Mathematical Programming 62 (1993) 95–117
North-Holland

Multicommodity network flows: The impact of formulation on decomposition

Kim L. Jones and Irvin J. Lustig

Department of Civil Engineering and Operations Research, Program in Statistics and Operations Research, Princeton University, Princeton, NJ, USA

Judith M. Farvolden

Department of Industrial Engineering, University of Toronto, Ont., Canada

Warren B. Powell

Department of Civil Engineering and Operations Research, Program in Statistics and Operations Research, Princeton University, Princeton, NJ, USA

Received 9 April 1992
Revised manuscript received 17 January 1993

This paper is dedicated to Phil Wolfe on the occasion of his 65th birthday.

This paper investigates the impact of problem formulation on Dantzig–Wolfe decomposition for the multicommodity network flow problem. These problems are formulated in three ways: origin-destination specific, destination specific, and product specific. The path-based origin-destination specific formulation is equivalent to the tree-based destination specific formulation by a simple transformation. Supersupply and superdemand nodes are appended to the tree-based product specific formulation to create an equivalent path-based product specific formulation. We show that solving the path-based problem formulations by decomposition results in substantially fewer master problem iterations and lower CPU times than by using decomposition on the equivalent tree-based formulations. Computational results on a series of multicommodity network flow problems are presented.

1. Introduction

Multicommodity network flow (MCNF) problems arise when several commodities share arcs in a network and compete for the capacity on these arcs. Capacitated MCNF problems arise in a variety of applications (Assad [2], Farvolden [8], Tomlin [17]), including the transporation of goods over a network, the flow of information through a communications network, and multi-item production planning. In transportation, the traffic assignment problems involves the routing of items through a capacitated network of transportation services, where each item must be moved from its origin to its destination. For this problem, a commodity can be a specific item,

Correspondence to: Irvin J. Lustig, Department of Civil Engineering and Operations Research, School of Engineering and Applied Sciences, Princeton University, Princeton, NJ 08544, USA.

the flow of items between a common origin and destination, the flow of items with a common destination, or the flow of items with a common origin. By contrast, the multi-fleet allocation problem involves the assignment of different types of vehicles to handle a scheduled set of transportation services, where a transportation service may be handled by more than one type of equipment. For this problem, a commodity is generally treated as the flow of a particular type of equipment.

When deciding on an appropriate formulation, the common practice has been to choose one that produces the smallest number of commodities, thereby minimizing the size of the corresponding linear program. If a Dantzig–Wolfe decomposition approach is being used to solve the problem, a reduction in the number of commodities decreases the number of subproblems that need to be solved and also reduces the size of the linear programming master problem. In this paper, we investigate the hypothesis that using such compact formulations actually produces a much more complex problem, producing the slow convergence that is characteristic of Dantzig–Wolfe decomposition. For example, the traffic assignment problem can be formulated by defining a commodity as the flow between a single origin and destination, instead of flow from a single origin to multiple destinations. The result is a significantly larger master problem, but we present evidence to suggest that the extreme points produced by this formulation offer considerably greater flexibility, producing substantial reductions in the number of master iterations required to reach convergence. This formulation also suggests that many more subproblems are solved at each iteration. However, it is possible to bypass this difficulty by solving the compact subproblem formulation and then decomposing the solution of the subproblem for addition to the master problem. Although it can be argued that a much larger master problem may result in increased CPU times, the total reduction in the number of master problem iterations compensates for this increase in size. Also, depending on the sophistication of the linear programming solver, larger and sparser master problems do not necessarily entail longer solution times at each iteration. Recent advances in solving large-scale linear programs enable one to solve much larger master problems, thereby making it possible to solve formulations with larger numbers of commodities. For this reason, we present both master problem iteration counts and CPU times to optimality for each of the test problems. Our test problems are chosen from applications that use time-space dynamic networks. However, our results should be applicable to other types of problems as well.

Relatively little attention has been devoted to the issue of problem formulation while considerably more attention is given to improvements in algorithms for a given solution method. For example, recent research into interior point methods has produced substantial advances for solving large MCNF problems directly. Choi and Goldfarb [5] exploit the special structure of the MCNF problem in a theoretical application of the primal dual interior point method. Carolan et al. [3] present a similar test of various interior point methods on AT&T's KORBX system. The latter offered the ability to directly solve very large-scale MCNF problems with interior point methods. More recently, Shultz and Meyer [16] and Pinar and Zenios [13]

show it is possible to solve very large MCNF problems using parallel computation techniques.

Large MCNF problems that arise in practice are unlikely to be solved directly as linear programs. However, the special structure of these problems makes decomposition an attractive solution method. Assad [2] uses decomposition to solve a set of MCNF problems where a commodity is associated with a single origin and single destination. Tomlin [17] uses a Dantzig–Wolfe [7] decomposition procedure in solving the traffic assignment and distribution problems in a path-based formulation.

Another common solution method for solving MCNF problems is the primal partitioning algorithm. This algorithm has traditionally been used with an arc-based formulation as seen in Ali et al. [1] and Kennington and Helgason [11]. More recently, Farvolden et al. [9] employs a path-based primal partitioning algorithm for the MCNF problem. This work is especially relevant to the research presented in this paper, as it addresses the issues involving a path-based formulation versus a tree-based formulation. Their results show that it is not only possible to solve large problems with the single origin/single destination formulation using primal partitioning, but that it can be beneficial.

The focus of this research is the investigation into the advantages of the path-based single origin and single destination formulation for the MCNF problem when Dantzig–Wolfe decomposition is applied. Under decomposition, the effects of problem formulation are fully exposed. It is hypothesized that even though the size of the single origin/single destination formulation explodes as the size of the problem grows, this formulation is *easier* to solve than a more compact representation of the problem within a decomposition framework. The intuition behind this hypothesis is twofold. First, there are *fewer* path-based extreme points than tree-based extreme points. Naturally this is dependent upon network structure; however, when this is the case, the decomposition of the path-based formulation will outperform the tree-based formulation. Second, the decomposition of tree structures into paths enables the decomposition method to consider a wider range of feasible solutions in the domain of the master problem. For instance, a path in a network that has a low cost and a large capacity is likely to have a positive flow. However, if this path were grouped into a tree structure, the flow over that path may be inhibited by constraints and the cost of the tree as a whole. Thus, not only are fewer extreme points generated as solutions of the subproblems, the subproblems are also easier to solve. This claim that the single origin/single destination formulation leads to faster convergence via decomposition is supported with computational results in Section 6.

2. Problem Formulation

Consider the following linear program:

$$\text{(MCNF):}\quad \text{minimize} \quad \sum_{\kappa=1}^{\mathscr{K}} (c^{\kappa})^{\mathrm{T}} x^{\kappa}$$

$$\begin{aligned} \text{subject to} \quad & B^{\kappa} x^{\kappa} = d^{\kappa}, \quad \kappa \in \mathscr{K}, \\ & \sum_{\kappa=1}^{\mathscr{K}} x_{ij}^{\kappa} \leqslant u_{ij} \quad \forall (i,j) \in A, \\ & x_{ij}^{\kappa} \geqslant 0 \quad \forall (i,j) \in A, \ \kappa \in \mathscr{K}. \end{aligned} \tag{1}$$

The linear program in (1) is a generic formulation for a MCNF problem where

$\mathscr{K}$ is the set of commodities;

B^{κ} is the node-arc incidence matrix over the network $G=(N, A)$ for commodity κ;

x^{κ} is a vector of arc flows for commodity κ;

c^{κ} is a vector of cost coefficients for commodity κ;

d^{κ} is a vector of supply/demand requirements for commodity κ; and,

u_{ij} is the upper bound on arc (i, j) over the commodities κ.

The linear program in (1) displays the typical block-angular structure of a MCNF problem in that the first set of constraints are the network flow conservation constraints for each commodity and the second set of constraints are the mutual capacity, or *bundle* constraints over the commodities. For ease of discussion, only models without individual capacity constraints on each x_{ij}^{κ} are considered.

We discuss three formulations of the MCNF problem where in each a commodity is defined in a unique way. In the first formulation, we define a commodity as a product that travels between a specific origin and a specific destination. This problem is termed the *origin-destination problem* (ODP) where κ represents the triplet (k, s, t) such that k is the product $(1 \leqslant k \leqslant K)$, $s \in S$ is the specific origin, and $t \in T$ is the specific destination. Here there are K products, $S \subseteq N$ is a set of origins, and $T \subseteq N$ is a set of destinations. The (ODP) is representative of a crew scheduling problem, where the identity of a crew member k must be maintained while satisfying origin and destination constraints. In the second formulation, we define a commodity as a product that travels to a specific destination from multiple origins, or vice versa, from a specific origin to multiple destinations. This problem is termed the *destination specific problem* (DSP) where the commodity κ is the pair (k, t) that identifies the product k with the specific destination t. The (DSP) is typically seen in the traffic assignment problem, where vehicles k must be routed through a network from multiple origins to a common destination t. Finally, in the third formulation, we define a commodity as a product that must travel through a network from multiple origins to multiple destinations. This problem is termed the *product specific problem* (PSP), where the commodity κ represents the singleton k as the product. The (PSP) is representative of the multi-fleet allocation problem where a specific equipment type k must handle a set of transportation services. The (PSP) is the formulation that is traditionally indicated when referring to a MCNF problem.

Note that the constraint matrix for all three formulations is of the form

$$\begin{pmatrix} B^1 & & & \\ & B^2 & & \\ & & \ddots & \\ & & & B^{|\mathscr{K}|} \\ I & I & \cdots & I \end{pmatrix}, \tag{2}$$

where

$$\mathcal{K} = \begin{cases} \{(k, s, t): d^{\kappa} = d^{k}_{st} \neq 0\} = \mathcal{K}(\text{ODP}) & \text{for (ODP)}, \\ \{(k, t): d^{\kappa} = d^{k}_{t} \neq 0\} = \mathcal{K}(\text{DSP}) & \text{for (DSP)}, \\ \{k: d^{\kappa} = d^{k} \neq 0\} = \mathcal{K}(\text{PSP}) & \text{for (PSP)}. \end{cases} \tag{3}$$

The differences among the formulations are the number of *primary blocks* $|\mathcal{K}|$ and the density of the right-hand side vector d^{κ}. The differences in $|\mathcal{K}|$ are noted in (3) where the (ODP) has potentially many more primary blocks than (DSP) and (PSP) due to the combinatorial aspects of all the products, origins, and destinations in the problem. If we assume that the underlying network is identical for all commodities and identical for each of the three formulations, then the node-arc incidence matrix $B^{\kappa} = B$ for all $\kappa \in \mathcal{K}$. Hence, the number of constraints in each primary block is the same for each of the three formulations, but the number of primary blocks varies according to $|\mathcal{K}|$.

The right-hand side vector d^{κ} varies significantly among the formulations. This is seen by investigating the number of non-zeros in each d^{κ} for the three formulations. For the (ODP) and a given triplet (k, s, t) corresponding to commodity κ, there are only two nonzeros in each $d^{\kappa} = d^{k}_{st}$ that correspond to the origin s and the destination t. For the (DSP) and a given pair (k, t) corresponding to commodity κ, each $d^{\kappa} = d^{k}_{t}$ contains multiple nonzeros for the supply constraints at the multiple origins and a single nonzero for the demand constraint at the specific destination t. Finally, for the (PSP) and a given product k corresponding to commodity κ, $d^{\kappa} = d^{k}$ contains multiple nonzeros for the supply/demand constraints at the multiple origins and multiple destinations. Hence, the (ODP) has potentially many more primary blocks than (DSP) or (PSP) in the linear programming formulation, but d^{k} is significantly sparser than in the other two formulations. This is an important concept in the formulation of the MCNF problem and plays a significant role in the decomposition procedure.

3. Decomposition

This section outlines the general mathematical descriptions of the master problem and the subproblems for the tree formulations discussed in Section 2. The reader is assumed to be familiar with Dantzig–Wolfe decomposition and is referred to [7] for an in-depth description of the algorithm.

3.1. Master problem formulation for (ODP), (DSP) and (PSP)

Because of the similarities in the constraint matrices of (ODP), (DSP), and (PSP), the Dantzig–Wolfe decomposition procedure yields similar master problem formulations. Recall that the number of constraints in the master problem is equal to the number of bundle constraints plus the number of convexity constraints. Also recall

that the number of convexity constraints is determined by the number of subproblems in the Dantzig–Wolfe decomposition algorithm. The following is the generic master problem formulation for the block angular system shown in (2). As before, the general set $\mathscr{K}$ is used in the master problem formulation to represent the set of commodities for each formulation. Since a column generation technique is employed for the decomposition algorithm, there is a varying number of columns associated with each $\kappa \in \mathscr{K}$ in the *restricted master problem* constructed at each iteration of the decomposition algorithm. Let P^κ be the total number of columns in the current restricted master problem associated with the element κ. Hence, the general linear program for the restricted master problem is

$$\begin{aligned} \text{(RMP):}\quad \text{minimize} \quad & \sum_{\kappa\in\mathscr{K}} \sum_{p\in P^\kappa} ((c^\kappa)^{\mathrm{T}} v_p^\kappa)\lambda_p^\kappa \\ \text{subject to} \quad & \sum_{\kappa\in\mathscr{K}} \sum_{p\in P^\kappa} (\hat{I}^\kappa v_p^\kappa)\lambda_p^\kappa \leqslant u, \\ & H\lambda = e, \\ & \lambda_p^\kappa \geqslant 0 \quad \text{for each } p\in P^\kappa,\ \kappa\in\mathscr{K}, \end{aligned} \tag{4}$$

where v_p^κ is the pth extreme point solution from the subproblem associated with the index κ. Note that $\hat{I}^\kappa$ is a diagonal matrix with elements

$$\hat{I}_{ii}^\kappa = \begin{cases} 1 & \text{if arc is bundled,} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

It is possible that $\hat{I}^\kappa$ has diagonal elements other than 0 or 1 that represent a weighting of the bundle constraints; however, for ease of discussion, we assume that the diagonal elements are as shown in (5). In many cases, $\hat{I}^\kappa = \hat{I}$ for all $\kappa \in \mathscr{K}$. The vector u is the vector of mutual capacities for the bundled arcs and e is a vector of ones representing the convexity constraints. The matrix H is a block diagonal matrix of e-vectors such that

$$H = \begin{bmatrix} (e^1)^{\mathrm{T}} & & & \\ & (e^2)^{\mathrm{T}} & & \\ & & \ddots & \\ & & & (e^{|\mathscr{K}|})^{\mathrm{T}} \end{bmatrix}, \tag{6}$$

where $e^\kappa \in \mathbb{R}^{P^\kappa}$, for all $\kappa \in \mathscr{K}$. That is, the number of ones in e^κ is equal to the number of columns P^κ in (RMP) that are associated with the subproblem κ.

Depending on the definition of $\mathscr{K}$ in (3), the number of convexity constraints in (RMP) may be vastly different in each formulation. It is important to point out that even though the number of constraints in (RMP) for the (ODP) may be much larger than (DSP) or (PSP), the density of each column v_p^κ is significantly sparser in (ODP) due to the nonzero structure of d^k. This is the topic of Section 3.2 where the nature of v_p^κ is outlined and discussed in full for each of the three formulations.

3.2. Subproblem formulation for (ODP), (DSP) and (PSP)

At each iteration of the Dantzig–Wolfe decomposition algorithm, a subproblem is solved for each $\kappa \in \mathcal{K}$. For the MCNF problem, each subproblem describes a single commodity network flow problem and is formulated as

$$\begin{aligned} \text{SUB}(\kappa, \pi): \quad & \text{minimize} \quad (c^\kappa - \hat{I}^\kappa \pi)^{\mathrm{T}} v^\kappa \\ & \text{subject to} \quad Bv^\kappa = d^\kappa, \\ & \qquad\qquad v_{ij}^\kappa \geqslant 0 \quad \text{for each } (i,j) \in A, \end{aligned} \tag{7}$$

where the elements of the vector π are the values of dual variables acting upon the bundle constraints in (4). The solution to the linear program in (7) varies among the three formulations and is uniquely determined by d^κ. Recall that for the (ODP), d^κ is a vector with two nonzeros that correspond to the constraints on the flow between a specific origin and a specific destination. The solution is thus a vector of arc flows determining the *shortest path* from origin s to destination t for the product k. For the (DSP), d^κ contains multiple positive elements for each of the supplies of k at multiple origins s, and a single negative element for the demand at the specific destination t. The solution to this subproblem is thus a vector of arc flows determining the *shortest path tree* from all sources to the destination. Finally for the (PSP), d^κ demonstrates that the product k must be routed from multiple origins to multiple destinations. The extreme point generated by this subproblem is the solution to a *minimum cost network flow problem*. Hence it is likely that the extreme point solution v_{st}^k for the (ODP) subproblem will be much sparser than that of the (DSP) or the (PSP). Also, because this (ODP) extreme point solution represents the flow along a single path, the column added to the master problem has a nonzero unit value corresponding to the convexity constraint, and all other nonzero values equal to d_{st}^k. Conversely, the extreme point solutions for (DSP) and (PSP) yield varying nonzero values within each column.

Because the set $\mathcal{K}$(ODP) has potentially many more elements than $\mathcal{K}$(DSP) or $\mathcal{K}$(PSP), there are many more subproblems and hence, many more possible columns added to the restricted master problem at each iteration of the decomposition procedure. It appears as though there is great potential for an explosion in the number of columns in the restricted master problem for the (ODP) formulation; however, we claim that for certain types of networks, there will be fewer total extreme points added to the (ODP) master problem than either the (DSP) or (PSP) master problems, because there will be fewer master problem iterations. To demonstrate this claim, let P_t^κ represent the number of extreme points associated with commodity κ on iteration t. Define $\mathcal{P}_t$ to be the maximum number of columns at decomposition iteration t associated with any element $\kappa \in \mathcal{K}$, i.e.,

$$\mathcal{P}_t = \max_{\kappa \in \mathcal{K}} \{P_t^\kappa\}. \tag{8}$$

Hence, for all formulations, there are potentially $|\mathscr{K}| \times \mathscr{P}_t$ columns in the restricted master problem at iteration t. Therefore, we claim that for large-scale networks, especially time-space dynamic networks,

$$|\mathscr{K}(\mathrm{ODP})| \times \mathscr{P}_{T_o}(\mathrm{ODP}) \ll |\mathscr{K}(\mathrm{DSP})| \times \mathscr{P}_{T_d}(\mathrm{DSP}), \tag{9}$$

where T_o is the number of iterations to optimality for (ODP) and T_d is the number of iterations to optimality for (DSP). This is apparent by investigating the total

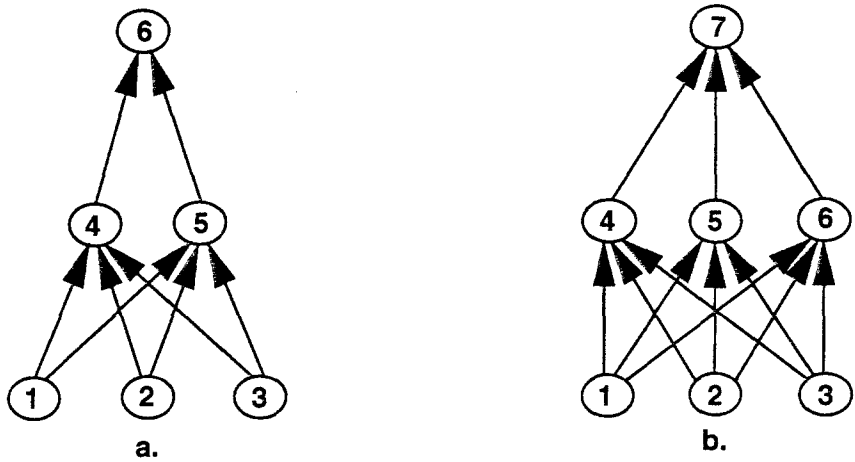

Fig. 1. Extreme point enumeration for paths versus trees.

number of extreme points in each formulation. The small example in Figure 1a is due to [8] and shows a dynamic network with a planning horizon of two time periods. This network has 6 possible paths and 8 possible trees. Extending this example, Figure 1b shows that the addition of one node increases the number of paths from 6 to 9 paths while the number of trees grows from 8 to 27. It is clear that for a problem with a long planning horizon and many nodes per time period, the difference in the number of paths versus the number of trees will be extremely large. Hence, even though there may be many more extreme point proposals at each iteration for the (ODP) formulation versus the (DSP) formulation, there will be fewer columns in the final restricted master problem due to this combinatorial effect of paths and trees. This is especially relevant to this research as most of the data sets are drawn from transportation related problems modeled as large-scale time-space dynamic networks. We support these claims with extensive numerical results in Section 6.

4. Comparisons of formulations

The focus of this research is to evaluate the effects of problem formulation on the Dantzig-Wolfe decomposition procedure specifically for MCNF problems. We seek to establish and isolate the differences between the path-based formulation of (ODP)

and the tree-based formulations of (DSP) and (PSP). In order for a valid comparison among the formulations, they must be mathematically equivalent problems. The Dantzig–Wolfe decomposition procedure produces a solution that is a convex combination of the extreme points obtained from the subproblems. Hence, if the subproblems are determined to be mathematically equivalent, then the entire formulations are equivalent.

To better distinguish between the solutions of the subproblems, let x^k be the collection of vectors of arc flows describing the tree solutions to the (PSP) minimum cost network flow subproblems, let y^{kt} be the collection of vectors of arc flows describing the shortest path tree solutions to the (DSP) subproblems, and finally, let z^{kst} be the vector of arc flows describing the solutions to the shortest path (ODP) subproblems. The relationships between (ODP) and (DSP) depend on the following lemma, stated in Rockafellar [15] and restated here in our notation.

Lemma 1. *If y^{kt} is feasible for* (DSP), *then there exists values of z^{kst} for each $s \in S$, $t \in T$, and $1 \leqslant k \leqslant K$, such that*

$$y^{kt} = \sum_{s \in S} z^{kst}. \qquad \square \tag{10}$$

This lemma says that flows aggregated in a tree structure from all sources into a destination can be decomposed into a set of feasible path flows from individual sources into that destination. The following corollaries are obvious consequences of this lemma.

Corollary 1. *If the costs on the arcs in the network are not dependent upon origin or destination constraints, and z^{kst} is optimal for* (ODP), *then y^{kt} given by* (10) *is optimal for* (DSP). $\square$

Corollary 2. *If the costs on the arcs in the network are not dependent upon origin or destination constraints, and y^{kt} is optimal for* (DSP), *then an optimal solution z^{kst} to* (ODP) *can be constructed from the optimal solution y^{kt} such that* (10) *holds.* $\square$

Given the cost requirement that c^k is dependent solely upon the product k (i.e., costs are not origin-destination specific), these corollaries state that one can solve (ODP) if the problem is formulated as a (DSP) and that the reverse situation also holds. For Dantzig–Wolfe decomposition, these corollaries enable the extreme point solutions for the (ODP) to be constructed from the optimal shortest path tree solution of the (DSP) subproblem. Hence, the number of subproblems solved at each iteration of the (ODP) is significantly reduced by solving only $|\mathscr{K}(\text{DSP})|$ subproblems, then decomposing the solutions into the optimal shortest paths for the equivalent (ODP) extreme points.

Note that corresponding theorems do not hold between the (DSP) and the (PSP) subproblems when this cost requirement is true. That is, an optimal solution x^k to the (PSP) subproblem cannot be constructed by aggregating the optimal solutions to the (DSP) subproblems (and hence, the optimal solutions to the (ODP) subproblems). We demonstrate this by example to compare the optimal solution of a set of (ODP) shortest path problems and the optimal solution to the (PSP) corresponding

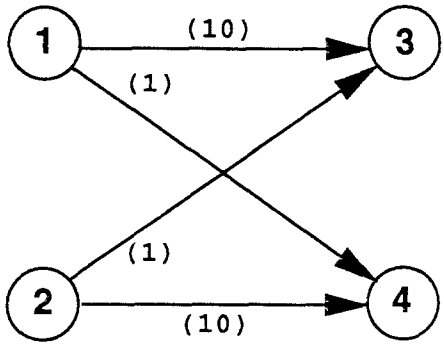

Fig. 2. A 4-node network.

minimum cost network flow problem. The small example in Figure 2 uses the (ODP) demand matrix

D^1	3	4
1	1	0
2	0	1

and is solved at a cost of 20 by routing flow over arcs (1, 3) and (2, 4). Since the (PSP) enforces only aggregate flow conservation at each node, the (PSP) formulation results in a solution that uses the arcs (1, 4) and (2, 3) at a cost of 2. Since the optimal flows for (ODP) are optimal for (DSP), this example serves as a proof by contradiction that corollaries for (DSP) and (PSP) that are similar to Corollaries 1 and 2 are invalid.

Hence, the comparison between the path-based (ODP) and the tree-based (DSP) is straight-forward if the costs in the network are not constrained by origin and destination requirements. This is a common assumption in MCNF problems and is true of the numerical examples presented in this research. However, the (PSP) must be transformed into a mathematically equivalent path-based formulation in order for the comparison to be valid. This is the topic of the next section, where we create an extended path-based (PSP) formulation that is equivalent to the tree-based (PSP). Section 6 presents results that compare the (ODP) to the (DSP) in a decomposition environment as well as a set of results that compare the (PSP) to its equivalent path-based formulation.

5. The extended product specific problem

The graph $G=(N, A)$ can be augmented with additional nodes and arcs with the goal of creating a path-based MCNF problem that is mathematically equivalent to the (PSP). Hence, we create K *supersupply* nodes $\hat{S}=\{\hat{s}_1, \ldots, \hat{s}_K\}$ and K *superdemand* nodes $\hat{T}=\{\hat{t}_1, \ldots, \hat{t}_K\}$ and require that the flow of commodity k travel between a specific origin $\hat{s}_k$ and a specific destination $\hat{t}_k$. Hence, a new graph $\hat{G}=(\hat{N}, \hat{A})$ is created where $\hat{N}=N \cup \hat{S} \cup \hat{T}$ and

$$\hat{A}=A \cup \{(\hat{s}, s): \hat{s} \in \hat{S}, s \in S\} \cup \{(t, \hat{t}): t \in T, \hat{t} \in \hat{T}\}, \tag{11}$$

such that S and T are the sets of original supply and demand nodes, respectively. This is demonstrated in Figure 3. This new problem is mathematically equivalent to the (PSP) if the total flow $\bar{d}^k$ from $\hat{s}_k$ to $\hat{t}_k$ is such that

$$\bar{d}^k=\sum_{s \in S} d_s^k=\sum_{t \in T} d_t^k, \tag{12}$$

and the flow along the new arcs is constrained to satisfy original supply and demand constraints. Here, d_s^k is the supply of commodity k available at the original origin node s. Similarly, d_t^k is the demand for commodity k at destination node t. The linear program to solve this problem is labelled (EPSP) to denote the extended (PSP) and is formulated as

$$\begin{aligned}
\text{(EPSP):}\quad \text{minimize}\quad & \sum_{k=1}^{K} (c^k)^{\mathrm{T}} x^k \\
\text{subject to}\quad & \sum_{k=1}^{K} x_{ij}^k \leqslant u_{ij} \quad \text{for each } (i,j) \in A, \\
& \bar{x}_{\hat{s}_k s}^k \leqslant d_s^k \quad \text{for each } \hat{s}_k \in \hat{S},\ s \in S,\ 1 \leqslant k \leqslant K, \\
& \bar{x}_{t \hat{t}_k}^k \leqslant d_t^k \quad \text{for each } t \in T,\ \hat{t}_k \in \hat{T},\ 1 \leqslant k \leqslant K, \\
& B x^k - \bar{x}_{\hat{s}_k \cdot}^k + \bar{x}_{\cdot \hat{t}_k}^k = 0 \quad \text{for each } 1 \leqslant k \leqslant K, \\
& \sum_{s \in S} \bar{x}_{\hat{s}_k s}^k = \bar{d}^k \quad \text{for each } 1 \leqslant k \leqslant K,\ \hat{s}_k \in \hat{S}, \\
& \sum_{t \in T} \bar{x}_{t \hat{t}_k}^k = -\bar{d}^k \quad \text{for each } 1 \leqslant k \leqslant K,\ \hat{t}_k \in \hat{T}, \\
& x_{ij}^k \geqslant 0 \quad \text{for each } (i,j) \in A,\quad 1 \leqslant k \leqslant K, \\
& \bar{x}_{\hat{s}_k s}^k \geqslant 0 \quad \text{for each } \hat{s}_k \in \hat{S},\ s \in S,\ 1 \leqslant k \leqslant K, \\
& \bar{x}_{t \hat{t}_k}^k \geqslant 0 \quad \text{for each } \hat{t}_k \in \hat{T},\ t \in T,\ 1 \leqslant k \leqslant K.
\end{aligned} \tag{13}$$

Here, $\bar{x}_{\hat{s}_k s}^k$ is the flow on the arc from $\hat{s}_k \in \hat{S}$ to $s \in S$ of commodity k. Similarly, $\bar{x}_{t \hat{t}_k}^k$ is the flow of commodity k on the arc from $t \in T$ to $\hat{t}_k \in \hat{T}$. The notation $-\bar{x}_{\hat{s}_k \cdot}^k + \bar{x}_{\cdot \hat{t}_k}^k$ is used to indicate that the flow into each node $s \in S$ from each node $\hat{s}_k \in \hat{S}$ plus the flow from each node $t \in T$ to each node $\hat{t}_k \in \hat{T}$ should be added to the network flow equations to maintain conservation of flow. The capacity constraints on the arcs from

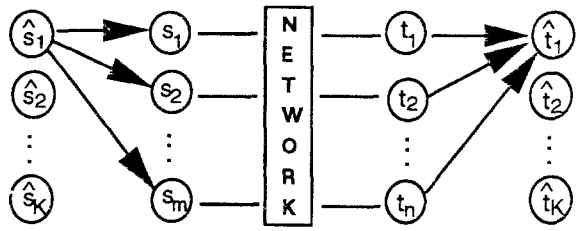

Fig. 3. Example of an (EPSP) network.

and to all supersupply and superdemand nodes are included in this linear program as inequality constraints. However, it is obvious that they will all be satisfied as equalities at optimality.

Note that the node-arc incidence matrix for the augmented graph is reflected in the last three equalities of (EPSP). This matrix is denoted as $\hat{B}$. Like (ODP), these equations have only two nonzeros on the right-hand side. However, the presence of the upper bound constraints on $\bar{x}^k_{\hat{s}_k s}$ and $\bar{x}^k_{t\hat{t}_k}$ differentiates this formulation from the (ODP). This difference is noted by considering the constraint matrix in a block form as

$$\begin{bmatrix} \hat{B}^1 & & & & & \\ & \hat{B}^2 & & & & \\ & & \ddots & & & \\ & & & \hat{B}^{K-1} & & \\ & & & & \hat{B}^K & \\ I & I & I & \cdots & I & \\ & I & & & & \\ & & I & & & \\ & & & \ddots & & \\ & & & & I & \end{bmatrix}, \tag{14}$$

where the matrices $\hat{B}^k$ represent the node-arc incidence matrices for each commodity k on the graph $\hat{G}$.

5.1. *Decomposition for (EPSP)*

When applying decomposition to (EPSP), the additional complicating constraints may be placed in either the master problem or in the subproblems. If they are included in the subproblems, the master problem formulation is identical to (4) and the subproblem for a specific commodity k imposes both upper and lower bounds on the additional arcs so that the flows along these arcs are *fixed*. When this is the

case, the solution to the (EPSP) subproblem is *identical* to that of the (PSP) subproblem and therefore nothing has been gained by transforming it to a path-based formulation. On the other hand, if the constraints on flow on arcs $(\hat{s}_k, s)$ and $(t, \hat{t}_k)$ are considered in the master problem, the solution to the subproblem is similar to that of (ODP) in that the solution is a single path between a specific supersupply node and a specific superdemand node. This latter formulation for the (EPSP) subproblem is the one we shall use. Because there are no individual capacity constraints in the subproblem, it is important to note that the solution is a single path which is *infeasible* for the original linear programming problem. Because there is an external flow $\bar{d}^k$ into supersupply node $\hat{s}_k$ in the subproblems that are solved, the solution must necessarily send the total flow $\bar{d}^k$ from supersupply node $\hat{s}_k$ to a single source node s. This is an infeasible flow for the original problem (PSP) that is rectified in the master problem by forcing feasibility on the capacity constraints for the arcs adjacent to all supersupply and superdemand nodes. Although the constraints on the arcs adjacent to the supersupply and superdemand nodes are included as inequality constraints in the restricted master problem for (EPSP), the decomposition procedure forces them to be satisfied as equalities because of the conditions on total flow in the subproblem. Hence, once a feasible solution to this (RMP) is obtained, it is guaranteed to satisfy the original supply/demand constraints in the (PSP) formulation.

The difficult part of this alternate formulation is obtaining an initial feasible solution. In order for a solution to be feasible, for each commodity k, as many paths must be generated in the subproblem as necessary to "touch" all original supply/demand nodes. That is, since the subproblems generate single paths from a supersupply node to a superdemand node, there must be at least one path passing through each of the original supply/demand nodes in the master problem. For example, consider a two commodity network with 3 supply nodes, 3 demand nodes and 2

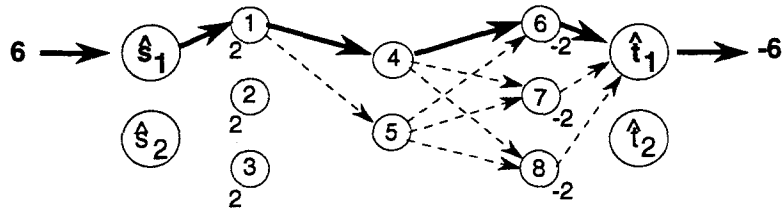

Fig. 4. (EPSP) subproblem example.

intermediate nodes. Figure 4 shows the subproblem solution for the first commodity indicated with bold arrows. This solution is included in the master problem as a column for the path $\hat{s}_1 \rightarrow \hat{t}_1$. The column will include two nonzeros in the bundling constraints of (RMP) associated with arcs $(\hat{s}_1, 1)$ and $(6, \hat{t}_1)$. However, in order to satisfy all of the supply/demand constraints for the first commodity, the master problem must also have columns with nonzeros associated with arcs $(\hat{s}_1, 2)$, $(\hat{s}_1, 3)$, $(7, \hat{t}_1)$ and $(8, \hat{t}_1)$. To satisfy these constraints, the subproblem for commodity 1 must

be solved *at least* 3 times, generating the paths

$$\hat{s}_1 \to 1 \to 4 \to 6 \to \hat{t}_1,$$

$$\hat{s}_1 \to 2 \to 4 \to 7 \to \hat{t}_1,$$

$$\hat{s}_1 \to 3 \to 5 \to 8 \to \hat{t}_1.$$

However, there are 18 possible paths that the subproblem may choose. This means that the subproblem may be solved anywhere from 3 to 18 times, and again, this is only for the first commodity. This "touching property" must be satisfied for *all* commodities before a feasible solution can be found for the restricted master problem. Even in this small example it is easy to see that it may be very time consuming to find an initial feasible solution. The question is, does the fast convergence rate of this path-based formulation outweigh the detriments of finding an initial feasible solution? The numerical results presented in Section 6 show that the path-based formulation of (EPSP) does in fact improve the convergence rate to optimality despite the difficulties in obtaining an initial feasible solution.

6. Numerical results

In this section, the effects of decomposition on the different formulations as discussed in the previous sections are compared and contrasted. Results are presented that support the claims that the larger and sparser path-based formulations of (ODP) and (EPSP) have better convergence rates via Dantzig–Wolfe decomposition than those of (DSP) and (PSP), respectively. This is demonstrated by comparing total master problem iteration counts and CPU times to optimality.

The decomposition code used in this study was written in the C programming language. The master problems are solved by *CPLEX* [6] and subproblems are solved by *Simpnet*, a C-based network simplex code written by Lustig [12]. The CPU times reported in this research are for an IBM RISC/System 6000 machine. A two phase solution technique for the problem is implemented where the first phase minimizes the sum of all infeasibilities in the master problem, and the second phase iterates until optimality. Tables are presented for the major iteration counts for both phase 1 and phase 2 (i.e., the number of times the master problem is solved), CPU time to optimality, and the maximum number of rows and columns in each master problem. As noted in Corollary 2, the optimal flow in a (DSP) formulation can be decomposed into a set of optimal path flows for the (ODP) formulation. Hence, in the (DSP)–(ODP) comparison, all subproblems are solved in the tree-based (DSP) formulation and the (DSP) trees are decomposed into (ODP) paths. The CPU times times reflect this in that the number of subproblems solved at each iteration is *identical* for the two formulations.

Table 1
(ODP) and (DSP) comparisons for Farvolden data sets

Problem	Nodes	Links	Bundled	$\|\mathscr{K}\|$		Phase 1 iter.		Phase 2 iter.	
				(DSP)	(ODP)	(DSP)	(ODP)	(DSP)	(ODP)
10term	190	510	146	10	317	9	3	18	4
10term.0	190	507	143	10	323	6	2	15	3
10term.50	190	498	134	10	323	7	3	12	3
10term.100	190	491	127	10	323	6	4	19	3
15term	285	796	253	15	530	7	3	20	4
15term.0	285	745	202	15	561	10	3	34	3

Table 1 presents results on problems drawn from Farvolden [8] where each problem can be treated as a (DSP) or an (ODP). These problems are drawn from the *LTL* transportation model where each product is associated with a unique destination node called a *terminal*. Each of these data sets is a time-space network with an 18 day planning horizon where each network contains two types of arcs: inventory arcs (no cost) and loaded movement arcs (capacitated, high cost). In Table 1, the value of $|\mathscr{K}|$ is presented for both (DSP) and (ODP) to indicate the difference in the number of (DSP) trees and (ODP) origin-destination pairs. Thus, in the Farvolden data sets, a (DSP) with 10 terminals is decomposed into an (ODP) with 317 origin-destination pairs. This difference is reflected in the number of rows in the master problems, as demonstrated in Table 2. Table 1 demonstrates that in every instance, solution of the (ODP) formulation requires fewer master problem iterations. This is especially noticeable as the size of the problem grows larger as in the 15 terminal problem. Table 3 displays the CPU times required to solve each of the problems in Table 1. Even though the size of the restricted master problem (as documented in Table 2) is significantly larger in the (ODP) formulation, the CPU times are smaller.

Table 2
Maximum number of rows and columns in the master problems for the Farvolden data sets

Problem	Master problem dimensions			
	(DSP)		(ODP)	
	Rows	Cols	Rows	Cols
10term	156	182	463	687
10term.0	153	135	466	502
10term.50	144	204	457	620
10term.100	137	223	450	771
15term	217	474	763	1311
15term.0	268	263	783	1056

Table 3
CPU time comparisons of (DSP) and (ODP) for the Farvolden data sets

Problem	CPU seconds		Ratio
	(DSP)	(ODP)	
10term	13.87	7.97	1.74
10term.0	8.94	6.64	1.35
10term.50	12.04	6.55	1.84
10term.100	12.59	7.25	1.74
15term	22.74	15.93	1.43
15term.0	33.39	16.08	2.08

It is clear that the reduced number of master problem iterations contributes greatly to the reduced CPU times for the solution of each (ODP) formulation.

The Chen [4] problems originate from the *stochastic multicommodity dynamic vehicle allocation* problem. These are also (DSP) formulated time-space networks that can be decomposed into an equivalent set of origin-destination pairs. As in the Farvolden data sets, each subproblem is solved using the (DSP) formulation and then decomposed into a set of paths for the (ODP) formulation. Again, Table 4 demonstrates that a significant change is seen in the number of master problem iterations required to solve each of the (DSP) and (ODP) formulations, especially as the size of the problem increases. Table 5 displays the CPU times required to solve each of the problems in Table 4. Note that the decomposition algorithm applied to the (ODP) formulations of *Chen2* and *Chen6* yield a reduction in CPU times of more than an order of magnitude. Table 6 lists the maximum size of the restricted master problems for the Chen data sets. The numerical results in Table 5 reflect that the number of columns in each master problem is limited to 10 times the number of rows for each formulation. Once this limit is reached, an effort is made to delete nonbasic columns from the current restricted master problem.

The next test set is from Assad [2] where each problem is originally formulated as an (ODP). For comparison purposes, only those data sets where it is possible to

Table 4
(ODP) and (DSP) comparisons for Chen data sets

Problem	Nodes	Links	Bundled	$\lvert\mathscr{K}\rvert$		Phase 1 iter.		Phase 2 iter.	
				(DSP)	(ODP)	(DSP)	(ODP)	(DSP)	(ODP)
Chen0	26	117	43	4	18	10	7	99	24
Chen3	31	149	56	15	71	12	8	74	31
Chen1	36	174	65	5	25	13	8	165	44
Chen2	41	358	155	7	70	23	7	356	34
Chen6	41	409	177	9	89	22	6	372	33
Chen4	55	420	176	15	133	25	8	318	61
Chen5	65	569	242	10	78	23	9	560	136

Table 5
CPU ratios (DSP): (ODP) for the Chen data sets

Problem	CPU seconds		Ratio
	(DSP)	(ODP)	
Chen0	8.16	2.63	3.10
Chen3	22.11	11.32	1.95
Chen1	30.15	11.08	2.72
Chen2	597.54	44.19	13.52
Chen6	841.64	64.32	13.08
Chen4	937.82	254.15	3.69
Chen5	2698.13	1184.54	2.28

Table 6
Maximum number of rows and columns in the master problems for the Chen data sets

Problem	Master problem dimensions			
	(DSP)		(ODP)	
	Rows	Cols	Rows	Cols
Chen0	47	400	61	419
Chen3	71	710*	127	1270*
Chen1	70	700*	80	800*
Chen2	162	1620*	225	2250*
Chen6	186	1860*	266	2660*
Chen4	191	1910*	309	2346
Chen5	252	2520*	320	3200*

* indicates the maximum number of allowable columns was reached.

aggregate, rather than decompose, groups of origin-destination pairs into corresponding (DSP) trees are reported. In this case, the (DSP) model actually has *flow from an origin* rather than flow into a destination. The sizes of these problems are much smaller than either of the Farvolden or Chen data sets, yet the results demonstrate again that the (ODP) formulation is solved in fewer master problem iterations than the (DSP) formulation. In Table 7, the networks labeled *assad1* and *assad3* are each solved with two different sets of origin-destination pairs. In each of these cases,

Table 7
(ODP) and (DSP) comparisons for Assad data sets

Problem	Nodes	Links	Bundled	$\lvert\mathcal{K}\rvert$		Phase 1 iter.		Phase 2 iter.	
				(DSP)	(ODP)	(DSP)	(ODP)	(DSP)	(ODP)
assad1.5k	47	98	98	3	10	6	3	12	1
assad1.6k	47	98	98	3	15	4	5	7	1
assad3.4k	85	204	204	6	18	7	5	13	5
assad3.7k	85	204	204	6	18	8	7	15	7

the total number of master problem iterations *increased* with a smaller $|\mathscr{K}|$ which is consistent with the results from the Farvolden and Chen data sets. This is another indication that the larger and sparser (ODP) formulation is easier to solve than the smaller and denser (DSP) formulation in a decomposition framework. The largest recorded CPU time in the set of solutions to the Assad data sets was 2.9 CPU seconds for the (DSP) formulation of *assad3.7k*.

The following results compare the iterative convergence rates for the solution of (PSP) and (EPSP) formulations when the decomposition algorithm is applied to the formulation of (EPSP) as outlined in Section 5.1. This set of test problems are also due to Chen [4] where each product k is associated with multiple origins and multiple destinations. Therefore, each is originally formulated as a (PSP) and is appended by supersource and superdemand nodes and arcs to create the corresponding (EPSP). Note that the size of the constraint matrix in both the master problem and subproblems are larger in the (EPSP) formulation due to the addition of the specially capacitated arcs. Table 8 shows the differences in size of the networks for (PSP) and (EPSP),

Table 8
(PSP) and (EPSP) comparisons for Chen data sets

Problem	$\lvert\mathscr{K}\rvert$	Nodes		Links		Phase 1 iter.		Phase 2 iter.	
		(PSP)	(EPSP)	(PSP)	(EPSP)	(PSP)	(EPSP)	(PSP)	(EPSP)
psp1	3	15	21	41	59	7	12	26	23
psp2	4	30	38	180	225	14	30	233	124
psp3	8	25	41	112	190	11	17	80	55
psp4	10	27	47	86	145	9	15	57	41
psp5	9	30	48	167	251	12	21	137	77
psp6	4	60	68	832	899	18	41	869	337
psp7	6	84	96	1435	1551	21	65	1743	542

as well as the iteration counts for the solutions of the decomposition algorithm applied to each formulation. As hypothesized earlier, more effort is required to obtain a feasible solution to the (EPSP) formulation for each of the test problems. However, the phase 2 performance of the algorithm for the (EPSP) formulation outperformed the algorithm applied to the (PSP) formulation in every instance. Also, the total iteration count of the algorithm applied to (EPSP) was less than the total iteration count of the algorithm applied to (PSP) in all instances except for the smallest problem, *psp1*. As the size of the problem increases, the performance of Dantzig–Wolfe decomposition on (EPSP) improves compared to that of (PSP). This gain in performance is especially noticeable in the larger data sets when comparing CPU times. Table 9 shows the CPU times required to solve each of the problems in Table 8. Note that as the size of the problem increases, the difference in the CPU times of the solutions is significant. This is especially noticeable upon inspection of *psp7* in Tables 8 and 9. Table 8 indicates that three times as many major iterations are required to solve *psp7* when the decomposition procedure is applied to the (PSP) formulation, compared to when it is applied to the (EPSP) formulation. However,

Table 9

CPU time comparisons of (PSP) and (EPSP) for the Chen data sets

Problem	CPU seconds		Ratio
	(PSP)	(EPSP)	
psp1	1.06	1.17	0.90
psp2	63.50	25.26	2.51
psp3	12.26	15.77	0.77
psp4	8.43	7.31	1.15
psp5	55.91	41.87	1.33
psp6	14959.04	902.74	16.57
psp7	494548.92	16149.69	30.62

Table 9 indicates that the CPU time required to solve the (EPSP) formulation was 30 times faster than the (PSP) formulation. Similarly, the algorithm applied to the (EPSP) formulation of *psp6* was 16 times faster in CPU seconds, even though the number of master problem iterations was only reduced by a factor of two. These observations strengthen our claim that the larger and sparser master problem is actually easier and faster to solve than the more traditional, compact representation.

From this it can be seen that even though more effort is required to obtain a feasible solution to the (EPSP) formulation, the path-based structure of the columns in the master problem yields faster convergence overall. Note that the number of rows in the restricted master problem of the (EPSP) formulation requires only $2|\mathscr{K}|$ additional rows as compared to the master problem of the (PSP) formulation. The results in Tables 8 and 10 shows that because of the decreased number of major iterations in the solution of the (EPSP) formulation, the final master problem has far fewer columns than that of the (PSP) formulation in half of the experiments, and is especially noticeable in the more difficult problems, *psp6* and *psp7*. This is clearly evident in the next two examples.

Table 10

Maximum number of rows and columns in the master problems for the Chen data sets

Problem	Master problem dimensions			
	(PSP)		(EPSP)	
	Rows	Cols	Rows	Cols
psp1	17	102	35	108
psp2	77	770*	122	565
psp3	51	510*	129	632
psp4	40	400*	99	504
psp5	78	780*	162	904
psp6	373	3488	440	1475
psp7	658	6580*	774	3557

* indicates the maximum number of columns was reached.

Table 11
(ODP) and (DSP) comparisons for Powell data set

Problem	Nodes	Links	Bundled	$\|\mathcal{K}\|$		Phase 1 iter.		Phase 2 iter.	
				(DSP)	(ODP)	(DSP)	(ODP)	(DSP)	(ODP)
veh.8	3071	6364	301	3	236	42	21	2109	15

The next two data sets demonstrate even more concretely than any of the previous examples that applying Dantzig–Wolfe decomposition to the path-based formulation of (ODP) is far superior to applying it to the (DSP) formulation in both iteration counts and CPU time. The first data set is drawn from a truckload transportation model as described by Powell [14], with an extension to allow multiple vehicle types. The problem is a time-space network with a planning horizon of eight days and is originally formulated as a (DSP) with three vehicle types k and a single destination node t. This particular data set is interesting in that each mutually capacited arc has an upper bound of one which makes the problem much more difficult and highly degenerate. Table 11 indicates the difference in $|\mathcal{K}|$ and shows the major iteration counts required to solve *veh.8* in both the (ODP) and (DSP) formulations. Note that the algorithm applied to the (ODP) formulation outperformed the algorithm applied to the (DSP) formulation in major iteration counts by nearly 60 times. However, Table 12 shows that in CPU seconds, solving the (ODP) formulation was over 400 times faster than solving the (DSP) formulation. Again note the difference in the sizes of the master problems. Table 13 shows that the (DSP) master problem reached its limit of 3040 columns, yet the solution of the (ODP) formulation reached the

Table 12
CPU time comparisons of (DSP) and (ODP) for the Powell data set

Problem	CPU seconds		Ratio
	(DSP)	(ODP)	
Veh.8	16164.69	38.47	420.19

Table 13
Maximum number of rows and columns in the master problems for the Powell data set

Problem	Master problem dimensions			
	(DSP)		(ODP)	
	Rows	Cols	Rows	Cols
veh.8	304	3040*	537	1472

* indicates the maximum number of allowable columns was reached.

Table 14
(ODP) and (DSP) comparisons for the ALK data sets

Problem	Nodes	Links	Bundled	$\|\mathscr{K}\|$		Phase 1 iter.		Phase 2 iter.	
				(DSP)	(ODP)	(DSP)	(ODP)	(DSP)	(ODP)
alk.half	1121	3194	1251	30	1013	32	7	>447	22
alk.two	4067	7090	4212	30	1596	72	13	≫350	133

optimal solution with only 1472 columns. This problem demonstrates a restricted master problem that has significantly fewer columns for the (ODP) formulation which is due entirely to the decreased number of major iterations while employing the column generation technique. Thus, even though many more columns may be added at each iteration of the algorithm using the (ODP) formulation, so many fewer iterations are required that the final restricted master problem has far fewer columns than the final restricted master problem in the equivalent (DSP) formulation.

The next data set comes from ALK [10] and is a railroad transportation problem. Unlike the other data sets that simulate real world problems, the railroad data is in fact real data supplied by ALK Inc. and the railroad company Union Pacific. These networks are again time-space networks where the products are different types of railcars that must be routed to a single destination node. These two data sets are the largest presented in this paper, with the largest having more than 4000 rows in the master problem. Table 14 shows the major iteration counts for solving each of the formulations. The algorithm applied to the (DSP) formulation did not reach optimality in both *alk.half* and *alk.two*. Table 15 shows the CPU seconds for each solution procedure and the objective value obtained. In both cases, the solution of the (ODP) formulation reached the optimal objective value. The algorithm applied to the *alk.half* formulated as a (DSP) was stopped after it was 3 orders of magnitude slower than the (ODP) formulation. Similarly, the algorithm applied to *alk.two* formulated as a (DSP) was stopped after nearly 450000 CPU seconds when it was clear that the optimal objective value could not be reached in a *reasonably* comparable length of time (for this reason, a "≫" is used in each of the tables to indicate that the solution of the (DSP) formulation was far from being optimal). Table 16 shows the difference in the sizes of the master problem at the time of completion. At the point of termination, the master problem of *alk.half* in the (ODP) formulation had fewer columns and

Table 15
CPU time comparisons of (DSP) and (ODP) for the ALK data sets

Problem	Obj. Value		CPU seconds		Ratio
	(DSP)	(ODP)	(DSP)	(ODP)	
alk.half	−9.89 e+07	−9.97 e+07*	>296553.60	292.23	>1014.79
alk.two	1.55 e+10	−4.98 e+08*	≫446233.80	36193.00	≫12.40

* indicates the optimal objective value was obtained.

Table 16

Maximum number of rows and columns in the master problems for the ALK data sets

Problem	Master problem dimensions			
	(DSP)		(ODP)	
	Rows	Cols	Rows	Cols
alk.half	1281	>9269	2264	5224
alk.two	4242	≫6951	5808	16431

a sparser constraint matrix than did the master problem of the (DSP) formulation. It is very likely that the master problem of the (DSP) formulation of *alk.two* would exhibit the same property if the algorithm had been allowed to iterate for a longer period of time.

7. Conclusion

This paper focuses on the effects of the formulation of a multicommodity network flow problem in the framework of Dantzig–Wolfe decomposition. Numerical experiments support the hypothesis that decomposition applied to the larger and sparser path-based formulations of (ODP) and (EPSP) yields faster convergence rates than decomposition applied to (DSP) and (PSP), respectively. These path-based formulations increase the number of rows in the master problem by introducing more convexity constraints in the case of (ODP) and more capacity constraints in the case of (EPSP). The reformulation of (DSP) into (ODP) causes only a small increase in the number of constraints in the master problem compared with the increase in the number of subproblem proposals generated at each iteration. For this reason, it may be argued that the huge increase in the number of subproblems may be a significant factor in the decision to use the (DSP) versus the (ODP) formulation. It must be emphasized that even though more subproblems are solved at each iteration when solving (ODP), there are a fewer number of total extreme points enumerated in this formulation. Also, it must be noted that the subproblems are easier to solve since they are simply shortest path problems. This hurdle of solving more subproblems at each iteration can be eliminated by solving the corresponding tree-based subproblem, then decomposing the solution into paths which are then added to the master problem.

The strength of the path-based formulation is most noted in the (EPSP) alternate formulation. This is a slight increase in the number of rows in the master problem with the additional complicating capacity constraints, yet the number of subproblems remains the same. Even though this reformulation of (PSP) causes an increase in the time it takes to obtain a feasible solution, the overall performance of (EPSP) was better than that of (PSP). Again, this problem may be remedied by solving a (PSP)

subproblem and decomposing it to all possible (EPSP) paths for the alternate formulation. This would result in the addition of multiple (EPSP) paths for each subproblem at each iteration.

Future research will examine these effects on other instances of multicommodity network flow problems in an attempt to improve the performance of decomposition procedures. The concept of changing the formulation to accomodate optimization algorithms may also be applicable to decomposition-like procedures such as those proposed in Shultz and Meyer [16] and Pinar and Zenios [13]. The "folklore" of linear programming has had a negative view towards decomposition procedures. Because of the developments in the linear programming community, this research indicates that decomposition procedures may be applicable if the right problem is solved using today's better optimization technology.

References

[1] A. Ali, R. Helgason, J. Kennington and H. Lall, "Computational comparison among three multicommodity network flow algorithms," *Operations Research* 28 (1980) 995–1000.

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

[3] W.J. Carolan, J.E. Hill, J.L. Kennington, S. Niemi and S.J. Wichmann, "An empirical evaluation of the KORBX algorithms for military airlift applications," *Operations Research* 38(2) (1990) 240–248.

[4] C.E. Chen, "A two-level decomposition algorithm for the stochastic multicommodity dynamic vehicle allocation model," PhD thesis, Department of Civil Engineering and Operations Research, Program in Statistics and Operations Research, Princeton University (Princeton, NJ, 1990).

[5] I.C. Choi and D. Goldfarb, "Solving multicommodity network flow problems by an interior point method," *SIAM Proceedings in Applied Mathematics* 46 (1990) 58–69.

[6] CPLEX Optimization, Inc., *Using the CPLEX Linear Optimizer, 1.2 edition* (Incline Village, NV).

[7] G.B. Dantzig and P. Wolfe, "The decomposition algorithm for linear programs," *Econometrica* 29 (1961) 767–778.

[8] J.M. Farvolden, "A primal partitioning solution for multicommodity network flow problems," PhD thesis, Department of Civil Engineering and Operations Research, Program in Statistics and Operations Research, Princeton University (Princeton, NJ, 1989).

[9] J.M. Farvolden, W.B. Powell and I.J. Lustig, "A primal partitioning solution for the arc-chain formulation of a multicommodity network flow problem," to appear in: *Operations Research.*

[10] ALK Inc., private communication (1991).

[11] J.L. Kennington and R.V. Helgason, *Algorithms for Network Programming* (Wiley, New York, 1980).

[12] I.J. Lustig, "The influence of computer language on computational comparisons: An example from network optimization," *ORSA Journal on Computing* 2(2) (1990) 152–161.

[13] M.C. Pinar and S.A. Zenios, "Parallel decomposition of multicommodity network flows using a linear-quadratic penalty algorithm," *ORSA Journal on Computing* 4(3) (1992) 235–249.

[14] W.B. Powell, "A review of sensitivity results for linear networks and a new approximation to reduce the effects of degeneracy," *Transportation Science* 23(4) (1989) 231–243.

[15] R.T. Rockafellar, *Network Flows and Monotropic Optimization* (Wiley, New York, 1984).

[16] G.L. Schultz and R.R. Meyer, "An interior point method for block angular optimization," *SIAM Journal on Optimization* 1(4) (1991).

[17] J.A. Tomlin, "A mathematical programming model for the combined distribution-assignment of traffic," *Transportation Science* 5 (1971) 122–140.