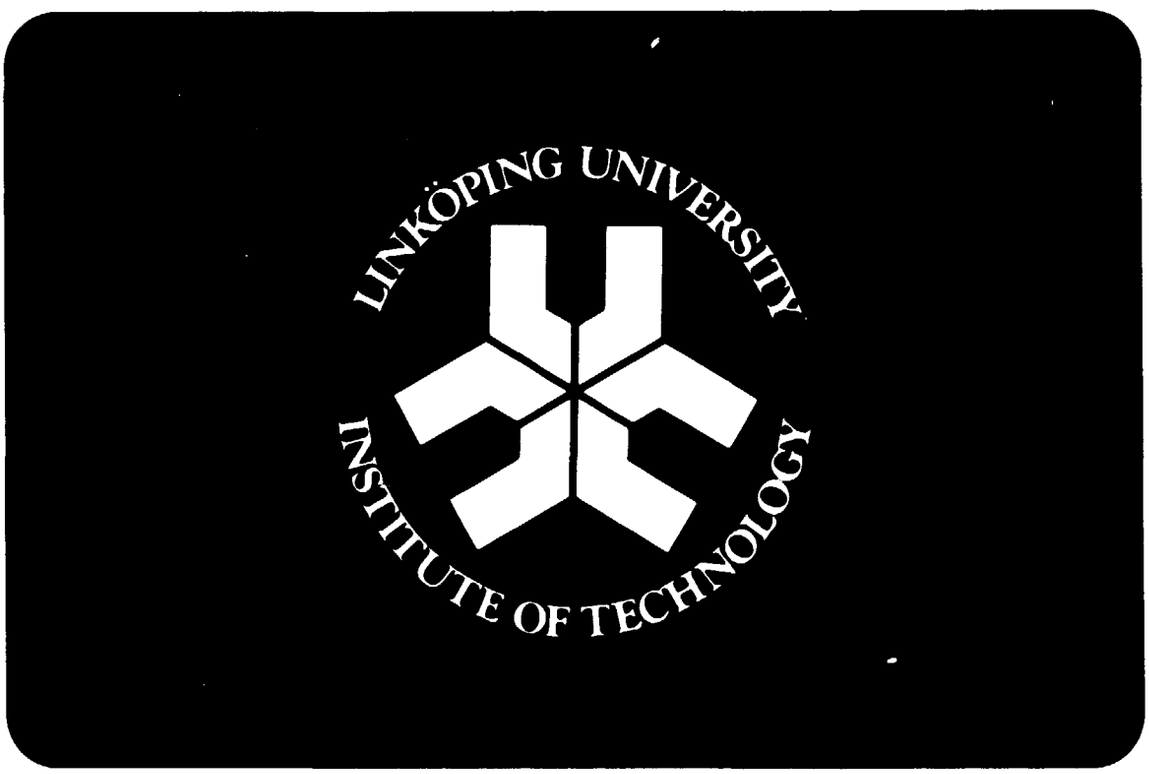
Department of Mathematics

Efficient Decomposition and Linearization Methods for the Stochastic Transportation Problem

Kaj Holmberg

LiTH-MAT-R-~~19~~93-10



Universitetet i Linkoping
581 83 Linkoping

Linkoping University
S-581 83 Linkoping, Sweden

Efficient Decomposition and Linearization Methods for the Stochastic Transportation Problem

Kaj Holmberg

LiTH-MAT-R-~~19~~93-10

Department of Mathematics
Linköping Institute of Technology
S-581 83 Linköping
Sweden

Abstract

The stochastic transportation problem can be formulated as a convex transportation problem with nonlinear objective function and linear constraints. We compare several different methods based on decomposition techniques and linearization techniques for this problem, trying to find the most efficient method or combination of methods. We discuss and test a separable programming approach, the Frank-Wolfe method with and without modifications, the new technique of mean value cross decomposition and the more well known Lagrangean relaxation with subgradient optimization, as well as combinations of these approaches. Computational tests are presented, indicating that some new combination methods are quite efficient for large scale problems.

Key Words: Transportation, Decomposition methods, Separable Programming, the Frank-Wolfe method

Acknowledgement: This work has been financed by the Swedish Research Council for Engineering Sciences.

1 Introduction

The Stochastic Transportation Problem, STP, has been treated in many papers, [8, 25, 26, 27, 5, 21], and can be solved by different methods. It is a transportation problem with stochastic demand, which is modelled by introducing shortage costs and holding costs. The resulting problem is a transportation problem with the demand constraints replaced by nonlinear costs as functions of the total inflow into each demand point. The STP is a problem with a nonlinear, convex objective function and structured linear constraints. The mathematical formulation of the STP is given in section 2.

In this paper we compare different methods based on linearization techniques and decomposition techniques for the STP. The aim is simply to find the most efficient method for the STP, so we try to make the comparison as fair and neutral as possible. We compare a separable programming approach, the Frank-Wolfe method with and without modifications, mean value cross decomposition and Lagrangean relaxation with subgradient optimization. The same amount of "fine-tuning" is made for all the methods, including the modified Frank-Wolfe method.

It turns out that several different combinations of these methods can be made very efficient. We do not claim to have tested every possible method for the STP; in fact we restrict ourselves to methods based on different linearization and decomposition approaches. However, this includes some of the methods previously claimed to be the most efficient for the STP, especially the Frank-Wolfe method.

The STP has previously been solved with the Frank-Wolfe method [9] in Cooper and LeBlanc [5] and LeBlanc, Helgason and Boyce [17], cross decomposition [22] in Holmberg and Jörnsten [14], the classical approach of separable programming in Holmberg [11] and the forest iteration method in Qi [21].

After Cooper and LeBlanc (1977) [5], the Frank-Wolfe method seems to have been accepted as the most efficient method for the STP. In Holmberg and Jörnsten (1984) [14], the cross decomposition method is shown to be superior to the Frank-Wolfe method. In that paper, separable programming is also found to be an interesting approach (more details can be found in [11]) worthy of a better fate than to be dismissed as in [5]. One should be aware of the development of new efficient network flow codes since 1977.

However, since the separable programming approach gives feasible solutions and upper bounds but no lower bounds on v^*, we have no good measure of how far from the optimum the obtained solutions lie. Due to that, separable programming cannot be compared on equal terms with the other two methods, both of them being iterative methods yielding both upper and lower bounds and able to continue iterating until any prespecified accuracy requirement is satisfied.

In LeBlanc, Helgason and Boyce (1985) [17], a simple modification is used to speed up the Frank-Wolfe method when applied to the STP. Since no comparison is made to cross decomposition, it is not at the present clear which of the methods is the most efficient for the STP.

The forest iteration method [21] iterates between "base forest triples", in order to find the optimal one. For each base forest triple, a number of line searches are made. Computational tests are reported, but no comparisons to other methods are done, and the sizes of the problems solved are much smaller than those previously solved in [5] and [14]. In section 4 it is argued why we do not believe this method to be more efficient than the (modified) Frank-Wolfe method. This is the reason why we have not made any computational comparisons with this method.

The new approach of mean value cross decomposition, presented in Holmberg [13], is a modification of ordinary cross decomposition [22], with the advantage of not using any master problem, which is why it might be an efficient method for the STP. This is discussed more in section 3.8.

In section 3.1 we describe the Frank-Wolfe method and some modifications applied to the STP. The application of separable programming is described in section 3.2. Lagrangean relaxation and subgradient optimization, as well as computational improvements are discussed in sections 3.5 and 3.6.

We have made computational tests comparing and combining the Frank-Wolfe method (with and without modifications), separable programming, Lagrangean relaxation with subgradient optimization and mean value cross decomposition. In section 4 it is shown that combinations of mean value cross decomposition and separable programming produces solution methods that are clearly more efficient than for example the Frank-Wolfe method.

2 The Stochastic Transportation Problem

The Stochastic Transportation Problem is a transportation problem with m supply points and n demand points. At each supply point, i, the supply is fixed and known, S_i, but at the demand points the demand is stochastic. At demand point j there is a probability density function, $\phi_j(d_j)$, for the demand, d_j, which gives an expected demand as $E[d_j] = \int_0^\infty v\phi_j(v)\,dv$, and a distribution function as $F_j(d_j) = \int_0^{d_j} \phi_j(v)\,dv$. At demand point j there is also a unit holding cost, $h_j > 0$ and a unit shortage cost, $p_j > 0$, which gives a total holding/shortage cost, f_j, as a function of the total amount, y_j, shipped into the demand point.

$$f_j(y_j) = p_j \int_{y_j}^{\infty} (v - y_j)\phi_j(v)\,dv + h_j \int_0^{y_j} (y_j - v)\phi_j(v)\,dv$$

This can also be expressed as

$$f_j(y_j) = p_j(E[d_j] - y_j) + (p_j + h_j) \int_0^{y_j} F_j(v)\,dv$$

The derivative of f_j (used later) is

$$\frac{df_j(y_j)}{dy_j} = -p_j + (p_j + h_j)F_j(y_j)$$

The second order derivative reveals that $f_j(y_j)$ is a convex function wherever $\phi_j(y_j) > 0$. The unconstrained minimum of $f_j(y_j)$ is obtained at the point $\check{y}_j$, where

$$F_j(\check{y}_j) = \frac{p_j}{p_j + h_j} \Rightarrow \check{y}_j = F_j^{-1}\left(\frac{p_j}{p_j + h_j}\right)$$

$\check{y}_j$ obviously is a positive, finite point.

Furthermore we have linear transportation costs, given by the unit cost c_{ij} from supply point i to demand point j. The main variables, x_{ij}, are the amounts transported from supply point i to demand point j.

The problem of minimizing the total costs without exceeding the supplies can be mathematically formulated as the stochastic transportation problem.

$$
\begin{aligned}
v^* = \quad \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j(y_j) & & \\
\text{s.t.} \quad & \sum_{i=1}^{m} x_{ij} = y_j & j = 1,\ldots,n \quad & (1) \\
& \sum_{j=1}^{n} x_{ij} \leq S_i & i = 1,\ldots,m \quad & (2) \\
& x_{ij} \geq 0 & \forall\, i,j \quad & (3)
\end{aligned}
\qquad \text{(STP)}
$$

The objective function is convex, and the constraints are linear. One could also add the following redundant constraints on y.

$$\sum_{j=1}^{n} y_j \leq S_{TOT},\ y_j \geq 0\ \forall j$$

where $S_{TOT} = \sum_{i=1}^{m} S_i$.

In an alternate formulation the y-variables are eliminated.

$$
\begin{aligned}
v^* = \quad \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j(\sum_{i=1}^{m} x_{ij}) & & \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} \leq S_i & i = 1,\ldots,m \quad & (1) \\
& x_{ij} \geq 0 & \forall\, i,j \quad & (2)
\end{aligned}
\qquad \text{(STP2)}
$$

This makes the linear constraints separable in i. Note, however, that the objective function is not additively separable in i.

3 Methods

The STP has been solved with the Frank-Wolfe method [5], cross decomposition [14], separable programming [11] and the forest iteration method [21]. We will here also test mean value cross decomposition and subgradient optimization of the Lagrange dual, as well as combinations of these approaches. We will below describe the methods in more detail.

First, however, we wish to mention the importance of upper and lower bounds on the optimal objective function value, v^*. Any feasible solution to the problem yields an upper bound, while lower bounds can be obtained by various relaxations of the problem. We are of course mostly interested in good feasible solutions to the problem. However, we also want to know the quality on an obtained feasible solution, and the superior measure of quality is obtained by a lower bound on v^*, since this yields the maximal difference in cost between the found solution and the optimal solution. This is important when you do not expect to solve the problem exactly, but only to some prespecified accuracy, which most often is the case with the STP (even in [21], due to the line searches) and similar nonlinear problems. Reasons for this are the limited accuray of computers and possible uncertainty of the cost coefficients.

So in this paper we only compare methods that yield both upper and lower bounds on v^*. Apart from such methods there are possibilities of obtaining "hopefully" good upper bounds and solution by different primal heuristics. But if they cannot be combined with some technique generating lower bounds, so that any specified accuracy can be obtained, they are not imple-

mented here. This also indicates the advantage of iterative methods, where better solutions can be obtained by increasing the number of iterations.

Our set of methods include combinations of methods that yield only upper bounds with methods that yield only lower bounds, and these combinations are actually the most promising ones.

3.1 The Frank-Wolfe method

One method that has been applied to the STP is based on iterative linear approximations of the objective function. It is the well known Frank-Wolfe method [9], first developed for quadratic programming. It was suggested for the STP by Cooper and LeBlanc [5].

The Frank-Wolfe method is applicable to nonlinear optimization problems with convex objective functions and linearly constrained feasible sets.

$$v^* = \min f(x) \text{ s.t. } Ax = b,\ x \geq 0$$

It is a primal search method. At each iteration point, $x^{(k)}$, the nonlinear objective function is linearized and approximated by $\tilde{f}(x) = f(x^{(k)}) + \nabla f(x^{(k)})^T(x - x^{(k)})$. The resulting LP-problem is then solved.

$$\min \nabla f(x^{(k)})^T x \text{ s.t. } Ax = b,\ x \geq 0$$

The solution, $\hat{x}$, to this LP-subproblem provides a lower bound on the optimal objective value, v^*, as $\underline{v} = f(x^{(k)}) + \nabla f(x^{(k)})^T(\hat{x} - x^{(k)})$, and also indicates the direction in which to move, $d^{(k)} = \hat{x} - x^{(k)}$. A line search is made in this direction, using the exact nonlinear objective function,

$$\min_{t \in [0,1]} f(x^{(k)} + td^{(k)})$$

yielding the steplength, $t^{(k)}$, and the next iteration point $x^{(k+1)} = x^{(k)} + t^{(k)}d^{(k)}$. The iteration points evaluated in the nonlinear objective function yields upper bounds, $\bar{v} = f(x^{(k)})$, and the algorithm is terminated when the relative difference between the bounds is smaller than a prespecified accuracy requirement. The method has asymptotic convergence in both the objective function value and the primal solution.

Applying the Frank-Wolfe method to the STP, the linear subproblem, obtained from linearizing formulation STP2 at $x^{(k)}$, separates into m trivially solvable knapsack problems. For each i, we solve

$$\begin{aligned} \hat{v}_i = \quad & \min \quad \sum_{j=1}^{n} \hat{c}_{ij} x_{ij} \\ & \text{s.t.} \quad \sum_{j=1}^{n} x_{ij} \leq S_i \qquad (1) \\ & \qquad x_{ij} \geq 0 \quad \forall\, j \quad (2) \end{aligned} \qquad (\text{FWS}_i)$$

where

$$\hat{c}_{ij} = c_{ij} + \frac{df_j(y_j^{(k)})}{dy_j} = c_{ij} - p_j + (p_j + h_j)F_j(y_j^{(k)})$$

and $y_j^{(k)} = \sum_{i=1}^{m} x_{ij}^{(k)}$. The optimal solution to FWS_i is obtained by simply finding $\hat{c}_{il} = \min_j \hat{c}_{ij}$ and letting $\hat{x}_{il} = S_i$ if $\hat{c}_{il} < 0$, $\hat{x}_{il} = 0$ if $\hat{c}_{il} \geq 0$, and $\hat{x}_{ij} = 0\ \forall j \neq l$. The lower bound on v^* is

obtained as

$$\underline{v} = \sum_{i=1}^{m} \hat{v}_i - \sum_{j=1}^{n} \left(\frac{df_j(y_j^{(k)})}{dy_j} y_j^{(k)} \right)$$

The main advantage of applying the Frank-Wolfe method to the STP is that the linear subproblems are so quickly solved. This is the reason why Cooper and LeBlanc [5] stated that the Frank-Wolfe method "is more efficient than any other technique" they are aware of for the STP.

3.1.1 Modifications of the Frank-Wolfe method

Often the Frank-Wolfe method exhibits a zig-zagging behaviour as the optimum is approached. A number of suggestions how improve the performance of the method have been made. One is to try to eliminate the zig-zagging by doing extra line searches in the direction from previous iterations points towards the new iteration point. These directions are called PARTAN-directions. However, in [17] this modification does not give any improvement for the STP. Instead the authors propose another, very simple modification.

Instead of solving the linear subproblem exactly, it is solved approximately. This is done by not *sending everything to the best destination, but dividing it evenly between the* K best destinations with negative modified costs, i.e. letting $\hat{x}_{ij} = S_i/K$ if $\hat{c}_{ij}$ is one of the K least coefficients (that are negative). If fewer than K cost coefficients are negative, we temporarily decrease K. This modification is claimed to improve the performance of the Frank-Wolfe method applied to the STP significantly. Practical questions are which values of K yield the most efficient method, and if one should change K during the solution process. To ensure convergence of the algorithm we must either use an ordinary Frank-Wolfe iteration (i.e. let $K = 1$) regularly, or let K decrease gradually down to 1. Some computational tests on these matters are reported in section 4.

Another simple modification, suggested in [24], is to make the steplength a little longer (multiplying by a factor of say 1.1) than the result of the line search.

We have tested the following specific methods:

FW1 The standard Frank-Wolfe method.

FW2 A modified Frank-Wolfe method. Divide S_i equally among the K destinations with least modified costs (provided they are negative). If the lower bound has not increased more than $K\varepsilon$ for NK iterations, decrease K with one.

FW3 A modified Frank-Wolfe method. Divide S_i equally among the K destinations with least modified costs (provided they are negative). Do an ordinary iteration ($K = 1$) every L-th iteration.

FW4 A modified version of FW2. Increase the steplength by multiplying the result of the line search with r (for example $r = 1.1$), if feasibility allows.

FW5 A modified version of FW2. Divide S_i among the K destinations with least modified costs (provided they are negative), so that the destination with the least cost gets $S_i/2$, the one with the next least cost gets $S_i/4$ and so on. The last assignment is modified so that the sum of the assignments equals S_i.

The methods FW2 and FW3 implement two different ways of ensuring convergence when using approximate solutions of FWS_i. The motivation behind FW5 is that the subproblem solution is better than that of FW2, but not so extreme as that of FW1.

Partial linearization [16, 18] is an extension of the Frank-Wolfe method, where only a part of the objective function is linearized. By some reformulations, this can be applied to the STP, which results in a nonlinear subproblem. However, some very preliminary tests indicate that this approach probably is not very efficient for the STP. Intuitively, partial linearization tries to improve the Frank-Wolfe method by strengthen the direction finding problem. This can be worthwhile when the Frank-Wolfe directions are too weak. However, when the Frank-Wolfe method is one of the best existing methods for the problem, as it is for the STP, the price for this strengthening, which is the increase in difficulty when solving the direction finding probelm, is too high.

3.2 Separable programming

Separable programming is a useful technique for problems with additively separable nonlinear functions. It is described in general text books such as [1]. Each nonlinear, convex function $f_j(y_j)$ is replaced by a piecewise linear approximation between 0 and $\check{y}_j$, which makes the whole problem linear. The function $f_j(y_j)$ is linearized between certain breakpoints, and if we use Q_j equidistant breakpoints, they will be placed at kT_j, for $k = 0, \ldots, Q_j$, where $T_j = \check{y}_j/Q_j$, i.e. T_j is the distance between subsequent breakpoints. (Counting $\check{y}_j$ as a breakpoint, makes the number of linear segments the same as the number of breakpoints.) In effect we make the substitution $y_j = \sum_{k=1}^{Q_j} z_{jk}$ and calculate the linearized cost coefficients

$$d_{jk} = \frac{f_j(kT_j) - f_j((k-1)T_j)}{T_j} = -p_j + \frac{(p_j + h_j)}{T_j} \int_{(k-1)T_j}^{kT_j} F_j(v)\, dv$$

The following linear approximation of STP is then obtained.

$$\begin{aligned} v^* = \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n}\sum_{k=1}^{Q_j} d_{jk}z_{jk} \\ \text{s.t.} \quad & \sum_{i=1}^{m} x_{ij} = \sum_{k=1}^{Q_j} z_{jk} \quad j = 1,\ldots,n \qquad (1) \\ & \sum_{j=1}^{n} x_{ij} \le S_i \qquad i = 1,\ldots,m \qquad (2) \\ & 0 \le z_{jk} \le T_j \qquad \forall\, j,k \qquad (3) \\ & x_{ij} \ge 0 \qquad \forall\, i,j \qquad (4) \end{aligned} \qquad \text{(SP)}$$

Of course the quality of the approximation is improved for increased numbers of breakpoints, but on the other hand this also makes the problem larger, and hence more time consuming to solve. In our computational tests we have used the same number of breakpoints for every function, $Q_j = Q\ \forall j$.

SP is a minimal cost network flow problem, with a very special network structure, consisting of a bipartite transportation network with in addition Q arcs leaving each destination point. The costs on these extra arcs are d_{jk} and the capacities are T_j. There are supply constraints, but no demand constraints. The size of the network is $m + n + nQ$ nodes and $mn + nQ$ arcs. See figure 1 for an example with $m = 3$, $n = 4$ and $Q = 2$.

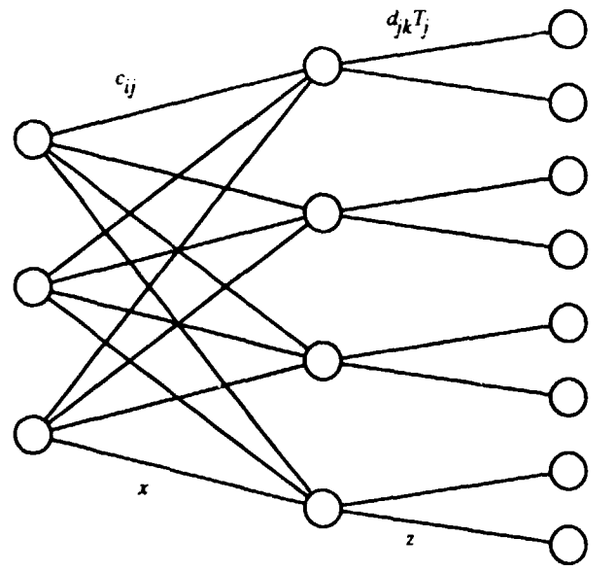


Figure 1: Extended network for separable programming.

This method yields feasible primal solutions, and our experience shows that the solutions are rather good, if the number of breakpoints is not too small. (This is discussed further in section 4.) Unfortunately this method does not give any lower bounds. Therefore we will consider combining this method with others that yield lower bounds.

We use the following notation. Solving SP with with the network code RELAX, Bertsekas and Tseng [3], is called **SP1**. Solving SP approximately with a greedy heuristic based on the special network structure is called **SP2**.

3.3 Heuristics

Heuristics can be used to get inital upper or lower bounds, and also starting solutions, both primal and dual. Most heuristics are not very good. However, one very simple, but interesting heuristic to get a dual starting solution, α, is to find the cheapest way from any origin to each destination, i.e. to set $\alpha_j = \min_i c_{ij}$. Another possibility is to start with the most expensive way to get to each destination, $\alpha_j = \max_i c_{ij}$. This of course often gives worse objective function values, but may nevertheless have advantages. This is dicussed more in section 3.6. In general, the best results were found by using a convex combination of these two extremes.

$$\alpha_j = \theta \min_i c_{ij} + (1-\theta) \max_i c_{ij}$$

We call this procedure **HEUR**.

3.4 Benders decomposition

Benders decomposition [2] is a method for problems with one linear part and one more difficult (originally integer) part. Here the y-variables can be regarded as "difficult". The primal subproblem, PS, is obtained by fixing y to $\bar{y}$. Ths turns STP into an ordinary transportation

problem, which can be solved efficiently (with for example the RELAX code by Bertsekas and Tseng [3]).

$$
\begin{aligned}
h(\bar{y}) = \quad \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j(\bar{y}_j) & & \\
\text{s.t.} \quad & \sum_{i=1}^{m} x_{ij} = \bar{y}_j \quad j = 1,\ldots,n & & (1) \\
& \sum_{j=1}^{n} x_{ij} \leq S_i \quad i = 1,\ldots,m & & (2) \\
& x_{ij} \geq 0 \quad \forall\, i,j & & (3)
\end{aligned}
\qquad \text{(PS)}
$$

We can note that PS has a feasible solution if and only if $\sum_{j=1}^{n} \bar{y}_j \leq S_{TOT}$, $\bar{y}_j \geq 0\ \forall j$.

In Benders decomposition we also need a master problem, which updates the values of $\bar{y}$. This master problem can be found in Holmberg and Jörnsten [14]. It uses the dual solutions of PS (α_j and β_i) to construct Benders "cuts". This method has finite, exact convergence, both for the objective function value and the primal solution. Unfortunately, the master problem is an unstructured nonlinear problem, and too difficult to solve, for Benders decomposition to be an efficient solution method for the STP.

3.5 Lagrangean relaxation

Lagrangean relaxation is a technique that often is used to get lower bounds on v^*. We construct a Lagrangean relaxation by relaxing constraint set 1 of STP, and fixing the Lagrangean multipliers, α_j to $\bar{\alpha}_j$. The following dual subproblem, DS, is then obtained.

$$
\begin{aligned}
g(\bar{\alpha}) = \quad \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j(y_j) + \sum_{j=1}^{n} \bar{\alpha}_j \left(\sum_{i=1}^{m} x_{ij} - y_j \right) & & \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} \leq S_i \quad i = 1,\ldots,m & & (1) \\
& x_{ij} \geq 0 \quad \forall\, i,j & & (2)
\end{aligned}
\qquad \text{(DS)}
$$

This problem separates in i, i.e. into one problem for each origin. Furthermore each such problem separates into one x-problem and one y-problem. The x-problem is of the same kind as FWS, above, i.e. a trivially solvable continuous knapsack problem.

$$
\begin{aligned}
g_i^1(\bar{\alpha}) = \quad \min \quad & \sum_{j=1}^{n} \hat{c}_{ij}x_{ij} & & \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} \leq S_i & & (1) \\
& x_{ij} \geq 0 \quad \forall\, j & & (2)
\end{aligned}
\qquad \text{(DS}x_i\text{)}
$$

where $\hat{c}_{ij} = c_{ij} + \bar{\alpha}_j$. The optimal solution to DSx_i is obtained by finding $\hat{c}_{il} = \min_j \hat{c}_{ij}$ and letting $\hat{x}_{il} = S_i$ if $\hat{c}_{il} < 0$, $\hat{x}_{il} = 0$ if $\hat{c}_{il} \geq 0$, and $\hat{x}_{ij} = 0\ \forall j \neq l$.

Adding the redundant constraints on y mentioned in section 2, the y-problem becomes

$$
\begin{aligned}
g^2(\bar{\alpha}) = \quad & \min \quad \sum_{j=1}^{n} f_j(y_j) - \sum_{j=1}^{n} \bar{\alpha}_j y_j \\
& \text{s.t.} \quad \sum_{j=1}^{n} y_j \leq S_{TOT} \qquad (1) \\
& \qquad y_j \geq 0 \qquad \forall\, j \quad (2)
\end{aligned}
\tag{DSy}
$$

If we omit constraint 1, DSy separates into a number of trivial one-dimensional problems (with explicit solution if $F_j(y_j)$ allows). If we include constraint 2, we can use Lagrange duality to solve the problem, since it is strictly convex problem (whenever $\phi_j(y_j) > 0$). We then get

$$
g^2(\bar{\alpha}) = \max_{\bar{\delta} \geq 0} \min_{y \geq 0} \sum_{j=1}^{n} f_j(y_j) - \sum_{j=1}^{n} \bar{\alpha}_j y_j - \bar{\delta} \left(\sum_{j=1}^{n} y_j - S_{TOT} \right) \tag{LDSy}
$$

Considering the inner minimization in y for a fixed $\bar{\delta}$, the problem separates into N one-dimensional problems. The derivative of the objective function with respect to y_j is

$$
\frac{df_j(y_j)}{dy_j} - \bar{\alpha}_j - \bar{\delta} = -p_j + (p_j + h_j)F_j(y_j) - \bar{\alpha}_j - \bar{\delta}
$$

To find the minimum we find the point, $\hat{y}_j$, where this derivative is equal to zero.

$$
(p_j + h_j)F_j(\hat{y}_j) = p_j + \bar{\alpha}_j + \bar{\delta} \Rightarrow \hat{y}_j = F_j^{-1}\left(\frac{p_j + \bar{\alpha}_j + \bar{\delta}}{p_j + h_j} \right)
$$

However, this is true only if the argument to F_j^{-1} lies between 0 and 1, i.e. if $-p_j \leq \bar{\alpha}_j + \bar{\delta} \leq h_j$. If the argument is negative, $\bar{\alpha}_j + \bar{\delta} < -p_j$, we have to truncate and set $\hat{y}_j = 0$. If the argument is greater than 1, the problem is unbounded and we should set $\hat{y}_j = \infty$. This is never optimal, so we could restrict the multipliers to $\bar{\alpha}_j + \bar{\delta} \leq h_j$.

The Lagrange dual can be solved by a line search in $\bar{\delta}$. We have used the bisection search method. Since the relaxed constraint is redundant, it is not always necessary to find the exact optimal value of $\bar{\delta}$. In some cases, it is more efficient to solve LDSy approximately, by not doing a line search at all. (Letting $\bar{\delta} = 0$ is the same as ignoring constraint 1.) However, as noted in section 3.4, constraint 1 must be included if the resulting y-solution is to be used in PS.

The lower bound on v^* is obtained as $\underline{v} = \sum_{i=1}^{m} g_i^1(\bar{\alpha}) + g^2(\bar{\alpha})$.

We can note that DSy is strictly convex, while DSx is an LP-problem. This means that if the optimal Lagrange multipliers are used in DS, we can expect to get the optimal y-solution, but not the optimal x-solution, due to the well-known lack of controllability in Lagrangean relaxation.

STP could be solved with (nonlinear) Dantzig-Wolfe decomposition [7], using the master problem given in Holmberg and Jörnsten [14] which is an unstructured LP-problem. However, we do not expect this to be a very efficient solution method, due to the difficulty of the master problem.

3.6 Subgradient Optimization

To solve the Lagrangean dual we wish to maximize the dual function over the Lagrangean multipliers, i.e. solve $\max g(\alpha)$. This can be done with subgradient optimization, [19, 20, 10]. A

subgradient of $g(\alpha)$ at $\bar{\alpha}$ is obtained as

$$\xi_j = \sum_{i=1}^{m} \hat{x}_{ij} - \hat{y}_j$$

where $\hat{x}$ and $\hat{y}$ are optimal solutions to DS at $\bar{\alpha}$. We then update the values of our Lagrange multipliers as

$$\bar{\alpha}_j^{(k+1)} = \bar{\alpha}_j^{(k)} + t_k \xi_j^{(k)}$$

where the steplength t_k is obtained by some standard steplength formula, such as

$$t_k = \lambda_k \frac{\bar{v} - g(\bar{\alpha}^{(k)})}{\|\xi^{(k)}\|^2}$$

where $0 < \varepsilon_1 \leq \lambda_k \leq 2 - \varepsilon_2$, $\varepsilon_2 > 0$ and $\bar{v}$ is an upper bound on v^*. If $\bar{v} = v^*$, one can prove asymptotic convergence for the objective function value ($g(\bar{\alpha}^{(k)}) \rightarrow v^*$), the dual α-solution and, due to the strict convexity of $f_j(y_j)$, the primal y-solution, but not the x-solution. If $\bar{v}$ is too large, we will take too long steps, which will result in slow convergence, so we obviously wish to use the best available upper bound. A strong and rather timeconsuming way to get an upper bound is to use the separable programming approach and solve SP. It is also possible to fix y and solve PS now and then during the procedure, in order to get upper bounds that might improve as the optimum is approached. An even quicker way is to use a simple primal heuristic.

We can use the heuristic HEUR to get a starting dual solution. Using $\theta = 1$ in HEUR, i.e. using the minimal costs, we get a very good lower bound when solving the first Lagrangean relaxation. However, when trying to improve this bound, we found that no improvement was obtained for a large number of iterations. We tried shortening the steplength quickly by decreasing λ_k quickly, but the results were still discouraging. The reason for this behaviour is, we believe, that this dual solution is very extreme, and lies at a nondifferentiable point of the dual function, $g(\alpha)$. This means that there is a large probability that the subgradients obtained are <u>not</u> ascent directions.

Using $\theta = 0$, i.e. the maximal costs, gave a much worse first iteration, but somewhat better convergence. However, the best results were found by using a convex combination of the two extremes. For $\theta = 0.1$, we managed to combine a rather good starting point with good convergence behaviour. Using this convex combination probably decreases the probability of the dual function being nondifferentiable at the starting point.

3.6.1 Modifications of subgradient optimization

Several modifications with the aim to improve the convergence in practice have been proposed. Especially interesting are the direction modifying techniques suggested in Camerini, Fratta and Maffioli [4] and Crowder [6]. The goal of these techniques is to decrease the zig-zagging. Computational results indicate that these modifications really help. We have found it best to use the modification in [6] with weight 0.6, i.e. add 0.6 times the old search direction to the new subgradient. We start with $\lambda_0 = 1.1$ and if no improvement is made in 5 iterations, the value of λ_k is halved. On the other hand, if a better upper bound is found, the value of λ_k is increased. Finally we note that is is most efficient <u>not</u> to do any line search in DS when using it in a subgradient method. (This means that constraint 1 in DSy is ignored.)

Methods implemented and tested are:

SO1 Subgradient optimization. The dual starting solution (and the upper bound) is obtained by the heuristic HEUR described in section 3.3.

SO2 Subgradient optimization with the primal subproblem. The dual starting solution is obtained by the heuristic HEUR. The primal subproblem PS is frequently solved for the y obtained by DSy, to get an upper bound.

SOSP1 Subgradient optimization and separable programming. The dual starting solution is obtained by the heuristic HEUR. The starting upper bound is obtained by solving the piecewise linearization SP.

SOSP2 Subgradient optimization and separable programming. The starting upper bound and the dual starting solution are obtained by solving the piecewise linearization SP.

The computational tests indicate that the first two methods are not competetive, i.e. separable programming and subgradient optimization are beneficial to use together.

3.7 Cross decomposition

In cross decomposition, Van Roy [22], one combines Benders decomposition and Lagrangean relaxation by replacing the difficult Benders master probem with the easier Lagrangean relaxation, DS, as much as possible. Here we must do a line search in DS, since the output is used in PS. In Holmberg and Jörnsten [14] this method is shown to be very efficient for the STP. However, we still had to use the Benders master problem a few times to solve the problem.

In this paper we have made some tests by only using the subproblem phase of cross decomposition, i.e. stopping when a master problem would have to be used. This method is denoted by **CS**, and has obviously no guaranteed convergence.

3.8 Mean value cross decomposition

Mean value cross decomposition is a new method, proposed in Holmberg [13] for linear problems. Convergence for LP-problems is shown in Holmberg [12] and convergence for general convex problems is shown in Vlahos [23]. It can be viewed as a modification of the ordinary cross decomposition method, and a generalization of the Kornai-Liptak method [15]. The main feature is that no master problems are needed.

In mean value cross decomposition we iterate between the two subproblems, PS (with line search) and DS, and as input to one we use the mean value of (a part of) the solutions of the other. Let, in iteration k, $\bar{\alpha}_j^k$ be the input to DS, $\bar{y}_j^k$ the input to PS, $\hat{\alpha}_j^k$ the solution of PS and $\hat{y}_j^k$ the solution of DS.

$$\bar{\alpha}_j^k = \frac{1}{k}\sum_{l=1}^{k}\hat{\alpha}_j^l \quad \text{and} \quad \bar{y}_j^k = \frac{1}{k}\sum_{l=1}^{k}\hat{y}_j^l$$

These mean values are updated iteratively as

$$\bar{\alpha}_j^{k+1} = \frac{1}{k}\hat{\alpha}_j^k + \frac{(k-1)}{k}\bar{\alpha}_j^{k-1}$$

and

$$\bar{y}_j^{k+1} = \frac{1}{k}\hat{y}_j^k + \frac{(k-1)}{k}\bar{y}_j^{k-1}$$

Note that $\bar{\alpha}$ and $\bar{y}$ are convex combinations of the subproblem solutions of PS and DS, and thus satisfy all constraints present in those problems. The method has asymptotic convergence for both primal and dual solutions as well as objective function value.

In this method the primal subproblem has to be solved for non-integral y's. Since most network codes work with integral data, this has to be accomplished by scaling and rounding. This introduces a small error, which however can be made small enough compared to the overall accuracy required.

Computational tests indicate that it is very important to start with fairly good solutions, since a bad starting solution may influence the mean values for many iterations. Therefore one should consider starting with heuristics to get good starting solutions. (Note that we may start with either a dual or a primal solution.) Another issue is whether one should include the starting solutions in the mean values or not.

Using the heuristic HEUR to get a starting solution to mean value cross decomposition, we found it best to use $\theta = 0.5$.

Methods implemented and tested are:

MVC1 Mean value cross decomposition, with the heuristic HEUR for the dual starting solution. The primal starting solution is not used in the mean values.

MVC2 Mean value cross decomposition with subgradient optimization. The starting dual solution is obtained by subgradient optimization (in at most 50 iterations). The starting primal solution is not used in the mean values.

MVC3 Mean value cross decomposition with separable programming. The starting dual solution is obtained as the dual solution to the piecewise linearization SP (with $Q = 6$). The corresponding primal solution is not used in the mean values.

MVC4 Mean value cross decomposition with separable programming. We start by solving the piecewise linearization SP (with $Q = 6$) and use both the primal and dual solution in the mean values.

We have also tried to replace PS in MVC2 by a piecewise linearization around $\bar{y}$, but that did not give any improvements in efficiency. Also letting the solution of SP be followed by subgradient optimization did not seem to improve the dual starting solution enough to make it worthwhile.

4 Computational results

We have solved a number of randomly generated problems in various sizes, from 10 origins and 100 destinations to 100 origins and 500 destinations. The largest problem has 50.000 x-variables and 500 y-variables, and is quite larger than the problems that are solved in the literature. The difficulty of the problems depends not only of the size of the problem, but also the relation between m and n (see table 4 in appendix B).

The supplies are drawn from a uniform distribution between 125 and 175. For each destination the shortage costs are drawn from a uniform distribution between 20 and 60 and the holding costs between 3 and 6. The probability density functions used are exponential distribution functions

(see appendix A), and the parameters λ_j are drawn from a uniform distribution between 0.005 and 0.025, which yields expectated demands in the interval between 40 and 200.

The coordinates for each point is generated randomly uniformly between 0 and 100, and the transportation cost coefficients are calculated as the Euclidean distance between the points multiplied by a constant. This constant is choosen such that the nonlinear costs in average is about 0.96 of the total costs at the optimum, which is how it was done in [5]. The relation between linear and nonlinear costs is important for the difficulty of the problem (in different ways for different methods). For a method based on linearization, the problem probably becomes easier if the importance of the linear part increases.

We have made some tests to see how changes in this constant affects the different methods, see figure 4. We have also scaled the supply up and down, in order to see how the methods are affected when the ratio between the total supply and the total expected demand varies, see figure 5.

The methods are run until the relative difference between the upper bound and the lower is less or equal to ε, which in our tests is set to 1%.

The methods we have tested are the following. We started with the ordinary Frank-Wolfe method, FW1, used in [5] (and there claimed to be the best). Then we introduced modifications similar to those suggested in [17], FW2 and FW3, which gave some improvement. A couple of other modifications were tested (FW4 and FW5) but did not give any significant improvements.

In table 1 the Frank-Wolfe methods FW1, FW2 and FW3 are compared. For FW2 the two cases $K = 3$, $N = 5$ and $K = 4$, $N = 3$ are investigated and for FW3 we have used $K = 2$, $L = 3$, since these setting were the best. We see that FW2 seems to be the best of the Frank-Wolfe methods, but the improvement compared to FW1 is not as large as indicated in [17]. (For a single problem, one might achieve larger improvements for certain parameter settings, but these settings will then increase the solution time for other problems.)

As for the forest iteration method, we note that in [21], problems of sizes up to $m = 34$ and $n = 44$ (which is much smaller than our problems) are solved. The numbers of iterations needed seem to be about the same size as m (when n is not much larger). The work done in one iteration seems to be more in the forest iteration method than in the Frank-Wolfe method, so since the number of iterations are in general <u>not</u> notably less than in FW2, we conclude that the forest iteration method cannot be more efficient than FW2. Therefore we have not implemented the forest iteration method.

The separable programming methods, SP1 and SP2, are not used by themselves, since they do not give any lower bounds, but computational tests indicate that the greedy heuristic SP2 is not strong enough.

In table 5 in appendix B we give (as an example) for problem p1 and method SP1, the size of the network, the total cost of the solution and the CPU-time needed to solve it, for different numbers of linear segments, Q. This is also shown in figure 2. While increasing the number of segments always leads to better solutions, the increase of CPU-time hardly makes it worthwhile after some limit.

We have implemented subgradient optimization with the upper bound and starting solutions obtained either by simple heuristics, SO1 and SO2, or by separable programming, SOSP1 and SOSP2. Here SO1 and SO2 are dominated by SOSP1 and SOSP2. Using the dual solution of

Problem: $m \times n$	Iter/Time FW1	Iter/Time (K/N) FW2: (3/5)	Iter/Time (K/N) FW2: (4/3)	Iter/Time (K/L) FW3: (2/3)
p1:25x200	37/6.21	24/4.59	30/5.89	17/3.10
p2:50x300	67/29.91	32/16.34	33/17.45	41/19.36
p3:10x100	17/0.75	23/1.16	32/1.68	14/0.68
p4:25x200	39/6.49	24/4.68	26/5.11	23/4.01
p5:50x300	60/26.23	34/17.53	35/18.65	38/17.99
p6:100x500	80/108.41	49/75.74	46/73.59	57/84.05
p7:50x200	88/26.01	62/20.31	53/18.03	64/21.17
p8:100x200	137/74.36	123/72.73	127/76.00	131/81.93
p9:40x400	38/18.14	17/9.85	18/10.82	24/12.84
p10:30x500	21/9.79	11/6.19	9/5.33	14/7.09
p11:70x300	90/53.14	61/39.98	60/41.03	66/43.14
p12:80x200	125/55.57	106/51.40	101/49.77	110/54.51
p13:100x200	152/82.79	141/82.60	140/83.21	144/85.73
p14:100x300	140/114.35	112/99.03	112/101.25	113/100.70
p15:100x400	106/115.21	80/95.86	78/95.83	83/98.57
p16:90x300	111/82.68	88/71.34	83/69.09	84/67.91

Table 1: Number of iterations and CPU-time in seconds to get a relative error less than 1%.

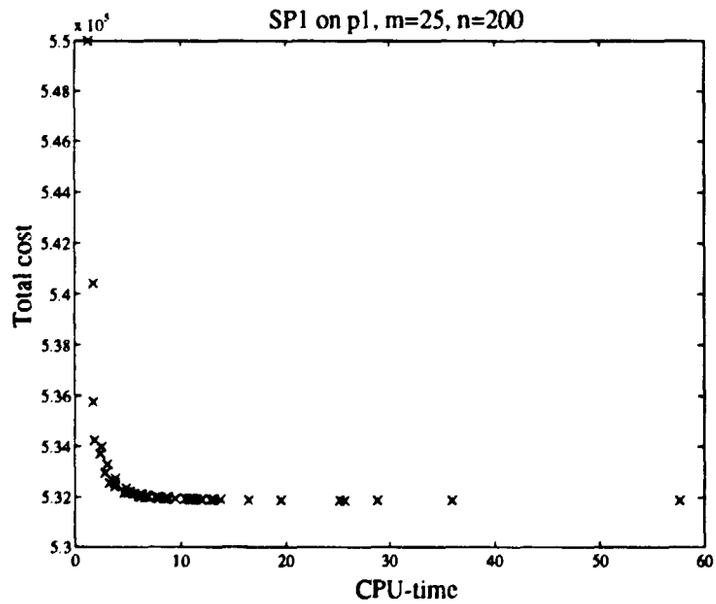


Figure 2: Total cost vs. CPU-time for different number of segments in SP. $Q = 3, 4, 5, \ldots, 49, 50, 60, 70, 80, 90, 100, 120, 150$.

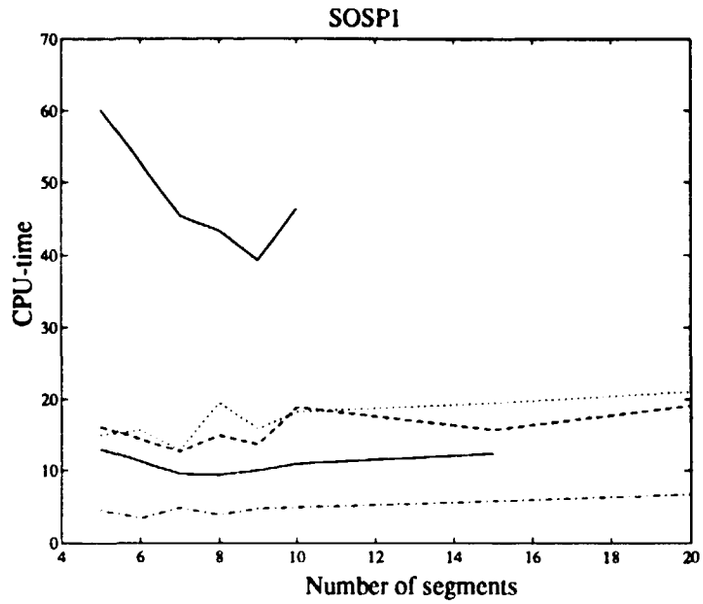


Figure 3: Comparison of CPU-times for different numbers of segments in the SP-part of SOSP1. p1: dash-dotted line, p2: dotted line, p5: dashed line and p6 and p7: solid lines.

SP in the Lagrangean relaxation directly (without any subgradient steps) does not produce a lower bound that is good enough (the relative error is not always less than 1%).

In table 6 in appendix B we give the number of iterations and the CPU-time needed to solve SOSP1, for different numbers of linear segments, Q. This is also shown in figure 3 for a few problems. Obviously the curves have some minimum at different numbers, but we have chosen to use $Q = 9$.

The subproblem phase in cross decomposition, CS, gave errors around 1% - 10%, but could not proceed from that. This makes the method too weak.

We have also implemented and tested mean value cross decomposition, MVC1, and different combinations of mean value cross decomposition and subgradient optimization or separable programming, MVC2, MVC3 and MVC4. These methods are the ones that we find most promising. In general, much of the solution time is spent in either the subgradient optimization part or the separable programming network problem, and the mean value cross decomposition framework is mainly used to wrap things up, add the missing bound and sharpen the existing bound.

In table 2 we compare the different mean value cross decomposition methods described in section 3.8. MVC3 and MVC4 seem to be the best methods, in spite of the fact that MVC1 is quicker on the first four problems as well as on some of the easiest ones. One can note that only one iteration is needed, and the main part of the CPU-time lies in solving SP.

In table 7 in appendix B the same comparison is made for problems p17 and p18, where the relation between the nonlinear and linear parts of the costs in the optimal solution is varied. Also the relation between the total supply and the total expected demand is varied.

In table 3 we compare some of the best methods of different types, MVC4, FW2, SOSP1 and

Problem: $m \times n$	Iter/Time MVC1	Iter/Time/SO MVC2	Iter/Time/SP MVC3	Iter/Time/SP MVC4
p1:25x200	1/1.00	1/3.40/2.74	1/2.04/1.95	1/2.00/1.91
p2:50x300	2/8.88	1/12.17/6.93	1/9.47/9.28	1/9.64/9.45
p3:10x100	2/0.50	1/1.01/0.80	1/0.63/0.59	1/0.65/0.60
p4:25x200	2/2.91	1/3.83/2.77	1/2.98/2.88	1/2.97/2.88
p5:50x300	4/22.30	1/12.63/6.85	1/7.18/7.00	1/7.24/7.06
p6:100x500	3/43.67	1/40.57/21.08	1/25.47/24.99	1/25.39/24.92
p7:50x200	4/10.92	1/6.93/4.51	1/5.60/5.43	1/5.62/5.46
p8:100x200	8/49.55	2/21.22/8.40	1/11.30/11.18	1/11.40/11.28
p9:40x400	2/10.51	1/12.98/7.54	1/10.10/9.90	1/10.12/9.92
p10:30x500	1/3.61	1/11.87/7.57	1/12.19/11.98	1/12.32/12.11
p11:70x300	4/33.38	1/17.52/9.27	1/12.51/12.23	1/12.52/12.23
p12:80x200	4/15.87	1/11.09/6.88	1/8.59/8.42	1/8.49/8.33
p13:100x200	8/40.67	1/15.30/8.46	1/7.88/7.69	1/7.75/7.57
p14:100x300	4/51.96	1/23.10/12.69	1/16.85/16.55	1/16.82/16.52
p15:100x400	3/36.68	1/29.29/16.85	1/28.75/28.33	1/28.64/28.22
p16:90x300	3/21.77	1/16.91/11.50	1/11.44/11.14	1/11.43/11.13
p17c:50x300	2/11.07	1/11.05/7.01	1/6.75/6.57	1/6.68/6.50
p18b:100x200	5/40.02	1/15.98/8.53	1/12.09/11.91	1/12.25/12.07

Table 2: Number of iterations and CPU-time in seconds to get a relative error less than 1%.

SOSP2. In table 8 in appendix B the comparison is made for problems p17 and p18 in the same way as in table 7. This is also illustrated in the figures 4 and 5, where we can see that MVC4 exhibits a very stable behaviour while FW2, MVC1 and SOSP1 fluctuates significantly. We can note the increase in time for FW2 and SOSP1 as the nonlinearity proportion increases above 0.96 (for problem p17 with $m = 50$, $n = 300$, which is the maximal size used in [5], the problem is easiest to solve for the quotient around 0.95).

5 Conclusions

We find that MVC4 is the best method, i.e. starting with solving a separable programming problem with 9 segments, and then using both the dual and the primal solutions as starting values for mean value cross decomposition. Other methods, such as MVC3, SOSP1 and SOSP2 are also quite efficient.

As examples of solutions times we solved a 100×500-problem (p6) in 25 seconds, a 100×200-problem (p13) in 8 seconds and a 25×200-problem (p1) in 2 seconds. The corresponding solution times for the original Frank-Wolfe method was 108, 83 and 6 seconds, which yields speed-up factors of about 4, 10 and 3. For the modified Frank-Wolfe method we got (for the standard setting $K = 4$ and $N = 3$) 74, 83 and 6 seconds. The average speed-up factor (quotient between CPU-times) between MVC4 and FW2 for the problems p1 - p16 is close to 4 (actually 3.82).

Our conclusion is thus that combining separable programming with mean value cross decomposition yields a very efficient solution method for the STP. Also combinations between separable programming and subgradient optimization and between subgradient optimization and mean

Problem: m×n	Iter/Time MVC4	Iter/Time FW2	Iter/Time SOSP1	Iter/Time (err) SOSP2
p1:25x200	1/2.00	30/5.97	30/4.62	1/3.36
p2:50x300	1/9.64	33/17.24	47/15.16	43/15.58
p3:10x100	1/0.65	32/1.69	30/1.06	1/0.73
p4:25x200	1/2.97	26/5.20	44/4.74	1/2.99
p5:50x300	1/7.24	35/18.31	29/13.21	48/16.32
p6:100x500	1/25.39	46/73.23	29/37.36	68/55.08
p7:50x200	1/5.62	53/17.93	47/9.63	65/11.66
p8:100x200	1/11.40	127/75.11	74/24.07	1/14.05
p9:40x400	1/10.12	18/10.42	45/17.07	1/12.23
p10:30x500	1/12.32	9/5.12	28/17.09	1/14.10
p11:70x300	1/12.52	60/40.33	39/18.99	67/24.22
p12:80x200	1/8.48	101/49.89	59/15.63	10001/1272.52 (1.57%)
p13:100x200	1/7.75	140/82.75	80/25.70	172/40.63 (1.22%)
p14:100x300	1/16.82	112/100.02	68/34.45	96/42.69
p15:100x400	1/28.64	78/95.15	49/44.96	10001/3133.13 (1.49%)
p16:90x300	1/11.43	83/68.46	50/24.67	53/25.95

Table 3: Number of iterations and CPU-time in seconds to get a relative error less than 1%.

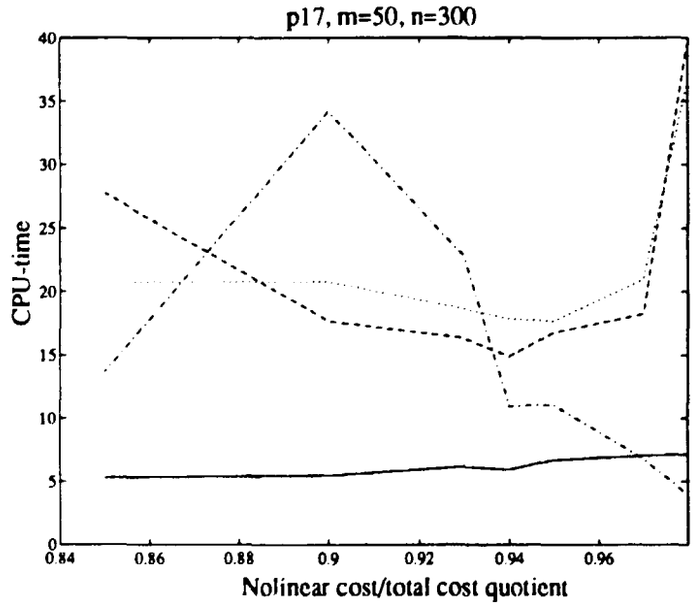


Figure 4: Comparison of CPU-times for different quotients between the nonlinear costs and the total costs at the optimum. MVC1: dash-dotted line, MVC4: solid line, SOSP1: dashed line and FW2: dotted line.

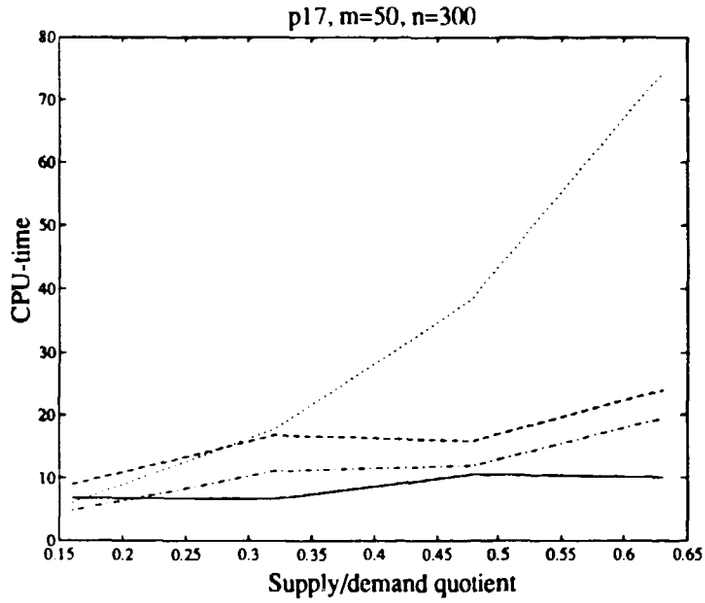


Figure 5: Comparison of CPU-times for different quotients between the total supply and the total expected demand. MVC1: dash-dotted line, MVC4: solid line, SOSP1: dashed line and FW2: dotted line.

value cross decomposition yield methods that are significantly better than the Frank-Wolfe method (with or without modifications).

Since we have compared (as fairly as possible) most solution methods proposed for this problem, except the forest iteration method (which is argued to be inferior to the modified Frank-Wolfe method), we believe that these result might be hard to beat.

As future research we plan to use these solution methods as subroutines when solving the stochastic location-transportation problem and generalizations thereof.

Bibliography

[1] M. S. Bazaraa and C. M. Shetty, Nonlinear Programming: Theory and Algorithms. John Wiley & Sons, New York, 1979.

[2] J. F. Benders, "Partitioning Procedures for Solving Mixed-Variables Programming Problems", Numerische Matematik, vol. 4 pp. 238–252, 1962.

[3] D. P. Bertsekas and P. Tseng, "Fortran Codes for Network Optimization: The Relax Codes for Linear Minimum Cost Network Flow problems", Annals of Operations Research, vol. 13 pp. 125–190, 1988.

[4] P. M. Camerini, L. Fratta, and F. Maffioli, "On Improving Relaxation Methods by Modified Gradient Techniques", Mathematical Programming Study, vol. 3 pp. 26–34, 1975.

[5] L. Cooper and L. J. LeBlanc, "Stochastic Transportation Problems and Other Network Related Convex Problems", Naval Research Logistics Quarterly, vol. 24 pp. 327–336, 1977.

[6] H. Crowder. Computational improvements for subgradient optimization. In Symposia Mathematica, Vol XIX, 357–372. Academic Press, London, 1976.

[7] G. B. Dantzig and P. Wolfe, "Decomposition Principle for Linear Programs", Operations Research, vol. 8 pp. 101–111, 1960.

[8] S. E. Elmaghraby, "Allocation Under Uncertainty when the Demand has Continuous D.F.", Management Science, vol. 6 pp. 270–294, 1960.

[9] M. Frank and P. Wolfe, "An Algorithm for Quadratic Programming", Naval Research Logistics Quarterly, vol. 3 pp. 95–110, 1956.

[10] M. Held, P. Wolfe, and H. P. Crowder, "Validation of Subgradient Optimization", Mathematical Programming, vol. 6 pp. 62–88, 1974.

[11] K. Holmberg. Separable programming applied to the stochastic transportation problem. Research Report LiTH-MAT-R-1984-15, Department of Mathematics, Linköping Institute of Technology, Sweden, 1984.

[12] K. Holmberg. A convergence proof for linear mean value cross decomposition. Research Report 90/12, Operationsanalyse, Matematisk Institut, Aarhus University, Denmark, 1990. Accepted for publication in *Zeitschrift für Operations Research.*

[13] K. Holmberg, "Linear Mean Value Cross Decomposition: A Generalization of the Kornai-Liptak Method", European Journal of Operational Research, vol. 62 pp. 55–73, 1992.

[14] K. Holmberg and K. Jörnsten, "Cross Decomposition Applied to the Stochastic Transportation Problem", European Journal of Operational Research, vol. 17 pp. 361–368, 1984.

[15] J. Kornai and T. Liptak, "Two-Level Planning", Econometrica, vol. 33 pp. 141–169, 1965.

[16] T. Larsson and A. Migdalas, "An Algorithm for Nonlinear Programs over Cartesian Product Sets", Optimization, vol. 21 pp. 535–542, 1990.

[17] L. J. LeBlanc, R. V. Helgason, and D. E. Boyce, "Improved efficiency of the Frank-Wolfe algorithm for convex network programs", Transportation Science, vol. 19 pp. 445–462, 1985.

[18] M. Patriksson. Partial linearization in nonlinear programming. Research Report LiTH-MAT-R-1991-11, Department of Mathematics, Linköping Institute of Technology, Sweden, 1991. Accepted for publication in *JOTA*.

[19] B. T. Poljak, "A General Method of Solving Extremum Problems", Soviet Mathematics Doklady, vol. 8 pp. 593-397, 1967.

[20] B. T. Poljak, "Minimization of Unsmooth Functionals", USSR Computational Mathematics and Mathematical Physics, vol. 9 pp. 14-29, 1969.

[21] L. Qi, "Forest Iteration Method for Stochastic Transportation Problem", Mathematical Programming Study. vol. 25 pp. 142-163, 1985.

[22] T. J. Van Roy, "Cross Decomposition for Mixed Integer Programming", Mathematical Programming, vol. 25 pp. 46-63, 1983.

[23] K. Vlahos. Convergence proof for mean cross decomposition applied to convex problems. Research paper, Decision Science Group, London Business School, England, 1991.

[24] A. Weintraub, C. Ortiz, and J. Gonzáles, "Accelerating Convergence of the Frank-Wolfe Algorithm", Transportation Research, vol. 19 pp. 113-122, 1985.

[25] A. C. Williams, "A Stochastic Transportation Problem", Operations Research, vol. 11 pp. 759-770, 1963.

[26] D. Wilson, "An a Priori Bounded Model for Transportation Problems with Stochastic Demand and Integer Solutions", AIIE Transactions, vol. 4 pp. 186-193, 1972.

[27] D. Wilson, "Tighter Bounds for Stochastic Transportation Models", AIIE Transactions, vol. 5 pp. 180-185, 1973.

Appendix

A Exponentially distributed demand

In our testexamples we have used exponentially distributed demand, which yields

$$\phi_j(d_j) = \lambda_j \exp(-\lambda_j d_j)$$

$$F_j(d_j) = 1 - \exp(-\lambda_j d_j)$$

$$E_j[d_j] = \frac{1}{\lambda_j}$$

$$f_j(y_j) = h_j \left(y_j - \frac{1}{\lambda_j} \right) + \left(\frac{p_j + h_j}{\lambda_j} \right) \exp(-\lambda_j y_j)$$

$$\check{y}_j = \frac{1}{\lambda_j} \ln \left(1 + \frac{p_j}{h_j} \right)$$

In the application of separable programming we get

$$T_j = \frac{1}{\lambda_j Q_j} \ln \left(1 + \frac{p_j}{h_j} \right)$$

and

$$d_{jk} = h_j + \frac{(p_j + h_j)}{\lambda_j T_j} (1 + \exp(\lambda_j T_j)) \exp(-\lambda_j k T_j)$$

In the solution of the Lagrangean relaxation we get, if $-p_j \leq \bar{\alpha}_j + \bar{\delta} \leq h_j$,

$$\hat{y}_j = -\frac{1}{\lambda_j} \ln \left(1 - \frac{p_j + \bar{\alpha}_j + \bar{\delta}}{p_j + h_j} \right)$$

B Tables of computational results

Problem	$m \times n$	Supply/Demand	Nonlin/total costs	m/n
p1	25x200	0.23	0.96	8
p2	50x300	0.32	0.96	6
p3	10x100	0.19	0.96	10
p4	25x200	0.24	0.97	8
p5	50x300	0.31	0.94	6
p6	100x500	0.37	0.96	5
p7	50x200	0.47	0.92	4
p8	100x200	0.91	0.86	2
p9	40x400	0.19	0.97	10
p10	30x500	0.11	0.99	6
p11	70x300	0.44	0.94	4.3
p12	80x200	0.74	0.87	2.5
p13	100x200	0.94	0.84	2
p14	100x300	0.66	0.88	3
p15	100x400	0.48	0.93	4
p16	90x300	0.57	0.91	3.3
p17a	50x300	0.32	0.98	6
p17b	50x300	0.32	0.97	6
p17c	50x300	0.32	0.95	6
p17d	50x300	0.32	0.94	6
p17e	50x300	0.32	0.93	6
p17f	50x300	0.32	0.90	6
p17g	50x300	0.32	0.85	6
p17a1	50x300	0.16	0.98	6
p17a2	50x300	0.48	0.91	6
p17a3	50x300	0.63	0.88	6
p18a	100x200	0.90	0.95	2
p18b	100x200	0.90	0.83	2
p18c	100x200	0.90	0.78	2
p18d	100x200	0.18	0.98	2
p18e	100x200	0.45	0.93	2

Table 4: Quotients: the total supply/the total expected demand, the nonlinear costs/the total costs at the optimum and n/m.

Q	Nodes x Arcs	Cost	Time
2	229x5428	564489	1.31
3	230x5629	549997	1.28
4	231x5830	540412	1.78
5	232x6031	535762	1.76
6	233x6232	534238	1.89
7	234x6433	533984	2.56
8	235x6634	533708	2.44
9	236x6835	533291	3.11
10	237x7036	532956	2.86
11	238x7237	532750	3.82
12	239x7438	532571	3.79
13	240x7639	532558	3.28
14	241x7840	532410	3.79
15	242x8041	532353	4.83
16	243x8242	532204	5.21
17	244x8443	532188	5.13
18	245x8644	532199	5.02
19	246x8845	532174	4.70
20	247x9046	532134	5.54
21	248x9247	532102	6.10
22	249x9448	532093	6.34
23	250x9649	532058	6.36
24	251x9850	532068	6.80
25	252x10051	532041	6.04
26	253x10252	532019	6.24
27	254x10453	531998	6.65
28	255x10654	532006	6.95
29	256x10855	531997	7.67
30	257x11056	531991	7.81
35	262x12061	531936	8.87
40	267x13066	531924	10.49
45	272x14071	531916	11.93
50	277x15076	531900	13.77
60	287x17086	531889	16.48
70	297x19096	531878	19.55
80	307x21106	531873	25.69
90	317x23116	531871	25.22
100	327x25126	531868	28.87
120	347x29146	531862	36.06
150	377x35176	531857	57.56
200	427x45226	531855	79.95

Table 5: Size of network in SP, cost and solution time for p1:25x200.

Q	5	7	8	9	10	15
Problem	Iter/Time					
p1:25x200	53/4.48	41/4.83	26/3.87	30/4.70	35/4.89	14/5.73
p2:50x300	61/14.93	37/12.75	47/19.48	47/15.84	51/18.24	22/19.43
p3:10x100	45/1.11	30/0.97	26/1.00	30/1.07	15/0.93	12/1.15
p4:25x200	63/5.29	37/4.30	36/4.38	44/4.85	36/6.87	19/5.52
p5:50x300	71/16.08	40/12.59	47/14.87	29/13.60	45/18.87	23/15.66
p6:100x500	98/60.06	42/45.39	32/43.34	29/39.29	28/46.41	
p7:50x200	97/12.92	48/9.48	46/9.37	47/9.97	42/10.88	28/12.29
p8:100x200	95/23.37	89/23.51	74/21.67	74/24.53	70/23.55	67/24.36
p9:40x400	51/17.89	40/14.19	28/13.61	45/17.40	33/15.29	22/18.37
p10:30x500	77/17.86	32/19.15	31/19.77	28/17.26	18/13.91	16/17.48
p11:70x300	80/23.45	38/18.90	38/19.4[illegible]	39/18.69	41/19.41	25/21.73
p12:80x200	81/17.61	82/17.55	66/16.37	59/15.65	55/17.51	54/18.85
p13:100x200	159/33.88	102/25.69	88/23.08	80/25.77	75/22.30	72/23.15
p14:100x300	388/108.20	94/49.94	74/41.59	68/35.04	66/37.76	58/40.11
p15:100x400	180/78.90	46/36.62	54/43.68	49/45.45	66/54.21	35/46.88
p16:90x300	115/35.73	58/27.54	53/28.30	50/24.97	45/24.02	39/29.44

Table 6: Number of iterations and CPU-time for different numbers of linear segments in SOSP1.

Problem: $m \times n$	Iter/Time MVC1	Iter/Time/SO MVC2	Iter/Time/SP MVC3	Iter/Time/SP MVC4
p17a:50x300	1/3.95	3/15.09/6.95	1/7.29/7.10	1/7.12/6.93
p17b:50x300	2/6.76	1/10.80/7.04	1/6.92/6.72	1/7 01/6.81
p17c:50x300	2/11.07	1/11.05/7.01	1/6.75/6.57	1/6.68/6.50
p17d:50x300	3/10.94	1/12.38/7.18	1/5.87/5.68	1/5.92/5.74
p17e:50x300	4/22.76	1/11.63/7.01	1/6.12/5.93	1/6.14/5.95
p17f:50x300	9/34.19	1/11.41/6.88	1/5.71/5.52	1/5.46/5.36
p17g:50x300	10/13.69	1/8.17/7.06	1/5.27/5.09	1/5.29/5.09
p17h:50x300	22/15.44	2/7.48/6.59	1/4.05/3.96	1/3.92/3.83
p17a1:50x300	1/4.84	1/10.49/7.18	1/6.76/6.59	1/6.78/6.62
p17a2:50x300	3/11.94	1/12.02/6.98	1/10.61/10.34	1/10.55/10.28
p17a3:50x300	4/19.37	2/12.33/6.85	1/10.11/9.87	1/10.03/9.79
p18a:100x200	2/9.67	2/16.29/8.51	1/10.37/10.17	1/10.56/10.36
p18b:100x200	5/40.02	1/15.98/8.53	1/12.09/11.91	1/12.25/12.07
p18c:100x200	15/61.74	1/16.89/8.69	1/12.73/12.62	1/12.91/12.78
p18d:100x200	2/5.98	1/13.46/8.57	1/8.32/8.15	1/8.18/8.02
p18e:100x200	3/15.16	1/16.06/8.66	1/10.88/10.62	1/11.06/10.84

Table 7: Number of iterations and CPU-time in seconds to get a relative error less than 1%.

Problem: $m \times n$	Iter/Time MVC4	Iter/Time FW2	Iter/Time SOSP1
p17a:50x300	1/7.12	76/36.69	75/39.77
p17b:50x300	2/7.01	41/20.95	65/18.22
p17c:50x300	2/6.68	34/17.69	46/16.77
p17d:50x300	3/5.92	34/17.85	37/14.89
p17e:50x300	4/6.14	36/18.62	54/16.33
p17f:50x300	9/5.46	39/20.77	69/17.66
p17g:50x300	10/5.29	41/20.73	174/27.75
p17a1:50x300	1/6.78	11/6.02	16/8.93
p17a2:50x300	3/10.55	79/38.68	35/15.84
p17a3:50x300	4/10.03	154/74.21	105/23.91
p18a:100x200	2/10.56	243/147.86	97/44.82
p18b:100x200	5/12.25	127/78.51	70/24.75
p18c:100x200	15/12.91	102/62.54	62/22.56
p18d:100x200	2/8.18	18/12.78	6/10.01
p18e:100x200	3/11.06	250/147.87	213/43.90

Table 8: Number of iterations and CPU-time in seconds to get a relative error less than 1%.

Avdelning, institution, fakultet
Division, department, faculty

Optimization
Department of Mathematics
Linköping Institute of Technology

ISBN:

ISSN: 0348-2960

Rapportnr:
Report no: LiTH-MAT-R 1993-10

Upplagans storlek:
Number of copies: 130

Datum:
Date:

Projekt:
Project:

Titel:
Title:

EFFICIENT DECOMPOSITION AND LINEARIZATION METHODS FOR THE STOCHASTIC TRANSPORTATION PROBLEM

Författare:
Author: Kaj Holmberg

Uppdragsgivare:
Commissioned by:

Dnr.:
Call no:

Rapporttyp:
Kind of report:

- ☐ Examensarbete/Final project
- ☐ Delrapport/Progress report
- ☐ Reserapport/Travel report
- ☐ Slutrapport/Final report
- ☒ Övrig rapport/Other kind of report

Rapportspråk:
Language:

- ☐ Svenska/Swedish
- ☒ Engelska/English
- ☐ _____

Sammanfattning (högst 150 ord):
Abstract (150 words):

The stochastic transportation problem can be formulated as a convex transportation problem with nonlinear objective function and linear constraints. We compare several different methods based on decomposition techniques and linearization techniques for this problem, trying to find the most efficient method or combination of methods. We discuss and test a separable programming approach, the Frank-Wolfe method with and without modifications, the new technique of mean value cross decomposition and the more well known Lagrangean relaxation with subgradient optimization, as well as combinations of these approaches. Computational tests are presented, indicating that some new combination methods are quite efficient for large scale problems.

Nyckelord *(högst 8)*
Keywords (8): Transportation, Decomposition methods, Separable Programming, the Frank-Wolfe method.

Bibliotekets anteckningar