Simplicial Decomposition with Disaggregated Representation for the Traffic Assignment Problem

by

Torbjörn Larsson
Michael Patriksson

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

LiTH-MAT-R-89-34

Revised December 1990
Second revision April 1991

Abstract

The class of simplicial decomposition (SD) schemes have shown to provide efficient tools for nonlinear network flows. When applied to the traffic assignment problem, shortest route subproblems are solved in order to generate extreme points of the polyhedron of feasible flows, and, alternately, master problems are solved over the convex hull of the generated extreme points. We review the development of simplicial decomposition and the closely related column generation methods for the traffic assignment problem; we then present a modified, disaggregated, representation of feasible solutions in SD algorithms for convex problems over Cartesian product sets, with application to the symmetric traffic assignment problem. The new algorithm, which is referred to as disaggregate simplicial decomposition (DSD), is given along with a specialized solution method for the disaggregate master problem. Numerical results for several well known test problems and a new one are presented. These experimentations indicate that only few shortest route searches are needed; this property is important for large-scale applications. The modification proposed maintains the advantages of SD, and the results show that the performance of the new algorithm is at least comparable to that of state-of-the-art codes for traffic assignment. Moreover, the reoptimization capabilities of the new scheme are significantly better; this is a main motive for considering it. The reoptimization facilities, especially with respect to changes in origin-destination flows and network topology, make the new approach highly profitable for more complex models, where traffic assignment problems arise as subproblems.

Key Words: Traffic Equilibrium, Frank-Wolfe Algorithm, Simplicial Decomposition, Column Generation, Cartesian Product Sets

Introduction

Consider a transportation network $\mathcal{G}=(\mathcal{N},\mathcal{A})$, where each directed arc $a \in \mathcal{A}$ is associated with a positive travel time, $t_a(\mathbf{f})$. This travel time, or transportation cost, measures the disutility of using the arc as a function of the network flow $\mathbf{f}$. The functions $t_a(\mathbf{f})$ are usually referred to as *arc performance functions*, and are monotone as a result of congestion. For certain pairs of origins and destinations, $(p,q) \in \mathcal{C}$, where $\mathcal{C} \subset \mathcal{N} \times \mathcal{N}$, there is a given positive flow demand d_{pq}. Each O-D pair (p,q) is associated with a specific commodity. We denote the commodity flow directed from node p to node q through arc a by f_{apq}, giving rise to the total arc flow equation $f_a = \sum_{(p,q)\in\mathcal{C}} f_{apq}$. To measure the commodity flow into and out of a specific node $i \in \mathcal{N}$, we define $\mathcal{W}_i$ and $\mathcal{V}_i$ to be the sets of arcs initiated and terminated at node i, respectively. The problem of determining a network flow fulfilling the travel demands and a prescribed performance criterion is referred to as the *traffic assignment problem.*

In this paper, we consider static traffic assignment problems, modelling peak-hour urban traffic. Two main principles of optimality are normally considered. These are attributed to WARDROP,[1] although PIGOU[2] had already discussed these principles in similar terms. The first principle of optimality is based on the intuitive behaviour of traffic, i.e., each user of the traffic network seeks to minimize his/her own travel time; it is therefore known as the principle of *user equilibrium*. With the assumptions that each function $t_a(\mathbf{f})$ is integrable and that the Jacobian of $\mathbf{t} = (t_a)$ is positive semidefinite for all feasible flows, Wardrop's first conditions of optimality can be formulated as a convex mathematical program (e.g. DAFERMOS[3]). First to formulate this program are BECKMANN *et al.*,[4] in the case of separable cost functions. The assumption that the travel cost functions are integrable is, however, in some applications too restrictive. Nonintegrable functions $t_a(\mathbf{f})$ correspond to an asymmetric Jacobian of travel costs. The problem is therefore known as the *asymmetric traffic assignment problem*, and may be formulated for instance, as a variational inequality (SMITH;[5] DAFERMOS[6]), a nonlinear complementarity problem (AASHTIANI;[7] AASHTIANI and MAGNANTI[8]) or as a, generally nonconvex, mathematical program by the use of so called *gap functions* (e.g. HEARN *et al.*[9]). If separable costs are considered, the corresponding mathematical program will have an objective with additive terms $g_a(f_a) = \int_0^{f_a} t_a(s)ds$.

The second optimality principle is known as the *system optimum* principle, and corresponds to the situation in which the whole transportation system's disutility is minimized. The flows corresponding to the system optimum must be imposed upon the users, thus giving a problem of *prescription*, as opposed to the problem of *description* in the case of user equilibrium. Under the assumption that each function $t_a(\mathbf{f})$ is monotone and convex, this principle can be shown to be equivalent to a convex mathematical program, and the objective will have additive terms $g_a(\mathbf{f}) = t_a(\mathbf{f})f_a$.

There are a number of different versions of the traffic assignment problem. It is often extended to incorporate elastic demands; mode choice and trip distribution are sometimes included in combined models. In this paper, we limit ourselves to integrable travel cost functions and consider the basic model of traffic assignment with fixed travel demands and travel cost functions without link interactions. The reader may note that the algorithm to be presented can be modified for use in elastic demand and combined models.

The fixed demand Traffic Assignment Problem may, in both cases of optimality principles discussed above, be stated as:

$$
\begin{array}{llrcll}
[\mathbf{TAP}] & \min & T(\mathbf{f}) = \displaystyle\sum_{a\in\mathcal{A}} g_a(f_a) & & & \\
& \text{s.t.} & \displaystyle\sum_{a\in\mathcal{W}_i} f_{apq} - \sum_{a\in\mathcal{V}_i} f_{apq} & = & \left\{\begin{array}{rl} d_{pq} & \text{if } i=p \\ -d_{pq} & \text{if } i=q \\ 0 & \text{otherwise} \end{array}\right. & \forall\, i\in\mathcal{N} \quad \forall\,(p,q)\in\mathcal{C} \\
& & \displaystyle\sum_{(p,q)\in\mathcal{C}} f_{apq} & = & f_a & \forall\, a\in\mathcal{A} \\
& & f_{apq} & \geq & 0 & \forall\, a\in\mathcal{A} \quad \forall\,(p,q)\in\mathcal{C}.
\end{array}
$$

We will refer to this formulation as the *arc-node* formulation.

The traffic assignment problem has received a lot of attention; partly because of its practical importance, partly because the size of real life problems makes it a challenge for algorithmic development. Methods applied to traffic assignment include linearization and cyclic decomposition methods, and dual approaches.

The next section offers a review of the basic linearization methods applied to the traffic assignment problem. In Section 2 we present the disaggregate simplicial decomposition approach. A specialized method for the restricted master problem is also given. The method combines a scaled reduced gradient algorithm with an approximate Newton method. Numerical experiments for some well known problems and a new large-scale network is reported in Section 3, and in Section 4 we reach some conclusions about the algorithmic performance and present suggestions for further research and developments.

1 Review of some linearization methods for traffic assignment

One class of algorithms commonly applied to the traffic assignment problem is based on iterative linear approximations of the objective; the basis for this algorithmic class is the well known Frank-Wolfe method,[10] first developed for quadratic programming. It was suggested for the traffic assignment problem by Bruynooghe *et al.*[11] and Murchland;[12] LeBlanc *et al.*[13] showed that it was a viable approach by solving a real-world problem. This class of methods also includes extensions such as PARTAN directions[14] and the related method of Lupi,[15] the modified search direction of Fukushima,[16] and the restricted simplicial decomposition method of Hearn *et al.*[17] The column generation methods applied to traffic assignment, such as the one given by Leventhal *et al.*,[18] also belong to this class. In the following, we will outline the development of the various methods in this class, and relate them to each other. Especially, we establish the strong relations between simplicial decomposition algorithms for problems defined over Cartesian product sets and the column generation methods.

1.1 The Frank-Wolfe algorithm

The feasible direction method of Frank and Wolfe[10] is applicable to any nonlinear optimization problem with a pseudoconvex objective and linearly constrained feasible set; its performance is well known due to extensive research into the method and its use in various applications. The method has several good qualities; it is easy to understand and simple to implement. The core storage needed is small, and the algorithm is able to utilize problem structure. Both these properties are advantageous because of the size of practical problems. For convex problems, the linear subproblem provides a lower bound on the optimal objective value; this convergence controllability is an advantage compared to many other nonlinear programming methods. For problems defined over Cartesian product sets, the algorithm is amenable to parallel computation in the subproblem phase.

As applied to the user equilibrium case of traffic assignment, the algorithm has a nice intuitive interpretation in behavioural terms. In iteration k, a feasible flow, $\mathbf{f}^{(k)}$, is observed in the network $\mathcal{G}$. For every pair (p,q) of origins and destinations, the shortest route from p to q is calculated given the current flow $\mathbf{f}^{(k)}$, and a portion of the flow is shifted to the shortest routes, resulting in a new feasible flow $\mathbf{f}^{(k+1)}$ with a decreased objective value. The algorithm is reasonable from an intuitive point of view: the shifting of some flow to the currently shortest routes is the result of the decisions of individual travellers to seek their own shortest route to their destinations. For comparison, the algorithm is described below.

The algorithm is initialized by the determination of a feasible solution $\mathbf{f}^{(0)}$, for instance as an *all-or-nothing* assignment. At iteration k, the linearized subproblem

$$[\mathbf{LP}] \qquad \min \ \underline{T}(\mathbf{y}) = T\left(\mathbf{f}^{(k)}\right) + \nabla T\left(\mathbf{f}^{(k)}\right)^{\mathrm{T}} \cdot \left(\mathbf{y} - \mathbf{f}^{(k)}\right)$$

$$\begin{aligned} \text{s.t.} \quad \sum_{a\in\mathcal{W}_i} y_{apq} - \sum_{a\in\mathcal{V}_i} y_{apq} &= \begin{cases} d_{pq} & \text{if } i = p \\ -d_{pq} & \text{if } i = q \\ 0 & \text{otherwise} \end{cases} && \forall\, i \in \mathcal{N} \quad \forall\,(p,q) \in \mathcal{C} \\ \sum_{(p,q)\in\mathcal{C}} y_{apq} &= y_a && \forall\, a \in \mathcal{A} \\ y_{apq} &\geq 0 && \forall\, a \in \mathcal{A} \quad \forall\,(p,q) \in \mathcal{C} \end{aligned}$$

is solved. This problem is equivalent to $|\mathcal{C}|$ shortest route problems with arc costs $g_a'\left(f_a^{(k)}\right)$. The result is a shortest route pattern $\hat{\mathbf{y}}^{(k)}$, and a lower bound, $\underline{T}\left(\hat{\mathbf{y}}^{(k)}\right)$, on the optimal objective value. A line search problem

$$[\mathbf{LS}] \qquad \min_{l\in[0,1]} \ \overline{T}(l) = T\left(\mathbf{f}^{(k)} + l \cdot (\hat{\mathbf{y}}^{(k)} - \mathbf{f}^{(k)})\right)$$

is solved, with a solution $l^{(k)}$ as the optimal steplength, which gives the new iteration point $\mathbf{f}^{(k+1)} = \mathbf{f}^{(k)} + l^{(k)} \cdot (\hat{\mathbf{y}}^{(k)} - \mathbf{f}^{(k)})$. The upper bound on the objective value is updated and the process is repeated. The algorithm is terminated when the relative difference between the bounds are smaller than an *a priori* set parameter. As suggested by LeBlanc *et al.*[13] one may alternatively terminate when the changes in the arc flows are small.

The Frank-Wolfe algorithm has been implemented in a number of program packages, for instance TRAFFIC.[19] The performance is therefore well known among scientists and practitioners. The algorithm is experienced to be efficient in the first iterations. The overall performance of the algorithm is, however, not fully satisfactory. The theoretical convergence rate is only arithmetic,[20, 21] and the algorithm is, in later iterations, characterized by jamming. The main reason for this behaviour is that the search direction $\hat{\mathbf{y}}^{(k)} - \mathbf{f}^{(k)}$ generated by the subproblem solution tend to be perpendicular to the steepest descent direction as k increases. (In the traffic assignment context, a discussion on this topic may be found in Lupi.[15]) Moreover, it has been found that, in practice, the algorithm may generate cyclic flows,[22] although an optimal solution can not contain cycles.[23] These cyclic flows are very unlikely to be removed and thus degrade the performance of the method.

Different suggestions have been made for changing the search directions or steplengths to gain in efficiency. For the general problem, examples of modifications include the *away steps* of Wolfe,[21] HOLLOWAY's extension[24] and the accelerated Frank-Wolfe algorithms of MEYER.[25] Modified Frank-Wolfe algorithms applied to traffic assignment include PARTAN directions,[14] the modified steplengths of WEINTRAUB *et al.*[26] and the modified search directions of Fukushima,[16] among others. Some modifications are reported to show significantly improved efficiency.

1.2 Simplicial decomposition algorithms

An important group of modifications of the Frank-Wolfe algorithm are the *simplicial decomposition* (SD) algorithms based on Carathéodory's theorem (see e.g. [27]). A direct consequence of this theorem is that any point in a bounded polyhedral set $\mathbf{X}$ can be described by a convex combination of its extreme points. In SD algorithms, extreme points are generated algorithmically by the solution of the linear Frank-Wolfe subproblem. Alternately, a so called *master problem*, defined by a restricted set of extreme points, is solved in order to generate a new iteration point.

For the ease of presentation, we here consider a general convex problem over a bounded polyhedral set, which is formulated as

[NLP]
$$\mathbf{x}^* \in \arg \min_{\mathbf{x}\in\mathbf{X}} T(\mathbf{x}).$$

To formalize the above, suppose that in the kth iteration, the set of stored extreme points is $\left\{\hat{\mathbf{y}}^{(1)}, \hat{\mathbf{y}}^{(2)}, \dots, \hat{\mathbf{y}}^{(l)}\right\}$, where $l \le k$. Then the master problem is

[MP]
$$\begin{aligned} \min \quad & T\left(\sum_{i=1}^{l} \lambda_i \hat{\mathbf{y}}^{(i)}\right) \\ \text{s.t.} \quad & \sum_{i=1}^{l} \lambda_i = 1 \\ & \lambda_i \ge 0 \quad i = 1, \dots, l. \end{aligned}$$

The master problem generalizes the line search in the Frank-Wolfe algorithm. For $l > 2$

the master problem is, theoretically, as difficult to solve as the original problem [**NLP**]. Due to the potential difficulty of [**MP**], suggestions have been made for approximate solutions (e.g. [25]). Its particular structure, however, makes it possible to solve the problem efficiently by using specialized methods (see e.g. BERTSEKAS[28, 29]).

In two papers, VON HOHENBALKEN[30, 31] gives the theoretical grounds for simplicial decomposition methods for minimizing pseudoconvex functions on polytopes. The presentation of von Hohenbalken is closely related to that of Holloway,[24] who develops an extension to the Frank-Wolfe algorithm by means of an inner linearization (i.e., inner representation) of the feasible set, followed by restriction. The Holloway technique is a special case of the *inner linearization/restriction* type algorithms defined by GEOFFRION.[32] As applied to [**NLP**], von Hohenbalken's algorithm is, in fact, an instance from the algorithmic class of Holloway.

Holloway shows that the extended Frank-Wolfe algorithm converges with a linear rate and von Hohenbalken[31] shows that, for the special case of SD, the convergence is finite even if extreme points with zero weights are removed from one master problem to the next. The convergence result obtained by von Hohenbalken allows for the use of *column dropping*, that is, the possibility of removing extreme points with small weights from the problem. This is essential in large-scale applications. Theoretically, the maximum number of extreme points needed in any master problem is bounded by $\dim(\mathbf{X}) + 1$ according to Carathéodory's theorem. This number is, however, often too large to be useful as a limit in practice, and a stronger bound has been obtained by HEARN *et al.*[33] in defining a *restricted simplicial decomposition* (RSD) algorithm. The restriction refers to the possibility of bounding the number of stored extreme points by a number, r, supplied *a priori* by the user. They are able to show finite convergence, provided that r is greater then the dimension of an optimal face of the feasible set.

SACHER[34] specializes SD to quadratic programming and considers an extended problem with an unbounded polyhedral feasible set. He uses column dropping in the same manner as von Hohenbalken. SHETTY and BEN DAYA[35] study the same problem as Sacher, but concentrate on the master problem, in general the computationally most demanding part of the decomposition procedure. Utilizing the structure of the quadratic [**MP**], they specialize the well known reduced gradient and gradient projection algorithms.

The simplicial decomposition algorithm is highly related to various methods in quadratic programming. COTTLE and DJANG[36] show that the least-distance method of WOLFE[37] produces the same sequence of points as SD when applied to this special quadratic program. PANG[38] extends this result to hold for any convex quadratic program, when the symmetric programming algorithm[39] is applied to the equivalent program, obtained from the (implicit) inner representation. See DJANG[40] for a thorough study on these topics.

When applied to convex uncapacitated multicommodity flow problems, the simplicial decomposition algorithm alternates between the generation of extremal flows, defined by patterns of shortest routes, and the solution of the original problem over a restricted feasible set, given by the convex hull of a number of extremal flows. The first to discuss this algorithmic approach is Murchland;[12] the first implementation is due to CANTOR and GERLA,[41] who apply the algorithm to the problem of optimal routing in computer communication networks. They propose the use of Rosen's gradient projection algorithm (see e.g. [27]) for [**MP**], which is attractive because of the easy projection onto the simple constraints. By the explicit use of Carathéodory's theorem, they bound the number of extreme points stored by $|\mathcal{A}|+1$. Best[42] studies the same problem as Cantor and Gerla, and proposes the use of a conjugate direction method[43] for [**MP**]. Florian[44] applies the away step procedure of Wolfe[21] on the convex hull of stored extremal flows. (The away step approach is not directly applicable to the original formulation of the traffic assignment problem, since it involves calculating the direction from the longest route pattern towards the current flow; the calculation of longest simple routes in a network with cycles is known to be $\mathcal{NP}$-hard.) When comparing the directional derivatives, the away step direction is always chosen instead of the Frank-Wolfe direction, indicating the superiority of Wolfe's approach over the Frank-Wolfe algorithm. Van Vliet and Dow[45] compare SD to an heuristic *quantal loading* procedure; their limited experimentations indicate that the heuristic is advantageous. They do not state, however, how the SD master problem is solved. Without this information a comparison cannot be conclusive. Fukushima[16] suggests storing a fixed number of extremal flows and exchanging the Frank-Wolfe search direction with a direction obtained from a convex combination of the stored subproblem solutions. This algorithm is highly related to the conceptual algorithm of Meyer.[25] Guélat[46] develops an SD algorithm, where [**MP**] is solved by a reduced gradient method. The algorithm is restarted with regular intervals by the dropping of all extremal flows stored.

The fact that the problem [**MP**] has relatively few variables makes it favourable to use second order methods for its solution; von Hohenbalken[31] is the first to suggest such methods. He considers the possibility of using quasi-Newton as well as Newton directions. Pang and Yu[47] approximate the master problem by a quadratic program, for which they apply the algorithm of Van de Panne and Whinston.[39] Hearn *et al.*[17] use the projected Newton method of Bertsekas[28, 29] on the master problem in their RSD algorithm, and also consider an approximation of the master problem as in [47]. They use the result in [33], restricting the number of retained extreme points by a user-specified parameter, r. (In order to enable the use of efficient Newton methods for the master problem, r must be kept small. On the other hand, a smaller value of r will give rise to a larger number of main iterations. The proper choice of r is therefore difficult to make.) Finally, we mention the Newton algorithm of Dembo and Tulowitzki,[48] in which the quadratic subproblem is solved by partan search, embedded in a simplicial decomposition scheme. The approaches of Guélat[46] and Hearn *et al.*[17] are considered to be the *state-of-the-art* algorithms for the symmetric traffic assignment problem.

In [49, 50, 47], SD algorithms are extended to the asymmetric traffic assignment problem, formulated as a variational inequality. The algorithm in [49] is combined with a column dropping scheme, governed by the gap function (see e.g. [9]). In order to ensure convergence, extreme caution must be taken in the column dropping scheme.

1.3 Column generation algorithms

The traffic assignment problem may alternatively be formulated using route flow variables. For each pair (p,q) of origins and destinations, we denote the set of simple routes from p to q by $\mathcal{R}_{pq}$ and the flow on route r from p to q by h_{pqr}. By defining an arc-route incidence matrix (δ_{pqra}) for $\mathcal{G}$, i.e., $\delta_{pqra} = 1$ if route $r \in \mathcal{R}_{pq}$ contains arc a, and 0 otherwise, arc flows may be calculated from route flows according to the following relation.

$$f_a = \sum_{(p,q)\in\mathcal{C}} \sum_{r\in\mathcal{R}_{pq}} \delta_{pqra} h_{pqr}$$

The traffic assignment problem is then equivalently formulated as

$$
\begin{array}{llrcll}
[\mathbf{TAP}] & \min & T(\mathbf{f}) & = & \displaystyle\sum_{a\in\mathcal{A}} g_a(f_a) & \\
& \text{s.t.} & \displaystyle\sum_{r\in\mathcal{R}_{pq}} h_{pqr} & = & d_{pq} & \forall\,(p,q)\in\mathcal{C} \\
& & h_{pqr} & \geq & 0 \quad \forall\, r\in\mathcal{R}_{pq} & \forall\,(p,q)\in\mathcal{C} \\
& & \displaystyle\sum_{(p,q)\in\mathcal{C}} \sum_{r\in\mathcal{R}_{pq}} \delta_{pqra} h_{pqr} & = & f_a \quad \forall\, a\in\mathcal{A}. &
\end{array}
$$

We will refer to this formulation as the *arc-route* formulation. It can be traced back to Gibert,[51] and Dafermos and Sparrow,[52] although route flow variable formulations are well known from linear multicommodity flows, see e.g. Ford and Fulkerson[53] and Tomlin.[54] The explicit use of this formulation requires that all possible routes are enumerated *a priori*; since, in general, the number of routes grows exponentially with the size of the network, this is obviously impractical for large-scale applications.

To prevent all routes being enumerated, Gibert suggests generating the routes *as needed*; later, Leventhal *et al.*[18] proposed the same technique, which works as follows. Assume that a nonempty subset of the routes, $\hat{\mathcal{R}}_{pq}$, between p and q, has been generated, and that the restricted problem has been solved. By traversing the corresponding flow through the network and finding the shortest routes for all O-D pairs, new favourable routes may be generated. This corresponds to the subproblem phase of the Frank-Wolfe algorithm. The restricted set of routes is augmented, and the process is repeated until no more favourable routes can be found. This procedure is known as *column generation*.

The column generation methods, as applied to the arc-route formulation of the traffic assignment problem, differ mostly with respect to the order in which the route generation, restriction and a cyclic decomposition is combined.[55] The basis for most column generation algorithms is the Gauss-Seidel type decomposition method of Dafermos and Sparrow.[52] The decomposition is made over origin-destination pairs, and a so called *equilibration operator* approach is used on *a priori* generated routes within each commodity. The operator shifts a portion of the flow from the most expensive route towards the least expensive one, so that the two routes have equal cost; if this is not possible, the most expensive route will be given a zero flow. This technique is applied until the O-D pair is (approximately) equilibrated, after which it is applied to the next O-D pair. Bruynooghe *et al.*[11] apply the same algorithmic scheme on the arc-node formulation where commodities are associated with origins.

To our knowledge, the first to consider a similar methodology in a column generation context is Gibert.[51] He observes that the calculation of longest routes in the algorithm of Bruynooghe *et al.* is cumbersome, due to the presence of cycles in the network. He instead proposes using the algorithm on a restricted set of routes, where this calculation is almost trivial. The first algorithm of Leventhal *et al.*[18] is, in essence, the same as the one given by Gibert. The algorithm extends the cyclic decomposition method of Dafermos and Sparrow to include column generation, and may therefore be seen as an extension of the column generation methods[53, 54] to nonlinear flows. In each main iteration, Leventhal *et al.* choose to augment the set of routes by one new route only; they are able to show finite convergence, even if routes with zero flows are dropped. For linear cost functions, they also propose the use of a quadratic programming method due to Van de Panne and Whinston[56] to solve the restricted problem. One of the algorithms presented by Nguyen[57] is very similar to the first method of Leventhal *et al.* Instead of applying the equilibration operator on the single-commodity problem, he uses a reduced gradient technique.

Several other researchers have presented similar algorithms for the arc-route formulation of the traffic assignment problem and the problem of optimal routing in computer communication networks. Common to these approaches is that they are based on different combinations of cyclic decomposition, column generation and restriction, and that they all solve shortest route problems more often than the algorithms outlined in the preceding paragraph. For large-scale problems, this will probably make these algorithms less efficient, due to the complexity of shortest route algorithms. For further reading, we refer to Florian[58] and Patriksson.[55]

Column generation methods have also been extended to variational inequality and nonlinear complementarity formulations of the asymmetric traffic assignment problem. The algorithm of Bertsekas and Gafni[59] is a Newton-type linearization method, based on diagonal approximations of the Jacobian. Aashtiani and Magnanti[60] also consider Newton-type methods, for a nonlinear complementarity formulation. They apply Lemke's algorithm[61] to the linear complementarity subproblem in a cyclic decomposition scheme. Smith[62] uses a column generation technique based on a reformulation of the variational inequality as a nonconvex mathematical program. A descent method is presented, but no numerical results are provided.

2 Disaggregate simplicial decomposition

Column generation is also a well known concept in large-scale linear programming; the most important technique of this type is the Dantzig-Wolfe decomposition principle.[63] This is a method for structured large-scale linear programs, usually over a Cartesian product set with additional coupling constraints. The advantage of utilizing several convexity constraints in the master problem, if possible, is well known in Dantzig-Wolfe decomposition.

Now, consider again the general problem [**NLP**], and assume that the feasible set is a *Cartesian product*, i.e., that $\mathbf{X} = \prod_{i=1}^{n} \mathbf{X}_i$, and $T(\mathbf{x}) = T(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$. Assume further that, for each separate set $\mathbf{X}_i$, m_i extreme points, denoted by $\left\{\hat{\mathbf{y}}_i^{(1)}, \hat{\mathbf{y}}_i^{(2)}, \ldots, \hat{\mathbf{y}}_i^{(m_i)}\right\}$, have been generated. If the strategy of using one convexity constraint per individual set $\mathbf{X}_i$ is applied in a simplicial decomposition algorithm, the master problem [**MP**] is replaced by

$$[\mathbf{DMP}] \qquad \begin{array}{lrcll} \min & \multicolumn{4}{l}{T\left(\sum_{j=1}^{m_1} \lambda_1^{(j)} \hat{\mathbf{y}}_1^{(j)}, \sum_{j=1}^{m_2} \lambda_2^{(j)} \hat{\mathbf{y}}_2^{(j)}, \ldots, \sum_{j=1}^{m_n} \lambda_n^{(j)} \hat{\mathbf{y}}_n^{(j)}\right)} \\ \text{s.t.} & \sum_{j=1}^{m_i} \lambda_i^{(j)} & = & 1 & \qquad i = 1, 2, \ldots n \\ & \lambda_i^{(j)} & \geq & 0 \quad j = 1, 2, \ldots, m_i & \quad i = 1, 2, \ldots n. \end{array}$$

Since each set $\mathbf{X}_i$ is represented separately, we will refer to this problem as the *disaggregate master problem*, and the decomposition scheme as the *disaggregate simplicial decomposition* (DSD) algorithm. Although the number of variables in [**DMP**] is, in general, larger than in [**MP**], the constraints are still highly structured, so that specialized methods may be employed for its solution.

2.1 Disaggregate simplicial decomposition for the traffic assignment problem

Consider again the problem [**TAP**]. Disregarding the implicit arc-flow defining constraints, the entire feasible set is actually a Cartesian product with respect to the separate commodities, that is, the constraint matrix has a block-diagonal shape. This enables the use of disaggregate simplicial decomposition by using one convexity constraint in a disaggregate master problem for each origin-destination pair. This improvement has, however, not been fully exploited for the symmetric traffic assignment problem.

In order to formulate the disaggregate master problem, we assume that a nonempty subset $\hat{\mathcal{R}}_{pq}$ of the set of simple routes $\mathcal{R}_{pq}$ is known. In the disaggregate master problem, each convexity variable λ_{pqi} will be associated with a route i in $\hat{\mathcal{R}}_{pq}$, and denote the portion of the demand d_{pq} distributed along route i. Let $\boldsymbol{\lambda}_{pq}$ be the vector of convexity variables λ_{pqi} for all $i \in \hat{\mathcal{R}}_{pq}$. The vector $\boldsymbol{\lambda}$ then consists of all such $\boldsymbol{\lambda}_{pq}$.

The arc flows may be calculated from the weighted route flows according to:

$$f_a = \sum_{(p,q)\in\mathcal{C}} d_{pq} \sum_{i\in\hat{\mathcal{R}}_{pq}} \delta_{pqia}\lambda_{pqi}. \tag{1}$$

The function $T(\mathbf{f})$ is, through (1), a function of the convexity variables $\boldsymbol{\lambda}$; the notation $T(\boldsymbol{\lambda})$ will therefore be used in the following.

Utilizing the disaggregated representation the restricted master problem becomes:

$$
\begin{array}{llrcll}
\textbf{[RMP]} & \min & T(\boldsymbol{\lambda}) & & & \\
& \text{s.t.} & \sum_{i \in \hat{\mathcal{R}}_{pq}} \lambda_{pqi} & = & 1 & \forall\, (p,q) \in \mathcal{C} \\
& & \lambda_{pqi} & \geq & 0 \quad \forall\, i \in \hat{\mathcal{R}}_{pq} & \forall\, (p,q) \in \mathcal{C}.
\end{array}
$$

The reader may note that, via the substitution $h_{pqi} = \lambda_{pqi} d_{pq}$ and (1), the problem [**RMP**] is equivalent to the arc-route formulation of the traffic assignment problem, with the exception that only a subset of the routes is available. The two formulations of the traffic assignment problem, either in terms of route flows, or in terms of arc flows, differ only in the description of the polyhedron of feasible flows. From this simple observation, we may conclude that the constraints of the arc-route formulation in fact define an inner representation of the feasible set in the arc-node formulation of [**TAP**]. Moreover, the disaggregate simplicial decomposition algorithm is the link between simplicial decomposition and column generation methods for traffic assignment.

The DSD algorithm works as follows. Suppose that a disaggregate master problem, defined by a restricted set of routes, $\hat{\mathcal{R}}_{pq}$, $\forall\, (p,q) \in \mathcal{C}$, has been solved. Given this solution, the shortest routes are calculated for all commodities. The sets $\hat{\mathcal{R}}_{pq}$ are augmented by the routes not contained in the sets already, and the procedure is repeated. The algorithm is valid; the proof given by von Hohenbalken[31] may be applied, although his approach includes the dropping of all columns with zero weights.

The disaggregate master problem is a convex program with very simple linear constraints. The same type of algorithms employed for [**MP**] may be used in the case of disaggregated representation. The larger number of variables of the disaggregate master problem, however, makes it necessary to approximate the Hessian matrix by its diagonal whenever Newton-type methods are used.

2.2 Algorithms for the disaggregate master problem

In this section, we will outline the two algorithms used for the solution of [**RMP**]. These algorithms are used in combination; a first order method based on scaled reduced gradients is employed to reach a near-optimal solution, which is sufficiently accurate in most applications. An approximate second order method, based on a quadratic, separable approximation of the objective, is utilized when a highly accurate solution is demanded.

We first present a scaled reduced gradient method especially adapted to [**RMP**]. (For a general description of reduced gradient methods, see e.g. Bazaraa and Shetty.[27])

- **Initialization:**

  Let $k = 0$, and choose a point $\boldsymbol{\lambda}^{(0)}$ such that

$$\begin{aligned} \sum_{i \in \hat{\mathcal{R}}_{pq}} \lambda_{pqi}^{(0)} &= 1 & & \forall\ (p,q) \in \mathcal{C} \\ \lambda_{pqi}^{(0)} &\geq 0 & \forall\ i \in \hat{\mathcal{R}}_{pq} & \forall\ (p,q) \in \mathcal{C}. \end{aligned}$$

- **Search direction generation:**

  Let

$$i_{pq} \in \arg\max_{i \in \hat{\mathcal{R}}_{pq}} \left\{ \lambda_{pqi}^{(k)} \right\}$$

  be the basic variable index and let N_{pq} denote the set of nonbasic variables for commodity (p, q), respectively. Compute

$$\mathbf{r}_{pq} = \nabla_{N_{pq}} T\left(\boldsymbol{\lambda}^{(k)}\right) - \nabla_{i_{pq}} T\left(\boldsymbol{\lambda}^{(k)}\right) \cdot \mathbf{1},$$

  where each component of $\nabla T\left(\boldsymbol{\lambda}^{(k)}\right)$ is calculated as

$$\frac{\partial}{\partial \lambda_{pqi}} T\left(\boldsymbol{\lambda}^{(k)}\right) = d_{pq} \sum_{a \in \mathcal{A}} \delta_{pqia} g_a^{'} \left(f_a^{(k)}\right).$$

  (The sum in the right hand side is, in the user equilibrium case, equal to the travel time on route i for origin-destination pair (p, q).) This calculation is made through the use of the relation (1), which defines the aggregate arc flows $f_a^{(k)}$. The reduced gradient is the vector of all $\mathbf{r}_{pq}$. The search direction is defined as the negative of a scaled reduced gradient, that is,

$$\begin{aligned} \overline{r}_{pqi}^{(k)} &= \begin{cases} -s_{pqi} r_{pqi} & \text{if} \quad i \neq i_{pq} \quad \text{and} \quad r_{pqi} \leq 0 \\ \\ -\lambda_{pqi}^{(k)} s_{pqi} r_{pqi} & \text{if} \quad i \neq i_{pq} \quad \text{and} \quad r_{pqi} > 0 \end{cases} \\ \overline{r}_{pqi_{pq}}^{(k)} &= -\sum_{i \neq i_{pq}} \overline{r}_{pqi}^{(k)}, \end{aligned}$$

  where $\mathbf{S} = \operatorname{diag}(s_{pqi})$ is a positive definite scaling matrix. (The use of the factors $\lambda_{pqi}^{(k)}$ in the definition of $\overline{\mathbf{r}}_{pq}^{(k)}$ is a well known technique for deflecting the search direction away from near-active nonnegativity constraints. In our implementation, we have chosen the scaling coefficients $s_{pqi} = 1/d_{pq},\ \forall\ i \in \hat{\mathcal{R}}_{pq}$. This choice of scaling coefficients corresponds, in the user equilibrium case, to using a search direction $\overline{\mathbf{r}}$ based directly on the route travel costs.)

- **Convergence test:**

  If $\overline{\mathbf{r}}^{(k)} = \mathbf{0}$, then terminate $\rightarrow \boldsymbol{\lambda}^{(k)}$ solves [**RMP**].

- **Line search:**

  Let $$l_{\max} = \min_{(p,q)\in\mathcal{C}} \left\{ \min_{i\in\hat{\mathcal{R}}_{pq}} \left\{ -\frac{\lambda_{pqi}^{(k)}}{\overline{r}_{pqi}^{(k)}} \;:\; \overline{r}_{pqi} < 0 \right\} \right\}.$$

  Solve the line search problem

  $$[\mathbf{LS}] \qquad \min_{l\in[0,l_{\max}]} \overline{T}(l) = T\left(\boldsymbol{\lambda}^{(k)} + l \cdot \overline{\mathbf{r}}^{(k)}\right),$$

  which gives the solution $l^{(k)}$. (This line search is performed in total arc flow variables by first calculating the arc flows corresponding to $l_{\max}$.)

- **New iteration point and convergence test:**

  Let $\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + l^{(k)} \cdot \overline{\mathbf{r}}^{(k)}$.

  Terminate if a stopping criterion is fulfilled $\rightarrow \boldsymbol{\lambda}^{(k)}$ solves [**RMP**] approximately.

  Let $k := k+1$.
  Go to the search direction generation phase.

The scaled reduced gradient method is efficient in finding near-optimal solutions, but more slowly convergent if a very high accuracy is required, since it is a first order method. Therefore, the reduced gradient method is replaced by a second order method after two successive iteration points have been found within a predetermined small distance. The second order method used is a constrained Newton algorithm, where the subproblem is approximated by utilizing only the diagonal of the Hessian matrix at each iteration point. A line search is then made towards the solution of the approximate subproblem. This approach has been considered for the arc-route formulation of the optimal routing problem, see e.g. Bertsekas.[64]

Due to the diagonal approximation of the Hessian, the quadratic programming subproblem is separable into $|\mathcal{C}|$ problems:

$$[\mathbf{QPS}_{pq}^{(k)}] \qquad \begin{aligned} \min \quad & \sum_{i\in\hat{\mathcal{R}}_{pq}} \left\{ a_{pqi}\left(\lambda_{pqi} - \lambda_{pqi}^{(k)}\right) + \frac{1}{2} b_{pqi}\left(\lambda_{pqi} - \lambda_{pqi}^{(k)}\right)^2 \right\} \\ \text{s.t.} \quad & \sum_{i\in\hat{\mathcal{R}}_{pq}} \lambda_{pqi} = 1 \\ & \lambda_{pqi} \geq 0 \quad \forall\, i \in \hat{\mathcal{R}}_{pq}, \end{aligned}$$

where

$$\begin{aligned} a_{pqi} &= d_{pq} \sum_{a\in\mathcal{A}} \delta_{pqia} g_a^{'}\left(f_a^{(k)}\right) \\ b_{pqi} &= d_{pq}^2 \sum_{a\in\mathcal{A}} \delta_{pqia} g_a^{''}\left(f_a^{(k)}\right). \end{aligned}$$

It is a well known fact that this type of problem is analytically solvable at low computational cost; this is done by the Lagrangean dualization of the convexity constraint and a line search on the piecewise quadratic one-dimensional dual function. The line search is performed by studying breakpoints of the piecewise linear derivative, see e.g. Brucker.[65]

3 Numerical experiments

In order to prove the feasibility and analyze the practical efficiency of the proposed algorithm, it was coded in double precision FORTRAN-77 on a SUN 4/390 computer and numerically tested on networks known from the literature, as well as a new network, modelling the city of Barcelona. The shortest route algorithm used is an implementation of Dijkstra's algorithm.[66] The starting point was always chosen as an all-or-nothing assignment at zero flows. Since it was found unfavourable to solve the line search problems accurately, we implemented an approximate Armijo-type line search[67] (with the acceptance parameter set to 0.2 or 0.3). We demanded a more accurate solution of the master problem for each consecutive main iteration; this guaranteed the global convergence of the algorithm, and ensured that new routes were added to the restricted set in every new master problem. In the reduced gradient algorithm, we therefore applied the stopping criterion $T^{(k+1)} > (1-\gamma)\, T^{(k)}$, where $T^{(k+1)}$ is the result from the kth Armijo line search in a certain master problem; the parameter γ was initiated to a value between 0.0001 and 0.1, and decreased in each main iteration. In the Newton phase, we terminated the algorithm if the distance between consecutive points was smaller than a parameter ε. This parameter was initiated to a value between 0.001 and 0.1, and decreased after each main iteration. The changeover from the reduced gradient method to the Newton method was based on the same criterion. In this case, however, the parameter was kept constant through all main iterations. After having made this changeover once, only the second order method was used for the remaining master problems. The global termination criterion was that the relative difference between the bounds was small enough.

The data structure used for storing routes is a vector containing arc indices of all routes generated so far. To keep track of a specific commodity flow pattern, the starting index and length of all routes available for each commodity are maintained. A similar choice of data structure can be found in Aashtiani and Magnanti.[60]

Below we outline some performance results. The first test problem is a small network of 19 arcs, 13 nodes and 4 commodities, and may be found in Nguyen and Dupuis.[68] The objective is quadratic. Results for the DSD algorithm and two related methods are shown in Table 1. The fourth column consists of accumulated figures of the number of routes generated, line searches made in the scaled reduced gradient and approximate second order methods, respectively, and the total number of function evaluations. (Note that the total number of routes generated also defines the number of variables in the last master problem.) The results from the Frank-Wolfe algorithm are found in Nguyen and Dupuis, and for the simplicial decomposition algorithm, the results were obtained from a double precision version of the RSD algorithm of Hearn *et al.*[17] (These results were kindly provided by Professor S. Lawphongpanich.) Our algorithm produced a lower bound of 85,028.03 at termination, giving a relative error of $5.3 \cdot 10^{-5}$ %. The true deviation from

Iter.	$T^{(\text{Iter})}$ Frank-Wolfe	$T^{(\text{Iter})}$ RSD	$T^{(\text{Iter})}$ DSD	Iteration History
0	125,500.00	125,500.00	125,500.00	4; 0; 0; 1
1	94,253.37	94,253.38	92,301.92	7; 10; 14; 56
2	86,847.30	86,560.91	85,054.16	9; 10; 24; 133
3	86,447.56	85,895.14	85,028.07	10; 10; 40; 243
4	86,229.62	85,513.68		
5	85,939.96	85,215.00		
6		85,047.23		
7		85,028.07		
8				
10	85,555.79			

Table 1: Comparison between various related methods for the Nguyen/Dupuis problem

optimality of the final solution is, however, only $2.6 \cdot 10^{-10}$ %. (This figure is obtained from a lower bound produced by another execution of the algorithm, where a much higher accuracy was demanded.) The CPU time used was 0.15 seconds. Hearn *et al.* report an objective value of 85,027.6, computed in single precision arithmetic, after 0.22 seconds, on an IBM 3081D computer. (Arithmetic in single precision is of course faster than in double precision, but may lead to slightly infeasible solutions, as is indicated by the results of Hearn *et al.*) This computer is roughly as fast as the one used by ourselves. (MATLAB benchmark tests performed at our department have demonstrated that the SUN 4/390 is about 1.3 times faster than the SUN SPARCstation 1, which, according to [69] is 1.3 times slower than the IBM 3081D.)

The second example is due to Steenbrink.[70] The network has 12 O-D pairs, 9 nodes and 36 arcs, and a quadratic objective function. The DSD algorithm terminates after six main iterations with an objective value of 16,957.6747, which gives a relative error of $2.4 \cdot 10^{-4}$ %. The true deviation from optimality is less than $4.0 \cdot 10^{-6}$ %. In the process, 38 routes are generated, and 223 function evaluations are performed within 37 line searches, of which 15 are performed within the reduced gradient phase. The CPU time used was 0.26 seconds. The simplicial decomposition algorithm of Pang and Yu[47] terminated after 19 main iterations with the same objective value as found by ourselves.

The third problem is taken from Barton and Hearn.[71] The network has 4 O-D pairs, 9 nodes and 18 arcs. The objective value at termination is 1,453.15222, obtained after five main iterations, with a relative error of $1.8 \cdot 10^{-4}$ %, the true deviation being less than $2.4 \cdot 10^{-7}$ %. A total number of 19 routes are generated, and 353 function evaluations are performed within 55 line searches, of which 5 are performed within the reduced gradient phase. The CPU time used was 0.17 seconds. Hearn *et al.* report an objective function value of 1,453.14 after 14 main iterations in 0.47 seconds.

The fourth problem is another well known numerical example, a highly aggregated network modelling the city of Sioux Falls, South Dakota. It consists of 76 arcs, 24 nodes and 528

Iter.	$T^{(\text{Iter})}$ Frank-Wolfe	$T^{(\text{Iter})}$ Fukushima	$T^{(\text{Iter})}$ DSD	Iteration History	% Cost Difference
0	159.897	159.897	158.58605	528; 0; 1	663
1	70.552	70.552	46.06718	895; 3; 4	33.4
2	58.517	58.517	42.55642	1020; 15; 18	7.12
3	55.503	55.503	42.32227	1058; 42; 73	0.614
4	52.274	52.274	42.31356	1070; 99; 268	0.0344
5	50.499	50.499			
6	47.085	47.085			
7	45.860	45.860			
8	45.318	45.318			
9	45.018	45.018			
20	43.035	42.731			
30	42.656	42.656			
40	42.506	42.347			
50	42.452	42.330			

Table 2: Comparison between various related methods for the Sioux Falls network

commodities of non-zero demand. Data for the problem may be found in LeBlanc *et al.*,[13] and results for the DSD algorithm and two related methods are given in Table 2. The last column of the table contains the average relative difference between the cost of the longest and the shortest route used, taken over all origin-destination pairs. This difference gives a notion of the violation of the Wardrop equilibrium conditions. (The calculation of this violation, which may be a natural termination criterion for user equilibrium problems, is not easily performed in an ordinary SD algorithm, whereas in DSD, it is very simple.) The column with data from the method of Fukushima[16] corresponds to his use of the 10 latest subproblem solutions to create the search direction. The result for the Frank-Wolfe method is taken from the same paper. At termination, DSD reports a lower bound of 42.311, and a relative error of $6.0 \cdot 10^{-3}$ %, the true error being less than $4.9 \cdot 10^{-4}$ %. The reduced gradient phase was active during the whole execution. (One may note that the number of line searches needed to obtain a certain upper bound on the objective value seems to grow more slowly than in the Frank-Wolfe algorithm.) The CPU time used in the DSD algorithm was 7.04 seconds.

The fifth test example is a version of the network of Hull, Canada. It consists of 501 nodes, 798 arcs and 142 origin-destination pairs of non-zero demand. The disaggregate simplicial decomposition algorithm was terminated after three main iterations, using 5.75 CPU seconds. The upper bound reported was 21,516.85, and the lower bound 21,516.64, making the solution accurate to a relative error of about $9.5 \cdot 10^{-4}$ %. The true error, however, was less than $6.8 \cdot 10^{-8}$ %. In the process 56 line searches were performed, of which 30 were made in the approximate Newton phase. A total of 158 function evaluations were made, and 307 routes were generated. Hearn *et al.*[17] solve a different version of the problem using RSD. They report a final relative error of $7.0 \cdot 10^{-3}$ %, using 15.08 CPU seconds.

The sixth test example is a version of the network of Winnipeg, Canada. This network has 1052 nodes, 2836 arcs and 4345 O-D pairs of non-zero demand. The DSD algorithm was executed using two accuracy levels. The first execution resulted in a solution with a relative accuracy of 0.50 %, with an upper and lower bound of 886,094.76 and 881,690.67, respectively. The true error was less than $9.3 \cdot 10^{-2}$ %. Using five main iterations, the algorithm generated 14,481 routes and performed 241 function evaluations within 132 line searches, in 310 CPU seconds. The second execution gave an upper bound of 885,656.67, and a lower bound of 884,134.68. Thus, the relative accuracy was 0.17 %, while the true error was less than $4.3 \cdot 10^{-2}$ %. The execution involved six main iterations, during which 16,744 routes were generated, of which 9,583 were actually used, 177 line searches were made using 380 function evaluations. The vector of arc indices used for storing routes contained about 500,000 elements at termination. The CPU time used was 457 seconds. The scaled reduced gradient method was used throughout both executions. Hearn *et al.* report a relative error of $8.4 \cdot 10^{-2}$ % in 485 CPU seconds for a slightly different version of the same problem.

The last test example is a network modelling the city of Barcelona. It consists of 1020 nodes, 2522 arcs and 7922 origin-destination pairs. The disaggregate simplicial decomposition algorithm was executed with two different levels of accuracy. The first execution was terminated after two main iterations requiring 85 CPU seconds, with a relative error of 2.64 %, the true error being less than 1.34 %. The final upper bound was 1,285,408.52. The number of routes generated was 17,589; at termination, 16,833 of these were used. The number of line searches and function evaluations was 30 and 45, respectively. The second execution was terminated after 271 CPU seconds and four main iterations, with a relative error of 0.99 %. The real deviation from optimality was less than 0.29 %. The upper bound was 1,272,128.59. The number of line searches and function evaluations was 89 and 155, respectively. A total of 20,528 routes were generated, of which 18,376 were actually used. The vector of arc indices used for storing routes contained 570,000 elements. In both executions, the master problem was always solved using scaled reduced gradient directions. For practical purposes, the accuracy obtained in the latter execution should be fully satisfactory. The best lower bound ever obtained for this problem was 1,268,414.61 (from another execution).

We conclude this section with the general observation that the quality of the upper bound at termination often is much better than is indicated by the lower bound. There are two reasons for this. Firstly, if the algorithm is terminated during the master problem phase, the final lower bound was computed using a tangential approximation at a point that might differ considerably from the final one. Secondly, the lower bounds provided by tangential approximations are often poor.

4 Conclusions, discussion and further research

Simplicial decomposition methods constitute an efficient approach to the solution of the problem of traffic assignment. The disaggregated representation of the feasible set results in a modification of the existing methods. Since the proposed algorithm is an extension of the Frank-Wolfe method and the class of simplicial decomposition methods, advantages of these methods are maintained; these include the utilization of the network structure and the effective bound on the optimal value obtained from the subproblem phase. The numerical experimentations verify that the proposed algorithm is a viable approach. Moreover, the results obtained for the Winnipeg and Barcelona networks show that it is practical for large-scale problems. In particular, the memory requirements are not prohibitively large for today's standard computer equipment. The computational efficiency of the algorithm is at least as high as that of RSD, in the cases where a comparison has been possible to make. To obtain a relatively high accuracy, it seems that the scaled reduced gradient approach for the master problem is quite sufficient; only for very accurate solutions, the second order method is used.

It is well known that, in the Frank-Wolfe algorithm, the vast majority of the computational effort is spent in the shortest route subproblem phase; this is true also for the RSD algorithm, see e.g. [17]. A natural strategy for constructing an efficient algorithm for the traffic assignment problem would therefore be to minimize the number of shortest route calculations necessary to solve the problem. Indeed, this was one of our main objectives when designing the disaggregate simplicial decomposition algorithm. The number of main iterations needed is, approximately, bounded by the maximal number of routes actually used in any origin-destination pair. (For example, the second execution of the DSD algorithm for the Winnipeg network resulted in six main iterations, corresponding to seven shortest route calculations, and seven routes were indeed used in 11 O-D pairs.) This implies that the number of main iterations, and therefore, shortest route problems, is very limited if the network is not heavily congested. Thus, the DSD algorithm seems to be optimal with respect to the number of shortest route calculations, and these computations were, in fact, found to be very limited compared to the line searches performed in the master problem.

As a consequence of the small number of main iterations, we believe that column dropping will not be necessary. The reader may also note, that the subproblem phase in DSD generates routes with a greater striving for the optimal solution, in the sense that the algorithm terminates when no more profitable routes can be obtained from the subproblem. In the ordinary simplicial decomposition scheme this is not the case, since the new route pattern may consist of routes that have been generated previously in different patterns.

As mentioned earlier, Hearn *et al.*[33] introduce a parameter r in simplicial decomposition algorithms, defining the maximum number of retained extreme points. In order to guarantee finite convergence, this parameter must be greater than the (*a priori* unknown) dimension of an optimal face of the feasible set. It is interesting to note that this parameter can in the case of DSD be given an interpretation in terms of equilibrium route flows. Through the disaggregated representation of the feasible flows, the dimension of an optimal face of the polyhedron of feasible flows for commodity (p,q) is equal to the number of routes actually used within the commodity in an equilibrium situation, minus one. A corresponding set of parameters r_{pq}, $(p,q) \in \mathcal{C}$, may be defined by letting r_{pq} be the maximum number of routes, in O-D pair (p,q), retained in the DSD algorithm. In order to ensure finite convergence of the DSD algorithm, for each O-D pair (p,q) the parameter r_{pq} must not be less than the number of routes actually used within commodity

(p, q) in an equilibrium situation - a quite obvious result.

Another main objective for developing the disaggregate simplicial decomposition algorithm is its excellent reoptimization capabilities. Reoptimization with respect to changes in arc performance functions, travel demands and network topology are easily handled through appropriate modifications of the latest master problem, and its solution. For changes in origin-destination flows, the reoptimization is easy, since the previously optimal master problem solution, $\boldsymbol{\lambda}^*$, is still feasible. In the case of network topology changes, i.e., the removal and addition of some arcs, it is only necessary to drop routes that uses an arc to be removed. In ordinary SD algorithms for traffic assignment, it is however necessary to drop *every* shortest route pattern that includes such an arc. The easy reoptimization is enabled due to the fact that more information is stored. The efficiency of the method and its easy reoptimization are of great importance when traffic assignment problems arise as subproblems in, for instance, time-sliced traffic assignment, prediction of intercity freight flows, equilibrium network design problems, and the estimation of origin-destination trip matrices.

The method is also of special interest when studying traffic flows in an urban area, employing different levels of network aggregation within the model. Assume that we want to give a detailed description of the traffic flows in a specific subarea, e.g. the heavily congested parts of the road network. The analysis can then be performed in two steps: first, we compute a traffic assignment solution for the whole urban area using a rough network model. The given route flows are then used to calculate an induced origin-destination trip matrix for the subarea, and a traffic assignment problem is solved for the smaller urban area in the more detailed network representation. Since the proposed method provides route flows, the induced trip matrix is easily calculated. (As opposed to DSD, the Frank-Wolfe and ordinary SD methods must be modified with an extra book-keeping in order to provide route flows.)

Since the proposed algorithm is computationally comparable to state-of-the-art SD algorithms for traffic assignment, it is our view that the much better reoptimization facilities motivates the use of DSD in applications where reoptimizations are to be made.

Subjects for further research are the application of simplicial decomposition with disaggregated representation to other classes of nonlinear programs over Cartesian product sets and also to more complex methods and algorithms, where the efficient reoptimization facilities can be exploited in an iterative solution scheme. Applications would include the above mentioned models, where traffic assignment problems may arise as subproblems; included are also nonlinear network flow models with a Dantzig-Wolfe structure of the feasible set, such as capacitated traffic assignment, where Lagrangean penalty methods have been proposed by e.g. Hearn and Ribera.[72] The implementation described above is still experimental, and will be further improved. The reoptimization facilities will also be investigated in the near future.

Acknowledgements

This research was done within a project (no. V1054, ASTERIX) of the European Community Research Programme DRIVE. The purpose of the ASTERIX project is to develop a simulation tool for testing and evaluating road transport informatics (RTI) systems for large urban areas. In such a simulation tool, there is a need for an efficient traffic assignment algorithm with good reoptimization facilities, since slightly different scenarios are to be analyzed sequentially.

The authors would like to thank Professor Anna Nagurney for her constructive criticism and support during the reviewing process, and two anonymous referees for several suggestions leading to clarifications in the presentation of this paper. We would also like to thank Professor Jaime Barcelo and Doctor Athanasios Migdalas for valuable discussions and suggestions for further research. Professor Michael Florian provided us with the networks of Hull and Winnipeg, and Professor Barcelo supplied the network of Barcelona. Olof Damberg helped us with some coding details. The research leading to this paper was supported by the Swedish Transport Research Board (TFB) (Dnr. 16/88-62).

References

[1] J.G. Wardrop, "Some theoretical aspects of road traffic research," *Proceedings of the Institute of Civil Engineers*, Part **II**, pp. 325-378 (1952).

[2] A.C. Pigou, *The Economics of Welfare*, MacMillan & Co., London, 1920.

[3] S.C. Dafermos, "The traffic assignment problem for multiclass-user transportation networks," *Transportation Science* **6**, pp. 73-87 (1972).

[4] M. Beckmann, C. McGuire and C.B. Winsten, *Studies in the Economics of Transportation*, Yale University Press, New Haven, CT, 1956.

[5] M.J. Smith, "The existence, uniqueness and stability of traffic equilibria," *Transportation Research* **13B**, pp. 295-304 (1979).

[6] S. Dafermos, "Traffic equilibrium and variational inequalities," *Transportation Science* **14**, pp. 42-54 (1980).

[7] H.Z. Aashtiani, "The multi-modal traffic assignment problem," Ph.D. dissertation, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA, 1979.

[8] H.Z. Aashtiani and T.L. Magnanti, "Equilibria on a congested transportation network," *SIAM Journal on Algebraic and Discrete Methods* **2**, pp. 213-226 (1981).

[9] D.W. Hearn, S. Lawphongpanich and S. Nguyen, "Convex programming formulations of the asymmetric traffic assignment problem," *Transportation Research* **18B**, pp. 357-365 (1984).

[10] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly* **3**, pp. 95-110 (1956).

[11] M. BRUYNOOGHE, A. GIBERT AND M. SAKAROVITCH, "Une méthode d'affectation du trafic," in *Proceedings of the Fourth International Symposium on the Theory of Traffic Flow*, Karlsruhe, 1968, W. Lentzback and P. Baron (eds.), pp. 198-204, Beiträge zur Theorie des Verkehrsflusses Strassenbau und Strassenverkehrstechnik Heft 86, Herausgeben von Bundesminister für Verkehr, Abteilung Strassenbau, Bonn, 1969.

[12] J.D. MURCHLAND, "Road network traffic distribution in equilibrium," in *Proceedings of Mathematical Models in the Social Sciences*, Matematisches Forschungsinstitut, Oberwolfach, 1969, R. Henn, H.P. Künzi and H. Schubert (eds.), pp. 145-183, II Oberwolfach-Tagung über Operations Research, Operations Research-Verfahren **8**, Anton Hain Verlag, Meisenheim am Glan, 1970.

[13] L.J. LEBLANC, E.K. MORLOK AND W.P. PIERSKALLA, "An efficient approach to solving the road network equilibrium traffic assignment problem," *Transportation Research* **9**, pp. 308-318 (1975).

[14] L.J. LEBLANC, R.V. HELGASON AND D.E. BOYCE, "Improved efficiency of the Frank-Wolfe algorithm for convex network programs," *Transportation Science* **19**, pp. 445-462 (1985).

[15] M. LUPI, "Convergence of the Frank-Wolfe algorithm in transportation networks," *Civil Engineering Systems* **3**, pp. 7-15 (1986).

[16] M. FUKUSHIMA, "A modified Frank-Wolfe algorithm for solving the traffic assignment problem," *Transportation Research* **18B**, pp. 169-177 (1984).

[17] D.W. HEARN, S. LAWPHONGPANICH AND J.A. VENTURA, "Restricted simplicial decomposition: computation and extensions," *Mathematical Programming Study* **31**, pp. 99-118 (1987).

[18] T. LEVENTHAL, G. NEMHAUSER AND L. TROTTER, JR., "A column generation algorithm for optimal traffic assignment," *Transportation Science* **7**, pp. 168-176 (1973).

[19] S. NGUYEN AND L. JAMES, "TRAFFIC - an equilibrium traffic assignment program," Publication no. 17, Centre de recherche sur les transports, Université de Montréal, 1975.

[20] M.D. CANON AND C.D. CULLUM, "A tight upper bound on the rate of convergence of the Frank-Wolfe algorithm," *SIAM Journal on Control* **6**, pp. 509-516 (1968).

[21] P. WOLFE, "Convergence theory in nonlinear programming," in *Integer and nonlinear programming*, J. Abadie (ed.), pp. 1-36, North-Holland, Amsterdam, 1970.

[22] B.N. JANSON AND C. ZOZAYA-GOROSTIZA, "The problem of cyclic flows in traffic assignment," *Transportation Research* **21B**, pp. 299-310 (1987).

[23] G.F. NEWELL, *Traffic Flow on Transportation Networks*, MIT Press, Cambridge, MA, 1980.

[24] C.A. HOLLOWAY, "An extension of the Frank and Wolfe method of feasible directions," *Mathematical Programming* **6**, pp. 14-27 (1974).

[25] G. MEYER, "Accelerated Frank-Wolfe algorithms," *SIAM Journal on Control* **12**, pp. 655-663 (1974).

[26] A. Weintraub, C. Ortiz and J. Gonzales, "Accelerating convergence of the Frank-Wolfe algorithm," *Transportation Research* **19B**, pp. 113-122 (1985).

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

[28] D.P. Bertsekas, "Projected Newton methods for optimization problems with simple constraints," *SIAM Journal on Control and Optimization* **20**, pp. 221-246 (1982).

[29] D.P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, San Diego, CA, 1982.

[30] B. von Hohenbalken, "A finite algorithm to maximize certain pseudoconcave functions on polytopes," *Mathematical Programming* **9**, pp. 189-206 (1975).

[31] B. von Hohenbalken, "Simplicial decomposition in nonlinear programming algorithms," *Mathematical Programming* **13**, pp. 49-68 (1977).

[32] A.M. Geoffrion, "Elements of large scale mathematical programming," *Management Science* **16**, pp. 652-691 (1970).

[33] D.W. Hearn, S. Lawphongpanich and J.A. Ventura, "Finiteness in restricted simplicial decomposition," *Operations Research Letters* **4**, pp. 125-130 (1985).

[34] R.S. Sacher, "A decomposition algorithm for quadratic programming," *Mathematical Programming* **18**, pp. 16-30 (1980).

[35] C.M. Shetty and M. Ben Daya, "A decomposition procedure for convex quadratic programs," *Naval Research Logistics Quarterly* **35**, pp. 111-118 (1988).

[36] R.W. Cottle and A. Djang, "Algorithmic equivalence in quadratic programming, I: A least-distance programming problem," *Journal of Optimization Theory and Applications* **28**, pp. 275-301 (1979).

[37] P. Wolfe, "Algorithm for a least-distance programming problem," *Mathematical Programming Study* **1**, pp. 190-205 (1974).

[38] J.-S. Pang, "An equivalence between two algorithms for quadratic programming," *Mathematical Programming* **20**, pp. 152-165 (1981).

[39] C. Van de Panne and A. Whinston, "The symmetric formulation of the simplex method for quadratic programming,", *Econometrica* **37**, pp. 507-527 (1969).

[40] A. Djang, "Algorithmic equivalence in quadratic programming," Ph.D. dissertation, Department of Operation Research, Stanford University, Stanford, CA, 1980.

[41] D.G. Cantor and M. Gerla, "Optimal routing in a packet switched network," *Transactions on Computers* **c-23**, pp. 1062-1069 (1974).

[42] M.J. Best, "Optimization of nonlinear criteria subject to flow constraints," in *Proceedings of the 1975 Midwest Symposium on Circuits and Systems*, Kansas City, KS, pp. 438-443, 1975.

[43] M.J. Best and K. Ritter, "A class of accelerated conjugate direction methods for linearly constrained minimization problems," *Mathematics of Computation* **30**, pp. 478-504 (1976).

[44] M. Florian, "An improved linear approximation algorithm for the network equilibrium (packet switching) problem," in *Proceedings of the 1977 IEEE Conference on Decision and Control*, New Orleans, TX, pp. 812-818, 1977.

[45] D. Van Vliet and P.D.C. Dow, "Capacity-restrained road assignment," *Traffic Engineering and Control* **20**, pp. 296-305 (1979).

[46] J. Guélat, "Algorithmes pour le problème d'affectation d'équilibre avec demandes fixes: comparaisons," Publication no. 299, Centre de recherche sur les transports, Université de Montréal, 1983.

[47] J.-S. Pang and C.-S. Yu, "Linearized simplicial decomposition methods for computing traffic equilibria on networks," *Networks* **14**, pp. 427-438 (1984).

[48] R.S. Dembo and U. Tulowitzki, "Computing equilibria on large multicommodity networks: an application of truncated quadratic programming algorithms," *Networks* **18**, pp. 273-284 (1988).

[49] S. Lawphongpanich and D.W. Hearn, "Simplicial decomposition of the asymmetric traffic assignment problem," *Transportation Research* **18B**, pp. 123-133 (1984).

[50] M.J. Smith, "An algorithm for solving asymmetric equilibrium problems with a continuous cost-flow function," *Transportation Research* **17B**, pp. 365-371 (1983).

[51] A. Gibert, "A method for the traffic assignment problem," Report LBS-TNT-95, Transportation Network Theory Unit, London Business School, London, 1968.

[52] S.C. Dafermos and F.T. Sparrow, "The traffic assignment problem for a general network," *Journal of Research of the National Bureau of Standards* **73B**, pp. 91-118 (1969).

[53] L.R. Ford and D.R. Fulkerson, "A suggested computation for maximal multicommodity network flows," *Management Science* **5**, pp. 97-101 (1958).

[54] J.A. Tomlin, "Minimum-cost multicommodity network flows," *Operations Research* **14**, pp. 45-51 (1966).

[55] M. Patriksson, "The traffic assignment problem - theory and algorithms," Report LiTH-MAT-R-90-29, Linköping Institute of Technology, Linköping, Sweden, 1990.

[56] C. Van de Panne and A. Whinston, "The simplex and the dual method for quadratic programming," *Operations Research Quarterly* **15**, pp. 355-388 (1964).

[57] S. Nguyen, "A mathematical programming approach to equilibrium methods of traffic assignment with fixed demands," Publication no. 138, Département d'informatique et de recherche opérationelle, Université de Montréal, 1973.

[58] M. Florian, "Nonlinear cost network models in transportation analysis," *Mathematical Programming Study* **26**, pp. 167-196 (1986).

[59] D.P. Bertsekas and E.M. Gafni, "Projection methods for variational inequalities with application to the traffic assignment problem," *Mathematical Programming Study* **17**, pp. 139-159 (1982).

[60] H.Z. Aashtiani and T.L. Magnanti, “A linearization and decomposition algorithm for computing urban traffic equilibria,” in *Proceedings of the 1982 IEEE International Large Scale Systems Symposium*, Virginia Beach, VA, pp. 8-19, 1982.

[61] C.E. Lemke, “Bimatrix equilibrium points and mathematical programming,” *Management Science* **11**, pp. 681-689 (1965).

[62] M.J. Smith, “The existence and calculation of traffic equilibria,” *Transportation Research* **17B**, pp. 291-303 (1983).

[63] G.B. Dantzig and P. Wolfe, “The decomposition algorithm for linear programming,” *Operations Research* **8**, pp. 101-111 (1960).

[64] D.P. Bertsekas, “A class of optimal routing algorithms for communication networks,” in *Proceedings of the 5th International Conference on Computer Communications*, Atlanta, GA, J. Salz (ed.), pp. 71-76, 1980.

[65] P. Brucker, “An O(n) algorithm for quadratic knapsack problems,” *Operations Research Letters* **3**, pp. 163-166 (1984).

[66] G. Gallo and S. Pallottino, “Shortest path algorithms,” in *Fortran Codes for Network Optimization*, B. Simeone *et al.* (eds.), pp. 3-79, Annals of Operations Research **13**, 1988.

[67] L. Armijo, “Minimization of functions having Lipschitz continuous partial derivatives,” *Pacific Journal of Mathematics* **16**, pp. 1-3 (1964).

[68] S. Nguyen and C. Dupuis, “An efficient method for computing traffic equilibria in networks with asymmetric transportation costs,” *Transportation Science* **18**, pp. 185-202 (1984).

[69] J.J. Dongarra, “Performance of various computers using standard linear equations software,” Report CS-89-85, Computer Science Department, University of Tennessee, Knoxville, TN, 1990.

[70] P.A. Steenbrink, *Optimization of Transport Networks*, John Wiley & Sons, London, 1974.

[71] R.R. Barton and D.W. Hearn, “Decomposition techniques for nonlinear cost multicommodity flow problems,” Research Report 79-2, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, 1979.

[72] D.W. Hearn and J. Ribera, “Bounded flow equilibrium problems by penalty methods,” in *Proceedings of the 1980 IEEE International Conference on Circuits and Computers*, pp. 162-166, 1980.