) in one sub-problem and (5-41) and (5-43) in another. Suppose that set $\{y_l \mid L\}$, which typically has unbounded cardinality, represents all vectors satisfying constraints (5-41) and (5-43). Let L represent any index set of vectors y_l and $\mathfrak{D}$ be the collection of all index sets. Applying DWDG for integer programs with $L \in \mathfrak{D}$, we obtain

$$(IMPI'_L) \quad max \quad \sum_{k \in K}(c^T x_k)\lambda_k + \sum_{l \in L}(c^T y_l)\mu_l \tag{5-44}$$

$$s.t. \quad \sum_{k \in K}(Ax_k)\lambda_k + \sum_{l \in L}(Ay_l)\mu_l \leq b \tag{5-45}$$

$$\sum_{k \in K}(x_k)\lambda_k + \sum_{l \in L}(y_l)\mu_l \geq 0 \tag{5-46}$$

$$\sum_{k \in K}\lambda_k = 1 \tag{5-47}$$

$$\lambda_k \geq 0, \ k \in K \tag{5-48}$$

$$\mu_l \quad \text{Unrestricted}, \ l \in L \tag{5-49}$$

$$\sum_{k \in K}(x_k)\lambda_k + \sum_{l \in L}(y_l)\mu_l \in \mathbb{Z}^n_+ \tag{5-50}.$$

Here, K is the same as in (IMP). DWDG deals with the linear relaxation (5-44)-(5-49), invoking (5-50) implicitly in the branching process. Letting $\omega \in \mathbb{R}^p_+$, $u \in \mathbb{R}^n_+$ and $\alpha \in \mathbb{R}$ correspond to dual variables associated with constraints (5-45), (5-46), and (5-47), respectively, we define sub-problem $(ISPI')$ as:

$$(ISPI') \quad min \quad (\omega^T A - u^T - c^T)x + \alpha$$
$$s.t. \quad \text{(5-40) and (5-42)}.$$

Constraints (5-40) and (5-42) in $(ISPI')$ are exactly the same as (5-34) and (5-35) **Error!**

Reference source not found.in (ISP), so sub-problems $(ISPI')$ and (ISP) have the

same set of constraints. In addition, a second sub-problem, (ISP'), is associated with

$(IMPI'_L)$:

$$(ISP') \qquad min \quad (\omega^T A - u^T - c^T)y$$
$$s.t. \qquad \text{(5-41) and (5-43)}.$$

Proposition 5.12: Problems $(IMPI'_L)$ and (IMP) are equivalent for each $L \in \mathfrak{D}$.

Proof: Trivial.∎

Analogous to Proposition 5.5, the following proposition deals with (MPI''_L),

which we obtain by including constraints $\sum_{l \in L}(g_l \mu_l) \le h$ in $(IMPI'_L)$:

$$(IMPI''_L) \qquad ma \sum_{k \in K}(c^T x_k)\lambda_k \quad + \quad \sum_{l \in L}(c^T y_l)\mu_l$$
$$s.t. \quad \text{(5-45)-(5-50)}$$
$$\sum_{l \in L}(g_l \mu_l) \quad \le h \qquad\qquad\qquad (5\text{-}51).$$

Proposition 5.13: If $\mu_L = 0_L$ satisfy constraints (5-51) (i.e., $h \ge 0$), then (IMP),

$(IMPI'_L)$, and $(IMPI''_L)$ are equivalent.

Remark 5.14: If $L = \varnothing$, constraints (5-46) are redundant for $(IMPI'_L)$.

5.3.3. Issues related to integer programs

B&P works with the linear relaxation of the master problem to obtain a bound at each

node of the B&B tree; it branches on variables x from the original problem (IP), which

are formulated as decision variables in the sub-problem instead of the master problem

(IMP). If the jth variable in vector $\boldsymbol{x}$, x_j, is selected to branch upon, an additional constraint associated with this variable must be included in the sub-problem (ISP) associated with each resulting child node. A B&P formulation obtained by applying DWDG to (IP) is analogous to a B&P formulation obtained by applying DWD to (IP').

Proposition 5.11 establishes that a feasible solution to (IP'), $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$, corresponds to feasible solution to (IP), $\overline{\boldsymbol{x}} + \overline{\boldsymbol{y}}$, with the same objective function value, and a feasible solution to (IP), $\overline{\boldsymbol{x}} + \overline{\boldsymbol{y}}$, corresponds to a feasible solution to (IP'), $(\overline{\boldsymbol{x}} + \overline{\boldsymbol{y}}, 0)$, with the same objective function value. Thus, any feasible solution $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$ to (IP') corresponds to the feasible solution to (IP'), $(\overline{\boldsymbol{x}} + \overline{\boldsymbol{y}}, 0)$, with the same objective function value. Feasible solutions with respect to (IP') in the form of $(\overline{\boldsymbol{x}} + \overline{\boldsymbol{y}}, 0)$ can be identified relatively easily in the context of B&P because x_j can be branched upon only if the corresponding y_j is fixed to zero. If the jth variable in vector $\boldsymbol{y}$, y_j, is fixed to zero in sub-problem (ISP'), the jth variable in vector $\boldsymbol{x}$, x_j, can be branched upon by including an additional constraint associated with this variable in the sub-problem (ISP') associated with each resulting child node. In the context of B&P, sub-problems (ISP) and (ISP') have the same set of constraints after branching on the same set of variables in both. However, DWDG requires the ability to generate columns from sub-problem (ISP') with $y_j = 0$. This is not difficult to implement in most applications because fixing $y_j = 0$ has the same effect as removing this variable from the sub-problem.

One advantage of DWD applied to integer programs is that it may obtain models with a stronger linear relaxation. An issue raised here is about the tightness of bounds provided by DWDG. Overall, the bound provided by DWDG is no worse than the one provided by the linear relaxation of (IP) and at most as good as the one provided by the linear relaxation of (IMP) (i.e., by DWD). Actually, the tightness of the bound provided by DWDG depends on $\{y_l \mid L\}$ in ($IMPI'_L$) (i.e., the selection of L) so that problem-specific knowledge may be needed to make an appropriate selection.

5.3.4. Case study 2: the integer cutting stock problem

Valerio de Carvalho (2005) introduced the following model of the cutting stock problem (CSP):

$$(CSP) \quad min \quad \sum_{k \in K} \left(c^T x_k \right) \lambda_k$$
$$s.t. \quad \sum_{k \in K} \left(-I x_k \right) \lambda_k \leq (-b) \qquad (5\text{-}52)$$
$$\lambda_k \in \mathbb{Z}_+^n, \qquad k \in K.$$

Here, I is the identity matrix, b represents the vector of non-negative demands, each x_k represents a cutting pattern, and variable λ_k represents the number of times cutting pattern k is used. Sub-problem ($CSP-SP$) generates cutting patterns:

$$(CSP-SP) \quad max \quad (\omega^T I - c^T) x$$
$$s.t. \quad w^T x \leq W \qquad (5\text{-}53)$$
$$x \in \mathbb{Z}_+^n \qquad (5\text{-}54).$$

Here, ω is the vector of dual variable values associated with constraints (5-52) and constraints (5-53) and (5-54) restrict cutting patterns.

Valerio de Carvalho (2005) proposed the use of dual cuts, equivalently including additional columns in the master problem:

$$(CSP') \quad min \quad \sum_{k \in K} \left(c^T x_k \right) \lambda_k \ + \ \sum_{l \in L} \left(c^T y_l \right) \mu_l$$

$$s.t. \quad \sum_{k \in K} \left(-I x_k \right) \lambda_k \ + \ \sum_{l \in L} \left(-I y_l \right) \mu_l \quad \leq (-b) \quad\quad (5\text{-}55)$$

$$\lambda_k \quad\quad\quad\quad \in \mathbb{Z}_+^n, \ k \in K \quad\quad (5\text{-}56)$$

$$\mu_l \quad \in \mathbb{Z}_+^n, \ l \in L \quad\quad (5\text{-}57).$$

Here, all symbols are the same as (CSP) and vectors $\{ x_k \mid K \}$ are also generated using $(CSP-SP)$. Valerio de Carvalho (2005) defined a family of valid dual cuts, each of which corresponds to a y_l vector and exploits the fact that *a given length can be cut and used to fulfill the demand for a smaller length* in the cutting stock problem (Valerio de Carvalho, 2005). $\{ y_l \mid L \}$ can also be obtained from the following sub-problem $(CSP-SP')$:

$$(CSP-SP') \quad max \quad (\omega^T I - c^T) y$$

$$s.t. \quad\quad w^T y \leq 0$$

$$y \in \mathbb{Z}^n.$$

Please note that Valerio de Carvalho (2005) only studied the linear relaxation of (CSP) in their computational tests; that is, they dealt with the linear relaxations of (5-56) and (5-57). To manage the number of columns in RMP, Valerio de Carvalho (2005) included only a polynomial number of these dual cuts with the feature that each element in vector y_l belongs to $\{-1, 0, 1\}$. Furthermore, the author proves that *the lower bound*

given by the optimal solution of the linear relaxation of the extended model with dual

cuts is equal to the one given by Glimore-Gomory model (Valerio de Carvalho, 2005).

The authors used a heuristic to generate a set of these dual cuts and a pre-processing step

to incorporate them in RMP, and demonstrated an average 75.9% reduction in the num-

ber of CG iterations and 94.8% reduction in the number degenerate pivots in an experi-

ment that involves 120 instances.

We introduce two reformulations, $(IMP)^{II}$, which comprises (5-36)-(5-39) except

the convexity constraint (5-37), and corresponding $(IMPI'_L)^{II}$, which comprise (5-44)-

(5-50) except (5-47). It is obvious that $(IMP)^{II}$ and $(IMPI'_L)^{II}$ are equivalent. We can

map the Glimore-Gomory model of the cutting stock problem to $(IMP)^{II}$.

Proposition 5.15: The work of Valerio de Carvalho (2005) can be viewed as a special

application of DWDG.

Proof: We can obtain $(IMPI'_L)^{II}$ corresponding to (CSP) as follows:

$$(IMPI'_L)^{II} \quad min \quad -\sum_{k \in K}\left(c^T x_k\right)\lambda_k \quad + \quad \left(-\sum_{l \in L}\left(c^T y_l\right)\mu_l\right)$$

$$s.t. \quad \text{(5-55), (5-56), and (5-57)}$$

$$\sum_{k \in K}\left(x_k\right)\lambda_k \quad + \quad \sum_{l \in L}\left(y_l\right)\mu_l \quad \geq 0 \qquad \text{(5-58)}.$$

Constraints (5-58) are implied by (5-55) because $\sum_{k \in K}\left(Ix_k\right)\lambda_k + \sum_{l \in L}\left(Iy_l\right)\mu_l \geq b \geq 0$.

Removing constraints (5-58) from $(IMPI'_L)^{II}$ yields (CSP'), which was proposed by

Valerio de Carvalho (2005). Because the sub-problem is of the "$\leq$" form, μ_l is non-

negative (see section 5.2.6). ■

One disadvantage of DWDG is that the large number of constraints (5-46) makes the master problem more challenging to solve. However, for some special applications (e.g., cutting stock problem), constraints (5-46) might be implied by constraints (5-45); such constraints (5-46) would be redundant and could be eliminated. Without constraints (5-46), the primal master problem in DWDG can be thought of as a relaxation of the primal master problem in DWD because the former includes additional variables $\{\mu_l \mid L\}$. The column associated with each variable μ_l corresponds to a cut in the space associated with the linear programming dual of the master problem. Adding variables μ_l does not change the optimal objective function value of the primal problem, so these dual cuts do not cut off all optimal dual solutions in the case in which there are alternative optimal dual solutions. But, they provide additional restrictions in the dual space, which may serve to improve convergence by limiting the range of values that associated dual variables can assume (Ben Amor et al., 2006).

Vanderbeck and Savelsbergh (2006) commented on the work of Valerio de Carvalho (2005), interpreting dual cuts in the cutting stock problem as exchange vectors, each of which is the difference between two vectors that represent feasible sub-problem solutions; that is, $\boldsymbol{y}_l = (\boldsymbol{x}_{k_1} - \boldsymbol{x}_{k_2})$ and $\boldsymbol{D}\boldsymbol{y}_l = \boldsymbol{D}(\boldsymbol{x}_{k_1} - \boldsymbol{x}_{k_2}) = \boldsymbol{d} - \boldsymbol{d} = \boldsymbol{0}$, so that $\boldsymbol{y}_l$ satisfies (5-41) and (5-43). Adding sub-problem solution vector $\boldsymbol{x}_k$ to a linear combination of a set L of exchange vectors (i.e., $\boldsymbol{x}_k + \sum_{l \in L} (\boldsymbol{y}_l) \bar{\mu}_l$) forms a particular linear combination that can generate another sub-problem solution so that the sub-problem solver need not be used to generate all sub-problem solutions (i.e., $\boldsymbol{D}\boldsymbol{x}_k + \sum_{l \in L} (\boldsymbol{D}\boldsymbol{y}_l) \bar{\mu}_l = \boldsymbol{d} + \boldsymbol{0} = \boldsymbol{d}$ and

$x_k + \sum_{l \in L}(y_l)\overline{\mu}_l \in \mathbb{Z}^n$). In the cutting stock problem, the particular linear combination of a feasible sub-problem solution and a set L of exchange vectors can be used to construct other *feasible* sub-problem solutions (i.e., $x_k + \sum_{l \in L}(y_l)\overline{\mu}_l \geq b \geq 0$, because the vector of right-hand-side coefficients, b, is always non-negative for the cutting stock problem). However, for other applications the feasibility of such combinations can not be guaranteed because $\exists (x_k + \sum_{l \in L}(y_l)\overline{\mu}_l \geq 0)$. In contrast, DWDG includes additional constraints (5-14) or (5-46) to guarantee that the particular linear combination (i.e., $x_k + \sum_{l \in L}(y_l)\overline{\mu}_l$) always generates other feasible sub-problem solutions. So, DWDG subsumes and generalizes the methods of Valerio de Carvalho (2005) and Vanderbeck and Savelsbergh (2006), which add dual cuts or exchange vectors, respectively.

5.3.5. Case study 3: production-assembly-distribution system design

Liang and Wilhelm (2007a) (i.e., section 4) applied dual cuts to the production-assembly-distribution system design problem, which employs two different types of sub-problems. One sub-problem type is a 0-1 knapsack, which allocates components to facility alternatives under a budget limitation. Space considerations preclude a detailed explanation of their model here; however, we outline the relationship of their work to DWDG in this subsection to relate that this special case of DWDG accelerates convergence and reduces run time.

Following the work of (Valerio de Carvalho, 2005), the authors designed dual cuts by using the symmetric difference of two feasible solutions from the knapsack type sub-problem. Each dual cut corresponds to one y_l column, which represents the reallo-

cation of a component to a different alternative to reduce cost relative to the budget limitation. Given a feasible solution representing an allocation of components to alternatives under the budget limitation, each reallocation yields another feasible allocation. Thus, fewer columns from the knapsack type sub-problem need be incorporated in RMP, improving convergence. A pre-processing step can generate a batch of these dual cuts and incorporate them in RMP. Furthermore, adding these dual cuts will not affect the tightness of the bound. Test results show an average reduction of 77.6% in the number of CG iterations and 94.1% in run time for each of 20 instances. This work can also be viewed as a special case that DWDG subsumes and generalizes.

5.3.6. Computational tests on the integer generalized assignment problem

This subsection describes the computational tests we conduct to evaluate the efficacy of DWDG in application to an integer program. For this purpose, we solve the generalized assignment problems that Savelsbergh (1997) used in his study. As in subsection 5.2.6, the master problem includes the (disjoint) assignment constraints $Ax = b$ and each sub-problem comprises a single (knapsack) constraint, which we now solve as an integer program using 0-1 knapsack code (Martello et al., 1999).

Our tests explore two issues that affect run time: first, how does DWDG affect the rate of convergence of CG at each node in the B&B tree; and, second, how does DWDG affect the tightness of bounds provided by RMP at each node.

Similar to the definition of y columns in subsection 5.2.6, each y column here represents that reallocation of agent i from job j to job $j+1$. The total number of y

Table 9. Comparison of DWD and DWDG applied to GAP

	DWD								DWDG							
	# Iterations		# Deg. Iter.		Run Time (s)		Bound Gap		# Iterations		# Deg. Iter.		Run Time (s)		Bound Gap	
	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
A.3.30	129.5	147	84.5	97	0.1142	0.125	0.00%	0.00%	54.2	65	27	36	0.0668	0.093	0.00%	0.00%
A.5.30	68.6	77	25.3	32	0.0688	0.078	0.00%	0.00%	37.2	41	12.9	19	0.0655	0.078	0.03%	0.25%
A.10.30	34	41	4.6	8	0.0393	0.047	0.00%	0.00%	23.5	30	4.4	11	0.0842	0.094	0.00%	0.00%
B.3.50	319.1	374	251.4	308	0.8220	1.000	0.28%	0.57%	113.2	167	77.4	134	0.2547	0.359	0.54%	1.26%
B.5.50	147.6	181	90.2	124	0.3782	0.484	0.03%	0.14%	68.3	80	34.2	45	0.2721	0.312	0.09%	0.24%
B.20.50	32.5	35	4.2	9	0.1034	0.110	0.02%	0.07%	22.5	24	3.6	6	0.4393	0.485	0.02%	0.10%
C.3.30	104.3	114	65.8	77	0.0965	0.109	0.23%	0.70%	50.8	57	25.6	33	0.0592	0.063	0.45%	0.96%
C.5.30	55.5	64	22.5	33	0.0530	0.062	0.15%	0.58%	33	41	11.9	18	0.0608	0.078	0.24%	0.62%
C.10.30	26.8	30	5	8	0.0315	0.032	0.11%	0.45%	19.3	21	4.2	7	0.0673	0.078	0.16%	0.48%
D.3.50	292	321	220.3	280	0.8138	0.969	0.20%	0.25%	176.3	231	126.5	183	0.5171	0.625	0.22%	0.30%
D.5.50	132.4	144	53.6	71	0.3486	0.391	0.17%	0.24%	79.6	92	34.5	45	0.3578	0.407	0.19%	0.29%
D.20.50	36.5	38	4.6	6	0.1467	0.156	0.58%	0.87%	25.3	26	4.1	6	0.6173	0.688	0.59%	0.87%

columns is $(n-1) \times m$; we generate them in a batch and add them to the initial RMP of DWDG. Also, we need to add constraints corresponding to (5-46) in ($IMPI'_L$) to RMP.

Table 9 presents test results. The first column defines each the instance using the tuple $(\#,m,n)$. Following Savelsbergh (1997) we use each parameter setting to generate 10 instances and record the average and maximum for three measures of performance: number of iterations, number of degenerate iterations, run time (in seconds), and bound provided at the root node, which we compare with the optimal value in terms of the per-cent gap. Eight columns give these measures for DWD; and eight, for DWDG. Results reinforce our analysis, clearly showing that DWDG needs fewer CG iterations (and fewer degenerate iterations) than DWD in all cases. However, DWDG may require a longer run time and a correspondingly larger portion of CG iterations that are degenerate in some cases, in particular, if m is large. The reason is the same as that in section 5.2.6. Also, we can see that the bound provided by DWDG is not as tight as that provided by DWD.

5.4. Summary

The section presents a generalization of DWD (DWDG) that achieves the goal of accel-erating convergence. It attains a set of primary research objectives for investigating DWDG in application to linear programs by establishing a theoretical basis, providing geometric insights through a numerical example, and showing how DWDG improves convergence through analytical means, subsuming and generalizing approaches that have been shown to improve convergence in special cases, and reporting successful computational experience. It also accomplishes its second set of research objectives, ex-

ploring equivalent issues related to DWDG in application to integer programs; in particular, it describes how DWDG subsumes and generalizes approaches that has been shown to improve the rate at which CG converges in application to the cutting stock and production-assembly-distribution system design problems and relates computational tests that provide insights into the efficacy of DWDG.

In application to a linear program, DWD and DWDG differ in the way they represent points in the polytope associated with the sub-problem. The former represents these points as a convex combination of the extreme points of that polytope, while the latter represents them using a mixture of both convex and linear combinations of relevant points. DWDG leads to a master problem with fewer columns, but at the cost of adding more rows. This paper shows that DWDG does, however, accelerate the rate of convergence, reducing the run time required to prescribe an optimal solution.

6. DUAL-ASCENT AND PRIMAL HEURISTICS FOR

PRODUCTION-ASSEMBLY-DISTRIBUTION SYSTEM DESIGN*

This section proposes two dual-ascent algorithms and uses each in combination with a primal drop heuristic to solve the uncapacitated production-assembly-distribution system design problem (UPADSPD), which is formulated as a mixed integer program. Computational results indicate that one combined heuristic finds solutions within 0.15% of optimality in most cases and within reasonable time, an efficacy suiting it well for actual large-scale applications.

6.1. Introduction

In this section, we propose two dual-ascent algorithms and use each in combination with a primal drop heuristic to solve UPADSDP, which is important as a basic model for designing enterprises and their supply chains, both domestic and global. UPADSDP extends the facility location problem by including bill-of-materials (BOM) relationships in a multi-echelon assembly system that operates over multiple time periods. And it is also a simplification of PADSDP by removing capacity-type constraints.

This paper is motivated by the need to optimize assembly systems and their supply chains so that companies can better compete in the global economy and by the challenges posed by the size of mixed-integer-programs that have been formulated to model

_____

UPADSDP. The primary contribution that this paper makes is the proposed solution method, which yields tight lower bounds (actually near-optimal solutions) within reasonable run times so that it is well suited for large-scale instances.

The body of this section is organized in five sections. Subsection 6.2 describes our model. Subsection 6.3 presents our dual-ascent algorithms and 6.4 presents the primal drop heuristic. And subsection 6.5 describes computational results. Finally, subsection 6.6 offers summaries.

6.2. UPADSDP Model

UPADSDP integrates three types of decisions (i.e., opening facilities, allocating components to facilities, and routing material flows) and holds the objective of minimizing the total cost associated with these decisions without considering the capacity-type constraints in PADSDP. Here we use the same index, set, parameter and variable definition in section 4. We now formulate our UPADSDP model:

Model ($\mathcal{PA}$)

$$Z^*(\mathcal{PA}) = \text{Min} \sum_{f \in F} G_f^O x_f \qquad + \sum_{e \in E} \sum_{p \in P_e} \sum_{f \in F_p} G_{pf}^A y_{pf} \qquad + \sum_{e \in E} \sum_{a \in A_e} G_a^V z_a \qquad (6\text{-}1)$$

$$-x_f \quad + y_{pf} \qquad\qquad \leq 0, \qquad\qquad \forall\, e \in E, p \in P_e, f \in F_p \qquad (6\text{-}2)$$

$$-\bar{D}_e y_{pf} + \sum_{t \in T} \sum_{a \in A_{pft}} z_a \quad \leq 0, \qquad\qquad \forall\, e \in E, p \in P_e, f \in F_p \qquad (6\text{-}3)$$

$$\sum_{a \in A_{pft}^+} z_a - \sum_{a \in A_{pft}^-} z_a \quad = D_{pft}, \qquad\qquad \forall\, e \in E, p \in P_e, f \in F_p, t \in T \quad (6\text{-}4)$$

$$x_f \qquad\qquad\qquad \in \{0,1\}, \qquad \forall\, f \in F \qquad (6\text{-}5)$$

$$y_{pf} \qquad\qquad\qquad \in \{0,1\}, \qquad \forall\, e \in E, p \in P_e, f \in F_p \qquad (6\text{-}6)$$

$$z_a \qquad\qquad \geq 0, \qquad\qquad \forall\, e \in E, a \in A_e \qquad (6\text{-}7).$$

We also assume that each end product has a BOM that is a tree structure and requires a unique set of components. Thus, sets P and A can be partitioned with respect to end product e into disjoint subsets P_e and A_e for each $e \in E$; correspondingly, objective (6-1) and constraints (6-3), (6-4), (6-6), and (6-7) are separable relative to e. The meaning of objective (6-1) and constraints (6-2)-(6-7) are the same as those in PADSDP.

An alternative formulation, ($\mathcal{PD}$), of UPADSDP is to use disaggregated flow variables $\sum_{t \in T, \{f \in F_e : D_{eft} > 0\}} z_a^{eft}$ to replace aggregate flow variable z_a in objective (6-1) and constraints (6-3), and (6-4). Similarly, constraints (6-3) and (6-4) can be disaggregated relative to z_a^{eft} as shown in (6-9) and (6-10). ($\mathcal{PD}$) can be represented as:

Model ($\mathcal{PD}$)

$$Z^*(\mathcal{PD}) = \text{Min} \sum_{f \in F} G_f^O x_f + \sum_{e \in E} \sum_{p \in P_e} \sum_{f \in F_p} G_{pf}^A y_{pf} + \sum_{e \in E} \sum_{a \in A_e} G_a^V \left(\sum_{t \in T, \{f \in F_e : D_{eft} > 0\}} z_a^{eft} \right) \quad (6\text{-}8)$$

s.t. (6-2), (6-5), (6-6)

$$-D_{ef't'} y_{pf} + \sum_{t \in T} \sum_{a \in A_{pft}} z_a^{ef't'} \leq 0, \ \forall \ e \in E, p \in P_e, f \in F_p, t' \in T, f' \in F_e : D_{ef't'} > 0 \quad (6\text{-}9)$$

$$\sum_{a \in A_{pft}^+} z_a^{ef't'} - \sum_{a \in A_{pft}^-} z_a^{ef't'} = D_{pf't'}, \ \forall \ e \in E, p \in P_e, f \in F_p, t \in T, t' \in T, f' \in F_e : D_{ef't'} > 0 \quad (6\text{-}10)$$

$$z_a^{eft} \geq 0, \ \forall \ e \in E, a \in A_e, t \in T, f \in F_e : D_{eft} > 0 \quad (6\text{-}11).$$

Model ($\mathcal{PD}$) comprises more constraints and continuous variables than ($\mathcal{PA}$). Letting CZ_e represent the number of customer zones for e, one z_a variable in ($\mathcal{PA}$) becomes $\sum_{e \in E} CZ_e \cdot |T|$ variables of type z_a^{eft} in ($\mathcal{PD}$) and one constraint (6-3) will become $\sum_{e \in E} CZ_e \cdot |T|$ constraints of type (6-9). Thus, the linear relaxation of ($\mathcal{PD}$) becomes more changeling because it is much larger. However, the linear relaxation of ($\mathcal{PD}$)

can provide a tighter bound than that of ($\mathcal{PA}$) because the former is a disaggregated version of the latter (Nemhauser and Wolsey, 1999).

We assume that G_f^O and G_{pf}^A are non-negative, reflecting actual costs. Otherwise, if these costs are negative, then the values of corresponding binary variables should each be set to one and they can be eliminated from the problem. Based on this assumption, we define the linear relaxation of ($\mathcal{PD}$), ($\mathcal{P}$), by relaxing constraints (6-5) and (6-6) as:

$$(6\text{-}12) \quad x_f \geq 0, \ \forall \ f \in F$$
$$(6\text{-}13) \quad y_{pf} \quad unrestriced, \ \forall \ e \in E, p \in P_e, f \in F_p.$$

Here we do not use $0 \leq x_f, y_{pf} \leq 1$ because $y_{pf} \geq 0$ is implied by (6-3) and (6-7). Because costs of x_f, y_{pf} are non-negative and the problem is to minimize the cost, x_f, y_{pf} will stay as small as possible in the optimal solution. So, $x_f \leq 1$ and $y_{pf} \leq 1$ are implied

We now introduce the linear programming dual of ($\mathcal{P}$), ($\mathcal{D}$). It is obvious that $Z^*(\mathcal{D})$ $= Z^*(\mathcal{P}) \leq Z^*(\mathcal{PD})$, where $Z^*(-)$ represents the optimal objective value of $(-)$.

Model ($\mathcal{D}$)

$$Z^*(\mathcal{D}) = \text{Max} \ + \sum_{e \in E} \sum_{p \in P_e} \sum_{f \in F_p} \sum_{t \in T} D_{pft} \left(\sum_{f' \in F_e : D_{ef't'} > 0} \sum_{t' \in T} \omega_{pft}^{f't'} \right) \tag{6-14}$$

$$\sum_{p \in P : f \in F_p} \mu_{pf} \leq G_f^O, \ \forall \ f \in F_l \tag{6-15}$$

$$\sum_{f' \in F_e : D_{ef't'} > 0} \sum_{t' \in T} \upsilon_{pf}^{f't'} - \mu_{pf} = G_{pf}^A, \ \forall \ e \in E, p \in P_e, f \in F_p \tag{6-16}$$

$$-\sum_{p \in P_e} \sum_{f \in P_f} \sum_{t \in T : a \in A_{pft}} (\upsilon_{pf}^{f't'} / D_{ef't'}) \ + \sum_{p \in P_e} \sum_{f \in P_f} \sum_{t \in T : a \in A_{pft}^+} \omega_{pft}^{f't'} - \sum_{p \in P_e} \sum_{f \in P_f} \sum_{t \in T : a \in A_{pft}^-} \omega_{pft}^{f't'} \leq G_a^V,$$
$$\forall \ e \in E, a \in A_e, t' \in T, f' \in F_e : D_{ef't'} > 0 \tag{6-17}$$

$$\mu_{pf} \geq 0, \qquad \forall \ e \in E, p \in P_e, f \in F_p \tag{6-18}$$

$$\upsilon_{pf}^{f't'} \geq 0, \qquad \forall \ e \in E, p \in P_e, f \in F_p, t' \in T, f' \in F_e : D_{ef't'} > 0 \tag{6-19}$$

$$\omega_{pft}^{f't'} \ unrestriced \ , \ \forall \ e \in E, p \in P_e, f \in F_p, t \in T, t' \in T, f' \in F_e : D_{ef't'} > 0 \ (6\text{-}20).$$

One important substructure in ($\mathcal{D}$) is defined as follows: Given the values of dual variables $\upsilon_{pf}^{f't'}$, (i.e., $\bar{\upsilon}_{pf}^{f't'}$), ($\mathcal{D}$) can be simplified as

Model (SP): $Z^*(\text{SP}) = \max \sum_{e \in E} \sum_{p \in P_e} \sum_{f \in F_p} \sum_{t \in T} D_{pft} (\sum_{f' \in \in F_e : D_{ef't'} > 0} \sum_{t' \in T} \omega_{pft}^{f't'})$, s.t.(6-17) and (6-20).

Please note that RHS associated with constraint (6-17) is $\sum_{p \in P_e, f \in P_f, t \in T : a \in A_{pft}} (\bar{\upsilon}_{pf}^{f't'} / D_{ef't'})$

$+G_a^V$, the cost per unit of flow on hyper-arc a. SP can be decomposed with respect to each end product e (i.e., SP(e)) so that each resulting sub-problem corresponds to the linear dual of a shortest path problem on the hypergraph. $Z^*(\text{SP}) = \sum_{e \in E} Z^*(\text{SP}(e))$.

SP(e) can be solved in polynomial time (Gallo and Pallottino, 1992) (see step 2-(a) in subsection 6.3.2) using an algorithm that is similar to a network labeling algorithm (e.g., Dijkstra algorithm). $\omega_{pft}^{f't'} |_{a \in A_{pft}^+}$ is the label on the head node of the hyper-arc, and

$\sum_{p \in P_e, f \in P_f, t \in T : a \in A_{pft}^-} \omega_{pft}^{f't'}$ is the summation of the labels on the tail nodes of the hyper-arc.

In the following subsections, we discuss our approaches for obtaining solutions to the mixed-integer program model ($\mathcal{PD}$). We measure the quality of our solution to each instance using the gap between the values of the solution our approach prescribes and the optimal solution.

6.3. Dual-ascent Solution Approach

In order to design our dual-ascent algorithm, we first study inherent relationships between different types of dual variables. Constraints (6-15) show that dual variable μ_{pf},

which is associated with constraint (6-2), plays the role of distributing fixed cost G_f^O to different components processed at facility f (i.e., $p \in P_f$). Given the value of variable μ_{pf}, $\bar{\mu}_{pf}$, dual variable $\upsilon_{pf}^{f't'}$, which is associated with constraint (6-9), distributes the fixed cost $G_{pf}^A + \bar{\mu}_{pf}$ that is associated with satisfying demands $D_{ef't'}$, $\forall f' \in F_e^D$, $t' \in T$, when component p is processed at f to assemble end product e. Thus, the total cost on arc a is equal to the variable cost, G_a^V, plus fixed cost, $\upsilon_{pf}^{f't'}/D_{ef't'}, a \in \bar{A}_{pf}$ (i.e., RHS in (SP(e))). The dual-ascent algorithm updates μ_{pf} and $\upsilon_{pf}^{f't'}$ as it seeks to distribute fixed costs G_f^O and G_{pf}^A on arc a so that the objective function value can be improved monotonically. In the following sections, we propose two algorithms to update μ_{pf} and $\upsilon_{pf}^{f't'}$.

6.3.1. Dual-ascent algorithm one (DAA1)

This algorithm is based on the following observation. Given values of $\upsilon_{pf}^{f't'}$ variables (also values of μ_{pf} variables by (6-16)), ($\mathcal{D}$) can be simplified as a set of sub-problems SP(e) $e \in E$, each of which can be solved as a shortest path problem on the hypergraph. Increasing the value of $\upsilon_{pf}^{f't'}$, $\bar{\upsilon}_{pf}^{f't'}$, increases the cost on arc a (i.e., $G_a^V + \sum_{pf} (\bar{\upsilon}_{pf}^{f't'}/D_{ef't'})$) and can increase the length of the shortest paths (note, there may exist multiple shortest paths on the hypergraph); that is, $\sum_{epft} D_{pft}(\sum_{f't'} \omega_{pft}^{f't'})$ (i.e., the objective value of ($\mathcal{D}$)) becomes larger. Because (6-16) holds at equality (i.e., $\sum_{f't'} \upsilon_{pf}^{f't'} = \mu_{pf} + G_{pf}^A$), μ_{pf} increases as $\upsilon_{pf}^{f't'}$ increases. However, μ_{pf} must satisfy

constraints (6-15) (i.e., $\sum_p \mu_{pf} \leq G_f^O$), so $\upsilon_{pf}^{f't'}$ and μ_{pf} can not be increased without bound. Thus, the summation on all $\upsilon_{pf}^{f't'}$ variables is equal to $\sum_{pf} (G_{pf}^A + \mu_{pf})$ by (6-16), which is always less than $\sum_{pf} G_{pf}^A + \sum_f G_f^O$ by (6-15). We start with setting all $\upsilon_{pf}^{f't'}$ and μ_{pf} variables to zero, then increase their values one by one until we can not increase them further because of restriction (6-15).

Furthermore, given any component p, the set of arcs associated with $\upsilon_{pf}^{f't'}$, $f \in F_p$ forms a valid arc cut (Liang and Wilhelm, 2007a); that is, component p must be supplied by at least one facility $f \in F_p$ and any shortest path must contain one of these arcs (i.e., $a \in \bigcup_{f \in F_p} \overline{A}_{pf}$). Increasing the values of all $\upsilon_{pf}^{f't'}$, $f \in F_p$ associated with a given p by one unit will increase the length of the shortest path by one unit. By (6-15) and (6-16), the summation of all $\upsilon_{pf}^{f't'}$ variables must be less than $\sum_{pf} G_{pf}^A + \sum_f G_f^O$, so increasing the values of $\upsilon_{pf}^{f't'}$, $f \in F_p$ for a given p by one unit will use $|F_p|$ units of resource of $\sum_{pf} G_{pf}^A + \sum_f G_f^O$ and improve the objective function value by one unit. However, it is not necessary to simultaneously change all $\upsilon_{pf}^{f't'}$, $f \in F_p$ for a give p. We only need to change those $\upsilon_{pf}^{f't'}$ on the shortest paths. Based on that, we then decide how many $\upsilon_{pf}^{f't'}$, $f \in F_p$ for a given p must be updated simultaneously to improve the objective function value. Please note that the model includes multiple components and it is attractive to select one p such that we only need to increase a small number of $\upsilon_{pf}^{f't'}$,

$f \in F_p$ simultaneously. Based on this, we decide the order in which to update $\upsilon_{pf}^{f't'}$ and μ_{pf}. The following section details DAA1.

The underlying material flow network can be represented as an acyclic hypergraph. DAA1 solves a set of shortest path problems, each on a hypergraph (i.e., SP(e), $\forall e \in E$). The shortest path algorithm on an acyclic hypergraph is similar to the one on an acyclic graph that numbers nodes in topological order, then goes through all nodes in that order, setting the label of each to represent the length of the shortest path from the source node to it (Gallo and Pallottino, 1992).

Step 0: Initialization

Set variables $\mu_{pf} = 0$ and $\upsilon_{pf}^{f't'} = 0$; and set $n = 1$ (n represents how many μ_{pf}, $f \in F_p$ update simultaneously for a given $p \in P$).

Step 1: Update labels $\overline{\omega}_{pft}^{f't'}$ and $\underline{\omega}_{pft}^{f't'}$ at each node of each material flow network for $D_{ef't'}$

Using shortest path algorithms to update two labels at each node to record the lengths of the shortest paths from the current node to the source and sink nodes, respectively, in all material networks associated with $D_{ef't'}$, $\forall e \in E, t' \in T, f' \in F_e : D_{ef't'} > 0$. See the following section for details.

Step 2: Update values of all μ_{pf} and $\upsilon_{pf}^{f't'}$ according to n and labels $\overline{\omega}_{pft}^{f't'}$ and $\underline{\omega}_{pft}^{f't'}$

For each material network associated with $D_{ef't'}$, first determine the duality gap of each node (i.e., $\overline{\omega}_{pft}^{f't'} - \underline{\omega}_{pft}^{f't'}$), which is used to set the value of $\Delta \upsilon_{pf}^{f't'}$ (step 2-(a) in subsection

6.3.2); then update $\Delta \upsilon_{pf}^{f't'}$ with the consideration of the dependence among $p \in P_e$ and

n (i.e., how many $\upsilon_{pf}^{f't'}$, $f \in F_p$ update simultaneously for a given p), (step 2-(b)&(c)

in subsection 6.3.2); next, accumulate $\Delta \upsilon_{pf}^{f't'}$ to set the value of $\Delta \mu_{pf}$ (step 2-(d) in sub-

section 6.3.2); after that, update $\Delta \mu_{pf}$ with the restriction of $(\sum_{p' \in P_f} \mu_{p'f}) + \Delta \mu_{pf} \leq G_f^O$

(step 2-(e) in subsection 6.3.2); finally, according to values $\Delta \mu_{pf}$, update $\Delta \upsilon_{pf}^{f't'}$ so that

$\sum_{f't'} \Delta \upsilon_{pf}^{f't'} = \Delta \mu_{pf}$, and update values of μ_{pf} and $\upsilon_{pf}^{f't'}$ by $\mu_{pf} \leftarrow \mu_{pf} + \Delta \mu_{pf}$, $\upsilon_{pf}^{f't'} \leftarrow$

$\upsilon_{pf}^{f't'} + \Delta \upsilon_{pf}^{f't'}$ (step (f)-(i) in subsection 6.3.2). See the following section for details.

If some $\Delta \upsilon_{pf}^{f't'} \neq 0$, go to Step 1; Else if $n+1$ does not exceed the largest possi-

ble number of facility alternatives for component p, $\max_{p \in P} |F_p|$, then set $n \leftarrow n+1$

and Return to Step 1.

6.3.2. Implementation details for DAA1

Step 1: Labeling nodes in the material network associated with one demand $D_{ef't'}$

a) Set the forward label $\bar{\omega}_{pft}^{f't'}$ on each node

Assign the forward label zero to each supplier node (i.e., node with incoming arcs) and

$+\infty$ to each other node. Then Go through all nodes according to the topological order,

set the forward label as follows:

For node corresponding to indices pft and $f't'$ (i.e., each node corresponds to one flow

conservation constraint (6-10)), check all its incoming arcs. Set $\bar{\omega}_{pft}^{f't'} =$

$\min_a (c_a + \sum \bar{\omega}_{p_1 f_1 t_1}^{f't'})$, where $\sum \bar{\omega}_{p_1 f_1 t_1}^{f't'}$ is the summation of all forward labels of the tail nodes of arc a.

b) Set the backward label $\underline{\omega}_{pft}^{f't'}$ on each node

For the only demand node, assign its forward label $\bar{\omega}_{pft}^{f't'}$ to the backward label. For all other nodes, assign backward labels to $-\infty$. Then go through all nodes according to reverse topological order, setting the backward label on each node as follows:

For the node corresponding to indices pft and $f't'$ (i.e., each node corresponds to one constraint (6-10)), check all its outgoing arcs. Set $\underline{\omega}_{pft}^{f't'} = \max_a (\underline{\omega}_{p_2 f_2 t_2}^{f't'} - c_a - \sum \bar{\omega}_{pft}^{f't'})$,

where $\underline{\omega}_{p_2 f_2 t_2}^{f't'}$ is the backward label of the head node of arc a and $\sum \bar{\omega}_{pft}^{f't'}$ is the summation of the forward labels of all other tail nodes of arc a.

Step 2: Update values of all μ_{pf} and $\upsilon_{pf}^{f't'}$ according to n and labels $\bar{\omega}_{pft}^{f't'}$ and $\underline{\omega}_{pft}^{f't'}$

At each material network associated with $D_{ef't'}$, we calculate all $\Delta\upsilon_{pf}^{f't'}$, $p \in P_e, f \in F_p$, using all $\bar{\omega}_{pft}^{f't'}$ and $\underline{\omega}_{pft}^{f't'}$

a) Determine the duality gap for each node in the material network associated with $D_{ef't'}$

If $\bar{\omega}_{pft}^{f't'} - \underline{\omega}_{pft}^{f't'} < 0$, error (there is a shorter path than current shortest path).

If $\bar{\omega}_{pft}^{f't'} - \underline{\omega}_{pft}^{f't'} = 0$, this node can be on the shortest path; we can increase the value of $\upsilon_{pf}^{f't'}$ to extend the length of the shortest path.

If $\bar{\omega}_{pft}^{f't'} - \underline{\omega}_{pft}^{f't'} > 0$, this node is not on the shortest path.

For each pf $f't'$, set $\Delta\upsilon_{pf}^{f't'} = \min_{t\in T}\{(\overline{\omega}_{pft}^{f't'} - \underline{\omega}_{pft}^{f't'})D_{ef't'}\}$. Component p must be processed by at least one facility alternative $f \in F_p$.

Given component p, $\Delta\upsilon_{pf}^{f't'} = 0$ for at least one alternative $f \in F_p$ and $\Delta\upsilon_{pf}^{f't'} \geq 0$ for other alternative; that is, the shortest path must contain one of the nodes with indices pft $f't'$, $f \in F_p$, $t \in T$); furthermore, updating the values of $\upsilon_{pf}^{f't'}$, $f \in F_p$ with $\Delta\upsilon_{pf}^{f't'} = 0$ simultaneously to the smallest non-zero $\Delta\upsilon_{pf}^{f't'}$ among alternative $f \in F_p$ (i.e., $\min_{f\in F_p}\{\Delta\upsilon_{pf}^{f't'} > 0\}$) can increase the length of the shortest path without changing the current shortest path.

b) Update $\Delta\upsilon_{pf}^{f't'}$, considering the dependence among $p \in P_e$

Components $p \in P_e$ in the same hypergraph are dependent (i.e., updating $\upsilon_{pf}^{f't'}$, $p \in P_e$ serially changes the current shortest path). To avoid this, updating $\Delta\upsilon_{pf}^{f't'} \leftarrow \Delta\upsilon_{pf}^{f't'} / |P_e|$.

c) Update $\Delta\upsilon_{pf}^{f't'}$, considering changes $\Delta\upsilon_{pf}^{f't'} = 0$, $f \in F_p$ simultaneously

For each p, no less than one $\Delta\upsilon_{pf}^{f't'} = 0$, $f \in F_p$. We need to update $\upsilon_{pf}^{f't'}$ for $f \in F_p$ simultaneously; otherwise, we can not improve the length of the shortest path. Given p, suppose n represents the number of $\upsilon_{pf}^{f't'}$, $f \in F_p$ that must be updated simultaneously. We sort $\Delta\upsilon_{pf}^{f't'}$, $f \in F_p$ in non-decreasing order. We can increase the value of the first n $\Delta\upsilon_{pf}^{f't'}$ until all of them are equal to the value of the $(n+1)^{th}$ $\Delta\upsilon_{pf}^{f't'}$. Thus, set the value of the first n $\Delta\upsilon_{pf}^{f't'}$ to the value of the $(n+1)^{th}$ $\Delta\upsilon_{pf}^{f't'}$, and set remaining $\Delta\upsilon_{pf}^{f't'}$ to zero.

Given $D_{ef't'}$, check upper bounds associated with $\Delta\mu_{pf}$ for all $p \in P_e, f \in F_p$.

For each p, do the following:

d) Given p, use $\Delta\upsilon_{pf}^{f't'}$ to obtain $\Delta\mu_{pf}$ s.t. $(\sum_{f"t"}\upsilon_{pf}^{f"t"}) + \Delta\upsilon_{pf}^{f't'} \le (u_{pf} + \Delta\mu_{pf}) + G_{pf}^{A}$ for

each , $f \in F_p$; i.e., $\Delta\mu_{pf} = \max\{(\sum_{f"t"}\upsilon_{pf}^{f"t"}) + \Delta\upsilon_{pf}^{f't'} - u_{pf} - G_{pf}^{A}, 0\}$. If $\exists f \in F_p$, such

that $\Delta\mu_{pf} > 0$, go to (b); else go to (h).

e) Given p, make sure that $(\sum_{p'\in P_f}\mu_{p'f}) + \Delta\mu_{pf} \le G_f^{O}$ for each $f \in F_p$

If $(\sum_{p'\in P_f}\mu_{p'f}) + \Delta\mu_{pf} \le G_f^{O}$ for all $f \in F_p$, go to (e); otherwise, for those $f \in F_p$ such

that $(\sum_{p'\in P_f}\mu_{p'f}) + \Delta\mu_{pf} > G_f^{O}$; calculate ratio r_1 such that $(\sum_{p'\in P_f}\mu_{p'f}) + r_1\Delta\mu_{pf} = G_f^{O}$,

where $r_1 = (G_f^{O} - \sum_{p\in P_f}\mu_{pf})/\sum_{p\in P_f}\Delta\mu_{pf}|_{>0}$; update the value of $\Delta\mu_{pf}$, $\Delta\mu_{pf} \leftarrow r_1\Delta\mu_{pf}$.

f) Given p, use $\Delta\mu_{pf}$ to update $\Delta\upsilon_{pf}^{f't'}$ for each $f \in F_p$

If $(\sum_{f"t"}\upsilon_{pf}^{f"t"}) + \Delta\upsilon_{pf}^{f't'} > (u_{pf} + \Delta\mu_{pf}) + G_{pf}^{A}$, calculate r_2 such that $(\sum_{f"t"}\upsilon_{pf}^{f"t"}) +$

$r_2\Delta\upsilon_{pf}^{f't'} = (u_{pf} + \Delta\mu_{pf}) + G_{pf}^{A}$, where $r_2 = ((u_{pf} + \Delta\mu_{pf}) + G_{pf}^{A} - (\sum_{f"t"}\upsilon_{pf}^{f"t"}))/\Delta\upsilon_{pf}^{f't'}$; re-

set the value of $\Delta\upsilon_{pf}^{f't'}$ as $\Delta\upsilon_{pf}^{f't'} \leftarrow r_2 \cdot \Delta\upsilon_{pf}^{f't'}$.

g) Given p, balance $\Delta\upsilon_{pf}^{f't'}$ for $f \in F_p$

Suppose n represents the number of $\Delta\upsilon_{pf}^{f't'}$ changed simultaneously in (c), all these n

$\Delta\upsilon_{pf}^{f't'}$ among $f \in F_p$ should change consistently. If one of them becomes smaller, all

others should become smaller too. Reset these n $\Delta\upsilon_{pf}^{f't'}$ by using the smallest $\Delta\upsilon_{pf}^{f't'}$

among them.

h) Given p, update $\upsilon_{pf}^{f't'}$: $\upsilon_{pf}^{f't'} \leftarrow \upsilon_{pf}^{f't'} + \Delta\upsilon_{pf}^{f't'}$ for $f \in F_p$.

i) Given p, update $\Delta\mu_{pf} = \max\{(\sum_{f''t''} \upsilon_{pf}^{f''t''}) + \Delta\upsilon_{pf}^{f't'} - \mu_{pf} - G_{pf}^A, 0\}$ and update μ_{pf}:

$\mu_{pf} \leftarrow \mu_{pf} + \Delta\mu_{pf}$ for $f \in F_p$.

6.3.3. Dual-ascent algorithm two (DAA2)

As we discuss in subsection 6.3.1, increasing the value of $\upsilon_{pf}^{f't'}$ can increase the length of the shortest paths and improve the objective function value. Meanwhile, decreasing the value of $\upsilon_{pf}^{f't'}$ may not affect the length of the shortest paths (i.e., the objective function value). The change of $\upsilon_{pf}^{f't'}$ leads to a change of μ_{pf} because of constraints (6-16) (i.e.,

$\sum_{f't'} \upsilon_{pf}^{f't'} = \mu_{pf} + G_{pf}^A$). And the changes of μ_{pf} variables must satisfy constraints (6-15)

(i.e., $\sum_p \mu_{pf} - G_f^O \leq 0$).

Figure 11 describes how to improve the objective function value without violate (6-15). The solid line represents the fact that increasing μ_{pf} can increase the length of the shortest path (i.e., the objective function value). The broken line represents that de-creasing μ_{pf} will not affect the objective function value. At node f_2, we can improve the objective function value by increasing the value of $\mu_{p_1 f_2}$, and decreasing the values of $\mu_{p_2 f_2}$, $\mu_{p_3 f_2}$, and $\mu_{p_4 f_2}$, so that $\sum_{p \in P_{f_2}} \mu_{pf} - G_{f_2}^O \leq 0$ is not violated; this does not affect the objective value. DAA2 works to update the value of $\upsilon_{pf}^{f't'}$ such that the values of μ_{pf} reach a balanced status; that is, μ_{pf} can not be increased or decreased further to improve the objective value. We describe DAA2 as follows.
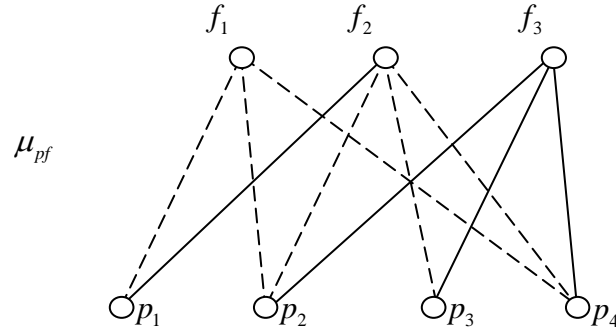
Figure 11. Balance status

Step 0: Initialization

Set values of variables $\mu_{pf} = G_f^O / |P_f|$ and $\upsilon_{pf}^{f't'} = (G_f^O / |P_f| + G_{pf}^A) \cdot (D_{ef't'} / \sum_{f't'} D_{ef't'})$.

Step 1: Update labels $\overline{\omega}_{pft}^{f't'}$ and $\underline{\omega}_{pft}^{f't'}$ at each node of each material network for

$D_{ef't'}$

Using shortest path algorithms to update two labels at each node, representing the

lengths of the shortest paths from the current node to the source and sink nodes, respec-

tively, in all material networks associated with $D_{ef't'}$, $\forall e \in E, t' \in T$, $f' \in F_e : D_{ef't'} > 0$.

(Same as DAA1).

Step 2: Update values of all μ_{pf} and $\upsilon_{pf}^{f't'}$ according to labels $\overline{\omega}_{pft}^{f't'}$ and $\underline{\omega}_{pft}^{f't'}$

For each material flow network associated with $D_{ef't'}$, first determine the duality gap for

each node and set the value of $\Delta \upsilon_{pf}^{f't'}$ based on $\overline{\omega}_{pft}^{f't'} - \underline{\omega}_{pft}^{f't'}$ (step 2-(a) in subsection

6.3.4); then update $\Delta\upsilon_{pf}^{f't'}$, considering the dependence among $p \in P_e$, and average

$\Delta\upsilon_{pf}^{f't'}$, $f \in F_p$ for given p, considering $\mu_{pf} - \Delta\mu_{pf} \geq 0$ (step 2-(b), (c), & (d) in subsec-

tion 6.3.4); next, sum $\Delta\upsilon_{pf}^{f't'}$ on f' and t' to set the value of $\Delta\mu_{pf}$ (i.e.,

$\sum_{f't'}\Delta\upsilon_{pf}^{f't'} = \Delta\mu_{pf}$) according to constraint (6-16) (step 2-(e) in subsection 6.3.4); after

that, update $\Delta\mu_{pf}$ for the balance (step 2-(f)&(g) in subsection 6.3.4); finally, according

to values $\Delta\mu_{pf}$, set $\Delta\upsilon_{pf}^{f't'}$, and update values of μ_{pf} and $\upsilon_{pf}^{f't'}$: $\mu_{pf} \leftarrow \mu_{pf} + \Delta\mu_{pf}$,

$\upsilon_{pf}^{f't'} \leftarrow \upsilon_{pf}^{f't'} + \Delta\upsilon_{pf}^{f't'}$ (step 2-(h)&(i) in subsection 6.3.4). See the following section for

details.

Stop if a balance is achieved for each p and f (i.e., all $\Delta\upsilon_{pf}^{f't'} = 0$), or, if the ob-

jective value does not improve appreciably over the last ten iterations (i.e., all $\Delta\upsilon_{pf}^{f't'}$'s

are small enough); otherwise, go to Step 1.

6.3.4. Implementation details for DAA2

Step 2: Update values of all μ_{pf} and $\upsilon_{pf}^{f't'}$ according to labels $\bar{\omega}_{pft}^{f't'}$ and $\underline{\omega}_{pft}^{f't'}$

Calculate all $\Delta\upsilon_{pf}^{f't'}$, $p \in P_e, f \in F_p$, using all $\bar{\omega}_{pft}^{f't'}$ and $\underline{\omega}_{pft}^{f't'}$ associated with $D_{ef't'}$

a) Determine the duality gap for the material flow network associated with $D_{ef't'}$, as

DAA1 does.

b) Update $\Delta\upsilon_{pf}^{f't'}$ considering the dependence among $p \in P_e$, as DAA1 does.

For each $p \in P_e, f \in F_p$, make sure that $\mu_{pf} - \Delta\mu_{pf} \geq 0$

c) Set $\Delta\mu_{pf} = \min(\mu_{pf}, \sum_{f't'}\Delta\upsilon_{pf}^{f't'})$ in order to make sure that $\mu_{pf} - \Delta\mu_{pf} \geq 0$.

If $\Delta\mu_{pf} = \mu_{pf}$, we must satisfy that $r_3 \sum_{f't'} \Delta\upsilon_{pf}^{f't'}|_{\geq 0} + \sum_{f't'} \Delta\upsilon_{pf}^{f't'}|_{<0} = \mu_{pf}$, where $r_3 =$

$(\mu_{pf} - \sum_{f't'} \Delta\upsilon_{pf}^{f't'}|_{<0}) / \sum_{f't'} \Delta\upsilon_{pf}^{f't'}|_{\geq 0}$. Update $\Delta\upsilon_{pf}^{f't'}$, which is negative: $\Delta\upsilon_{pf}^{f't'} \leftarrow$

$r_3 \Delta\upsilon_{pf}^{f't'}$.

Given $p \in P_e$, average all $\Delta\upsilon_{pf}^{f't'}$ for $f \in F_p$

d) $\Delta\upsilon_{pf}^{f't'} = \Delta\upsilon_{p}^{f't'} - \Delta\upsilon_{pf}^{f't'}$ with $\Delta\upsilon_{p}^{f't'} \equiv \sum_{f \in F_p} \Delta\upsilon_{pf}^{f't'} / |F_p|$, such that $\sum_{f \in F_p} \Delta\upsilon_{pf}^{f't'} = 0$.

If $\Delta\upsilon_{pf}^{f't'} \geq 0$, increase $\upsilon_{pf}^{f't'}$ to make the shortest path longer.

If $\Delta\upsilon_{pf}^{f't'} < 0$, reduce $\upsilon_{pf}^{f't'}$; this will not change the length of the shortest path.

For each $p \in P_e, f \in F_p$, calculate $\Delta\mu_{pf}$

e) Calculate $\Delta\mu_{pf} = \sum_{f't'} \Delta\upsilon_{pf}^{f't'}$; note $\sum_{f \in F_p} \Delta\mu_{pf} = \sum_{f't'} \sum_{f \in F_p} \Delta\upsilon_{pf}^{f't'} = 0$.

We can guarantee that $\mu_{pf} + \Delta\mu_{pf} \geq 0$ because of (c).

Check balances for each $p \in P_e, f \in F_p$

f) Balance at p

If $\Delta\mu_{pf} \geq 0$, increase μ_{pf}. (Tight status)

If $\Delta\mu_{pf} < 0$, decrease μ_{pf} if possible. (Loose status)

Given p for at least one $f \in F_p$, $\Delta\mu_{pf} \geq 0$; and for others, $\Delta\mu_{pf} < 0$ (note: p must be

processed at one $f \in F_p$ for which $\Delta\mu_{pf} \geq 0$).

Given p, if for all $f \in F_p$, $\Delta\mu_{pf} = 0$, we have reached balance for this p.

g) Balance at f: $\sum_f \sum_p \Delta\mu_{pf} = \sum_p \sum_f \Delta\mu_{pf} = 0$

Case 1: $\sum_{p\in P_f}\Delta\mu_{pf}=0$, balance; use $\Delta\mu_{pf}$ to update μ_{pf}: $\mu_{pf}\leftarrow\mu_{pf}+\Delta\mu_{pf}$;

Case 2: $\sum_{p\in P_f}\Delta\mu_{pf}<0$ (loose status), use $\Delta\mu_{pf}$ to update μ_{pf}: $\mu_{pf}\leftarrow\mu_{pf}+\Delta\mu_{pf}$;

Case 3: $\sum_{p\in P_f}\Delta\mu_{pf}>0$ (tight status), if $\sum_{p\in P_f}(\mu_{pf}+\Delta\mu_{pf})\geq G_f^O$, update $\Delta\mu_{pf}$: $\Delta\mu_{pf}$

$\leftarrow r_4\Delta\mu_{pf}$ using ratio r_4 (if $\Delta\mu_{pf}>0$) such that $\sum_{p\in P_f}(\mu_{pf}+\Delta\mu_{pf})=G_f^O$, where

$r_4=(G_f^O-\sum_{p\in P_f}\mu_{pf}-\sum_{p\in P_f}\Delta\mu_{pf}|_{\leq 0})/\sum_{p\in P_f}\Delta\mu_{pf}|_{>0}$; use $\Delta\mu_{pf}$ to update μ_{pf}: μ_{pf}

$\leftarrow\mu_{pf}+\Delta\mu_{pf}$.

h) Given $\Delta\mu_{pf}$, update $\Delta\upsilon_{pf}^{f't'}$ to satisfy that $\Delta\mu_{pf}=\sum_{f't'}\Delta\upsilon_{pf}^{f't'}$

After the updation in (f) and (g), it is not always true that $\Delta\mu_{pf}=\sum_{f't'}\Delta\upsilon_{pf}^{f't'}$. If it is not,

update $\Delta\upsilon_{pf}^{f't'}$: $\Delta\upsilon_{pf}^{f't'}\leftarrow r_5\Delta\upsilon_{pf}^{f't'}$, where $r_5=(\mu_{pf}-\sum_{f't'}\Delta\upsilon_{pf}^{f't'}|_{<0})/\sum_{f't'}\Delta\upsilon_{pf}^{f't'}|_{\geq 0}$ so that

$\Delta\mu_{pf}=\sum_{f't'}\Delta\upsilon_{pf}^{f't'}$.

i) Given p, update $\upsilon_{pf}^{f't'}$ and μ_{pf}: $\upsilon_{pf}^{f't'}\leftarrow\upsilon_{pf}^{f't'}+\Delta\upsilon_{pf}^{f't'}$ and $\mu_{pf}\leftarrow\mu_{pf}+\Delta\mu_{pf}$ for

$f\in F_p$

6.4. Primal Drop Heuristic

The objective in constructing a primal solution is to minimize the gap between it and a

known dual solution. Because the gap can be measured as the total violation of comple-

mentary slackness (Gao and Robinson, 1996), the constructed primal solution should

violate as few as possible of the complementary slackness conditions associated with the

current dual solution. Intuitively, the meaning of complementary slackness in our appli-

cation can be explained as follows. If the value of dual variable $\upsilon_{pf}^{f't'}$ is zero, we know that no arcs associated with this dual variable will remain on any shortest path (otherwise, if its value is changed, the value of this dual variable can be increased to improve the objective function value). If no flows of component p go through facility alternative f, y_{pf} can be set to zero in the primal solution. Based on this observation, it seems that the condition to decide whether y_{pf} is set to zero or one can depend on whether the material flows go through those arcs associated with $\upsilon_{pf}^{f't'}$. We can use the solution from the dual-ascent algorithm to obtain this information.

Gao and Robinson (1996) discussed a *drop heuristic* (DH) applied to a two-echelon location problem. We use this idea to our application. According to the values of dual variables $\upsilon_{pf}^{f't'}$, we can fix a set of y_{pf} variables to zero. And x_f can be fixed to zero if all variables y_{pf}, $p \in P_f$, are fixed to zero. Remaining x_f and y_{pf} variables will be set to one temporarily. We then solve a set of SP(e)s to obtain values of z_a variables based on the temporary setting of x_f and y_{pf} to obtain a primal objective function value. DH processes x_f and y_{pf} variables that have been given the temporary value of one (i.e., *drops* the value from one to zero), one by one to improve the objective function value. That is, if dropping x_f or y_{pf} can improve the objective function value, we fix it to zero. Please note that each time one x_f or y_{pf} is dropped, we must solve a set of SP(e)s to obtain corresponding values of z_a and the objective function. If SP(e) happens to be infeasible, the corresponding objective function value is set as $+\infty$ (a very

high total costs). The drop procedure stops when the objective function value can not be improved further by dropping an x_f or a y_{pf} variable. Thus, all x_f and y_{pf}, which are not fixed to zero, will be set to one permanently. We now detail DH.

This heuristic can be viewed as a greedy algorithm. It starts with an initial feasible primal solution, then greedily fixes a y_{pf} or an x_f to zero to obtain the largest possible improvement of the objective function value. It stops if the objective function value can not be improved by dropping a y_{pf} or an x_f.

Main Procedure:

Step 0: Check y_{pf} associated with arcs on the shortest paths in the material flow network.

For each pf, calculate $\min_{f't'}\{\min_{t\in T}\{(\overline{\omega}_{pft}^{f't'} - \underline{\omega}_{pft}^{f't'})D_{ef't'}\}\}$ based on the values of $\overline{\omega}_{pft}^{f't'}$, $\underline{\omega}_{pft}^{f't'}$ at the last iteration of the dual-ascent algorithm. If $\min_{f't'}\{\min_{t\in T}\{(\overline{\omega}_{pft}^{f't'} - \underline{\omega}_{pft}^{f't'})D_{ef't'}\}\} > 0$ (i.e., no arcs $a \in \overline{A}_{pf}$ are in the shortest paths), fix $y_{pf} = 0$. The meaning is that since no material flow goes through nodes with indices pf because they are not on the shortest path, p is not processed at f (i.e., $y_{pf} = 0$). Fix $x_f = 0$ if all variables y_{pf} associated with x_f (i.e., $p \in P_f$) are fixed to zero.

Step 1: Construct a primal solution based on the fixing in step 0.

Step 0 fixes certain y_{pf} and x_f variables to zero; set all other y_{pf} and x_f variables to one temporarily. Based on these values, calculate the flow variable values and obtain the current objective value.

For each p, arcs associated with all $f \in F_p$ form a valid arc cut. Component p must be

supplied by at least one of the facility $f \in F_p$ and any shortest path must contain these

arcs (i.e., $a \in \bar{A}_{pf}$). That is, for at least one pf for $f \in F_p$,

$\min_{f't'}\{\min_{t\in T}\{(\bar{\omega}_{pft}^{f't'} - \underline{\omega}_{pft}^{f't'})D_{ef't'}\}\} = 0$. Thus, for each p, at least one facility alterna-

tive $f \in F_p$ will be opened. Thus, it is guaranteed that the initial solution constructed by

DH is feasible.

Step 2: Find a y_{pf} variable to drop (determine if its value can be flipped to zero).

For each y_{pf} with value one, calculate the difference between the objective function

value with $y_{pf} = 1$, $Z_{PD}^*(y_{pf}=1)$, and the objective function value with $y_{pf} = 0$,

$Z_{PD}^*(y_{pf}=0)$. This difference represents the improvement that the objective function

value would realize if y_{pf} were fixed to zero. If fixing $y_{pf} = 0$ would lead to an infeasi-

ble primal solution, fix this $y_{pf} = 1$ and the corresponding $x_f = 1$ permanently; or, if the

difference is positive (i.e., $Z_{PD}^*(y_{pf}=1) - Z_{PD}^*(y_{pf}=0)>0$), record the improvement.

Among all y_{pf} variables with temporary value one, find y_{pf} with $pf = $

$\arg\max\{Z_{PD}^*(y_{pf}=1) - Z_{PD}^*(y_{pf}=0)>0\}$ (i.e., the y_{pf} with the largest improvement).

This step is to find a pair pf such that p will not be processed at f (i.e., $y_{pf} = 0$) to

achieve the largest possible reduction of the total costs.

Step 3: Find x_f variable to drop.

For each x_f with value one, calculate the difference between the objective function val-

ue with $x_f = 1$, $Z^*_{PD}(x_f=1)$, and the objective function value with $x_f = 0$, $Z^*_{PD}(x_f=0)$.

The intuitive meaning of this difference is the same as that in step 2. If fixing $x_f = 0$ would lead to an infeasible primal solution, fix this $x_f = 1$ permanently; or, if the difference is positive ((i.e., $Z^*_{PD}(x_f=1) - Z^*_{PD}(x_f=0) > 0$), record the improvement.

Among all x_f variables with a temporary value of one, find x_f with $f = \arg\max\{Z^*_{PD}(x_f=1) - Z^*_{PD}(x_f=0) > 0\}$ (i.e., the x_f that gives the largest improvement in objective function value).

This step finds which facility f can be closed to obtain the largest reduction in the total costs.

Step 4: Stop criteria.

If neither step 2 nor 3 finds an improvement, stop; otherwise, compare the improvements from steps 2 and 3 and identify the better one. Fix the corresponding y_{pf} or x_f to zero, then go to step 1.

If we can find a y_{pf} in step 2 or an x_f in step 3, we select the one with larger reduction in cost. Otherwise, we can not improve the current objective function value further by simply flipping (or dropping) one y_{pf} or one x_f to zero.

6.5. Computational Results

The objectives of our tests are to investigate the solution qualities (i.e., the gap between primal and dual solutions) and run times for DAA1 (DAA2) each used in combination

with DH (i.e., DAA1-DH and DAA2-DH). We conduct all experiments on a PC with a

2.8 GHz Pentium IV processor and 1 GB of RAM using Visual Studio C++ 6.0.

We use four factors to describe each case, the number of time periods, $|T|$ (10 or

20); the number of end products, $|E|$ (4 or 8); the number of components in the BOM

for end products, $BOMS$ (7, 15, or 31); and the number of facility alternatives for each

component, FA (4 or 8). We also assume that the BOM is a binary tree. Columns 2-5 of

Table 10 detail the level of each factor associated with each case. Specifying one level of

each of the four factors defines a case and determines the number of variables and con-

straints in model ($\mathcal{PD}$).

Table 10. Factor levels and problem size of each test set

Scenario	$|T|$	$|E|$	$BOMS$	FA	BIN	CONT	ROW
1	10	4	7	4	146	277560	66072
2	20	4	7	4	146	1123760	262672
3	10	4	15	4	294	564760	137224
4	20	4	15	4	294	2282560	543344
5	10	4	15	8	587	3916560	558848
6	20	4	15	8	587	15791680	2235488
7	10	4	31	8	1176	7877520	1151776
8	20	8	31	8	2344	63514080	9237216

Column 6-8 of Table 10 give the number of binary variables (BIN), number of

continuous variables (CONT), and number of constraints (ROW) for each case.

CONT=$|E| \cdot BOMS \cdot FA^3 \cdot |T|^2$ so it is a function of all factors. For example, model ($\mathcal{PD}$)

for a system that includes 8 end products, 32 components for each end product, 8 facility

alternatives for each component, and 20 time periods, includes more than 60 million

$(8 \cdot 32 \cdot 8^3 \cdot 20^2)$ continuous variables. Since optimizing a mixed- integer-program including so many continuous variables could be expected to be quite challenging, investigating DAA1-DH and DAA2-DH is reasonable.

We design eight cases based on real-world considerations, including prices in different locations, facility alternatives, processing times for components, transport times, and demands, which are related to the population. Each facility alternative is defined by its location and the prices of labor, land, and capital required to open it. Component processing times determine workloads for producing, assembling, stocking, and transporting. For each case, we generate eight instances, which all have the same factor-level selection but each instance has a unique set of random number seeds to generate parameters and sets (e.g., G_f^O, P, etc.) used in the model.

We analyze run time and solution quality in terms of the gap between primal and dual solutions for DAA1-DH and DAA2-DH, $GAP(P-D)$,

$$GAP(P-D) = (100)(Z_{DDA\#-DH}^* - Z_{DDA\#}^*) / Z_{DDA\#-DH}^* \qquad (6\text{-}21).$$

DDA# represents DDA1 or DDA2. $Z_{DDA\#-DH}^*$ is the objective function value of the primal solution obtained by DDA1-DH or DDA2-DH. And $Z_{DDA\#}^*$ is the objective function value of dual solution obtained by DAA1 or DAA2. The initial primal solution in Step 0 of DH depends on the dual values at the last iteration of the dual-ascent algorithm. Thus, different dual-ascent algorithms may lead DH to prescribe different primal solutions.

Table 11 (Table 12) records measures of the run times for solving instances in each case, including the range and average of the run times for DAA1, DAA2, DAA1-

DH, and DAA2-DH. DAA1-DH is not competitive with respect to run time; it requires a run time that increases dramatically with instance size because it invokes steps 1 and 2 many times to update values of dual variables $\upsilon_{pf}^{f't'}$ and μ_{pf}. In comparison, DAA2-DH is much faster than DAA1-DH. Figure 12 plots the average run times of DAA2 and for DH in combination with DAA2 (i.e, DAA2-DH) versus the number of continuous variables; it also includes corresponding regression models. Run times for DAA2 and DH are approximately linear with the number of continuous variables. This is reasonable because the main part of each heuristic involves obtaining forward and backward labels for all nodes in the network, which requires step 2 to process all arcs in the network. On average, DAA2 and DH need around 150 seconds to solve the largest of our test in stances, which involve more than 60 million continuous variables. Considering these results and the resulting linear regression model, DAA2-DH could be used to solve larger instances in actual applications.

Table 11. Run times for DAA1 and DH

| Scenario	DAA1		DH	
	Range (seconds)	Average (seconds)	Range (seconds)	Average (seconds)
1	2.52 - 5.55	3.36	0.13 - 0.73	0.26
2	1.77 - 19.13	11.50	0.39 - 3.27	1.20
3	0.72 - 21.05	9.15	0.25 - 1.09	0.66
4	3.16 - 93.03	31.02	1.33 - 11.16	3.81
5	69.77 - 193.83	101.61	3.91 - 52.77	19.45
6	62.64 - 1045.33	624.40	32.28 - 278.56	170.94
7	311.67 - 944.36	782.85	45.41 - 145.88	80.68
8	272.86 - 8939.75	4483.95	139.92 - 1520.02	591.45

Table 12. Run times for DAA2 and DH

Scenario	DAA2		DH	
	Range (seconds)	Average (seconds)	Range (seconds)	Average (seconds)
1	0.31 - 0.61	0.45	0.16 - 0.45	0.24
2	1.06 - 2.95	1.87	0.25 - 2.23	0.79
3	0.44 - 2.80	1.22	0.25 - 1.36	0.76
4	2.45 - 8.50	3.93	1.73 - 11.50	4.18
5	5.08 - 9.84	7.17	6.17 - 43.44	14.41
6	31.61 - 47.28	38.94	22.11 - 312.34	119.11
7	12.86 - 28.66	19.03	22.69 - 67.56	46.02
8	123.59 - 174.22	147.80	140.66 - 1162.47	417.20

DAA1 and DAA2 each provides a dual feasible solution, while DH provides a feasible primal solution as well as an estimate on its quality (i.e., the primal-dual gap). Table 13 gives the gap between the primal and dual solutions prescribed by DAA1-DH and DAA2-DH, reporting the range and average of the gap (defined as (6-21)) over all instances associated with each case.

Table 13. Gaps for DAA1-DH and DAA2-DH

Scenario	DAA1-DH $GAP(P-D)$		DAA2-DH $GAP(P-D)$	
	Range	Average	Range	Average
1	0.01% - 0.12%	0.06%	0.02% - 0.14%	0.08%
2	0.01% - 0.06%	0.04%	0.01% - 0.07%	0.04%
3	0.00% - 0.06%	0.02%	0.00% - 0.06%	0.04%
4	0.00% - 0.06%	0.02%	0.00% - 0.05%	0.03%
5	0.02% - 0.09%	0.04%	0.02% - 0.11%	0.05%
6	0.02% - 0.04%	0.03%	0.01% - 0.11%	0.07%
7	0.02% - 0.05%	0.04%	0.02% - 0.09%	0.04%
8	0.00% - 0.02%	0.01%	0.00% - 0.05%	0.02%

Overall, DAA1-DH provides a small gap than DAA2-DH. Both DAA1 and DAA2 work well in combination with DH and both combinations prescribe primal solutions within a 0.08% gap. The worst average gap in our tests is only 0.15%.
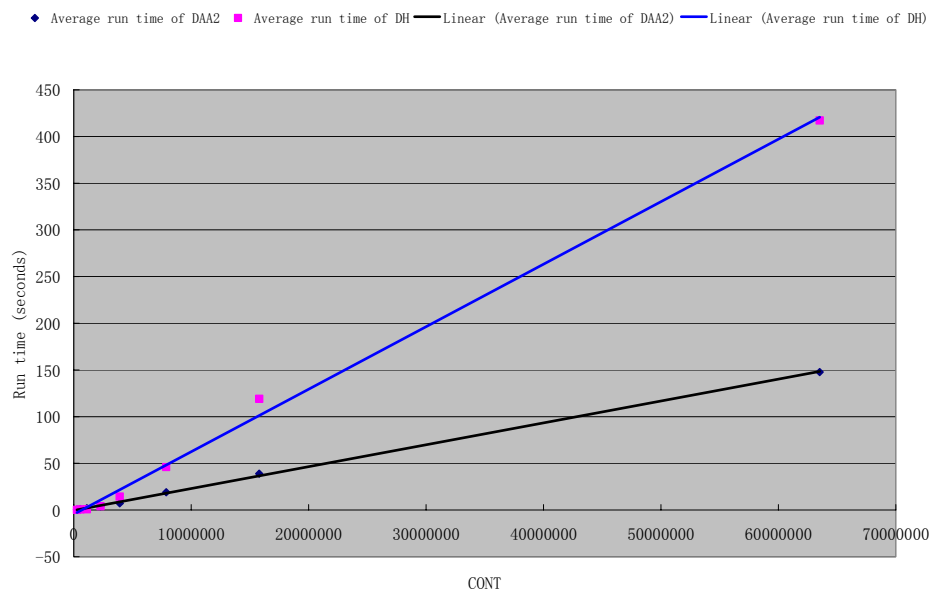
Figure 12. Run time vs number of continuous variables

Table 14 gives measures (range and average) of the run times CPLEX required to solve cases 1-4 of the mixed-integer program model ($\mathcal{PD}$). Here, CPLEX solves all cases to optimality rather than solving the linear relaxation of ($\mathcal{PD}$). CPLEX can not solve cases 5-8 because they require prohibitive amounts of storage capacity. It is obvious that CPLEX takes longer to solve these cases than DAA2-DH (see in Table 12)). Table 14 also presents, for each case, the gap between the optimal primal solution ob-

tained using CPLEX and the corresponding primal solution obtained using DAA2-DH, which is defined as

$$GAP(P-O) = (100)(Z^*_{DDA2-DH} - Z^*_{PD}) / Z^*_{PD} \qquad (6\text{-}22).$$

Because the optimal objective function value is within the range of the primal and dual bound, the gap defined in (6-22) is always smaller than the one in (6-21). Gaps reported in Table 14 are fairly small. The average gap is 0.01% and the largest gap is 0.04%. It seems that the DAA2-DH is a reasonable solution approach, which gives a favorable trade-off between run time and solution quality.

Table 14. Run time for CPLEX and the gap between heuristic and optima solutions

Scenario	Run Times		$GAP(P-O)$	
	Range (seconds)	Average (seconds)	Range	Average
1	3.11 - 4.08	3.48	0.00% - 0.04%	0.01%
2	17.45 - 23.91	19.70	0.00% - 0.03%	0.01%
3	7.81 - 9.72	8.68	0.00% - 0.03%	0.01%
4	44.11 - 63.06	53.95	0.00% - 0.03%	0.01%

6.6. Summary

This section presents two dual-ascent algorithms and uses each in combination with a primal drop heuristic to solve the uncapacitated production-assembly-distribution system design problem (UPADSDP). Tests show that the resulting heuristic (DAA2-DH) is effective. It can solve instances that include more than one thousand binary and sixty million continuous variables within twenty minutes. For all instances, the gaps between the

primal and dual solution are less than 0.15%. Thus, the proposed method can be used to solve large-scale instances within reasonable time.

7. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

This paper mainly discussed the model and solution approach to PADSDP. It achieves its objective. Relative to section 3, we formulate the design of the production-assembly-distribution system as a mixed integer program, which is challenging to solve. We employ CG to solve PADSDP. Relative to section 4, we compare five decomposition schemes and two acceleration techniques in application to PADSDP. We identify the impacts of various measures associated with schemes and propose conjectures that lend insights into the rate of convergence. Computational tests provide empirical evidence that supports our qualitative analyses. Relative to section 5, we extended one of the acceleration techniques to a more general concept to improve the rate of convergence of CG. We established a theoretical foundation and provided geometric insights through a small example. Finally, Relative to section 6, we discussed the dual-ascent and primal heuristics applied to UPADSDP. Computational results indicate that this heuristic can find solutions with good quality and within reasonable time, the efficiency and large-scale suiting it well for actual applications.

Relative to section 4, future research can be directed to devising special techniques for implementing DWD in the B&P framework, e.g., by specifying i) branching rules; ii) methods to fix variables; iii) techniques to re-optimize nodes in the B&B tree; and iv) steps to initialize. All of these issues are also affected by the decomposition schemes. The solution approach to PADSDP here can be extended to deal with PADSDPN by Wilhelm et al. (2005). The extended model would include additional con-

straints and variables to represent the international business environment but does not affect the fundamental decomposition schemes.

Relative to section 5, future research can contribute by devising implementation techniques to apply DWDG to linear programs using different strategies for managing the numbers of variables and constraints to quantify the trade off between problem formulation and the run time needed to prescribe an optimal solution. In addition, future research could seek methods to assure that DWDG will prescribe tight bounds, allowing effective solution of integer programs consistently. For example, methods to select a subset of $\{y_l \mid L\}$ may achieve a trade off between accelerating convergence and tightening bounds. Further, the set $\{y_l \mid L\}$ can be determined dynamically at each node of the B&B tree, depending upon the variables that are fixed. Finally, computational tests could evaluate the efficacy of DWDG in a broader range of applications.

Relative to section 6, future research to extend on i) DAA2-DH can be embedded in a B&B framework to obtain tighter bounds and to fix binary variables at each node in the B&B tree; ii) DAA2-DH can be extended to deal with capacitated PADSDP.

REFERENCES

Aardal, K. (1998) Capacitated facility location: separation algorithms and computational experience, *Mathematical Programming*, **81**, 149-175.

Aardal, K., Pochet, Y. and Wolsey, L. A. (1995) Capacitated facility location: valid inequalities and facets, *Mathematical Methods of Operations Research*, **20**, 562-582.

Ahuja, R. K. (1997) *Flows and Paths, Annotated Bibliographies in Combinatorial Optimization*, John Wiley & Sons, New York, 283-309.

Ahuja, R. K., Magnanti, T. L. and Orlin, J. B. (1993) *Network Flows: Theory, Algorithms and Applications*, Prentice-Hall, Englewood Cliffs, NJ.

Aikens, C. H. (1985) Facility location models for distribution planning, *European Journal of Operational Research*, **22**, 263-279.

Akinc, U. and Khumawala, B. M. (1977) An efficient branch-and-bound algorithm for the capacitated warehouse location problems, *Management Science*, **23**, 585-594.

Alonso-Ayuso, A., Escudero, L. F., Garin, A., Ortuno, M. T. and Perez, G. (2003) An approach for strategic supply planning under uncertainty based on stochastic 0-1 programming, *Journal of Global Optimization*, **26**, 97-124.

Alvarez, A. M., Gonzalez-Velarde, J. L. and De-Alba, K. (2005) GRASP embedded scatter search for the multicommodity capacitated network design problem, *Journal of Heuristics*, **11**, 233-257.

Alvelos, F. and Valerio de Carvalho, J. M. (2007) An extended model and a column generation algorithm for the planar multicommodity flow problem, *Networks*, **50**(1), 3-16.

Ambrosino, D. and Scutella, M. G. (2005) Distribution network design: New problems and related models, *European Journal of Operational Research*, **165**, 610-624.

Appelgren, L. H. (1969) A column generation algorithm for a ship scheduling problem, *Transportation Science*, **3**, 53-68.

Assad, A. A. (1978) Multicommodity network flows – a survey, *Networks*, **8**, 37-91.

Balakrishnan, A., Magnanti, T. L. and Mirchandani, P. (1994) A dual-based algorithm for multi-level network design, *Management Science*, **40**(5), 567-581.

Balakrishnan, A., Magnanti, T. L. and Wong, R. T. (1989) A dual-ascent procedure for large-scale uncapacitated network design, *Operations Research*, **37**(5), 716-740.

Barahona, F. and Jensen, D. (1998) Plant location with minimum inventory, *Mathematical Programming*, **83**, 101-111.

Barnhart, C., Hane, C. A., Johnson, E. L. and Sigismondi, G. (1995) A column generation and partitioning approach for multicommodity flow problems, *Telecommunication Systems*, **3**, 239-258.

Barnhart, C., Hane, C. A. and Vance, P. H. (1997) Integer multicommodity flow problems, *Lecture Notes in Economics and Mathematical Systems*, **450**, 17-31.

Barnhart, C. and Sheffi, Y. (1993) A network-based primal-dual heuristic for the solution of multi-commodity network flow problems, *Transportation Science*, **27**(2), 102-117.

Barnhart, C. (1993) Dual-ascent methods for large-scale multicommodity flow problems, *Naval Research Logistics*, 40, 305-324.

Bartmess, A. and Cerny, K. (1993) Building competitive advantage through a global network of capabilities, *California Management Review* (Winter), 78-103.

Bazaraa, M.D., Jarvis, J. J. and Sherali, H. D. (2005) *Linear Programming and Network Flows*. 3rd Ed., Wiley, New York.

Ben Amor, H., Desrosiers, J. and Frangioni, A. (2004) Stabilization in column generation, *Les Cahiers du GERAD*, G–2004–62.

Ben Amor, H., Desrosiers, J. and Valerio de Carvalho, J. M. (2006) Dual-optimal inequalities for stabilized column generation, *Operations Research*, **54**, 454-463.

Ben Amor, H. and Valerio de Carvalho, J. M. (2005) *Cutting Stock Problems, Column Generation*, Desaulniers, G., Desrosiers, J., Solomon, M. M. (Editors) Kluwer, Boston, 131-161.

Bumb, A. and Kern, W. (2001) A simple dual ascent algorithm for the multilevel facility location problem, approximation, randomization, and combinatorial optimization: Algorithms and Techniques *Lecture Notes in Computer Science*, **2129**, 55-62.

Cambini, R., Gallo, G. and Scutella, M. G. (1997) Flows on hypergraph, *Mathematical Programming*, **78**, 195-217.

Cappanera, P. and Frangioni, A. (2003) Symmetric and asymmetric parallelization of a cost-decompostion algorithm for multicommodity flow problems, *INFORMS Journal on Computing*, **15**(4), 369-384.

Chabrier, A., Danna, E., Pape, C. L. and Perron, L. (2004) Solving a network design problem, *Annals of Operations Research*, **130**, 217-239.

Christofides, N. and Beasley, J. E. (1983) Extensions to a Lagrangian relaxation approach for the capacitated warehouse location problem, *European Journal of Operational Research*, **12**, 19-28.

Cho, D. C., Johnson, E. L., Padberg, M. W. and Rao, M. R. (1983) On the uncapacitated plant location problem I: valid inequalities and facets, *Mathematical Methods of Operations Research*, **8**, 579-589.

Cho, D. C., Padberg, M. W. and Rao, M. R. (1983) On the uncapacitated plant location problem II: facets and lifting theorems, *Mathematical Methods of Operations Research*, **8**, 590-612.

Clarke, L. W. and Gong, P. (1996) Capacitated network design with column generation, ISyE published research papers, Georgia Institute of Technology, http://crier.isye.gatech.edu/apps/research-papers/papers/lec9606.pdf.

Cornuejols, G. (2001) Combinatorial Optimization–Packing and Covering, SIAM, Philadelphia.

Costa, A. M. (2005) A survey on Benders decomposition applied to fixed-charge network design problems, *Computers & Operations Research*, **32**, 1429-1450.

Crainic, T. G., Frangioni, A. and Gendron, B. (2001) Bundled-based relaxation methods for multicommodity capacitated fixed charge network design, *Discrete Applied Mathematics*, **112**, 73-99.

Crainic, T. G. and Delorme, L. (1993) Dual-ascent procedures for multicommodity location-allocation problems with balancing requirements, *Transportation Science*, **27**(2), 90-101.

Craninic, T. G., Gendreau, M. and Farvolden, J. M. (2000) A simplex-base tabu search method for capacitated network design, *INFORMS Journal on Computing*, **12**(3), 223-236.

Cohen, M. A. and Lee, H. L. (1988) Strategic analysis of integrated production-distribution systems, *Operations Research*, **36**, 216-228.

Cohen, M. A., Fisher, M. and Jaikumar, R. (1989) International manufacturing and distribution networks: a normative model framework, In: *Managing International Manufacturing*, Ferdow, K. (Editor), Elsevier Science Publishers, North Holland, 67-93.

Cournuejols, G., Nemhauser, G. L. and Wolsey, L. A. (1990) *The Uncapacitated Facility Location Problem, Discrete Location Theory*, Wiley-Interscience, New York.

Damiano, B. D. and Little, J. B. (1988) *A Course in Linear Algebra*, Harcourt Brace Jovanvich, San Diego.

Dantzig, G. B. and Wolfe, P. (1960) Decomposition principle for linear programs, *Operations Research*, **8**, 101-111.

Daskin, M. (1995) *Network and Discrete Location*, Wiley, New York.

Dasci, A. and Verter, V. (2001) The plant location and technology acquisition problem, *IIE Transactions*, **33**, 963-973.

De Leone, R., Gaudioso, M. and Monaco, M. F. (1993) Nonsmooth optimization methods for parallel decomposition of multicommodity flow problems, *Annuals of Operations Research*, **44**, 299-311.

Desrosier, J., Soumis, F. and Desrochers, M. (1984) Routing with time windows by column generation, *Networks*, **14**, 545-565.

Desaulniers, G., Desrosiers, J. and Solomon, M. M. (2001) Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems, In: *Essays and Surveys in Metaheuristics*, Ribeiro, C. C. and Hansen, P. (Editors), Kluwer, Boston, 309-324.

Desaulniers, G., Desrosiers, J. and Solomon, M. M. (Editors) (2005) *Column Generation*, Kluwer, Boston.

Drenzner, Z. (1995) *Facility Location*, Springer, Berlin.

Dudzinski, K. and Walukiewicz, S. (1987) Exact methods for knapsack problem and its generalizations, *European Journal of Operational Research*, **28**, 3-21.

Elhedhli, S. and Goffin, J. L. (2005) Efficient production-distribution system design, *Management Science*, **51**(7), 1151-1164.

Erlenkotter, D. (1978) A dual based procedure for uncapacitated facility location, *Operations Research*, **26**, 992-1009.

Ferris, M. C. and Horn, J. D. (1998) Partitioning mathematical problems for parallel solution, *Mathematical Programming*, **80**, 35-61.

Ford, L. R. and Fulkerson, D. R. (1958) A suggested computation for maximal multi-commodity network flows, *Management Science*, **5**, 97-101.

Fourer, R., Gay, D. M. and Kernighan, B. W. (2003) *AMPL: A Modeling Language for Mathematical Programming*, 2nd Edition. Pacific Grove, CA.

Frangioni, A. (2002) Generalized bundle methods, *SIAM Journal on Optimization*, **13**(1), 117-156.

Frangioni, A. and Gallo, G. (1999) A bundle type dual-ascent approach to linear mulit-commodity min-cost flow problems, *INFORMS Journal on Computing*, **11**(4), 370-393.

Gallo, G. and Pallottino, S. (1992) Hypergraph Models and Algorithms for the Assembly Problem, Technical Report TR-6/92, Dip. di Informatica, Univ.di Pisa.

Gao, L. L. and Robinson, E. P. (1992) A dual-based optimization procedure for the 2-echelon uncapacitated facility location problem, *Naval Research Logistics*, **39**(2), 191-212.

Gendron, B. (2002) A note on "A dual ascent approach to the fixed-charge capacitated network design problem", *European Journal of Operational Research*, **138**, 671-675.

Gendron, B., Crainic, T. G. and Frangioni, A. (1999) *Multicommodity Capacitated Network Design, Telecommunications Network Planning*, Soriano, P., Sansò, B. (Editor), Kluwer Academic Publisher, Boston, 1-19.

Gendron, B. and Crainic, T. G. (1994), Relaxation for Multicommodity Capacitated Network Design Problems, publication CRT-965, Centre de recherché sur les transports, University de Montreal.

Gendron, B. and Crainic, T. G. (1996) Bounding Procedures for Multicommodity Capacitated Fixed Charge Network Design Problems, publication CRT-96-06, Centre de recherché sur les transports, University de Montreal.

Geoffrion, A. M. (1974) Lagrangian relaxation for integer programming, *Mathematical Programming Study*, **2**, 82-114.

Geoffrion, A. M. and Graves, G. W. (1974) Multicommodity distribution system design by Benders decomposition, *Management Science*, **20**, 824-844.

Geunes, J. and Pardalos, P. M. (2003) Network optimization in supply chain management and financial engineering: an annotated bibliography, *Networks*, **42**(2), 66–84.

Gilmore, P. C. and Gomory, R. E. (1961) A linear programming approach to the cutting stock problem, *Operations Research*, **11**, 849-859.

Glover, F., Sherali, H. D. and Lee, Y. (1997) Generating cuts from surrogate constraint analysis for zero-one and multiple choice programming, *Computational Optimization and Applications*, **8**, 151-172.

Goffin, J. L., Gondzio, J., Sarkissian, R. and Vial, J. P. (1996) Solving nonlinear multicommodity flow problems by the analytic center cutting plane method, *Mathematical Programming*, **76**, 131-154.

Guignard, M. (1988) A Lagrangian dual ascent algorithm for simple plant location-problems, *European Journal of Operational Research*, **35**(2), 193-200.

Guignard, M. and Jonhnson, D. S. (1979) *Computers and intractability, A guide to the theory of NP-completeness*, Freeman, New York.

Guignard, M. and Opaswongkarn, K. (1990) Lagrangian dual ascent algorithms for computing bounds in capacitated plant location-problems, *European Journal of Operational Research*, **46**(1), 73-83.

Guignard, M. and Spielberg, K. (1979) A direct dual method for the mixed plant location problem with some side constraints, *Mathematical Programming*, **17**, 198-228.

Guignard, M. and Rosenwein, M. B. (1989) An application-oriented guide for designing Lagrangian dual ascent algorithms, *European Journal of Operational Research*, **43**(2), 197-205.

Hale, T. S. (2003) Location science research: A review, *Annals of Operations Research*, **123**, 21-35.

Herrmann, J. W., Ioannou, G., Minis, I. and Proth, J. M. (1996) A dual ascent approach to the fixed-charge capacitated network design problem, *European Journal of Operational Research*, **95**(3), 476-490.

Henningsson, M, Holmberg, K., Ronnqvist, M. and Varbrand, P. (2002) Ring network design by Lagrangean based column generation, *Telecommunication Systems*, **21**, 301-318.

Hinojosa, Y., Puerto, J. and Fernandez, F. R. (2000) A multiperiod two echelon multi-commodity capacitated plant location problem, *European Journal of Operational Research*, **123**, 271-291.

Hodder, J. E. and Dincer, M. C. (1986) A multi-factor model for international plant location and financing under uncertainty, *Computers & Operations Research*, **13**(5), 601-609.

Holmberg, K. (2001) Experiments with primal-dual decomposition and sub-gradient methods for the uncapacitated facility location problem, *Optimization*, **49**(5-6), 495-516.

Holmberg, K. and Yuan, D. (2003) A multicommodity network-flow problem with side constraints on paths solved by column generation, *INFORMS Journal on Computing*, **15**(1), 42-57.

Hu, J. and Johnson, L. E. (1999) Computational results with a primal-dual subproblem simplex method, *Operations Research Letters*, **25**, 149-157.

Huchzermeier, A. H. (1991) Global manufacturing strategic planning under exchange rate uncertainty, Ph. D. dissertation, University of Pennsylvania, Philadelphia.

Jayaraman, V and Rose, A. (2003) A simulated annealing methodology to distribution network design and management, *European Journal of Operational Research*, **144**, 629-645.

Jones, K. L., Lustig, I. J., Farvolden, J. M. and Powell, W. B. (1993) Multicommodity network flow: The impact of formulation on decomposition, *Mathematical Programming*, **62**, 95-117.

Kennington, J. L. (1978) A survey of linear cost multicommodity network flows, *Operations Research*, **26**, 209-236.

Klose, A. and Drexl, A. (2005) Facility location models for distribution system design, *European Journal of Operational Research*, **162**, 4-29.

Kouvelis, P. and Rosenblatt, M. J. (1997) *Development of International Facility Networks with Financing, Trade Tariff and Tax Consideration*, John M. Olin School of Business, Washington University, St. Louis.

Krarup, J. and Pruzan, P. M. (1983) The simple plant location problem: Survey and synthesis, *European Journal of Operational Research*, **12**, 36-81.

Liang, D. and Wilhelm, W. E. (2007a) Decomposition schemes and acceleration techniques in application to production-assembly-distribution system design, *Computers & Operations Research*, accepted, available online

Liang, D. and Wilhelm, W. E. (2007b) A generalization of column generation to Accelerate Convergence, submitted to *Mathematical Programming*.

Liang, D. and Wilhelm, W. E. (2007c) Dual-ascent and primal heuristics for production-assembly-distribution system design, submitted to *Discrete Optimization*.

Lee, C. Y. (1991) An optimal algorithm for the multiproduct capacitated facility location problem with a choice of facility type, *Computers & Operations Research*, **18**(2), 167-182.

Lemarechal, C. (2001) *Lagrangian Relaxation, Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions*, Springer-Verlag, Berlin.

Desrosiers, J. and Lubbecke, M. E. (2002) Selected topics in column generation, *Operations Research*, **53**, 1007-1023.

Magnanti, T. L. and Raghavan, S. (2005) Strong formulations for network design problems with connectivity requirements, *Networks*, **45**, 61-79.

Magnanti, T. L. and Wong, R. T. (1984) Network design and transportation planning: models and algorithms, *Transportation Science*, **18**(1), 1-55.

Martello, S., Pisinger, D. and Toth, P. (1999) Dynamic programming and strong bounds for the 0-1 knapsack problem, *Management Science*, **45**, 414-424.

Martin, R. K. (1987) Generating alternative mixed-integer programming models using variable redefinition, *Operations Research*, **35**(6), 820-831.

Mazzol, J. B. and Neebe, A. W. (1999) Lagrangian-relaxation-based solution procedures for multiproduct capacitated facility location problem with choice of facility type, *European Journal of Operational Research*, **115**, 285-299.

Magnanti, T. L. and Wong, R. T. (1990) *Decomposition Methods for Facility Location Problems, Discrete Location Theory*. Wiley, New York, 209-262.

McBride, R. D. and Mamer, J. W. (2004) Implementing an LU factorization for the embedded network simplex algorithm, *INFORMS Journal on Computing*, **16**(2), 109-119.

Melkonian,V and Tardos, T. (2005) Primal-dual-based algorithms for a directed network design problem, *INFORMS Journal on Computing*, **17**, 159-174.

Melkote, S. and Daskin, M. (2001) Capacitated facility location / network design problems, *European Journal of Operational Research*, **129**, 481-495.

Melo, M. T., Nickel, S. and Saldanha da Gama, F. (2005) Dynamic multi-commodity capacitated facility location: a mathematical modeling framework for strategic supply chain planning, *Computers & Operations Research*, **33**, 181-208.

Nazareth, J. (1987) *Computer Solutions of Linear Programs*, Oxford University Press, Oxford, UK.

Nemhauser, G. L. and Wolsey, L. A. (1999) *Integer and Combinatorial Optimization*, Wiley Interscience, New York.

Pisinger, D. (1995) A minimal algorithm for the multiple-choice knapsack problem, *European Journal of Operational Research*, **83**, 394-410.

Robinson, E. P. and Gao, L. L. (1996) A dual ascent procedure for multi-product dynamic demand coordinated replenishment with backlogging, *Management Science*, **42**(11), 1556-1564.

Rosen J. B. (1964) Primal partition programming for block diagonal matrices, *Numerishe Mahematik*, **6**, 250-260.

Santoso, T., Ahmed, S., Goetschalckx, M. and Shapiro, A. (2005) A stochastic programming approach for supply chain network design under uncertainty, *European Journal of Operational Research*, **167**, 96-115.

Sarmiento, A. M. and Nagi, R. (1999) A review of integrated analysis of production–distribution systems, *IIE Transactions*, **31**, 1061-1074.

Savelsbergh, M. W. P. (1997) A branch and price algorithm for the generalized assignment problem, *Operations Research*, **6**, 831-841.

Schmidt, G. and Wilhelm, W. E. (2000) Strategic, tactical, and operational decisions related to multi-national logistics networks: a review and discussion of modeling issues, *International Journal of Production Research*, **38**(7), 1501-1523.

Shaw, D. X. (1993) Limited Generation Technique for Several Telecommunication Network Design Problems, Technical Report, http://palette.ecn.purdue.edu/~shawdx2 /paper.html.

Shaw, D. X. (1999) A unified limited column generation approach for facility location problems on trees, *Annuals of Operations Research*, **87**, 363-382.

Sherali, H. D., Lee, Y. and Kim, Y. (2005) Partial convexification cuts for 0-1 mixed-integer programs, *European Journal of Operations Research*, 625-648.

Shetty, B. and Muthukrishnan, R. (1990) A parallel projection for the multicommodity network model, *Journal of the Operational Research Society*, **41**, 837-842.

Shi, L., Meyer, R. R., Bozbay, M. and Miller, A. J. (2004) A nested partitions framework for solving large-scale multicommodity facility location problems, *Journal of Systems Science and System Engineering*, **13**(2), 158-179.

Sridharan, R. (1995) The capacitated plant location problem, *European Journal of Operational Research*, **87**, 203-213.

Tayur, S., Ganeshan, R. and Magazine, M. (Editors) (1999) *Quantitative Models for Supply Chain Management*, Kluwer, Boston, 669-702.

Teo, C. P. and Shu, J. (2004) Warehouse-retailer network design problem, *Operations Research*, **52**(3), 396-408.

Turkay, B and Artac, T. (2005) Optimal distribution network design using genetic algorithms, *Electric Power Components and Systems*, **33**, 513-524.

Valerio de Carvalho, J. M. (2005) Using extra dual cuts to accelerate column generation, *INFORMS Journal on Computing*, **17**(2), 175-182.

Van Roy, T. J. (1983) Cross decomposition for mixed integer programming, *Mathematical Programming*, **25**, 46-63.

Van Roy, T. J. (1986) A cross decomposition algorithm for capacitated facility location, *Operations Research*, **34**, 145-163.

Van Roy, T. J. and Erlenkotter, D. (1982) A dual-based procedure for dynamic facility location, *Management Science*, **28**, 1091-1105.

Vanderbeck, F. (2002) A generic view of Dantzig-Wolfe decomposition in mixed integer programming, *Operations Research Letters*, **34**(3), 296-306.

Vanderbeck, F. (2005) Implementing Mixed Integer Column Generation, In: *Column Generation*, Desaulniers, G., Desrosiers, J., Solomon, M. M. (Editors), Boston, Kluwer, 331-358.

Verter, V. and Dincer, M. C. (1992) An integrated evaluation of location, capacity acquisition, and technology selection for designing global manufacturing strategies, *European Journal of Operational Research*, **60**, 1-18.

Vidal, C. and Goetschalckx, M. (2002) Modeling and design of global logistics systems: A review of integrated strategic and tactical models and design algorithms, *European Journal of Operational Research*, **143**, 1-18.

Vidal, C. J. and Goetschalckx, M. (2001) A global supply chain model with transfer pricing and transportation cost allocation, *European Journal of Operational Research*, **129**, 134-1158.

Vidal, C. J. and Goetschalckx, M. (forthcoming) The role and limitation of quantitative techniques in the strategic design of global logistics systems, Special Issue on Manufacturing in the global economy, *Technological and Forecasting and Social Change*.

Vidal, C. J. and Goetschalckx, M. (1997) Strategic production – distribution models: a critical review with emphasis on global supply chain models, *European Journal of Operational Research*, **98**, 1-18.

Villavicencio, J. (2005) Solving multicommodity flow problems by an approximation scheme, *SIAM Journal on Optimization*, **15**(4), 971-986.

Villeneuve, D., Desrosiers, J., Lubbecke, M. E. and Soumis, F. (2003) On compact formulations for integer programs solved by column generation, *Les Cahiers du GERAD*, G–2003–06.

Wilhelm, W. E (2001) A technical review of column generation in integer programming, *Optimization and Engineering*, **2**, 159-200.

Wilhelm, W. E., Liang, D., Vasudeva, B. R. T., Warrier, D., Zhu, X. and Bulusu, S. (2005) Design of international assembly systems and their supply chains under NAFTA, *Transportation Research Part E*, **41**, 467-493.

Zhu, X. (2005) The Dynamic, Resource-constrained Shortest-path Problem on an Acyclic Graph with Application in Column Generation and a Literature Review on Sequence-dependent Scheduling, Ph.D. Dissertation, Texas A&M University, College Station, Texas.

VITA

The author of this dissertation, Dong Liang, received his Bachelor of Economics from Shanghai Jiao Tong University, China in 1999, and his Master of Science from Shanghai Jiao Tong University in 2002. His research interests include integer programming, column generation, and combinatorial optimization.

Dong Liang may be reached at 9220 Centennial Dr., Keller, TX 76248. His email is liangdong77@hotmail.com.

The typist for this dissertation was Dong Liang.