Available online at www.sciencedirect.com

ScienceDirect

European Journal of Operational Research 175 (2006) 1577–1587

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Issues in the implementation of the DSD algorithm for the traffic assignment problem

Pablo A. Lotito

METALAU Project, INRIA, Rocquencourt, BP 105, Le Chesnay Cedex F-78153, France
INRETS, Arcueil, France

Available online 11 April 2005

Abstract

In this paper we consider the practical implementation of the disaggregated simplicial decomposition (DSD) algorithm for the traffic assignment problem. It is a column generation method that at each step has to solve a huge number of quadratic knapsack problems (QKP). We propose a Newton-like method to solve the QKP when the quadratic functional is convex but not necessarily strictly. Our O(n) algorithm does not improve the complexity of the current methods but extends them to a more general case and is better suited for reoptimization and so a good option for the DSD algorithm. It also allows the solution of many QKP's simultaneously in a vectorial or parallel way.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Large scale optimization; Non linear programming; Quadratic programming; Traffic; Network flows

1. Introduction

Given a transportation network, the traffic assignment problem consists in determining a flow that satisfies a given demand between certain pairs of origins and destinations and a certain equilibrium condition. The demand of transport can be fixed or variable with respect to travel times, this is the elastic demand case.

When the condition is that *no traveller can improve his travel time by unilaterally changing routes*, which is the Wardrop Principle (in Section 2 we give a mathematical formulation), the solution is said to be a user equilibrium (UE). This definition assumes that all the users of the network behave identically and have complete information about route costs. To relax that, the travel times can be defined as a deterministic term plus a random variable. Now the Wardrop principle has its stochastic version wich is called stochastic

E-mail address: pablo.lotito@inrets.fr

user equilibrium (SUE) (see [15,8]). The word Static may be added to stress the fact that network conditions are constant, in contrast with dynamic user equilibriums (DUE) where demand and travel time parameters can vary with time (for a definition see [1]).

Under certain assumptions the UE problem becomes a convex optimization problem. These assumptions require that the network be strongly connected and the travel time on each arc be a positive function of the flow over that arc (separable case).

More general cases have been proposed, they consider that travel time functions depend on the flow over many arcs or there are different classes of flows that affect differently the travel time. The difficulty for the solution of this problems is that in general they are no more equivalent to an optimization problem but to a variational inequality. Under certain hypothesis this variational inequality can be solved with an iterative algorithm that at each step solve a standard UE.

The UE problem can appear also as a subproblem when considering bilevel problems where the lower level problem is a traffic assignment problem. Even if the UE may seem very simple to be realistic our interest in its numerical solution stems from the fact that to solve other problems, as the ones mentioned above, many instances of it must be efficiently solved. Efficiency will depend on the specific problem because sometimes we will need a fast solution less approximated and sometimes a very precise solution no matter the time.

Among the different algorithms to obtain the UE (see [13] for a review) the so called DSD, developed by Larsson and Patriksson and presented in [7], appears among the more efficient ones. This algorithm has as a subproblem, a family of quadratic knapsack problems, i.e., the minimization of a quadratic objective function subject to an only one linear constraint (the knapsack constraint). Clearly, the lowest the computation time of these subproblems the better the performance of the global problem. To solve them, the authors suggested to use the algorithm presented by Brucker in [2].

More precisely, we will call QKP the following problem:

$$\text{(QKP)} \qquad \begin{aligned} \min \quad & \sum_{i\in I} a_i x_i + \frac{1}{2} b_i x_i^2, \\ \text{s.t.} \quad & \sum_{i\in I} x_i = 1, \quad x_i \geqslant 0, \ \forall i \in I, \end{aligned} \tag{1}$$

where I is a finite set of indices.

If we define the functions $f_i(x)$ as $a_i x_i + \frac{1}{2} b_i x_i^2$ when $x \geqslant 0$ and $+\infty$ otherwise and replace the knapsack constraint by $\sum_{i\in I} x_i = y$, the QKP is equivalent to compute the inf-convolution of the functions f_i evaluated at $y = 1$. Using the Fenchel transform and some properties of the dual function (see [4] or [14]) we obtain an analytic formula for the QKP solution, $x_i = f_i^F(p)$. The value of p can be obtained solving the equation defined by the knapsack constraint, that is looking for a zero of the function $\sum_{i\in I} f_i^F(p) - 1$. A derivation of this formula in a more general case without the Fenchel transform is presented in the Section 3.1.

So far, the solution of the QKP is equivalent to search a zero of a function that in our case is piece-wise linear, convex and monotone. The method of Brucker performs a binary search on the index set of critical values of this function. This $O(n)$ algorithm extends and improves the results obtained in [6] where an $O(n \log n)$ algorithm is presented to solve a less general problem. In [12] the authors present a *randomized* version with worst case second order complexity but with expected linear time and a small time constant. In [10] the authors develop four algorithms for the solution of a more general problem. Instead of a quadratic objective function they consider strictly convex, continuously differentiable functions. Their algorithms are designed to be suitable for parallel implementation. One of these algorithms can be considered similar to Brucker's, another can be seen as a Newton method to find a zero of the function mentioned above.

Based in those previous works we propose an algorithm that uses a Newton search directive and that can be used also in the case of non-strict convexity (all the fore-mentioned algorithms consider strictly convex objective functions). The algorithm presented here has shown to be more efficient in large scale problems and better suited for reoptimization. Indeed, when the master algorithm is near to converge, the successive QKP's are similar and using the solution obtained in the last iteration as starting point of the new one makes them converge in only one iteration, i.e., the expected time tends to 1, in contrast to the binary-search based algorithms.

Finally our algorithm can be easily parallelized and used vectorially for the ease of programming in vectorial languages as Scilab and Matlab. The parallelization of the whole DSD algorithm can be found in [5].

We begin by defining the traffic assignment problem. After that we describe the DSD algorithm and we explain with more detail the subproblem that needs the solution of many quadratic knapsack problems. Then we state the quadratic knapsack problem and derive its analytical solution. Then, using the results exposed, we propose the algorithm analyzing its performance and showing some numerical experiments.

2. The traffic assignment problem

We present the traffic assignment problem (TAP) as it is stated in [7]. Let us consider a traffic network represented by the graph $(\mathscr{N},\mathscr{A})$ where $\mathscr{N}$ is the set of nodes and $\mathscr{A}$ is the set of directed arcs (links). Each arc a is associated with the positive real number $t_a(f)$ which is the travel time of the link as a function of the network flow f. For certain pairs of nodes (p,q) there is a positive fixed[1] flow demand d_{pq} from p to q. We call $\mathscr{C} \subset \mathscr{N} \times \mathscr{N}$ the set of origin-destination pairs and we associate each pair with a commodity. The flow of the commodity d_{pq} over the arc a will be called f_{apq}; then the total flow over the arc a is

$$f_a = \sum_{(p,q)\in\mathscr{C}} f_{apq}. \tag{2}$$

The TAP is now to determine a network flow fulfilling the travel demands and a prescribed performance criterion.

Two of the most common performance criteria considered are attributed to Wardrop (see [15]). The first one, based on the intuitive behavior of traffic, states that the users seek to minimize their own travel times and it is known as the *user equilibrium.* The second one, known as the *system optimum*, calls for the minimization of the total travel time. Two mathematical formulations of these principles can be given: the arc-node formulation and the route-arc formulation. We will work with the second one as it is better adapted for our algorithmic formulation (see [7]).

For each pair (p,q) in $\mathscr{C}$ we call $\mathscr{R}_{pq}$ the set of available simple routes from p to q and δ the incidence matrix of the arc a in the route $r \in \mathscr{R}_{pq}$, i.e., $\delta_{ra} = 1$ if the route r uses the arc a and $\delta_{ra} = 0$ otherwise. We define λ_r the portion of the demand of the commodity (p,q) using the route $r \in \mathscr{R}_{pq}$. Now the flow of the commodity (p,q) in the arc a is

$$f_a = \sum_{(p,q)\in\mathscr{C}} d_{pq} \sum_{r\in\mathscr{R}_{pq}} \delta_{ra}\lambda_r. \tag{3}$$

In the case that the travel time on the arc a depends only on the total flow on that arc and using the previous definitions the TAP, for both Wardrop's principles can be equivalently formulated as

[1] As the elastic demand case can be easily converted into a fixed demand case (see [15]) we will not consider it.

$$\begin{aligned}
\min \quad & T(f) = \sum_{a\in\mathscr{A}} g_a(f_a), \\
\text{s.t.} \quad & \sum_{r\in\mathscr{R}_{pq}} \lambda_r = 1, \quad \forall (p,q) \in \mathscr{C}, \\
& \lambda_r \geqslant 0, \quad \forall r \in \mathscr{R}_{pq}, \quad \forall (p,q)\in\mathscr{C}, \\
& \sum_{(p,q)\in\mathscr{C}} d_{pq} \sum_{r\in\mathscr{R}_{pq}} \delta_{ra}\lambda_r = f_a, \quad \forall a \in \mathscr{A},
\end{aligned} \tag{4}$$

where $g_a(x) = x_a t_a(x)$ for the system equilibrium and $g_a(x) = \int_0^{x_a} t_a(s)\,\mathrm{d}s$ if we want to obtain the user equilibrium.

For a given problem, to determine the complete set of routes can be a very difficult task since the number of routes grows exponentially with the size of the network. That's why route-based methods were considered not viable in the past. At present the advance on computing power together with a column generation method (generating the routes when they are needed) make viable a route-based formulation (see [3] for a analysis on the viability of route-based algorithms). This is the principal idea of the DSD algorithm (see [7]).

2.1. The DSD algorithm

Considering the equivalent formulation (4) and replacing the value of f_a in the formulation we have to solve

$$\begin{aligned}
\min \quad & T(\lambda) = \sum_{a\in\mathscr{A}} g_a\left(\sum_{(p,q)\in\mathscr{C}} d_{pq} \sum_{r\in\mathscr{R}_{pq}} \delta_{ra}\lambda_r\right), \\
\text{s.t.} \quad & \sum_{r\in\mathscr{R}_{pq}} \lambda_r = 1, \quad \forall (p,q) \in \mathscr{C}, \\
& \lambda_r \geqslant 0, \quad \forall r \in \mathscr{R}_{pq}, \quad \forall (p,q)\in\mathscr{C}.
\end{aligned} \tag{5}$$

As we have mentioned before, we will not work with the complete set $\mathscr{R}_{pq}$, instead we consider a smaller set of routes $\widehat{\mathscr{R}}_{pq}$. We will call master problem (MP) the restriction of (5) to this set. Following Larsson and Patriksson [7] the DSD algorithm works as follows:

"Suppose that a disaggregated master problem, defined by a restricted set of routes, $\widehat{\mathscr{R}}_{pq}, \forall (p,q) \in \mathscr{C}$, has been solved. Given this solution, the shortest routes are calculated for all commodities. The sets $\widehat{\mathscr{R}}_{pq}$ are augmented by the routes not contained in the sets already, and the procedure is repeated."

As the computation of the shortest routes is very well documented (see [11]) we will focus on the solution of the MP.

In [7] the MP is solved using a combination of a first order method (scaled reduced gradient method) with a second order method (restricted quasi Newton method) but one of the authors suggested in a personal communication that using only the second order method was enough.

To solve the MP we make, at the k-iteration, a second order approximation of $T(\lambda)$ around the current point λ_k:

$$T(\lambda) \simeq T(\lambda_k) + \nabla T(\lambda_k)(\lambda - \lambda_k) + \frac{1}{2}(\lambda - \lambda_k)^{\mathrm{T}} H(\lambda_k)(\lambda - \lambda_k), \tag{6}$$

where $H(\lambda_k)$ is the Hessian matrix of $T(\lambda)$ evaluated at λ_k. To obtain the next point λ_{k+1} we minimize the right-hand size of (6) restricted to the convexity and positivity constraints and where the Hessian matrix is replaced by its diagonal. Following the previous considerations we have to solve the problem

$$\begin{aligned} \min \quad & \nabla T(\lambda_k)\lambda + \frac{1}{2}(\lambda - \lambda_k)^{\mathrm{T}} D_k (\lambda - \lambda_k), \\ \text{s.t.} \quad & \sum_{r \in \mathscr{R}_{pq}} \lambda_r = 1, \quad \forall (p,q) \in \mathscr{C}, \\ & \lambda_r \geqslant 0, \quad \forall r \in \mathscr{R}_{pq}, \quad \forall (p,q) \in \mathscr{C}, \end{aligned} \tag{7}$$

where D_k is the diagonal of the Hessian of T computed at λ_k. The solution to this problem gives λ^* and λ_{k+1} is obtained through a linear search in the direction $d_k = \lambda^* - \lambda_k$, i. e., $\lambda_{k+1} = \lambda_k + \alpha d_k$ where α minimizes $T(\lambda_k + \alpha d_k)$.

2.2. The associated quadratic knapsack problem

Looking with more detail at the problem (7) we can observe that it can be decomposed for each commodity (p,q), i.e., we can solve independently for each $(p,q) \in \mathscr{C}$ the problem

$$\begin{aligned} \min \quad & \sum_r a_r \lambda_r + \frac{1}{2} b_r (\lambda_r - \mu_r)^2, \\ \text{s.t.} \quad & \sum_{r \in \mathscr{R}_{pq}} \lambda_r = 1, \\ & \lambda_r \geqslant 0, \quad \forall r \in \mathscr{R}_{pq}, \end{aligned} \tag{8}$$

where $\mu = \lambda_k$ and

$$a_r \triangleq d_{pq} \sum_{a \in A} g_a'(f_a) \delta_{ra}, \tag{9}$$

$$b_r \triangleq d_{pq}^2 \sum_{a \in A} g_a''(f_a) \delta_{ra} \tag{10}$$

come from the gradient and the Hessian matrix of T evaluated at λ_k.

The problem in (8) is then a QKP where $I = \mathscr{R}_{pq}$. As the feasible set is compact and the objective function is continuous it will always have at least one solution. If all the coefficients b_i are greater than zero, the objective function will be strictly convex and the solution will be unique. It will be the case if all the travel time functions are strictly increasing with respect to the flow the, as we can see in the definition (10).

Even if it can sound non realistic, for practical purposes we may want to consider arcs with constant travel times (dummy links for example). In order to be able to solve those cases without treating some arcs separately we will consider the possibility of having some b_i equal to zero. Consequently, the objective function will be convex but not strictly and we can have more than one solution.

3. Numerical solution of the QKP

We start by giving an analytical characterization of the solution of the QKP that will help us to design our numerical algorithm.

3.1. Analytical solution of the QKP

We will consider the QKP problem (8) for a fixed commodity (p,q) so we will drop the corresponding indices.

Introducing the appropriate multiplier π for the convexity constraint and ρ_i, $i \in I$, for the constraints $\lambda_i > 0$ we have the following optimality conditions:

$$\begin{cases} a_i + b_i(\lambda_i - \mu_i) + \pi - \rho_i = 0, & \forall i \in I, \\ \sum_{i\in I}\lambda_i = 1, \\ \rho_i \geqslant 0, & \forall i \in I, \\ \lambda_i \geqslant 0, & \forall i \in I, \\ \lambda_i\rho_i = 0, & \forall i \in I, \end{cases} \tag{11}$$

which, due to the convexity of the objective function and the linearity of the constraints, are necessary and sufficient for the global optimum.

In the equation system (11), the coefficients are a_i, b_i, μ_i, for $i \in I$ and the unknowns are λ_i, ρ_i, for $i \in I$ and π. To solve the system we will consider all the equations but the second obtaining a solution that depends on π (the multiplier associated to the knapsack constraint). Then we will search the value of π that makes the remaining equation be verified.

So, considering π as a parameter we solve the following reduced system:

$$\begin{cases} a_i + b_i(\lambda_i - \mu_i) + \pi - \rho_i = 0, & \forall i \in I, \\ \rho_i \geqslant 0, \quad \lambda_i \geqslant 0, & \forall i \in I, \\ \rho_i\lambda_i = 0, & \forall i \in I. \end{cases} \tag{12}$$

For the sake of clarity let us define $I_0 = \{i : b_i = 0\}$ and $I_+ = \{i : b_i > 0\}$. Also we define $c_i = b_i\mu_i - a_i$ and $\pi_0 = \max\{-a_i : i \in I_0\}$, if $I_0 = \emptyset$ we set $\pi_0 = -\infty$.

Proposition 1. *If $\pi < \pi_0$, the system (12) is incompatible. If $\pi \geqslant \pi_0$, the solutions of the system (12) (in terms of π) are given by*

$$\rho_i(\pi) = \begin{cases} \max(0, \pi - c_i), & \text{if } b_i > 0; \\ \pi + a_i, & \text{if } b_i = 0. \end{cases} \tag{13}$$

$$\lambda_i(\pi) = \begin{cases} \frac{1}{b_i}\max(0, c_i - \pi), & \text{if } b_i > 0; \\ 0, & \text{if } b_i = 0 \text{ and } \pi + a_i > 0; \\ \text{any non negative number}, & \text{if } b_i = 0 \text{ and } \pi + a_i = 0. \end{cases} \tag{14}$$

Proof. If $\pi < \pi_0$ then I_0 is not empty and for $i \in I_0$ from the first equation we have $a_i + \pi = \rho_i \geqslant 0$, this implies that $\pi \geqslant \pi_0$ and so no solution can exist.

If $i \in I_0$, then $\rho_i = \pi + a_i$ which proves the second part of (13). Since $\rho_i \geqslant 0$ we need that $\pi \geqslant -a_i$ for all $i \in I_0$. To find λ_i the only restrictions in this case are $\lambda_i\rho_i = 0$ and $\lambda_i \geqslant 0$ so if $\rho_i > 0$ then $\lambda_i = 0$ and if $\rho_i = 0$ then λ_i can be any non negative real number.

If $i \in I_+$ then from the first inequality in (12) $b_i\lambda_i - \rho_i = c_i - \pi$. If $c_i - \pi \geqslant 0$ then $b_i\lambda_i \geqslant \rho_i$ and the third inequality in (12) implies that $\rho_i = 0$. Then $\lambda_i = \frac{c_i - \pi}{b_i}$. Analogously, if $c_i - \pi \leqslant 0$ then $\rho_i = \pi - c_i$ and then $\lambda_i = 0$. □

Now to solve the original inequality system we have to find $\pi \geqslant \pi_0$ such that $\lambda(\pi)$ verify the convexity condition in (8), i.e.,

$$\sum_{i\in I}\lambda_i(\pi) = 1. \tag{15}$$

Let us define for $I_+ \neq \emptyset$ the function $f : \mathbb{R} \to \mathbb{R}$ as

$$f(\pi) = \sum_{i \in I_+} \lambda_i(\pi) - 1, \tag{16}$$

then solution of (15) is a zero of f.

Using (14) and (16) we see that f is continuous, non increasing, convex and piece-wise linear. Furthermore $\lim_{\pi \to \infty} f(\pi) = -1$ and f is strictly decreasing for $\pi < \max\{c_i : i \in I_+\}$. The function f is continuous, convex, non increasing and piece-wise linear because it is the sum of the functions $\lambda_i(\pi)$ with those properties. To compute the limit let us note that $\lambda_i(\pi) = 0$ for $\pi > c_i$, then, as there are a finite number of c_i, $f(\pi) = 0$ for π big enough. If $\pi < c_i$ for some $i \in I_+$ then $\lambda_i(\pi)$ is strictly decreasing for $\pi < c_i$ and so is the function f.

The value of f at π_0 determines the number of solutions of the system (11). In fact, if $f(\pi_0) > 0$ (which includes the case $I_0 = \emptyset$, because in this case $f(\pi_0) = +\infty$), as it is continuous and for π big enough f is negative it has a zero.

When $I_0 = \emptyset$ (strict convex case) the existence of a solution of the QKP as a zero of f is presented in [2,6,12]. This case can be considered included in the previous one if we assume $\pi_0 = -\infty$ and $f(\pi_0) = +\infty > 0$.

We resume these observations in the following Proposition.

Proposition 2. *If $I_0 = \emptyset$ or $f(\pi_0) > 0$ there is a unique solution of the system (11). The solution is given by $(\pi, \lambda_i(\pi), \rho_i(\pi))$ where $\pi > \pi_0$ is the only zero of f.*

The case not contemplated in the previous proposition is when $I_0 \neq \emptyset$ and $f(\pi_0) \leqslant 0$. In this case we have that there is at least one $i \in I_0$ such that $\pi_0 = -a_i$. Let us define $I_{00} = \{i \in I_0 : \pi_0 = -a_i\}$ and $I_{0+} = \{i \in I_0 : \pi_0 > -a_i\}$, then by the definition of π_0 we have that $I_{00} \neq \emptyset$.

As $f(\pi_0) \leqslant 0$ there is no hope to find a zero of f greater than π_0, hence the only possible solution is when $\pi = \pi_0$. In the following proposition we prove that also in this case there is always a solution.

Proposition 3. *If $I_+ = \emptyset$ or $f(\pi_0) \leqslant 0$ there is at least one solution of the system (11). The solution is given by $(\pi_0, \lambda_i(\pi_0), \rho_i(\pi_0))$ and the λ_i are uniquely determined only when $\#(I_{00}) = 1$.*

Proof. For $i \in I_+ \cup I_{0+}$ the λ_i are uniquely determined but we are free to choose any non negative value for λ_i when $i \in I_{00}$. As we want that Eq. (15) be satisfied, we need that

$$\sum_{i \in I_{00}} \lambda_i(\pi_0) + \sum_{i \in I_{0+}} \lambda_i(\pi_0) + \sum_{i \in I_+} \lambda_i(\pi_0) = 1, \tag{17}$$

but since $\lambda_i(\pi_0) = 0$ for $i \in I_{0+}$ Eq. (17) is the same as

$$\sum_{i \in I_{00}} \lambda_i(\pi_0) = 1 - \sum_{i \in I_+} \lambda_i(\pi_0), \tag{18}$$

and using the definition of f we have finally that the undefined λ_i verify

$$\sum_{i \in I_{00}} \lambda_i(\pi_0) = -f(\pi_0). \tag{19}$$

Then, as $f(\pi_0) \leqslant 0$, we can always find non negative values for λ_i such that (15) be verified. In the case that $\#(I_{00}) = 1$ there is only one λ_i undefined and its value must be $-f(\pi_0)$. $\square$

3.2. A Newton-like algorithm

Now to solve the problem we have to find a zero of f that we call π^*, then the sought value of λ is computed as it is indicated in the Proposition 1. To do so we propose an algorithm that first computes π_0 and $f(\pi_0)$. Then, if $f(\pi_0) > 0$, it makes a Newton search to find π^*.

It is worth to mention that a similar procedure is proposed in [10] but, once again, we extend it to allow the non strict case.

```
Init Compute $I_0$ and $I_+$
If $I_0 \neq \emptyset$ then $\pi_0 = \max\{-a_i : i \in I_0\}$
    If $I_+ \neq \emptyset$ then compute $f(\pi_0)$
        If $f(\pi_0) \leqslant 0$ then $\pi^* = \pi_0$
        else go to Find $\pi^*$
    else $\pi^* = \pi_0$
else $\pi_0 = \min(c_i - b_i)$, go to Find $\pi^*$
Find $\pi^*$
    Set $k = 0$, $\pi_k = \pi_0$
While $f(\pi_k) \neq 0$
    $\pi_{k+1} = \pi_k - f(\pi_k)/f'(\pi_k)$
    $k = k+1$
End while
$\pi^* = \pi_k$
```

For $\pi \neq c_i$ we have that

$$f'(\pi) = -\sum_{i \in I_+} \frac{1}{b_i} \chi(c_i - \pi), \tag{20}$$

where $\chi(x) = 1$ if $x \geqslant 0$ and 0 otherwise. The points $c_i = b_i\mu_i - a_i$ are the points of discontinuity of the derivative of $\lambda_i(\pi)$ so the lateral derivatives of f at those points are different ($f'(c_i+) \neq f'(c_i-)$). In order to accelerate the convergence and having in mind that f does not increase, we choose the right derivative if the function is positive and the left derivative otherwise.

Given the properties of f, it is piece-wise linear and non increasing, the convergence of the Newton phase is exact in a finite number of steps. This number of steps is bounded by # (I) (the number of routes).

3.3. Other algorithms

As we mentioned before other algorithms have been proposed to solve the QKP. All of them can be seen as a search for the zero of the function f.

In [6] they propose a binary search that consider the breakpoints of the (picewise linear) function f looking for the interval where function has a zero. Once the interval has been identified, the value of π results from an immediate calculation. This leads to an O($n \log n$)-algorithm. In [2] the binary search is improved using the a linear algorithm for the computation of the median of a set obtaining an O(n)-algorithm. In [12] they replace the calculation of the median by randomly chosen element. They observe that even if the complexity is higher the expected time is linear and the time constant is very small.

In [10] the authors analyze four algorithms and their parallel implementations. One of them is based on binary search and an other in a Newton search. Among them the most efficient results to be the Newton search. And the same qualities are observed in their parallel implementations.

3.4. Numerical experiences

We have numerically compared three algorithms, the one of Brucker, its randomized version and the one that we proposed. The three algorithms have been evaluated for different sizes $n = 100 \times 2^k$ where $k = 1, \ldots, 10$. For each value of n, we have averaged the time over 5000 randomly chosen n-dimensional problems. For each problem we have repeated the algorithm 5000 times to have a more robust measure of the processing time. In the Table we present the results in milliseconds obtained on a PC Pentium-IV 2.4 GHz with 528 Mb of RAM.

In the Table 1 and in the Fig. 1 we can see that the Newton algorithm is more efficient than the others. In the Fig. 1 we plot the time versus the size of the problem in a log–log scale, we can observe that the obtained complexity is almost linear and that time constant is better for the Newton algorithm.

This analysis has been made without considering the fact that the QKP will be solved as a subalgorithm of a master algorithm. When we consider the DSD or any other column generation algorithm that uses QKP at each step, the performance of the Newton method can be even better with respect to the others. These results have been observed that using the toolbox CiudadSim of Scilab.

Table 1
Numerical comparisons of the Newton algorithm and other two based on binary search

n	Brucker	R-Brucker	Newton
100	0.000094	0.000106	0.000031
200	0.000192	0.000192	0.000062
400	0.000311	0.00043	0.000148
800	0.000772	0.000828	0.000276
1600	0.002182	0.001794	0.000533
3200	0.004449	0.003278	0.001171
6400	0.007747	0.005614	0.003818
12800	0.026175	0.019991	0.006472
25600	0.062243	0.051367	0.018907
51200	0.15196	0.108777	0.04045
102400	0.34324	0.284508	0.117464

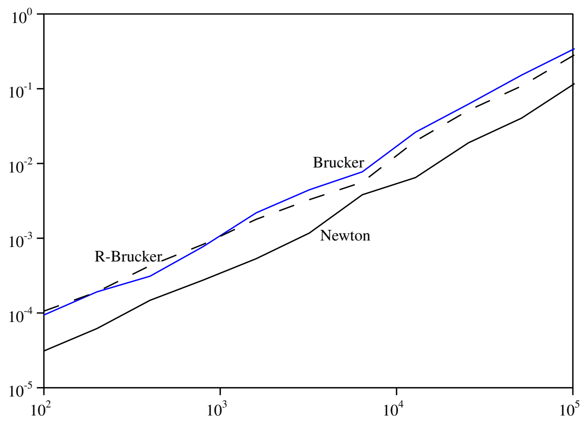

Fig. 1. Performance comparisons for the three algorithms.

Scilab is a free distributed software with vectorial syntax similar to Matlab. The toolbox CiudadSim [9] has been developed to solve traffic equilibrium problems with the algorithms found in the literature. It also has a graphical interface that facilitates the input of the data and the visualisation of the results.

Using this toolbox we have verified that the use of the Newton algorithm is very efficient because after a few master iterations the number of Newton iterations is limited to 2. In contrast the Brucker and R-Brucker algorithms keep making the corresponding number of iterations.

We have used two networks, the so called Sioux Falls network (24 nodes, 76 arcs and 528 OD pairs) and the Chicago network (546 nodes, 2176 arcs and 93135 OD pairs). The data for the Chicago network have been obtained from the web page *Transportation Network Test Problems* http://www.bgu.ac.il/bargera/tntp/. In the Sioux Falls network we obtained a gain of 26% in the computation time, in the Chicago network the gain was round 31%, the difference can be explained by the huge number of QKP that must be solved in the second example.

3.5. Vectorial implementation

This subsection is addressed to the users of softwares with vectorial syntax as Scilab or Matlab. They know the importance of coding vectorially if they don't want to interface lower level languages. In fact, as these languages are interpreted, the difference in performance between a vectorial operation and the same made with a for loop can be astronomic.

It is easy to see that the Newton algorithm can be done vectorially, i.e., for all the commodities at the same time. More precisely, suppose that we have to solve n_d QKP's, so we have to find the vector $\Pi = (\pi^{pq})$ such that $F(\Pi) := (f_{pq}(\pi^{pq})) = 0$. Now, the Newton directive is separable in each pq so we have

$$\Pi_{k+1} = \Pi_k - F(\pi_k) ø F'(\Pi_k), \tag{21}$$

where the symbol ø means the component-wise division (the operation "./" in Matlab/Scilab). The derivative of F can also be written in vectorial way as

$$F'(\Pi) = -\delta^t \chi(c - \delta\Pi), \tag{22}$$

where the coefficients of the matrix δ are $\delta_{i,pq} = 1$ if i corresponds to pq and $b_i > 0$ and 0 otherwise.

On the other hand, the algorithms based on binary search do not allow this vectorization, at least in a so simple way. The difficulty appears when we try to code the binary search.

4. Conclusion

We have presented an algorithm that allows to solve many quadratic knapsack problems in linear time with better performance than the ones based on binary searches and presented in [2,6,12].

The algorithm presented here is a better choice for the implementation of the Disaggregated Simplicial Decomposition algorithm for the traffic assignment problem because of its reoptimisation capabilities.

Acknowledgements

The author would like to thank the referees for their corrections and interesting remarks, and Dr. Jean-Pierre Quadrat for many useful discussions.

References

[1] D. Boyce, B. Ran, Modeling Dynamic Transportation Networks: An Intelligent Transportation System Oriented Approach, Springer-Verlag, 1996.

[2] P. Brucker, An O(n) algorithm for quadratic knapsack problems, Operations Research Letters 3 (3) (1984) 163–166.

[3] A. Chen, D.-H. Lee, R. Jayakrishnan, Computational study of state-of-the-art path-based traffic assignment algorithms, Mathematics and Computers in Simulation 59 (6) (2002) 509–518.

[4] G. Cohen, Convexité et optimisation, École Nationale des Ponts et Chaussées, 2000, Available from: <http://www-rocq.inria.fr/metalau/cohen/enseign-f.html>.

[5] O. Damberg, A. Migdalas, Distributed disaggregated simplicial decomposition—a parallel algorithm for traffic assignment, Lecture Notes in Economics and Mathematical Systems, vol. 450, Springer-Verlag, Berlin, 1997, pp. 346–373.

[6] R. Helgason, J. Kennington, H. Lall, A polynomially bounded algorithm for a singly constrained quadratic program, Mathematical Programming (18) (1980) 338–343.

[7] T. Larsson, M. Patriksson, Simplicial decomposition with disaggregated representation for the traffic assignment problem, Transportation Science 26 (1992) 4–17.

[8] Lotito, P.A., Mancinelli, E.M., Quadrat, J.-P., 2002. Traffic assignment and Gibbs–Maslov semirings. Technical Report RR-4809, INRIA, 2002. Available from: <http://www.inria.fr/rrrt/rr-4809.html>.

[9] Lotito, P.A., Mancinelli, E.M., Quadrat, J.-P., 2002. The traffic assignment toolboxes of scilab. Technical Report RT-0281, INRIA, 2002. Available from: <http://www.inria.fr/rrrt/rt-0281.html>.

[10] S.S. Nielsen, S.A. Zenios, Massively parallel algorithms for single constrained convex programs, ORSA Journal on Computing 4 (2) (1992) 166–181.

[11] S. Pallottino, M.G. Scutellà, Shortest path algorithms in transportation models: classical and innovative aspects, Technical Report TR-97-06, 14, 1997.

[12] P. Pardalos, N. Kovoor, An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds, Mathematical Programming 46 (1990) 321–328.

[13] M. Patriksson, The Traffic Assignment Problem—Models and Methods, VSP, Utrecht, 1994.

[14] R.T. Rockafellar, J.B. Wets, Variational Analysis, Springer-Verlag, New York, 1997.

[15] Y. Sheffi, Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods, Prentice-Hall, Englewood Cliffs, NJ, 1985.