Applications of Simplicial Decomposition with Nonlinear Column Generation to Nonlinear Network Flows

Torbjörn Larsson, Michael Patriksson, and Clas Rydergren

Department of Mathematics, Linköping University, S-581 83 Linköping, Sweden

Abstract. The simplicial decomposition method for linearly constrained nonlinear programs has been proven to be efficient for certain classes of structured large-scale problems. This method alternates between a multi-dimensional search over a restriction of the feasible set and an augmentation of the restriction through the solution of a linear column generation problem. The quality of the columns generated may however be very poor, since the first-order approximation of the objective function is used globally.

As a means for improving the quality of the columns in a simplicial decomposition method, we consider using *nonlinear* (e.g., second-order) approximations of the objective function in the augmentation phase. This modification leads to the generation of non-extreme point columns. The new method is applied to two nonlinear network flow models: the traffic assignment problem and the stochastic transportation problem. Numerical experiments indicate that significant time and storage savings can be gained from the proposed modification. These observations are particularly interesting due to the fact that similar nonlinearities may be incorporated in any column generation scheme.

1 Introduction

Linearly constrained, pseudo-convex minimization problems arise in a large variety of applications in the engineering and decision sciences. The successful outcome of such applications often depends on the availability of efficient and robust computer codes, and there are several general-purpose codes available (see, e.g., Moré and Wright, 1993; and Nash, 1995, for surveys of such codes). General-purpose codes are, however, insufficient for solving (very) large-scale problems. The rescue from this situation is the fact that such problems usually have some inherent structure that may be exploited in the construction of special-purpose methods.

A possible principle for the construction of such algorithms is that of *simplicial decomposition*, developed in the works of Holloway (1974) and von Hohenbalken (1977). The foundation for this class of algorithms is Carathéodory's Theorem, by which a bounded polyhedral set may be given an inner representation in terms of its extreme points. A simplicial decomposition scheme alternates between a multi-dimensional search over a restriction of the feasible set which is spanned by a set of known extreme points (the *restricted master problem*), and the solution of a linear program which is defined by a first-order approximation of the objective function (the *column generation subproblem*). Unless the solution to the current master problem is optimal in the original problem, any solution to this linear program defines a larger restriction.

This method is best suited for, and has been most successful in, applications where the linear constraints have structures that can be exploited in the solution of the linear subproblems; examples of such structures are constraints defining network flow polyhedra (e.g., Hearn et al., 1987; and Mulvey et al., 1990) or Cartesian product sets (Larsson and Patriksson, 1992). Further, whenever the objective function is convex, the linear subproblem provides a lower bound on the optimal value of the original problem; the availability of such bounds is a simple and direct means for monitoring the convergence of the algorithm and for constructing termination criteria.

The purpose of this work is to demonstrate that the potential of simplicial decomposition methods has not yet been fully explored; specifically, we establish through two applications that the quality of the columns generated may be significantly improved by using *strictly convex* column generation problems. Moreover, the construction of the nonlinear subproblems and the choices of algorithms for their solution may be tailored to structures in the original problem, thus enabling the efficient generation of columns of high quality. The result of this is much less iterations and, as a consequence, a much smaller final master problem; further, it results in a better overall performance of the algorithm. This work also illustrates the potential of using nonlinear subproblems in column generation methods (and, the dually related cutting plane methods) in general.

The remainder of the paper is organized as follows. In the next section, we review the development of simplicial decomposition methods. In Section 3, the new column generation principle is introduced, the resulting simplicial decomposition algorithm is presented and its convergence is established. Section 4 presents results from numerical experiments with two nonlinear network flow problems. In Section 5, we draw some general conclusions.

2 Background and Motivation

Consider the problem

[NLP]

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

where the feasible set $X \subset \mathbb{R}^n$ is a nonempty and bounded polyhedron described by a set of linear constraints, and the objective function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is pseudoconvex and continuously differentiable on X. Let f^* and X^* denote the optimal value and the set of optimal solutions to [NLP], respectively.

In order to provide a background to and a motivation for the proposed method, we review below linearization based approaches for the problem [NLP].

2.1 The Frank–Wolfe Method

The Frank–Wolfe method is a well-known solution method for the above problem and is frequently used for the solution of certain nonlinear network flow problems (e.g., LeBlanc et al., 1975; LeBlanc et al., 1985; and Patriksson, 1994). The Frank–Wolfe method alternates between the solution of a linear approximation of the original problem and a line search towards the solution to the linear subproblem. Given an iterate x^t, the linear subproblem is

[LP]

$$\min_{x \in X} \underline{f}(x) = f(x^t) + \nabla f(x^t)^T (x - x^t),$$

an optimal solution of which is denoted by y^t. The next iterate, x^{t+1}, is obtained through the minimization of f along the line segment between x^t and y^t.

In cases where the objective function is convex, $\underline{f}(y^t) \leq f^*$ holds, that is, the solution to [LP] provides a lower bound on the optimal value of [NLP]. This lower bound converges to f^* and is therefore a natural convergence measure for the algorithm.

The Frank–Wolfe method has reached some popularity, partly due to its simplicity. Further, any recognizable structure in the constraints of [NLP] can be exploited in the solution of the subproblem [LP]; whenever the set X is defined by network constraints, for example, [LP] may be solved by specialized network flow methods. Moreover, the storage requirements of the algorithm are small. The rate of improvement in the objective value in the initial iterations is usually quite high, which makes the method useful in contexts where rough estimates of the optimal solution are satisfactory. Asymptotically, however, the rate of convergence is only sublinear (Canon and Cullum, 1968; and Wolfe, 1970), due to the inherent property that

$$\{\nabla f(x^t)^T (y^t - x^t)\} \to 0, \quad \text{whereas} \quad \{y^t - x^t\} \not\to 0; \tag{1}$$

this property stems from the global use in [LP] of the (local) first-order approximation of f.

The poor convergence of the Frank–Wolfe method has led to the development of several modifications and extensions of the original scheme. Examples of simple modifications are the Partan directions approach (LeBlanc et al., 1985) and the step length enlargement modification of Weintraub et al. (1985).

An extension of the Frank–Wolfe method which has a bearing to the principle behind the proposed algorithm is the *regularized* Frank–Wolfe algorithm (Migdalas, 1994). In this algorithm, the linear objective of [LP] is regularized through the addition of a strictly convex function; this function acts as a penalty for the subproblem solution to lie far from the current iterate, thereby preventing the generation of extreme point subproblem solutions as well as invalidating the second statement in property (1). (An interpretation of the penalty function is that of introducing an implicit trust region which restricts the use of the linear approximation to a neighbourhood of the current iterate.)

Most improvements suggested for the Frank–Wolfe algorithm are based on modifications of its subproblem phase in order to construct search directions with better descent properties. The major contribution among the research which aims at improving the line search phase of the algorithm is the development of simplicial decomposition methods.

2.2 The Simplicial Decomposition Method

The class of simplicial decomposition methods is founded on Carathéodory's Theorem, which states that any point in a nonempty, compact and convex set lies in the relative interior of a simplex whose vertices are extreme points of the set. An immediate consequence of this result is that any point in a bounded polyhedral set can be expressed as a convex combination of its extreme points, that is, that the set can be given an inner representation in terms of its vertices.

In [NLP], let the extreme points of X be denoted by $x^{(p)}$, $p \in \mathcal{P}$. A point $x \in \mathbb{R}^n$ is then contained in X if and only if

$$x = \sum_{p \in \mathcal{P}} \lambda_p x^{(p)}; \qquad \sum_{p \in \mathcal{P}} \lambda_p = 1; \qquad \lambda_p \geq 0, \quad p \in \mathcal{P}$$

holds for some vector $\lambda \in \mathbb{R}^{|\mathcal{P}|}$ of weights; [NLP] may then be equivalently stated in terms of these convexity weights as the (*complete*) *master problem*

[MP]

$$\begin{aligned} \min\ & f\left(\sum_{p \in \mathcal{P}} \lambda_p x^{(p)}\right) \\ \text{s.t.}\ & \sum_{p \in \mathcal{P}} \lambda_p = 1, \\ & \lambda_p \geq 0, \quad p \in \mathcal{P}. \end{aligned}$$

This formulation has very simple constraints, but requires the explicit enumeration of the extreme points; further, normally the number of extreme points is very large, and they are a priori unknown. The extreme points of X that are necessary to span an optimal solution to [NLP] are therefore generated algorithmically, as follows.

Assume that, in some iteration t, $t+1$ extreme points y^i of X are known. Introducing the notation

$$x(\lambda) = \lambda_x x^t + \sum_{i=0}^{t} \lambda_i y^i,$$

x^{t+1} solves the corresponding *restricted master problem*

[RMP]

$$\begin{aligned}
&\min f(x(\lambda)) \\
&\text{s.t. } \lambda_x + \sum_{i=0}^{t} \lambda_i = 1, \\
&\qquad \lambda_x, \lambda_i \geq 0, \quad i = 0, \ldots, t.
\end{aligned}$$

We let $X_{t+1} = \text{conv}\,(x^t \cup_{i=0}^t y^i)$ denote the feasible set of [RMP], and X_{t+1}^* its set of optimal solutions. Given an optimal solution $x^{t+1} \in X_{t+1}^*$ to [RMP], a new column (i.e., an extreme point) is generated by solving the linear program [LP]. If the solution, y^{t+1}, to [LP] is already contained in X_{t+1}, then x^{t+1} solves [NLP] (this is essentially the termination criterion of the Frank–Wolfe algorithm); otherwise, y^{t+1} is added to the set of extreme points, defining X_{t+2}, and the process is repeated with $t := t + 1$.

We note that as long as all columns generated are retained in [RMP], x^t is contained in the convex hull of $y^0, \ldots, y^t$, and therefore is redundant in the formulation of [RMP]; we choose to include the iterate x^t in [RMP] partly for clarity, but mainly because it is necessary in order to ensure convergence whenever columns with non-zero weights are dropped from the set of columns retained.

Since the number of extreme points of X is finite, the simplicial decomposition method has finite convergence; this result holds also if columns that receive zero weights in a master problem are removed (von Hohenbalken, 1977).

In principle, the problem [RMP] is as difficult to solve as the original problem [NLP]. The special structure of its constraints ([RMP] is essentially unconstrained), however, enables the construction of specialized methods for its solution, and a new master problem is efficiently reoptimized starting from the (approximate) solution to the previous master problem. The convergence of an algorithm for [RMP] may be monitored by the use of a lower bound, equal to that provided in a Frank–Wolfe algorithm applied to [RMP]; the simplicity of the constraints of [RMP] makes its computation very simple: given a point λ, feasible in [RMP], a lower bound on the optimal objective value of [RMP] is

$$f(x(\lambda)) - \nabla_\lambda f(x(\lambda))^T \lambda + \min_{i \in \{x,0,1,\ldots,t\}} \left\{ \frac{\partial f(x(\lambda))}{\partial \lambda_i} \right\}.$$

In order to be able to solve [RMP] repeatedly with a rapidly convergent algorithm, it is essential that the number of columns retained remains small. A modification of the original algorithm was introduced and studied by Hearn et al. (1987), wherein the number of extreme points stored is bounded above by a user-specified parameter. Convergence remains finite in the number of restricted master problems if the value of this parameter is higher than the dimension of the optimal face of X; this number (which also may be high) is, however, unknown a priori, and, further, if the parameter is given a lower value, then the asymptotic convergence rate deteriorates to that of the Frank–Wolfe algorithm. (When the value of this parameter is 1, [RMP] reduces to the line search problem of the Frank–Wolfe method.)

The simplicial decomposition algorithm utilizes the Frank–Wolfe subproblem [LP] in its column generation phase and hence it inherits the ability to exploit structures in the constraints defining X; it however (to some extent) also inherits the drawbacks of the Frank–Wolfe algorithm that stem from the global use of the linear approximation of f in this subproblem. The consequence for the simplicial decomposition algorithm is that the addition of a new extreme point may yield an arbitrarily small reduction in the objective value, since the directional derivative may be close to zero [cf. (1)].

A natural way to avoid these drawbacks is to improve the approximation of f used in the column generation phase. A column generation principle based on *nonlinear* approximations of f will yield (non-extremal) columns of higher quality than those resulting from the linear programming based principle of this section; the fact that the descent properties are much better also indicates that the improvements in the objective function value in each iteration will be much larger, and that therefore less iterations will be needed to reach a given accuracy. Provided that the columns can be generated with a computational effort that is not much higher we can also expect to gain in the total computational effort.

3 Simplicial Decomposition with Nonlinear Column Generation

3.1 The Column Generation Phase

The new algorithm, which we shall refer to as *simplicial decomposition with nonlinear column generation* (or, SDN for short), is the combination of the simplicial decomposition principle and the nonlinear search direction finding subproblem of a primal descent algorithm for [NLP]. We have here chosen to consider the search direction subproblem of the (conceptually simple) regularization of the Frank–Wolfe method presented by Migdalas (1994). The subproblem of this algorithm is discussed below.

Let $\varphi : X \times X \mapsto \mathbb{R}$ be a continuous *regularization* function on $X \times X$ with the following properties:

1. $\varphi(\cdot, y)$ is strictly convex and continuously differentiable on X for every $y \in X$;
2. $\nabla_x \varphi(x, y) = 0$ holds if and only if $x = y$.

Given an iterate x^t the subproblem is

[RLP]

$$\min_{x \in X} \nabla f(x^t)^T x + \varphi(x, x^t).$$

Due to the compactness of X and the strict convexity of $\varphi(\cdot, x^t)$, the solution to [RLP], $\hat{y}^t$, is unique. It further has the property of identifying an optimal solution to [NLP]: x^t is optimal in [NLP] if and only if x^t solves [RLP], that is, $\hat{y}^t = x^t$ (see Lemma 1).

In our numerical experiments, we have concentrated on utilizing *quadratic* subproblems. The function φ then has the form

$$\varphi(x, y) = \frac{1}{2}(x - y)^T Q(y)(x - y),$$

where $Q(y)$ is a positive definite and symmetric matrix for every y in X. (In cases where the matrix Q depends on entities associated with the objective function f, the requirements 1.–2. made on φ above introduce *implicit* conditions on f. In such cases, the properties of f in a given problem [NLP] therefore induce restrictions on the possible choices of a regularization function.)

Three such functions have been used in our experiments:

1. Let $Q(y) \equiv \gamma I$, $\gamma > 0$; this results in
$$\varphi(x, y) = \frac{\gamma}{2}\|x - y\|_2^2, \qquad \gamma > 0,$$
which yields the subproblem of the *Goldstein–Levitin–Polyak gradient projection algorithm* (Goldstein, 1964; and Levitin and Polyak, 1966).
2. Assume that f is twice continuously differentiable on X. Let $Q(y) = \nabla^2 f(y)$; this results in
$$\varphi(x, y) = \frac{1}{2}(x - y)^T \nabla^2 f(y)(x - y),$$
which yields the subproblem of the *Newton algorithm*.
3. Let $Q(y) = \text{diag } \nabla^2 f(y)$; this results in
$$\varphi(x, y) = \frac{1}{2}(x - y)^T [\text{diag } \nabla^2 f(y)](x - y),$$
which yields the subproblem of the *diagonalized Newton algorithm*.

The resulting subproblems yield directions that have much better descent properties than those of the Frank–Wolfe algorithm. This is evidenced, for example, by the stronger convergence properties of descent methods based on the above subproblems, with convergence rates ranging from linear to quadratic. An intuitive explanation for this fact is made by McCormick and Tapia (1972); they show that, for a value of γ large enough, the search direction obtained in the Goldstein–Levitin–Polyak gradient projection algorithm satisfies
$$\hat{y}^t - x^t = \frac{1}{\gamma} P_{T_X(x^t)} \left(-\nabla f(x^t) \right),$$
where T_X denotes the tangent cone operator for X and P denotes Euclidean projection, that is, the search direction obtained is parallel to the direction of steepest feasible descent. (The other two subproblems yield search directions that are the steepest feasible with respect to other norms.)

The meaning of introducing the regularization function in [LP] may also be characterized as follows. The solution $\hat{y}^t$ to [RLP] also solves a modified version of [LP], where a *trust region* $\{ x \in \mathbb{R}^n \mid \varphi(x, x^t) \leq \varphi(\hat{y}^t, x^t) \}$ is added; further, [RLP] is the result of a Lagrangean relaxation of a similar trust region restriction. In [RLP], the regularization function acts as a penalty for moving far from that trust region.

When modifying the simplicial decomposition method of Section 2.2 we will utilize the regularized subproblem solutions $\hat{y}^t$ in place of the solutions to the linear program [LP]. While the solution to [LP] is an extreme point, a solution to [RLP] may be in the relative interior of X; in order to augment the set X_t as much as possible, we shall not store $\hat{y}^t$ but its extension on the relative boundary of X, that is, we shall store

$$y^t = x^t + \ell_t(\hat{y}^t - x^t); \qquad \ell_t = \max\{\, \ell \mid x^t + \ell(\hat{y}^t - x^t) \in X \,\}. \tag{2}$$

In the first iterations, $\ell > 1$ may be a possible choice; as $\{x^t\}$ tends towards an optimal solution, however, the subproblem solutions $\hat{y}^t$ will eventually lie on the optimal face of X (Patriksson, 1993, Theorem 4.6), and thus an extension of $\hat{y}^t$ will not be feasible unless also x^t identifies this face.

3.2 A Summary of Simplicial Decomposition with Nonlinear Column Generation

We summarize the (conceptual) simplicial decomposition scheme with nonlinear column generation below.

0. (*Initialization*): Choose a regularization function φ, an initial point $x^0 \in X$, and let $t = 0$.
1. (*Regularized subproblem*): Find the solution, $\hat{y}^t$, to [RLP].
2. (*Termination criterion*): If x^t solves [RLP], then terminate (x^t solves [NLP]). Otherwise, continue.
3. (*Set augmentation*): Let y^t be given by (2), and $X_{t+1} = \operatorname{conv}\left(x^t \cup_{i=0}^{t} y^i\right)$.
4. (*Multidimensional search*): Find a solution x^{t+1} to [RMP].
5. (*Update*): Let $t := t + 1$ and go to Step 1.

The simplicial decomposition method thus introduced is related to the methods previously discussed as shown in Table 1.

Table 1. Method relationships.

		Subproblem	
		Nonlinear	*Linear*
Search	*Multi-dim.*	SDN	SD
	One-dim.	Reg. F–W	F–W

3.3 Convergence of the Simplicial Decomposition Method with Nonlinear Column Generation

Our first result is standard for many descent methods in nonlinear programming, and validates the termination criterion in Step 2.

Lemma 1 (A fixed point characterization of solutions to [NLP]). *The point* x^t *solves* [NLP] *if and only if* x^t *solves* [RLP].

Proof. The solution $\hat{y}^t$ to [RLP] is characterized by the following variational inequality.

$$[\nabla f(x^t) + \nabla_x \varphi(\hat{y}^t, x^t)]^T (x - \hat{y}^t) \geq 0, \qquad \forall x \in X \tag{3}$$

If $\hat{y}^t = x^t$, then the above, together with the property that $\nabla_x \varphi(x^t, x^t) = 0$, implies that $\nabla f(x^t)^T (x - x^t) \geq 0$ for all $x \in X$; by the pseudo-convexity of f on X, this in turn implies that x^t is optimal in [NLP].

The converse result follows analogously. □

Our second lemma establishes that if x^t is *not* optimal in [NLP], then the subproblem [RLP] yields an augmentation of the current restriction. Hence, Step 3 of our algorithm is validated.

Lemma 2 (The solution to [RLP] yields augmentation). *Let* $x^t \in X_t^*$. *If* x^t *does not solve* [RLP], *then* $X_t \cap X^* = \emptyset$ *and* $\hat{y}^t \in X \setminus X_t$.

Proof. Letting $x = x^t$ in (3) yields

$$[\nabla f(x^t) + \nabla_x \varphi(\hat{y}^t, x^t)]^T (x^t - \hat{y}^t) \geq 0.$$

The strict convexity of $\varphi(\cdot, x^t)$ is equivalent to the strict monotonicity of its gradient; utilizing this result in the above inequality, together with the property that $\nabla_x \varphi(x^t, x^t) = 0$, then yields

$$\nabla f(x^t)^T (\hat{y}^t - x^t) < 0, \tag{4}$$

whenever $\hat{y}^t \neq x^t$. But since x^t minimizes f over X_t,

$$\nabla f(x^t)^T (x - x^t) \geq 0, \qquad \forall x \in X_t,$$

from which follows that $\hat{y}^t \notin X_t$. □

The proof shows that the direction of $\hat{y}^t - x^t$ is a feasible direction of descent with respect to f whenever x^t is not optimal, whence it follows that $f(x^{t+1}) < f(x^t)$.

Below follows the main theoretical result of the paper.

Theorem 3 (Convergence of the algorithm). *If the sequence* $\{x^t\}$ *of iterates generated by the algorithm is finite, then the last iterate solves* [NLP]. *If the sequence is infinite, then*

$$\left\{ \min_{z \in X^*} \|z - x^t\| \right\} \to 0.$$

Proof. If the sequence $\{x^t\}$ of iterates is finite, then the algorithm has terminated in Step 2, with the conclusion following from Lemma 1. In the following, we assume that $\{x^t\}$ is infinite.

The sequence $\{X_t\}$ consists of nonempty, compact, convex and increasing subsets of X; according to Salinetti and Wets (1979) it has a set limit, $\tilde{X} \subseteq X$, which also is nonempty, compact and convex.

Assume that, for some $\varepsilon \geq 0$, the sequence $\{x^t\}$ contains an infinite subsequence $\{x^t\}_{t \in \mathcal{T}}$, $\mathcal{T} \subseteq \mathbb{N}$, such that $\min_{z \in X^*} \|z - x^t\| \geq \varepsilon$ for all $t \in \mathcal{T}$. This sequence has an accumulation point, $\tilde{x} \in X$, which is the limit point of a convergent subsequence, $\{x^t\}_{t \in \tilde{\mathcal{T}}}$, $\tilde{\mathcal{T}} \subseteq \mathcal{T}$.

Let $\tilde{p} \in \tilde{X}$ be arbitrary, which then is the limit of a sequence $\{p^t\}$ of points $p^t \in X_t$ (Aubin and Frankowska, 1990, Proposition 1.1.2). Since for all t, $x^t \in X_t^*$, it follows that $f(p^t) \geq f(x^t)$, and that $f(\tilde{p}) = \lim_{t \in \tilde{\mathcal{T}}} f(p^t) \geq \lim_{t \in \tilde{\mathcal{T}}} f(x^t) = f(\tilde{x})$, the equalities following from the continuity of f. Hence, $\tilde{x} \in \tilde{X}^*$.

Next, consider the sequence $\{\hat{y}^t\}$ of solutions to [RLP], and let $\tilde{y}$ be an accumulation point of $\{\hat{y}^t\}_{t \in \tilde{\mathcal{T}}}$. From the continuity of ∇f and φ, $\tilde{y}$ solves [RLP] defined in $\tilde{x}$. Further, since $\hat{y}^t \in X_{t+1}$ for all t it follows that $\tilde{y} \in \tilde{X}$. But since also $\tilde{x} \in \tilde{X}$, Lemma 2 yields that $\tilde{y} = \tilde{x}$, and Lemma 1 that, therefore, $\tilde{x} \in X^*$.

Hence, $\varepsilon = 0$ must hold, and the theorem follows. □

We note that the theorem can alternatively be established using the descent properties of the regularized Frank–Wolfe method. We have, however, chosen a proof which closely follows the process of the column generation algorithm, and also is relatively independent of the choice of nonlinear column generation subproblem. Further, this proof technique should be possible to use when establishing nonlinear column generation methods in more general settings.

We end this section with some remarks on the choice of regularization functions and algorithms for [RLP].

Many functions φ satisfy the requirements stated above, and φ can (and should) therefore be chosen to adapt to problem structures. For example, if X is a Cartesian product of sets in smaller dimensions, the choice of an additively separable function $\varphi(\cdot, y)$ would result in [RLP] separating into a number of independent subproblems. The freedom of choice in φ can also be used to introduce second-order terms from f into [RLP], thereby retaining more of the original objective function than is the case in [LP]. (The choice of $Q(y)$ as the diagonal of the Hessian matrix is an example with both these properties.)

The function φ must also be chosen such that the resulting subproblem [RLP] is efficiently solvable; otherwise, there will be little advantage derived from the generation of columns of higher quality. Note, however, from the proof of Lemma 2 that the subproblem [RLP] need not be solved exactly in order to obtain augmentation; it is sufficient to find a descent direction [cf. (4)]. This result allows for the subproblems to be truncated before reaching their solution, a strategy that can be used to reduce the computational effort spent on the solution of [RLP]. (The trade-off between the time spent on [RLP] and the quality of the column generated is studied in our applications.)

4 Applications and Experiments

The simplicial decomposition method with nonlinear column generation has been applied to the stochastic transportation problem and the traffic assignment problem, with the purpose of making a preliminary analysis and evaluation of the method.

There are two reasons for our choice of applications. First, both problems are nonlinear network flow problems, and therefore amenable to the solution with methods based on linearizations of the objective function (e.g., LeBlanc et al., 1985). Second, the two problems have different properties in the respect that the computational burden of their respective linear subproblems is very different; the linear approximation of the stochastic transportation problem leads to a set of easily solvable continuous knapsack problems, whereas the linearized traffic assignment problem separates into a number of shortest route calculations, which are computationally demanding compared with the continuous knapsack problems. This difference in computational burden is of some importance also in the proposed simplicial decomposition method.

Our implementations of the SDN method for the two applications were made in Fortran 77, and the numerical experiments were conducted on a DEC 5500 computer. The master problems which arise during the course of the method are solved by a straightforward constrained Newton scheme with an Armijo line search (with an acceptance parameter of 0.2); the quadratic subproblems are solved using the NAG library routine `e04ncf`. The solution procedure for the master problem is terminated when the relative difference between the upper and lower bounds on its optimal value is small enough. The first master problem is solved to a relative accuracy of 10%, and the accuracy demanded is then decreased successively until it has reached the level of one tenth of the relative accuracy demanded for the solution to the original problem. At termination of each master problem solution, all columns having (near-)zero weights are removed.

The experiments performed are similar for both applications. For each of them we first present results of experiments the purpose of which is to evaluate the quality of the columns generated using a nonlinear subproblem, as compared to the quality of the columns obtained from the traditional linear subproblem. Second, we present results of experiments which aim at evaluating the overall efficiency of the SDN method, as compared to that of the standard SD method. This includes an investigation of the effect of different degrees of truncation of the solution of the nonlinear column generation problems. Finally, we present results of experiments with a version of the SDN method in which the size of the master problem is restricted. At the end of this section we make some comments about the similarities and differences between the results for the two applications.

4.1 The Stochastic Transportation Problem

The stochastic transportation problem is used to determine the optimal quantities of a commodity to be shipped from a number of supply points, or sources, to a number of demand points, or destinations, at which the amounts of goods demanded are a priori unknown and described by random variables. In this section we specialize the SDN method to this problem and describe the principles for the generation of test problems.

Suppose that there are m points with known supplies S_i, $i = 1, 2, \ldots, m$, and n points with demands D_j, $j = 1, 2, \ldots, n$, each of which is a random variable with density function $f_j(t)$. Let c_{ij} denote the unit shipping cost from supply point i to demand point j, and let h_j and p_j denote the unit holding and shortage costs, respectively, at demand point j. The problem of determining the shipping volumes which yield the minimal expected total cost is usually formulated as below, where each variable x_{ij} describes the shipping volume from source i to destination j, and the variables y_j, $j = 1, 2, \ldots, n$, are auxiliary quantities which describe the total amounts of goods shipped to the various destinations.

[STP]

$$
\begin{aligned}
\min f(x,y) = {} & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n}\left(h_j\int_0^{y_i}(y_j - t)f_j(t)dt + p_j\int_{y_j}^{\infty}(t - y_j)f_j(t)dt\right) \\
\text{s.t. } & \sum_{i=1}^{m} x_{ij} - y_j = 0, \qquad j = 1, \ldots, n, \\
& \sum_{j=1}^{n} x_{ij} \le S_i, \quad i = 1, \ldots, m, \\
& x_{ij} \ge 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.
\end{aligned}
$$

Note however that the elimination of the auxiliary demand variables and the definitional constraints reveal that the problem can alternatively be stated, in terms of the shipping variables only, as the minimization of a nonseparable objective function over a feasible set which is a Cartesian product. Because of its very simple structure, this problem has been approached in many different ways; see, for example, Cooper and LeBlanc (1977), Holmberg and Jörnsten (1984), LeBlanc et al. (1985), and Holmberg (1995).

When applying the SDN method to the stochastic transportation problem, we obtain a column generation subproblem which is most conveniently stated as

[RLP]

$$
\begin{aligned}
\min \ & \sum_{i=1}^{m}\sum_{j=1}^{n} \overline{c}_{ij}^{t} x_{ij} + \varphi(x, x^t) \\
\text{s.t. } & \sum_{j=1}^{n} x_{ij} \le S_i, \quad i = 1, \ldots, m, \\
& x_{ij} \ge 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n,
\end{aligned}
$$

where

$$
\overline{c}_{ij}^{t} = \frac{\partial f(x^t, y^t)}{\partial x_{ij}} + \frac{\partial f(x^t, y^t)}{\partial y_j}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n,
$$

with

$$y_j^t = \sum_{i=1}^m x_{ij}^t, \quad j = 1, \ldots, n.$$

If the regularization term is dropped from the objective function, then the subproblem separates into m continuous knapsack problems, which are computational very cheap to solve; this property is exploited in an application of the Frank–Wolfe method to [STP].

We have in our experiments applied the three examples of regularization functions given in Section 3. When the Newton regularization is used, the subproblem is a non-trivial quadratic program; it is solved by applying a fixed number of iterations of the Frank–Wolfe method. The linear problems that arise in this application of the Frank–Wolfe method are, of course, also continuous knapsack problems.

The Euclidean norm and the diagonalized Newton regularizations both yield a subproblem that separates into m (separable) quadratic knapsack problems; each of these is easily solved via its one-dimensional Lagrangean dual problem. For example, when using the Euclidean norm regularization, the dual problem corresponding to a particular supply point is given by

[DRLP$_i$]

$$\max h_i(u_i) = \min \sum_{j=1}^n \overline{c}_{ij}^t x_{ij} + \frac{\gamma}{2}\sum_{j=1}^n (x_{ij} - x_{ij}^t)^2 + u_i \left(\sum_{j=1}^n x_{ij} - S_i \right)$$
$$\text{s.t. } x_{ij} \geq 0, \quad j = 1, \ldots, n.$$

The dual objective function is concave, piecewise quadratic, and everywhere differentiable, and the dual problem is easily solved through a search for a zero to the piecewise linear derivative

$$h_i'(u_i) = \sum_{j=1}^n x_{ij}(u_i) - S_i,$$

where

$$x_{ij}(u_i) = \max \left\{ 0, x_{ij}^t - \frac{1}{\gamma}(\overline{c}_{ij}^t + u_i) \right\}, \quad j = 1, \ldots, n,$$

is the solution to the Lagrangean subproblem. Further, if u_i^t solves [DRLP$_i$], then $x_{ij}(u_i^t)$, $j = 1, \ldots, n$, solves the quadratic knapsack problem corresponding to the supply point under consideration.

The test problems are randomly generated in accordance with two principles used earlier, resulting in problem instances of different sizes and with different cost structures.

The first principle is adopted from Cooper and LeBlanc (1977). These problems have 200 destinations and from 5 up to 25 sources. The locations of the supply and demand points are drawn from a uniform distribution over a $[0, 100] \times$

$[0, 100]$ grid, and the unit transportation costs are 1.1 times the Euclidean distances. (The value of this scaling constant is chosen so that a certain prespecified portion of the optimal objective value stems from the nonlinear term.) The unit shortage and holding costs are drawn from uniform distributions over the intervals $[20, 60]$ and $[3, 6]$, respectively. Each demand is exponentially distributed, that is, $f_j(t) = \lambda_j e^{-\lambda_j t}$, with an intensity λ_j which is drawn from the interval $[\frac{1}{200}, \frac{1}{40}]$; the supplies are uniformly distributed random numbers from the interval $[125, 175]$.

The second type of test problems (LeBlanc et al., 1985) have 100 destinations and from 5 up to 25 sources. The locations of the supply and demand points are randomly generated as stated above, but the scaling constant used in the calculation of the unit transportation costs is in this case replaced with random values drawn from the interval $[0.4, 0.8]$. The shortage and holding costs are random numbers within the intervals $[75, 115]$ and $[3, 6]$, respectively. The demands are exponentially distributed with intensities drawn from the interval $[\frac{1}{50}, \frac{1}{40}]$. Each supply is drawn from a uniform distribution over the interval $[300, 500]$.

For a test problem of size 25×200 generated using the principle given by Cooper and LeBlanc, about 96% of the optimal objective value arises from the nonlinear shortage and holding cost term, whereas for a problem of size 25×100 generated using the second principle, this figure is about 60%.

An initial solution to [STP] is obtained as follows. First, the objective function is minimized with respect to the auxiliary demand variables only, after which the solution obtained is rescaled into a solution in which all supply available is used. The resulting values of the auxiliary demand variables are used to set up a transportation problem, which is then solved approximately using the minimum cost rule. (For the test problems which are highly nonlinear, that is, for those generated using the principle of Cooper and LeBlanc, the initial solutions found by this procedure are of good quality. For the second class of test problems, they are not as good.)

Two sets of test problems were generated according to the respective principles given above, each of them consisting of five subsets of problems having 5, 10, 15, 20, and 25 sources, respectively, each of which contained 10 instances; the experimental results presented are averages for the 10 instances.

4.2 Experimental Results

The aim of introducing a regularization of the subproblem of a simplicial decomposition scheme is the prospect of generating columns of higher quality; we start the review of our numerical study by presenting some numerical evidence for the fulfillment of this aim. In this experiment we applied the standard SD scheme and the SDN method with the diagonalized Newton regularization (SDN:DN for short) to the largest test problem (that is, with 25 sources) among those generated using the principle of LeBlanc et al. (1985). (The reason for choosing the SDN:DN version of the SDN method will be apparent in the sequel.) Figure 1 shows the average values of the maximal sizes of the master problems in the SD and SDN methods versus the relative accuracy demanded. (When a relative accuracy of 0.1% is demanded, on average 130 iterations is needed for SDN:DN and 190 iterations for the standard SD method.) An explanation of the observed performance is that the high quality of the columns generated by [RLP], as compared to the quality of the columns provided by [LP], results in that any accuracy can be reached by utilizing fewer columns. A further explanation is that the high quality of the columns obtained from [RLP] results in a more frequent dropping of previously generated columns from the master problem. This claim might appear counter-intuitive, but it is justified by the fact that the columns generated during the course of the SDN method actually are of *increasing* quality (cf. the fixed point characterization in Lemma 1); this property does *not* hold in the SD method (cf. the discussion at the end of Section 2.2).

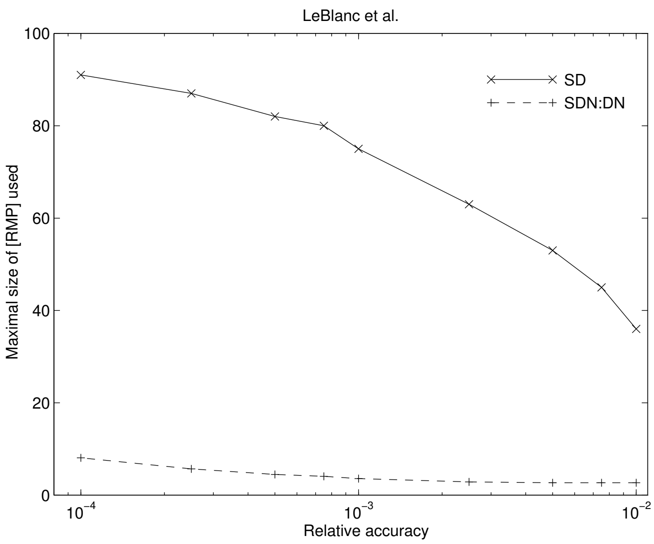

Fig. 1. Maximal size of [RMP] versus relative accuracy.

In the second experiment, we study the extension of the subproblem solutions made in Step 3 of the SDN algorithm [cf. the value of ℓ_t in (2)], for one instance of [STP] generated according to the principle of LeBlanc et al. (1985) and having 25 sources and 100 destinations. The result, which is shown in Figure 2, clearly demonstrates that the subproblem solution, in comparison with its corresponding iterate, almost always lies in a face of X of smaller dimension.

The third experiment aims at illustrating how the practical performance of the SDN method depends on the tradeoff between the respective computational efforts spent in its master and column generation phases. This tradeoff is varied by changing the number of Frank–Wolfe iterations used in the approximate solution of the Newton regularized column generation problems. The experiment was carried out for the largest problems in each of the two sets of test problems.

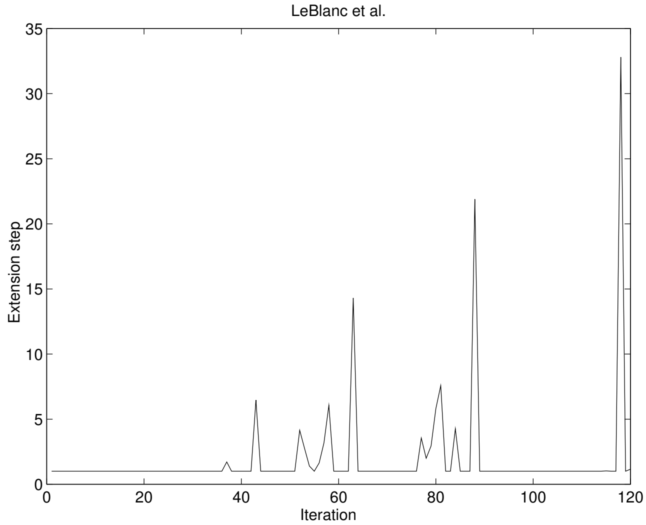

Fig. 2. Extension of $\hat{y}^t$ versus iteration.

Figure 3 shows the total computing times versus the number of Frank–Wolfe iterations made in each subproblem for the first set of test problems (Cooper and LeBlanc, 1977). Figure 4 contains the corresponding results for the largest problems in the second set of test problems. We conclude that there is in this application indeed a non-trivial, and even pronounced, tradeoff between the computational burden of the master and column generation phases, respectively, of the SDN method.

In the fourth experiment we compared the practical performance of the SD scheme versus that of the SDN method, using the Euclidean norm, Newton, and diagonalized Newton regularizations, respectively. (The two former schemes are referred to as SDN:GP and SDN:N, respectively, for short.) The parameter γ in the Euclidean norm regularization function was calibrated to the value 0.2, and 15 Frank–Wolfe iterations were used when solving the Newton regularized column generation problem. The relative accuracies of 1.0%, 0.1%, and 0.01% were considered.

The computing times for the Cooper and LeBlanc test problems solved to a relative accuracy of 0.01% are presented in Figure 5. Table 2 contains the computing times of the SD method relative those of the SDN method for the three accuracies studied.

Clearly, for these test problems the advantage of using a nonlinear column generation problem grows when higher accuracy is demanded.

In Figure 6 we show the computing times for the LeBlanc et al. test problems when solved to a relative accuracy of 0.01%. Table 3 gives the ratios of the computing times for the SD and SDN methods for the three accuracies studied.

For the largest test problems in this set we also present the computing times

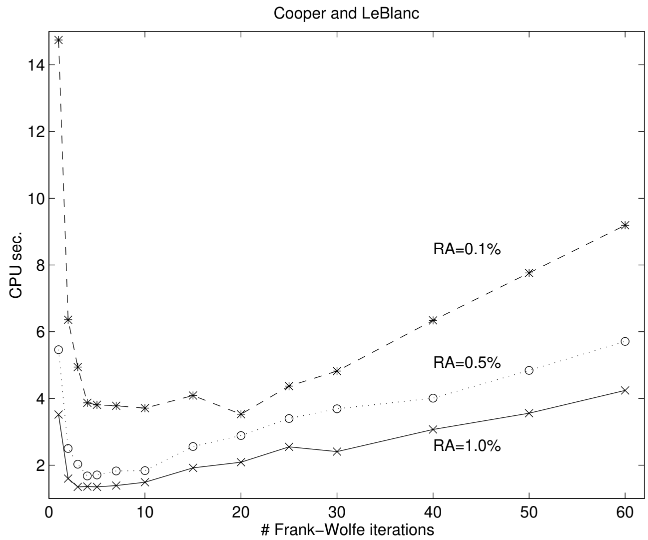

Fig. 3. Computing time versus effort spent in subproblem.

Table 2. Speedup for SDN:DN compared to SD; Cooper and LeBlanc.

	Relative accuracy		
# Sources	1.0%	0.1%	0.01%
5	1.6	2.9	4.1
10	2.3	3.9	10.2
15	2.4	4.2	11.6
20	2.7	4.7	12.9
25	3.0	4.2	11.9

of the SD and SDN:DN methods for a wider range of accuracies; these computing times are given in Figure 7.

In the last experiment with this application we made a comparison of the effect of restricting the number of columns retained in the master problems of the SD and SDN:DN methods, respectively. Specifically, the largest test problems of the second type were solved to a relative accuracy of 0.1% for different values of the maximal number of columns allowed in the master problem. Figure 8 displays the computing times of the SD method relative to that of the SDN:DN method versus the degree of restriction. As can be expected (cf. Figure 1), the advantage of invoking a nonlinear column generation problem is greatest when then master problem is strongly restricted.

To summarize the experimental results presented in this section, we merely conclude that a simplicial decomposition solution scheme for the stochastic transportation problem may undoubtedly be enhanced significantly through the in-

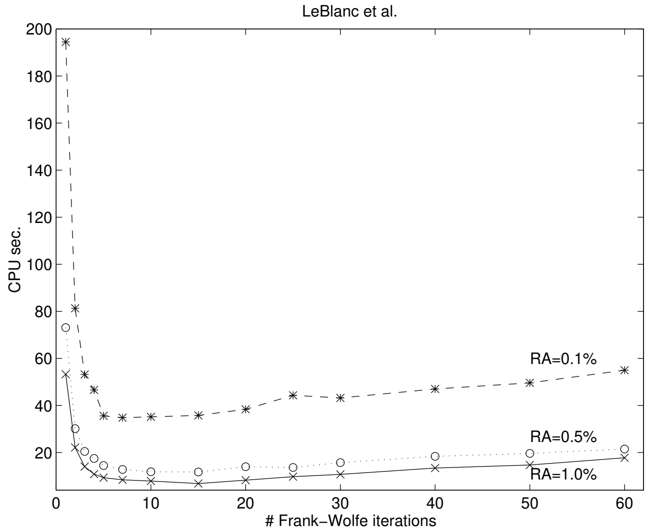

Fig. 4. Computing time versus effort spent in subproblem.

Table 3. Speedup for SDN:DN compared to SD; LeBlanc et al.

	Relative accuracy		
# Sources	1.0%	0.1%	0.01%
5	17.0	29.0	45.7
10	13.7	18.6	20.0
15	8.3	16.1	14.4
20	8.3	11.7	10.9
25	8.8	12.1	9.4

troduction of a nonlinear column generation problem.

4.3 The Traffic Assignment Problem

The traffic assignment problem is routinely solved in transportation planning contexts to analyze, predict, and describe traffic flows in an urban transportation network. The traffic assignment problem is to determine the flows in this network resulting from decisions made by the travellers regarding which routes to use to their respective destinations given the current travel conditions. The behavioural principle most commonly utilized is that a traveller chooses a shortest route to his/her destination; this is known as Wardrop's principle of user equilibrium, which is characterized by a situation in which the routes utilized between an origin and a destination have equal travel costs.

Consider a transportation network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ of nodes and directed links. Between certain pairs $(p, q) \in \mathcal{C}$ of nodes there is a known demand d_{pq} for

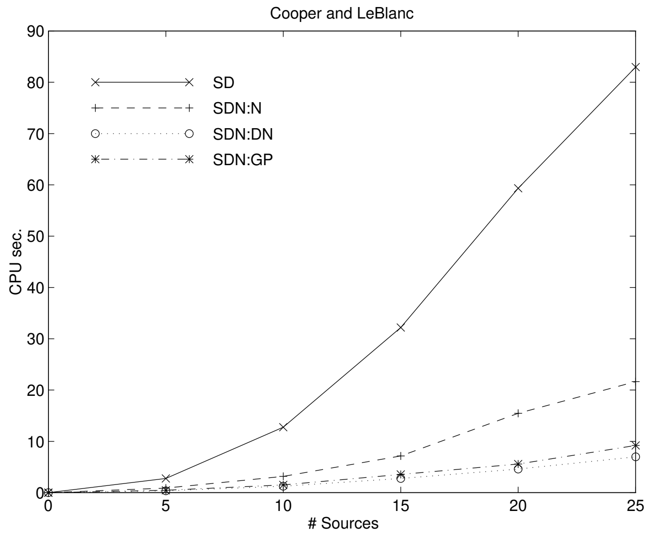

Fig. 5. Computing time versus problem size for a relative accuracy of 0.01%.

transportation; further, each link $a \in \mathcal{A}$ is associated with a positive, continuous and strictly increasing travel cost function c_a of the total flow f_a on the link. The flow which originates at node p and terminates at node q and passes link a is denoted by f_{apq}. We also denote by $\mathcal{W}_i$ and $\mathcal{V}_i$, respectively, the sets of links initiated and terminated at node $i \in \mathcal{N}$. The traffic assignment problem then is

[TAP]

$$
\begin{aligned}
\min T(f) &= \sum_{a \in \mathcal{A}} \int_0^{f_a} c_a(s)\, ds \\
\text{s.t.} \sum_{a \in \mathcal{W}_i} f_{apq} - \sum_{a \in \mathcal{V}_i} f_{apq} &= \begin{cases} d_{pq}, & \text{if } i = p, \\ -d_{pq}, & \text{if } i = q, \\ 0, & \text{otherwise}, \end{cases} \qquad i \in \mathcal{N}, \quad (p,q) \in \mathcal{C}, \\
\sum_{(p,q) \in \mathcal{C}} f_{apq} &= f_a, \qquad a \in \mathcal{A}, \\
f_{apq} &\geq 0, \qquad a \in \mathcal{A}, \quad (p,q) \in \mathcal{C}.
\end{aligned}
$$

The constraints of [TAP] define an uncapacitated multicommodity network, that is, they define a Cartesian product of feasible sets with respect to the origin–destination (O–D) pairs; this fact is revealed if the link flow definitional constraints are eliminated from the problem, which then is stated in terms of O–D link flows only. The objective function is strictly convex in the total link flows, and is therefore nonseparable in the O–D flows. The problem [TAP] hence has a structure similar to that of the stochastic transportation problem. The complexity of [TAP] is, however, much higher, for the following reasons. First, the

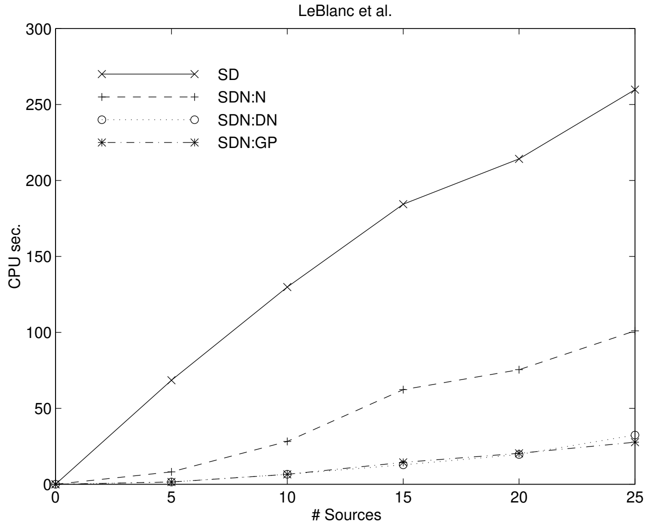

Fig. 6. Computing time versus problem size for a relative accuracy of 0.01%.

network defined in [TAP] is more general; second, the sizes of traffic assignment problems occurring in practice is very large: typical problem sizes are of the order of 1000, 3000 and 10,000 in the number of nodes, links and O–D pairs, respectively.

Given an iterate f^t, the nonlinear column generation subproblem is

[RLP]

$$
\begin{aligned}
&\min \nabla T(f^t)^T f + \varphi(f, f^t) \\
&\text{s.t.} \sum_{a \in \mathcal{W}_i} f_{apq} - \sum_{a \in \mathcal{V}_i} f_{apq} = \begin{cases} d_{pq}, & \text{if } i = p, \\ -d_{pq}, & \text{if } i = q, \\ 0, & \text{otherwise}, \end{cases} \qquad i \in \mathcal{N}, \quad (p,q) \in \mathcal{C}, \\
&\qquad \sum_{(p,q) \in \mathcal{C}} f_{apq} = f_a, \qquad a \in \mathcal{A}, \\
&\qquad f_{apq} \geq 0, \qquad a \in \mathcal{A}, \quad (p,q) \in \mathcal{C}.
\end{aligned}
$$

The objective function of [TAP] is separable with respect to the links in the network; the second and third regularization functions given in Section 3 therefore coincide. All the test results presented are based on their use. Hence, when referring to the method SDN, we shall always mean SDN:N.

The linear subproblem [LP] used in the column generation phase of the simplicial decomposition algorithm of Section 2.2 is in this application equivalent to a set of shortest route problems, with link travel costs given by the vector $c(f^t)$. In the presence of the regularization function, however, the nonlinear problem [RLP] must be solved using some iterative scheme. To this end, we choose to apply a fixed number of Frank–Wolfe iterations. Given an iterate f^t, the first

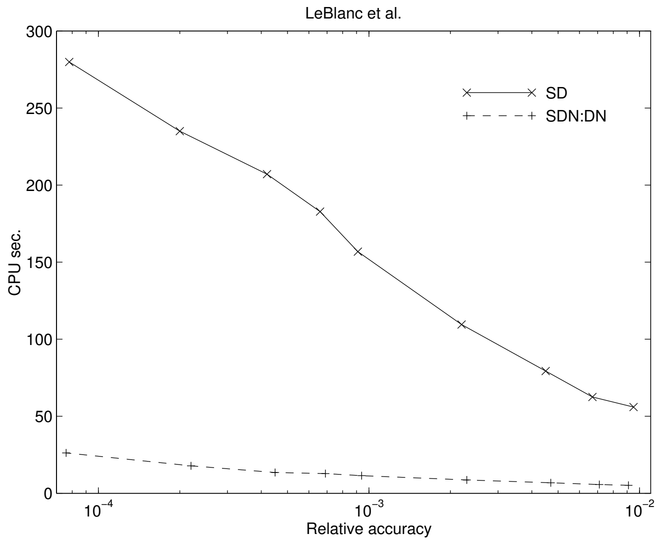

Fig. 7. Computing time versus relative accuracy required.

subproblem in this scheme is, by the second requirement of φ, the problem [LP], which provides a lower bound on T^*; later subproblems may involve negative link costs; whenever this occurs, the solution of [RLP] is terminated.

The test problems are one small and two medium-size network models, data for which are given in Table 4.

Table 4. Traffic network models.

Network	# Nodes	# Links	# O–D pairs
Sioux Falls	24	76	528
Winnipeg	1052	2836	4344
Barcelona	1020	2522	7922

4.4 Experimental Results

Numerical experiments were conducted with the three network models to study the effect of the accuracy by which [RLP] is solved on the quality of the columns generated and on the resulting solution times.

In the first experiment, we solved the Winnipeg network model with the simplicial decomposition method of Section 2.2 and two versions of the simplicial decomposition method with nonlinear column generation, wherein [RLP] is

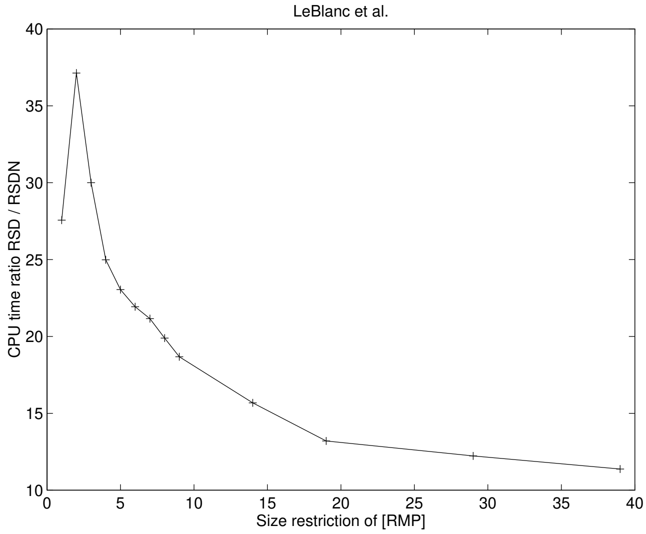

Fig. 8. Relative computing time versus master problem size restriction.

solved using 3 and 30 Frank–Wolfe iterations, respectively. (Note that the classical simplicial decomposition method is obtained by setting this number to 1.) During each of these three executions we recorded the number of columns needed to reach preselected accuracies in the solutions. The result of this experiment is shown in Figure 9.

The figure clearly shows that the quality of the columns increases with the accuracy by which [RLP] is solved. The outcome of this experiment is also a support for our claim that the simplicial decomposition scheme to some extent inherits the drawback of the Frank–Wolfe algorithm which originates in the solution of linear subproblems: the figure illustrates that the quality of each new column is inferior to the earlier ones, with the result that the number of necessary iterations increases very rapidly when demanding higher accuracy.

To investigate whether an increase in quality of the columns generated results in a computational advantage, we solved the three network models with the SDN method, and recorded the CPU times required to reach relative accuracies in the interval [0.1%–1.0%] as functions of the number of Frank–Wolfe iterations performed on each subproblem [RLP]. The results of this experiment are shown in Figures 10–12.

The results may be summarized by the conclusion that a very accurate solution of [RLP] is not meaningful. Also when demanding a high accuracy, the optimal number of iterations of the Frank–Wolfe algorithm never exceeds 6; the optimal choice also is stable with varying accuracy and does not vary much with the network models.

In terms of CPU time, traditional simplicial decomposition wins in one case only, when a low accuracy is required in the Barcelona network model; in all

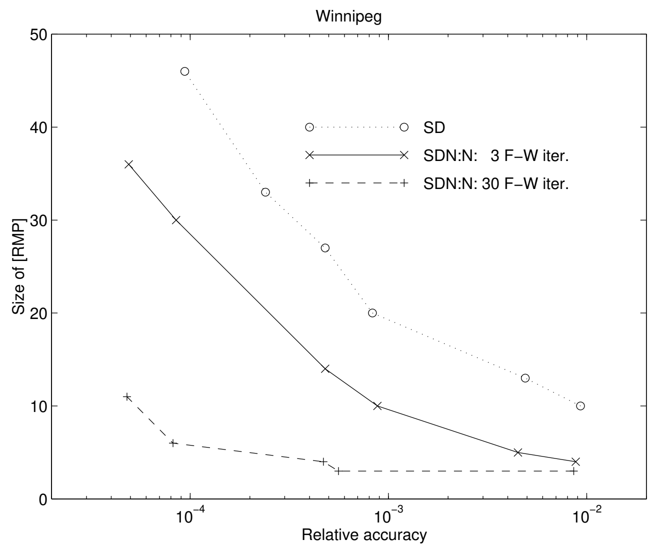

Fig. 9. Size of [RMP] versus relative accuracy required.

other cases, this method has a slower convergence than SDN, and the difference in convergence rate increases with the accuracy required. (For a relative accuracy of 0.1%, the ratio of the CPU time required for SD to that of SDN is 2.5, 1.5, and 1.2, respectively.) This difference however decreases with the size of the networks and numbers of O–D pairs; this is explained by the fact that the shortest route subproblems then increase in complexity.

It should be noted that the outcome of an experiment of this kind necessarily is dependent on the algorithm chosen for the solution of [RLP]. Our choice was to use the Frank–Wolfe algorithm which, as has been remarked upon earlier, is good for reaching rough solutions only. An implementation of the SDN method using a more rapidly convergent algorithm for [RLP] is likely to favour a more accurate solution of [RLP] than the present one.

We have observed that when the number of Frank–Wolfe iterations is optimal, about 80% of the total computing time in the SDN method is spent on solving shortest route problems. A potentially very large improvement of the algorithm would result from utilizing a reoptimizing procedure for these problems.

It has been noted elsewhere (e.g., Larsson and Patriksson, 1992) that the lower bounds provided by [LP] are sometimes of poor quality; the feasible solutions generated in our experiments are consistently much more accurate than is indicated by these bounds. For example, while an execution with the Winnipeg network model is terminated after 300 CPU seconds with a relative accuracy reported to be 0.018%, the true accuracy of the solution is, in fact, 0.002%.

The next line of experiments was based on a modification of the SDN method, wherein we introduced an upper bound, r, on the number of columns retained in any restricted master problem. These experiments were conducted with the

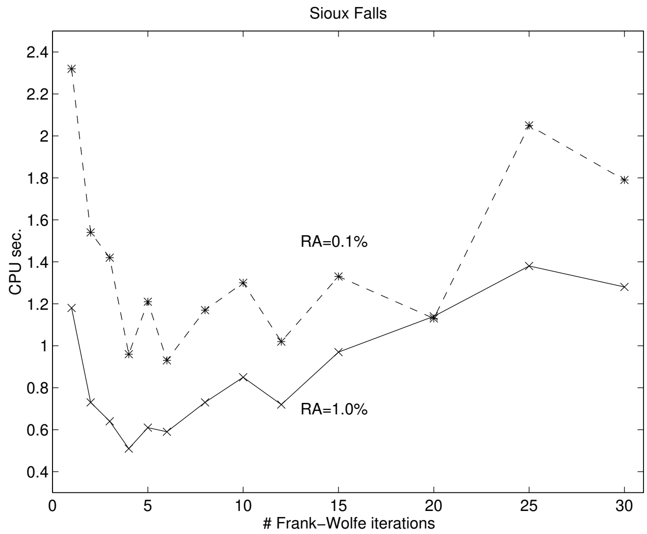

Fig. 10. Computing time versus effort spent in subproblem.

Winnipeg network model only.

In our first such experiment, we investigated the accuracy by which the subproblems [RLP] should be solved in order to maximize the efficiency of the SDN algorithm, given different values of the restriction parameter r. We executed the SDN method with different values of r and numbers of Frank–Wolfe iterations per subproblem [RLP], and recorded the CPU times needed to reach a solution with an accuracy of 0.1%. The result of this experiment is shown in Figure 13.

We first remark that with any value of $r > 12$, no columns with non-zero weights are removed, and hence the solutions obtained coincide with that of the basic algorithm. With decreasing values of r, the optimal number of Frank–Wolfe iterations increases (from 3 to 6 as r decreases to 1); this indicates that the quality of the columns generated becomes more significant for the convergence rate of the algorithm as the maximum size of the restricted master problems becomes more limited. Again, with reference to the poor convergence of the Frank–Wolfe algorithm, we venture to suggest that with a more efficient algorithm for [RLP], the accuracy by which [RLP] should be solved would be greater.

We finally investigated the efficiency of the SDN method for a low value of the restriction parameter ($r = 4$); for three different levels of accuracy demanded, we executed the algorithm with different numbers of Frank–Wolfe iterations applied in each subproblem [RLP]. The result is shown in Figure 14.

The intuitive result that the quality of the columns generated and retained becomes more and more important for the efficiency of a simplicial decomposition algorithm when higher accuracy is required, or the number of columns retained is more restricted, is again corroborated, as the figure shows that the number of Frank–Wolfe iterations should increase (from 3 to 6) as the demand for accuracy

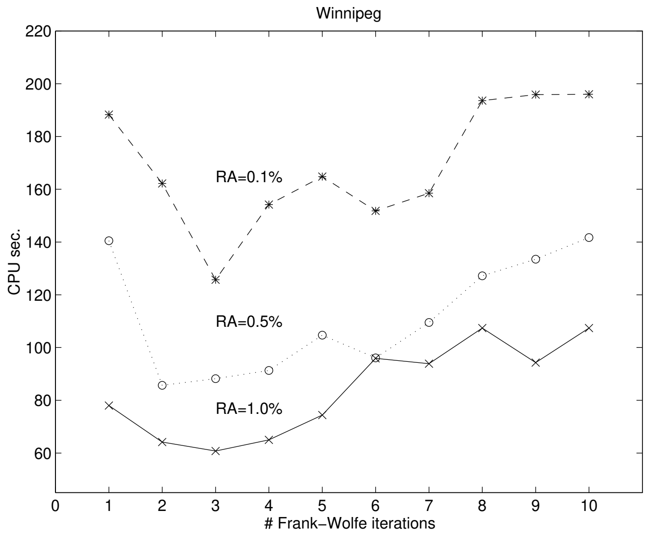

Fig. 11. Computing time versus effort spent in subproblem.

increases.

5 Conclusion

The general conclusion that can be drawn from the experiments conducted is that solution methods founded on the simplicial decomposition principle may benefit substantially from the generation of columns of higher quality; the means for achieving this goal considered in this paper is the introduction of a nonlinear column generation problem. This approach may be beneficial in other column generation schemes (and dually related constraint generation schemes) as well; our finding warrants an exploration of several paths of research, both in theoretical and applied directions.

Acknowledgement

We thank the anonymous reviewer for comments that resulted in an improved presentation and the addition of a numerical result.

References

Aubin, J-P., Frankowska, H.: Set-Valued Analysis. Birkhäuser, Boston, MA, 1990

Canon, M.D., Cullum, C.D.: A tight upper bound on the rate of convergence of the Frank–Wolfe algorithm. SIAM Journal on Control **6** (1968) 509–516

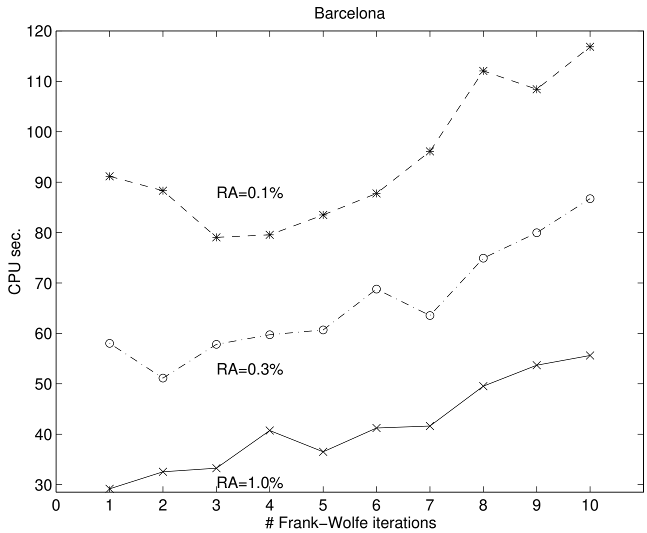

Fig. 12. Computing time versus effort spent in subproblem.

Cooper, L., LeBlanc, L.J.: Stochastic transportation problems and other network related convex problems. Naval Research Logistics Quarterly **24** (1977) 327–337

Dembo, R.S., Mulvey, J.M., Zenois, S.A.: Large-scale nonlinear network models and their applications. Operations Research **37** (1989) 353–372

Goldstein, A.A.: Convex programming in Hilbert space. Bulletin of the American Mathematical Society **70** (1964) 709–710

Hearn, D.W., Lawphongpanich, S., Ventura, J.A.: Restricted simplicial decomposition: Computation and extensions. Mathematical Programming Study **31** (1987) 99–118

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

Holmberg, K.: Efficient decomposition and linearization methods for the stochastic transportation problem. Computational Optimization and Applications **4** (1995) 293–316

Holmberg, K., Jörnsten, K.O.: Cross decomposition applied to the stochastic transportation problem. European Journal of Operational Research **17** (1984) 361–368

Larsson, T., Patriksson, M.: Simplicial decomposition with disaggregated representation for the traffic assignment problem. Transportation Science **26** (1984) 4–17

LeBlanc, L.J., Helgason, R.V. Boyce, D.E.: Improved efficiency of the Frank–Wolfe algorithm for convex network programs. Transportation Science **19** (1985) 445–462

LeBlanc, L.J., Morlok, E.K., Pierskalla, W.P.: An efficient approach to solving the road network equilibrium traffic assignment problem. Transportation Research **9** (1975) 309–318

Levitin, E.S., Polyak, B.T.: Constrained minimization methods. USSR Computational Mathematics and Mathematical Physics **6** (1966) 1–50

McCormick, C.P., Tapia, R.A.: The gradient projection method under mild differentiability conditions. SIAM Journal on Control **10** (1982) 93–98

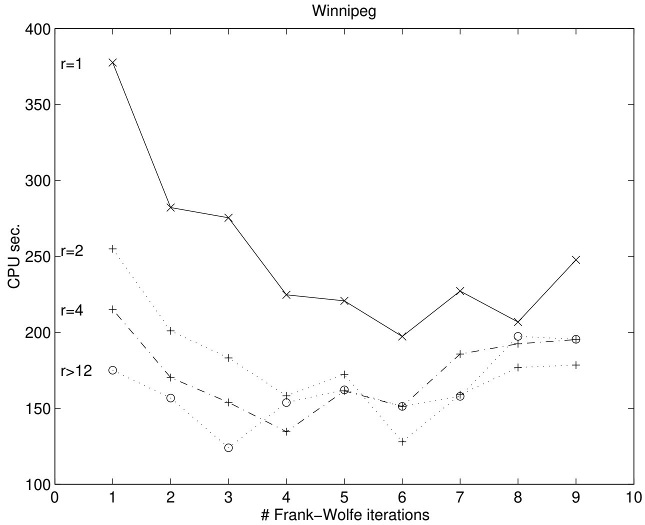

Fig. 13. Computing time versus effort spent in subproblem for different restrictions and a relative accuracy of 0.1%.

Migdalas, A.: A regularization of the Frank–Wolfe method and unification of certain nonlinear programming methods. Mathematical Programming **65** (1994) 331–345

Moré, J.J., Wright, S.J.: Optimization Software Guide. Frontiers in Applied Mathematics **14**, SIAM, Philadelphia, PA, 1993

Mulvey, J.M., Zenois, S.A., Ahlfeld, D.P.: Simplicial decomposition for convex generalized networks. Journal of Information & Optimization Sciences **11** (1990) 359–387

Nash, S. G.: Software Survey—NLP. OR/MS Today **22** (April, 1995) 60–71

Patriksson, M.: A unified framework of descent algorithms for nonlinear programs and variational inequalities. Doctoral dissertation, Department of Mathematics, Linköping University, Linköping, Sweden, 1993

Patriksson, M.: The Traffic Assignment Problem—Models and Methods. VSP BV, Utrecht, The Netherlands, 1994

Salinetti, G., Wets, R.J-B.: On the convergence of convex sets in finite dimensions. SIAM Review **21** (1979) 18–33

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

Weintraub, A., Ortiz, C., González, J.: Accelerating convergence of the Frank–Wolfe algorithm. Transportation Research **19B** (1985) 113–122

Wolfe, P.: Convergence theory in nonlinear programming. In: Integer and Nonlinear Programming, J. Abadie, ed., North-Holland, Amsterdam (1970) 1–36

This article was processed using the LaTeX macro package with LLNCS style

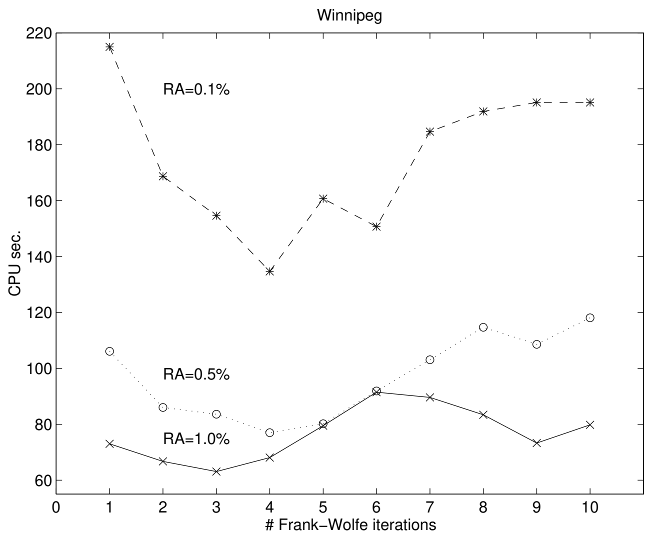

Fig. 14. Computing time versus effort spent in subproblem for different relative accuracies required and with master problem size restriction $r = 4$.