Discussion Paper No. 727

COMMUTERS' PATHS WITH PENALTIES FOR EARLY OR LATE ARRIVAL TIME

by

André de Palma*
Pierre Hansen**
and
Martine Labbé***

April 1987

*Department of Civil Engineering, Northwestern University, Evanston, Illinois 60201, USA.

**RUTCOR, Rutgers University, New Brunswick, New Jersey 08903, USA.

***G.R.E.S.G., Universite Louis Pasteur (Strasbourg I), FRANCE and CORE Universite Catholique de Louvain, Louvain la Neuve, BELGIUM.

The authors thank Y. Pochet for his comments on a first draft of the paper. The research of the third author was supported by the Air Force Office of Scientific Research Grant N^{o} AFOSR0271 to Rutgers University

1. INTRODUCTION

Innumerable commuters choose and/or follow daily some path from their origin to their destinataion in a transportation network. It is therefore of interest, both for decisional and for descriptive purposes, to provide a method of computing best such paths. Improved paths could be found and the discrepancies between these paths and those actually choosen could be evaluated. Moreover, a tool would thus be provided for an empirical study of the distribution of departure times and for a simulation approach to the dynamic network equilibrium problem.

A simplistic approach would be to express this problem as a classical shortest path one. Then, a positive length representing a cost or a travel time is assigned to each arc of the network. A path from an origin to a destination with minimum total length can then be found by the well known algorithm of Dijkstra (1959) or some variant thereof (see Deo and Pang (1984) for a recent survey).

There exists however a large number of situations of practical interest for which some assumptions of the classical shortest path problem are not adequate.

First, using that model implies the assumption that the travel time associated to each arc is constant, i.e. independent of the time of the day. This assumption is clearly not satisfied in dense urban areas where travel times vary significantly between peak and off-peak hours. Consequently, in the model we introduce, at each arc is associated a constant cost and a travel time which is a function of the arrival time at the origin node of the arc.

Second, when considering the shortest path problem for commuters, it appears that road users try also to minimize their schedule delay (i.e. the difference between the desired and actual arrival times at destination). Note that schedule delay is also an important factor when describing the travel behaviour of individuals going to scheduled events at sport arenas, movie theaters and alike. Again, the assumptions of the classical shortest path problem do not allow to take such a factor into account.

The problem we consider here, consists in determining a path linking an origin to a destination for a given departure time from the origin and which minimizes the following objective function :

$$z = TCC + \alpha(TTT) + \beta(ESD) + \gamma(LSD)$$

where

- TCC is the total constant cost,
- TTT is the total travel time,
- ESD is the early schedule delay (zero for on time or late arrivals and positive otherwise),
- LSD is the late schedule delay (zero for on time or early arrivals and positive otherwise),
- α (≥ 0) is the cost per unit of travel time,
- β (≥ 0) is the cost per unit of waiting time in case of early arrival,
- γ (≥ 0) is the cost per unit of lost time in case of late arrival.

We call this problem the **Generalized Shortest Path Problem** (GSPP). It includes among others the constrained shortest path problem (see Handler and Zang (1980)) and the shortest path problem with time dependent travel times.

The paper is organized as follows. In the next section, we present a mathematical programming formulation of GSPP and show that it is NP-hard. Section 3 then provides some properties of GSPP and discusses some special cases of GSPP which are shown to be polynomial. In Section 4, a pseudo-polynomial algorithm for solving GSPP is presented and it is illustrated by a short example.

2. THE GENERAL MODEL

Let $N = (V,A)$ be an oriented network with vertex set V and arc set A. There are n vertices and m arcs. Vertex v_n is the destination and v_1 the departure vertex or origin. At each arc $(v_k,v_\ell) \in A$ are associated a constant cost $c_{k\ell} (\geq 0)$ for using the arc and a travel time $t_{k\ell}(\tau_k)$ (≥ 0) where τ_k denotes the arrival time at v_k, the origin of the arc. Let $\tau^* \pm \Delta$ be the desired arrival time interval at v_n.

We now provide a mathematical programming formulation of the generalized shortest path problem (GSPP) for a given departure time τ_1 at the origin v_1.

$$\text{Min } z = \sum_{j,k|(v_j,v_k)\in A} [c_{jk} + \alpha\, t_{jk}(\tau_j)]\, x_{jk}$$
$$+ \beta \max\{0\,;\, \tau^* - \Delta - \tau_1 - \sum_{j,k|(v_j,v_k)\in A} t_{jk}(\tau_j) x_{jk}\} \tag{1}$$
$$+ \gamma \max\{0\,;\, \tau_1 + \sum_{j,k|(v_j,v_k)\in A} t_{jk}(\tau_j)\, x_{jk} - \tau^* - \Delta\}$$

s.t.

$$\sum_{j|(v_1,v_j)\in A} x_{1j} = 1 \tag{2}$$

$$\sum_{j|(v_j,v_k)\in A} x_{jk} - \sum_{\ell|(v_k,v_\ell)\in A} x_{k\ell} = 0 \quad , \quad k = 2,\ldots,n-1 \tag{3}$$

$$\sum_{j|(v_j,v_n)\in A} x_{jn} = 1 \tag{4}$$

$$\sum_{j|(v_j,v_k)\in A} [\tau_j + t_{jk}(\tau_j)]\, x_{jk} = \tau_k \quad ; \quad k = 2,\ldots,n\,. \tag{5}$$

$$x_{jk} \in \{0,1\} \quad \forall\, j,k \,|\, (v_j,v_k) \in A\,. \tag{6}$$

The first term of the objective function is the cost associated with the path (the fixed cost plus the cost associated with the travel time). The second term is the penalty for early arrival and the third term the penalty for late arrival.

As in classical shortest path problems, $x_{jk} = 1$ if and only if the arc (v_j,v_k) belongs to the optimal path. Constraints (2) to (4) imply that the path links v_1 to v_n and for any intermediate vertex v_k of the path there are exactly one arc arriving in v_k and one arc leaving v_k. Finally, constraint (5) defines the arrival time τ_k at each vertex v_k. If v_k does not belong to the optimal path then all $x_{jk} = 0$ and thus $\tau_k = 0$. If v_k belongs to the optimal path then $\tau_k = \tau_j + t_{jk}(\tau_j)$ where v_j is the predecessor of v_k on the optimal path.

We make the following two assumptions :

<u>H1</u> : $\forall\, i,j$ such that $(v_i,v_j) \in A$ and $\forall\, \tau_j$:

$$\frac{dt_{ij}(\tau_i)}{d\tau_i} \geq -1\,.$$

<u>H2</u> : $\alpha \geq \beta$.

Assumption H1 means that it is impossible to arrive earlier at the destination by leaving the origin later, i.e. the arrival time τ_j at any vertex v_j is an increasing function of the departure time τ_i at the origin v_i. Indeed, if a path contains only one arc (v_i, v_j) then, using H1, we obtain that :

$$\frac{d\tau_j}{d\tau_i} = \frac{d(\tau_i + t_{ij}(\tau_i))}{d\tau_i} \geq 0 .$$

To show that this property holds for any path we use induction : consider a path linking v_i and v_j and containing p arcs. Let τ_i be the departure time at v_i and τ_j the arrival time at v_j using this path. Furthermore, let v_k be the extremity of the subpath containing p-1 arcs and τ_k the arrival time at v_k using that subpath. If $\frac{d\tau_k}{d\tau_i} \geq 0$ then, since $\tau_j = \tau_k + t_{kj}(\tau_k)$, we obtain :

$$\frac{d\tau_j}{d\tau_i} = \left(1 + \frac{dt_{kj}(\tau_k)}{d\tau_k}\right) \cdot \frac{d\tau_k}{d\tau_i} \geq 0 \quad , \text{ using H1 .}$$

Notice that this assumption corresponds to real world traffic behaviour (see Ben-Akiva and de Palma (1986)) and is thus not restrictive.

Assumption H2 states that the penalty associated with one unit of travel time is larger than the penalty associated with one unit of waiting time at the destination. Empirical studies show that this inequality holds (see e.g. Small (1982)).

We now prove :

THEOREM 1.

GSPP is NP-hard .

Proof.

Consider the constrained shortest path problem (CSPP) :

$$\underset{j,k|(v_j,v_k)\in A}{\text{Min}} \; c_{jk}\, x_{jk}$$

$$\sum_{j,k|(v_j,v_k)\in A} t_{jk}\, x_{jk} \leq b$$

$$\sum_{j|(v_1,v_j)\in A} x_{1j} = 1$$

$$\sum_{j|(v_j,v_k)\in A} x_{jk} - \sum_{\ell|(v_k,v_\ell)\in A} x_{k\ell} = 0 \quad , \quad k = 2,\ldots,n-1$$

$$\sum_{j|(v_j,v_n)\in A} x_{jn} = 1$$

$$x_{jk} \in \{0,1\} \ , \ \forall\ j,k \mid (v_j,v_k) \in A \ .$$

This problem has been proved to be NP-hard by Megiddo (see Handler and Zang (1980)). Furthermore, it is a subcase of GSPP. To see this, set, in GSPP, $\tau^* - \tau_1 = b$, $\Delta = 0$, $\alpha = \beta = 0$, γ arbitrarily large and assume that the travel times $t_{jk}(\tau_j)$ are independent of τ_j (i.e. they are constant). The resulting GSPP is equivalent to CSPP. Now, since GSPP contains a subcase which is NP-hard, it is itself NP-hard.

□

3. PROPERTIES AND SPECIAL CASES

We now present properties of GSPP. For the first property we need some additional definitions. Let $P(v_1,v_n)$ be a path linking v_1 to v_n . The values of the variables x_{jk} , $(v_j,v_k) \in A$, corresponding to that path are such that $x_{jk} = 1$ if and only if (v_j,v_k) is an arc of $P(v_1,v_n)$ and they satisfy (2) to (4). Furthermore, for a given departure time τ_1 at v_1 , the arrival time τ_k at any intermediate vertex v_k of $P(v_1,v_n)$ is given by (5) and the arrival time τ_n at v_n is :

$$\tau_n = \tau_1 + \sum_{j,k|(v_j,v_k)\in A} t_{jk}(\tau_j)\ x_{jk} \ .$$

For a desired arrival time interval $\tau^* \pm \Delta$, $P(v_1,v_n)$ is said to be **on time** if and only if

$$\tau^* - \Delta \le \tau_1 + \sum_{j,k|(v_j,v_k)\in A} t_{jk}(\tau_j)\ x_{jk} = \tau^* + \Delta \ .$$

Similarly, $P(v_1,v_n)$ is **early** (respectively **late**) if and only if

$$\tau_1 + \sum_{j,k|(v_j,v_k)\in A} t_{jk}(\tau_j)\ x_{jk} \le \tau^* - \Delta \quad \text{(respectively} \ge \tau^* + \Delta) \ .$$

Finally, we define three problems related to GSPP :

the on **time** problem (TP) :

$$\min z_t = \sum_{j,k|(v_j,v_k)\in A} (c_{jk} + \alpha t_{jk}(\tau_j))\, x_{jk} \tag{7}$$

s.t. (2) to (6) ,

the early problem (EP) :

$$\min z_e = \sum_{j,k|(v_j,v_k)\in A} (c_{jk} + (\alpha - \beta) t_{jk}(\tau_j))\, x_{jk} \tag{8}$$

s.t. (2) to (6) ,

and the late problem (LP) :

$$\min z_\ell = \sum_{j,k|(v_j,v_k)\in A} (c_{jk} + (\alpha + \gamma) t_{jk}(\tau_j))\, x_{jk} \tag{9}$$

s.t. (2) to (6) .

THEOREM 2.

If the optimal solution of TP (respectively EP or LP) is on time (respectively early or late) then it is an optimal solution of GSPP.

Proof.

Let $F_t^*(v_1,v_n)$, $F_e^*(v_1,v_n)$ and $F_\ell^*(v_1,v_n)$ be optimal solutions of TP , EP and LP respectively. Consider a path $P(v_1,v_n)$ with corresponding x_{jk} . We have :

$$\begin{aligned} z(P(v_1,v_n)) = & \sum_{j,k|(v_j,v_k)\in A} (c_{jk} + \alpha t_{jk}(\tau_j))\, x_{jk} + \beta \max \{0 , \\ & \tau^* - \Delta - \tau_1 - \sum_{j,k|(v_j,v_k)\in A} t_{jk}(\tau_j)\, x_{jk}\} + \gamma \max \{0 , \\ & \tau_1 + \sum_{j,k|(v_j,v_k)\in A} t_{jk}(\tau_j)\, x_{jk} - \tau^* - \Delta\} \\ \geq & \sum_{j,k|(v_j,v_k)\in A} (c_{jk} + \alpha t_{jk}(\tau_j))\, x_{jk} + \beta \,(\tau^* - \Delta - \tau_1 \\ & - \sum_{j,k|(v_j,v_k)\in A} t_{jk}(\tau_j)\, x_{jk}) \\ = & \ z_e(P(v_1,v_n)) + \beta(\tau^* - \Delta - \tau_1) \end{aligned}$$

$$\geq z_e(P_e^*(v_1,v_n)) + \beta(\tau^* - \Delta - \tau_1) \text{ , by definition of } P_e^*(v_1,v_n) \text{ ,}$$

$$= z(P_e^*(v_1,v_n)) \text{ if } P_e^*(v_1,v_n) \text{ is early.}$$

The proofs that $z(P(v_1,v_n)) \geq z(P_t^*(v_1,v_n))$ and $z(P(v_1,v_n)) \geq z(P_\ell^*(v_1,v_n))$ when $P_t^*(v_1,v_n)$ is on time and $P_\ell^*(v_1,v_n)$ late are similar and are left to the reader.

□

COROLLARY 2.1.

If $c_{jk} = 0$ for every $(v_j,v_k) \in A$, then GSPP is equivalent to :

$$\text{Min} \sum_{j,k|(v_j,v_k)\in A} t_{jk}(\tau_j)\, x_{jk} \tag{10}$$

s.t. (2) to (6) .

Proof

A direct consequence of the fact that if all the costs are equal to zero, then TP, EP and LP are equivalent to (10) up to a multiplicative constant in the objective function.

□

This last corollary and assumption H1 imply that GSPP with zero costs can easily be solved using Dijkstra's algorithm.

COROLLARY 2.2

If the travel times are independent of the time of the day, i.e. if for every $(v_j,v_k) \in A$: $t_{jk}(\tau_j) = t_{jk}$ and if there exists a non negative constant a such that for every $(v_j,v_k) \in A$: $c_{jk} = at_{jk}$ then GSPP reduces to the classical shortest path problem :

$$\text{Min} \sum_{j,k|(v_j,v_k)\in A} t_{jk}\, x_{jk}$$

s.t. (2), (3), (4) and (6) .

Proof

A direct consequence of the fact that also in this case TP, EP and LP reduce to the classical shortest path problem.

□

Notice that even if the travel times are independent of the time of the day, the optimal solution of GSPP is not necessarily an optimal solution of TP, EP or LP when the costs are not proportional to the travel times. As an example, consider the network of Figure 1 with $\alpha = 2$, $\beta = \gamma = 1$, $\tau^* = 10$, $\Delta = 2$ and $\tau_1 = 0$. There are three paths from v_1 to v_5 :
$P_1 = (v_1, v_2) \cup (v_2, v_5)$; $P_2 = (v_1, v_3) \cup (v_3, v_5)$ and $P_3 = (v_1, v_4) \cup (v_4, v_5)$.
Let c_i and t_i denote the cost and the travel time respectively of path P_i , $i = 1,2,3$. We have : $c_1 = 10$, $c_2 = 4$, $c_3 = 6.5$, $t_1 = 5$, $t_2 = 10$ and $t_3 = 8$. Path P_1 is thus early and the paths P_2 and P_3 are on time. The values of the objective functions of EP, TP, LP and GSPP for P_1 , P_2 and P_3 are given in Table 1 . It appears that the optimal solution of EP is P_2 , the optimal solution of TP and LP is P_1 and the optimal solution of GSPP is P_3

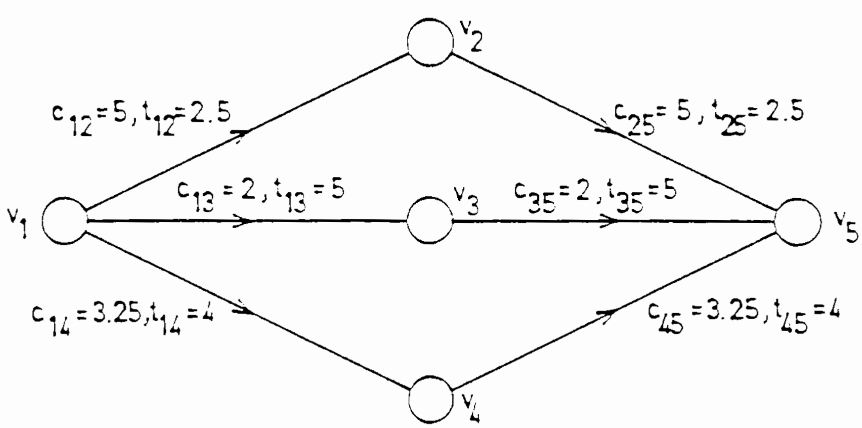

FIGURE 1 : A NETWORK WHERE THE OPTIMAL SOLUTION OF GSPP IS NOT OPTIMAL FOR EP , TP OR LP.

	P_1	P_2	P_3	Optimal solution
EP	15	14	14.5	P_2 (on time)
TP	20	24	22.5	P_1 (early)
LP	25	34	30.5	P_1 (early)
GSSP	23	24	22.5	P_3 (on time)

Table 1 : Values of the objective functions of EP, TP, LP and GSPP for the network of Figure 1.

Finally, with corollaries 2.1 and 2.2 we have identified special cases of GSPP which can be solved using Dijkstra's algorithm. This does unfortunately not hold for the general case even where there is no schedule delay. As an example, consider the network of Figure 2 with $\tau_1 = 0$. Assume that $\alpha = 1$ and $\beta = \gamma = 0$. There are two paths linking v_1 to v_5 : $P_1 = (v_1,v_2) \cup (v_2,v_4) \cup (v_4,v_5)$ and $P_2 = (v_1,v_3) \cup (v_3,v_4) \cup (v_4,v_5)$. The travel time along (v_4,v_5) is the only one that depends on the time of the day so that $t_{45}(1) = 2.75$ and $t_{45}(3) = 1$. The subpath $(v_1,v_2) \cup (v_2,v_4)$ is optimal to reach v_4 since

$$c_{12} + c_{24} + t_{12} + t_{24} = 3.75 < c_{13} + c_{34} + t_{13} + t_{34} = 5 .$$

However path P_1 is not optimal as

$$c_{12} + c_{24} + c_{45} + t_{12} + t_{24} + t_{45}(t_{12} + t_{24}) = 6.5$$

$$> c_{13} + c_{34} + c_{45} + t_{13} + t_{34} + t_{45}(t_{13} + t_{34}) = 6 .$$

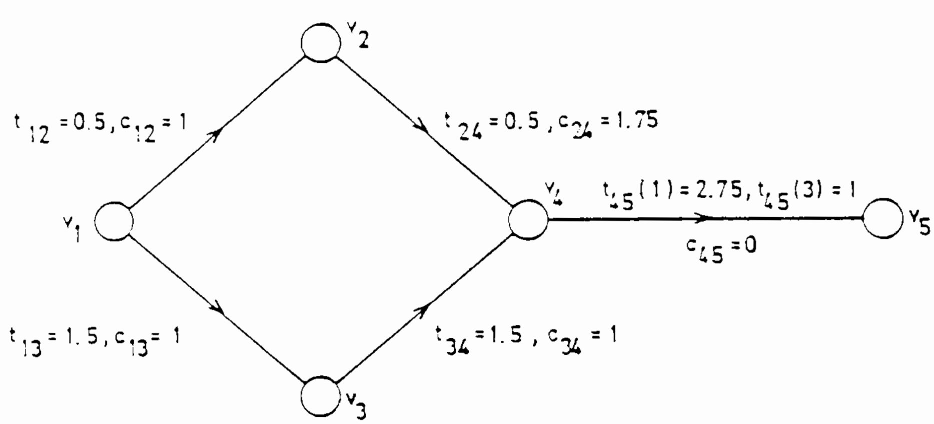

FIGURE 2 : A NETWORK FOR WHICH DIJKSTRA'S ALGORITHM CANNOT BE USED

Now, we present some properties concerning the efficiency of the optimal solution of GSPP. Given a pair of objective functions, a path $P(v_1,v_n)$ is efficient if and only if no other path $P'(v_1,v_n)$ has a better value for one criterion and a not worse value for the other one. If $P(v_1,v_n)$ is not efficient then it is **dominated** by some other path $P'(v_1,v_n)$.

THEOREM 3.

There exists an optimal solution $P^*(v_1,v_n)$ of GSPP such that any subpath $P^*(v_p,v_q)$ of $P^*(v_1,v_n)$ is efficient for the two criteria :

$$\text{Min} \sum c_{jk} \quad \text{and} \quad \text{Min} \sum t_{jk}(\tau_j)$$

where τ_p is the arrival time at v_p using the subpath $P^*(v_1,v_p)$.

Proof.

Let $P^*(v_1,v_n)$ be an optimal path to GSPP. Assume that there exists a path $\bar{P}(v_p,v_q)$ which dominates the subpath $P^*(v_p,v_q)$ of $P^*(v_1,v_n)$. We shall prove that the path $\bar{P}(v_1,v_n)$ defined as

$$\bar{P}(v_1,v_n) = P^*(v_1,v_p) \cup \bar{P}(v_p,v_q) \cup P^*(v_q,v_n)$$

is such that $z(\bar{P}(v_1,v_n)) \le z(P^*(v_1,v_n))$. This will imply that $\bar{P}(v_1,v_n)$ is also an optimal solution to GSPP. Then by iterating on all the dominated subpath of $P^*(v_1,v_n)$ we will obtain an optimal path to GSPP which does not contain any dominated subpath.

Since $\bar{P}(v_p,v_q)$ dominates $P^*(v_p,v_q)$ we have :

$$\sum_{(v_j,v_k)\in\bar{P}(v_p,v_q)} c_{jk} \le \sum_{(v_j,v_k)\in P^*(v_p,v_q)} c_{jk} \quad \text{, and} \tag{10}$$

$$\sum_{(v_j,v_k)\in\bar{P}(v_p,v_q)} t_{jk}(\tau_j) \le \sum_{(v_j,v_k)\in P^*(v_p,v_q)} t_{jk}(\tau_j) \quad , \tag{11}$$

with at least one strict inequality (τ_p is the arrival time at v_p using path $P^*(v_1,v_p)$). Given the definition of $\bar{P}(v_1,v_n)$, we obtain using (10) that

$$\sum_{(v_j,v_k)\in\bar{P}(v_1,v_n)} c_{jk} \le \sum_{(v_j,v_k)\in P^*(v_1,v_n)} c_{jk} \quad , \tag{12}$$

$$\sum_{(v_j,v_k)\in\bar{P}(v_1,v_n)} t_{jk}(\tau_j) \le \sum_{(v_j,v_k)\in P^*(v_1,v_n)} t_{jk}(\tau_j) \ . \tag{13}$$

Hence,

$$\begin{aligned}
z(\bar{P}(v_1,v_n)) = ;\ & \sum_{(v_j,v_k)\in\bar{P}(v_1,v_n)} c_{jk} + \alpha \sum_{(v_j,v_k)\in\bar{P}(v_1,v_n)} t_{jk}(\tau_j) \\
& + \beta \max \{0 \ ; \ \tau^* - \Delta - \tau_1 - \sum_{(v_j,v_k)\in\bar{P}(v_1,v_n)} t_{jk}(\tau_j)\} \\
& + \gamma \max \{0 \ ; \ \tau_1 + \sum_{(v_j,v_k)\in\bar{P}(v_1,v_n)} t_{jk}(\tau_j) - \tau^* - \Delta\} \\
= & \sum_{(v_j,v_k)\in\bar{P}(v_1,v_n)} c_{jk} + \max \{\alpha \sum_{(v_j,v_k)\in\bar{P}(v_1,v_n)} t_{jk}(\tau_j) \ ; \\
& (\alpha - \beta) \sum_{(v_j,v_k)\in\bar{P}(v_1,v_n)} t_{jk}(\tau_j) + \beta\ (\tau^* - \Delta - \tau_1)\} \\
& + \gamma \max \{0 \ ; \ \tau_1 + \sum_{(v_j,v_k)\in\bar{P}(v_1,v_n)} t_{jk}(\tau_j) - \tau^* - \Delta\} \\
\le & \sum_{(v_j,v_k)\in P^*(v_1,v_n)} c_{jk} + \max \{\alpha \sum_{(v_j,v_k)\in P^*(v_1,v_n)} t_{jk}(\tau_j) \ ; \\
& (\alpha - \beta) \sum_{(v_j,v_k)\in P^*(v_1,v_n)} t_{jk}(\tau_j) + \beta\ (\tau^* - \Delta - \tau_1)\} \\
& + \gamma \max \{0 \ ; \ \tau_1 + \sum_{(v_j,v_k)\in P^*(v_1,v_n)} t_{jk}(\tau_j) - \tau^* - \Delta\} \ ; \\
& \qquad \text{using (12), (13) and since } \alpha \le \beta \ ; \\
= & \ z(P^*(v_1,v_n)) \ ;
\end{aligned}$$

which completes the proof.

□

Theorem 3 provides the grounds for an algorithm for solving GSPP. It will be presented in the next section.

When the travel times are independent of the time of the day, we have a stronger property.

THEOREM 4.

If for each arc $(v_j, v_k) \in A$, $t_{jk}(\tau_j) = t_{jk}$ then there exists an optimal solution $P^*(v_1, v_n)$ to GSPP such that any subpath $P^*(v_p, v_q)$ of $P^*(v_1, v_n)$ is efficient for the two criteria :

$$\text{Min} \sum (c_{jk} + (\alpha - \beta)t_{jk}) \quad \text{and} \quad \text{Min} \sum t_{jk} .$$

Proof.

We proceed as for Theorem 3. Let $\bar{P}(v_p, v_q)$ be a path which dominates a subpath $P^*(v_p, v_q)$. Using the same notations, we shall prove that $z(\bar{P}(v_1, v_n)) \leq z(P^*(v_1, v_n))$.

Since $\bar{P}(v_p, v_q)$ dominates $P^*(v_p, v_q)$ and given the definition of $\bar{P}(v_1, v_n)$ we have that :

$$\sum_{(v_j, v_k) \in \bar{P}(v_1, v_n)} t_{jk} \leq \sum_{(v_j, v_k) \in P^*(v_1, v_n)} t_{jk} \quad , \text{ and} \tag{14}$$

$$\sum_{(v_j, v_k) \in \bar{P}(v_1, v_n)} (c_{jk} + (\alpha - \beta)t_{jk} \leq \sum_{(v_j, v_k) \in P^*(v_1, v_n)} (c_{jk} + (\alpha - \beta)t_{jk}). \tag{15}$$

Furthermore, combining (14) and (15) we obtain :

$$\sum_{(v_j, v_k) \in \bar{P}(v_1, v_n)} (c_{jk} + \alpha t_{jk}) \leq \sum_{(v_j, v_k) \in P^*(v_1, v_n)} (c_{jk} + \alpha t_{jk}) . \tag{16}$$

Hence,

$$\begin{aligned} z(\bar{P}(v_1, v_n)) &= \sum_{(v_j, v_k) \in \bar{P}(v_1, v_n)} c_{jk} + \alpha \sum_{(v_j, v_k) \in \bar{P}(v_1, v_n)} t_{jk} \\ &\quad + \beta \max \{0 \; ; \; \tau^* - \Delta - \tau_1 - \sum_{(v_j, v_k) \in \bar{P}(v_1, v_n)} t_{jk}\} \\ &\quad + \gamma \max \{0 \; ; \; \tau_1 + \sum_{(v_j, v_k) \in \bar{P}(v_1, v_n)} t_{jk} - \tau^* - \Delta\} \\ &= \max \{ \sum_{(v_j, v_k) \in \bar{P}(v_1, v_n)} (c_{jk} + \alpha t_{jk}) \; ; \\ &\quad \sum_{(v_j, v_k) \in \bar{P}(v_1, v_n)} (c_{jk} + (\alpha - \beta)t_{jk}) + \beta(\tau^* - \Delta - \tau_1)\} \end{aligned}$$

$$+ \gamma \max \{0 \; ; \; \tau_1 + \sum_{(v_j,v_k) \in \bar{P}(v_1,v_n)} t_{jk} - \tau^* - \Delta\}$$

$$\leq' \max \{ \sum_{(v_j,v_k) \in P^*(v_1,v_n)} (c_{jk} + \alpha t_{jk}) \; ;$$

$$\sum_{(v_j,v_k) \in P^*(v_1,v_n)} (c_{jk} + (\alpha - \beta) t_{jk}) + \beta(\tau^* - \Delta - \tau_1)\}$$

$$+ \gamma \max \{0 \; ; \; \tau_1 + \sum_{(v_j,v_k) \in P^*(v_1,v_n)} t_{jk} + \tau^* - \Delta\} ,$$

using (14), (15) and (16) ;

$$= z(P^*(v_1,v_n)) .$$

□

When some travel times depend on the time of the day, Theorem 4 does unfortunately not hold. As an example, consider again the network of Figure 2 with $\tau_1 = 0$, $\tau^* = 10$, $\Delta = 0$, $\alpha = 2$ and $\beta = \gamma = 1$. The two paths linking v_1 to v_5 are early. Furthermore the subpath $(v_1,v_2) \cup (v_2,v_4)$ dominates the subpath $(v_1,v_3) \cup (v_3,v_4)$ (in the sense of Theorem 4) as

$$c_{12} + c_{24} + (\alpha - \beta) [t_{12} + t_{24}] = 3.75$$

$$< c_{13} + c_{34} + (\alpha - \beta) [t_{13} + t_{34}] = 5 \; ; \text{ and}$$

$$t_{12} + t_{24} = 1 < t_{13} + t_{34} = 3 .$$

However it is the path $P_2 = (v_1,v_3) \cup (v_3,v_4) \cup (v_4,v_5)$ which is optimal. Indeed,

$$z(P_2) = c_{13} + c_{34} + c_{45} + (\alpha - \beta) (t_{13} + t_{34} + t_{45}(3)) + \beta(\tau^*) = 16, \text{ and}$$

$$z(P_1) = c_{12} + c_{24} + c_{45} + (\alpha - \beta) (t_{12} + t_{24} + t_{45}(1)) + \beta(\tau^*) = 16.5 .$$

4. THE ALGORITHM

In this section we present an algorithm for solving GSPP for a given departure time τ_1 at the origin vertex v_1. The data of the problem are the following : the network $N(V,A)$ and for each arc $(v_j,v_k) \in A$, a non negative cost c_{jk} and a non negative piecewise linear travel time function $t_{jk}(\tau_j)$. For simplicity, each such function is represented by a linked list of its break-points :

$$L_{jk} = \langle (a_1,t_{jk}(a_1));\ldots;(a_i,t_{jk}(a_i));\ldots;(a_{p_{jk}},t_{jk}(a_{p_{jk}})) \rangle .$$

Then for τ_j such that $a_i \le \tau_j \le a_{i+1}$:

$$t_{jk}(\tau_j) = t_{jk}(a_i) + \frac{t_{jk}(a_{i+1}) - t_{jk}(a_i)}{a_{i+1} - a_i} (\tau_j - a_i) .$$

The algorithm is of A*-type (see e.g. Pearl (1984)). In a first step, we compute for each vertex v_k a lower bound on the cost and on the travel time corresponding to the best path to go from v_k to v_n. The lower bound $\underline{c}_k$ on the cost is obtained by applying Dijkstra's algorithm backwards. Since the travel time along an arc depends on the departure time at the origin vertex of that arc, we proceed as follows to obtain a lower bound on the travel time. For T given arrival times τ^i, $i = 1,\ldots,T$ at v_n (e.g. from 5 to 5 minutes) we apply Dijkstra's algorithm backwards to obtain the latest departure time τ_k^i at each vertex v_k which allows to arrive at v_n at τ^i. Thus for these so-obtained departure times τ_k^i, a lower bound on the travel time is given by :

$$\underline{t}_k(\tau_k^i) = \tau^i - \tau_k^i .$$

For a departure time τ_k which has not been obtained by this last procedure, i.e. $\tau_k^i < \tau_k < \tau_k^{i+1}$, a lower bound on the travel time is obtained by

$$\underline{t}_k(\tau_k) = \tau^i - \tau_k .$$

This expression is justified by the fact that one cannot arrive earlier by leaving later (assumption H1). Again, practically, we will keep for each vertex v_k a linked list containing the so-obtained departure times and the corresponding arrival times at v_n :

$$\underline{L}_k = \langle (\tau_k^1,\tau_1);\ldots;(\tau_k^i,\tau^i);\ldots;(\tau_k^T,\tau^T) \rangle .$$

Once we have all the lower bounds, we proceed to the main part of the

- i : the number of the path;
- $j(i)$: the index k of the extremity vertex of the path;
- $p(i)$: the number of the path having for extremity the vertex which preceeds $j(i)$ in path i ;
- $\lambda(i)$: the cost of the path;
- $\mu(i)$: the travel time of the path given a departure time τ_1 at v_1 ;
- $b(i)$: a lower bound on the objective function corresponding to that path;

  $\lambda(i) + \underline{c}_k + \alpha[\mu(i) + \underline{t}_k(\tau_1 + \mu(i))]$
  $+ \beta \max \{0 , \tau^* - \Delta - \tau_1 - \mu(i) - \underline{t}_k(\tau_1 + \mu(i))\}$
  $+ \gamma \max \{0 , \tau_1 + \mu(i) + \underline{t}_k(\tau_1 + \mu(i)) - \tau^* - \Delta\}$;
- $e(i)$: an estimation of the exact value of the objective function for that path ;

  $\lambda(i) + \rho\underline{c}_k + \alpha[\mu(i) + \sigma\underline{t}_k(\tau_1 + \mu(i))]$
  $+ \beta \max \{0 ; \tau^* - \Delta - \tau_1 - \mu(i) - \sigma\underline{t}_k(\tau_1 + \mu(i))\}$
  $+ \gamma \max \{0 ; \tau_1 + \mu(i) + \sigma\underline{t}_k(\tau_1 + \mu(i)) - \tau^* - \Delta\}$
  where ρ and σ are two constants to be determined empirically.

During the whole procedure, the 6-tuples of the paths that have not been dominated yet are stored in a list P .

At each iteration, we select a path, that will be denoted by i^* , in the set R of unselected path such that :

$$e(i^*) = \min\{e(i) , i \in R\}$$

(ties are broken in favor of the path with minimum $b(i)$). We remove i^* from R. Then for each vertex v_k such that $(v_j,v_k) \in A$ where $j = j(i^*)$, we obtain a new path, say i . For each such path, we compute the values of $j(i)$, $p(i)$, $\lambda(i)$, $\mu(i)$, $b(i)$ and $e(i)$. If $b(i) > z_{opt}$, where z_{opt} is the value of the objective function for the best path linking v_1 to v_n found so far, we eliminate path i . If $b(i) \leq z_{opt}$ and $j(i) = n$, we insert this path in the list P and update z_{opt} . If $b(i) \leq z_{opt}$ and $j(i) < n$, we compare this path i with all the other paths i' from P such that $j(i') = j(i)$. If i is dominated, we eliminate it. Otherwise, we insert it in the list P of undominated paths and in the set R of unselected paths by choosing for i the smallest number not yet used. Furthermore, we eliminate from P and R all the paths that i dominates. Finally, the algorithm stops when R is empty.

ALGORITHM : Optimal path for GSPP.

Step 1 : Data

Read the data, i.e. τ_1 , τ^* , Δ , α , β , γ , σ , ρ , $N = (V,A)$, and for each $(v_j,v_k) \in A$: c_{jk} and $L_{jk} = \langle (a_1,t_{jk}(a_1));\ldots;(a_i,t_{jk}(a_i));\ldots;(a_{p_{jk}},t_{jk}(a_{p_{jk}})) \rangle$.

Step 2 : Lower bounds on the costs

(a) Set $\underline{c}_n = 0$ and $\underline{c}_k = +\infty$ for $k = 1,\ldots,n-1$. Set $S = \{1,2,\ldots,n\}$.

(b) If $S = \phi$, stop. Otherwise, determine k^* such that $\underline{c}_{k^*} = \min\{\underline{c}_k , k \in S\}$ and eliminate k^* from S .

(c) For each v_j such that $(v_j,v_{k^*}) \in A$, if $\underline{c}_j > \underline{c}_{k^*} + c_{jk^*}$, set $\underline{c}_j = \underline{c}_{k^*} + c_{jk^*}$. Return to (b).

Step 3 : Lower bounds on the travel times

Choose a set of T values τ^i , $i = 1,\ldots,T$ of arrival times at v_n .
For $i = 1,\ldots,T$, apply the following procedure.

(a) Set $\tau_n^i = \tau^i$ and $\tau_k^i = -\infty$ for $k = 1,\ldots,n-1$. Set $S = \{1,2,\ldots,n\}$.

(b) If $S = \phi$, then for each vertex v_k , $k = 1,\ldots,n$, add the couple (τ_k^i,τ^i) in the list $\underline{L}_k$. Then, proceed to the next τ^i , unless $i = T$, and return to (a). Otherwise, determine k^* such that $\tau_{k^*}^i = \max\{\tau_k^i , k \in S\}$ and eliminate k^* from S .

(c) For each v_j such that $(v_j,v_{k^*}) \in A$, determine in L_{jk^*} the element a_q such that

$$a_q + t_{jk^*}(a_q) \leq \tau_{k^*}^i < a_{q+1} + t_{jk^*}(a_{q+1}) .$$

If

$$a_q + \frac{a_{q+1} - a_q}{a_{q+1} + t_{jk^*}(a_{q+1}) - a_q - t_{jk^*}(a_{q+1})} (\tau_{k^*}^i - a_q - t_{jk^*}(a_q)) > \tau_j^i$$

then set

$$\tau_j^1 = a_q + \frac{a_{q+1} - a_q}{a_{q+1} + t_{jk*}(a_{q+1}) - a_q - t_{jk*}(a_q)} (\tau_{k*}^1 - a_q - t_{jk*}(a_q)) .$$

Return to (b) .

Step 4 : Initialization

Set $i = 1$, $j(1) = 1$, $p(1) = 0$, $\lambda(1) = \mu(1) = 0$ and $z_{opt} = +\infty$.
Determine (τ_1^q, τ^q) in $\underline{L}_1$ such that $\tau_1^q \leq \tau_1 < \tau_1^{q+1}$. Then compute :

$$\underline{t}_1(\tau_1) = \tau^q - \tau_1 ,$$

$$b(1) = \underline{c}_1 + \alpha\underline{t}_1(\tau^1) + \beta \max \{0 ; \tau^* - \Delta - \tau_1 - \underline{t}_1(\tau_1)\}$$

$$+ \gamma \max \{0 ; \tau_1 + \underline{t}_1(\tau_1) - \tau^* - \Delta\} , \text{ and}$$

$$e(1) = \rho\underline{c}_1 + \alpha\sigma\underline{t}_1(\tau_1) + \beta \max \{0 ; \tau^* - \Delta - \tau_1 - \sigma\underline{t}_1(\tau_1)\}$$

$$+ \gamma \max \{0 ; \tau_1 + \sigma\underline{t}_1(\tau_1) - \tau^* - \Delta\} .$$

Set $R = \{1\}$ and insert the 6-tuple $(j(1) , p(1) , \lambda(1) , \mu(1) , b(1) , e(1))$ in P .

Step 5 : Selection of the path with smallest estimation and test for ending

If $R = \phi$, go to step 7 (all the optimal paths sought for have been found). Otherwise, compute $\underline{e} = \min \{e(i) , i \in R\}$ and select i^* such that $b(i^*) = \min \{b(i) , i \in R$ and $e(i) = \underline{e}\}$. Delete i^* from R . If $b(i^*) > z_{opt}$, erase the corresponding 6-tuple from P and return to the beginning of this step. Otherwise, proceed to step 6.

Step 6 : Computation of new labels

For each v_k such that $(v_j, v_k) \in A$ and $j = j(i^*)$, compute

$$\mu(i) = \mu(i^*) + t_{jk}(\tau_1 + \mu(i^*)) ,$$

determine (τ_k^q, τ^q) in $\underline{L}_k$ such that

$$\tau_k^q \leq \tau_1 + \mu(i) < \tau_k^{q+1}$$

and set

$$\underline{t}_k(\tau_1 + \mu(i)) = \tau^q - \tau_1 - \mu(i) .$$

Then, consider the 6-tuple composed of :

$$j(i) = k ,$$

$$p(i) = i^* ,$$

$$\lambda(i) = \lambda(i^*) + c_{jk} ,$$

$$\mu(i) = \mu(i^*) + t_{jk}(\tau_1 + \mu(i^*)) ,$$

$$b(i) = \lambda(i) + \underline{c}_k + \alpha[\mu(i) + \underline{t}_k(\tau_1 + \mu(i))] + \beta \max \{0 ; \tau^* - \Delta - \tau_1 - \mu(i) - \underline{t}_k(\tau_1 + \mu(i))\} + \gamma \max \{0 , \tau_1 + \mu(i) + \underline{t}_k(\tau_1 + \mu(i)) - \tau^* - \Delta\} ,$$ and

$$e(i) = \lambda(i) + \rho\underline{c}_k + \alpha[\mu(i) + \sigma\underline{t}_k(\tau_1 + \mu(i))] + \beta \max \{0 ; \tau^* - \Delta - \tau_1 - \mu(i) - \sigma\underline{t}_k(\tau_1 + \mu(i))\} + \gamma \max \{0 ; \tau_1 + \mu(i) + \sigma\underline{t}_k(\tau_1 + \mu(i)) - \tau^* - \Delta\} .$$

If $b(i) > z_{opt}$ then erase the 6-tuple.
If $j(i) = n$ and $b(i) < z_{opt}$, set $z_{opt} = b(i)$ and eliminate from P all the paths i' such that $j(i') = n$ and $b(i') > z_{opt}$. Add the new 6-tuple in list P ; choose for i the first value not yet used.
If $j(i) = n$ and $b(i) = z_{opt}$, add the new 6-tuple in list P, choose for i the first value not yet used.
If $j(i) < n$, compare the 6-tuple with the undominated 6-tuples i' of list P such that $j(i') = j(i)$. If some of them are dominated by the new 6-tuple (i.e. if $\lambda(i') \geq \lambda(i)$ and $\mu(i') \geq \mu(i)$ with at least one strict inequality), erase them from P and R (if still in R).

least one strict inequality), then erase the new 6-tuple; otherwise add it to list P ; choose for i the first value not yet used; set $R = R \cup \{i\}$. Return to step 5.

Step 7 : Output of the optimal paths for GSPP

Each path i^* in P such that $j(i^*) = n$ is optimal. Use the $j(i)$'s and $p(i)$'s to recompose backwards the list of vertices of that path.

□

THEOREM 5.

The previous algorithm solves GSPP in $O(nmc \log n^2c)$ time .

Proof.

To prove the algorithm's correctness, we first note that the first four steps are preliminary ones. Step 1 involves reading of the data. Step 2 consists in a backwards application of the classical Dijkstra's algorithm. Step 3 uses T times a backwards variant of Dijkstra's algorithm in which travel times, along arcs, corresponding to the arrival times at the extremity vertices are computed. That this algorithm gives shortest time paths is a direct consequence of assumption H1 and of the rules of Dijkstra's algorithm. Step 4 consists in a few straightforward initializations of parameters for the main algorithm which comprises steps 5 to 7. In the latter, efficient paths are systematically constructed by labeling vertices which follow immediately the last vertex of a selected path. Subpaths are eliminated by using the lower bound $b(i)$ described above or because they are dominated.

Assuming, with very little loss in generality, the costs c_{jk} to be integer, the number of efficient paths from v_1 to all other vertices v_j is bounded by n^2c , where $c = \max \{c_{jk}, (v_j,v_k) \in A\}$ (indeed, the cost of any elementary path from v_1 to a vertex v_j is not greater than $(n-1) \cdot c$, all efficient paths are elementary and there are n vertices). The algorithm therefore yields an optimal solution in a finite time.

Regarding complexity, we note that step 2 is in $O(m \log n)$ (cf Gondran and Minoux (1979)). Similarly, step 3 is in $O(Tp\, m \log n)$ where p denotes the maximum number of break-points in a list $\underline{L}_{jk}$. Step 4 is in $O(T)$. Using a heap to store the unselected paths yields an $O(n^2c \log n^2c)$ implementation of step 5 (the selection of a path in the heap is in $O(n^2c)$ and this must be performed at most n^2c times). As the total number of arcs is m, step 6 takes O (nmc max {log n^2c, p, T}) time. Indeed, each time a new 6-tuple is computed, O (max {p,T}) operations are needed and then O (log (nc)) operations to check dominance and O (log (n^2c)) to update the heap containing the estimations $e(i)$. Finally, step 7 is in $O(n^2c)$ as there are at most $(n-1)$. c efficient paths from v_1 to v_n. Assuming p and T constant the total complexity is thus in O (nmc log n^2c).

□

EXAMPLE.

Consider the network of Figure 3. The costs c_{jk} are indicated along the arcs. The travel time functions are depicted in Figure 4. For ease of exposition, the arrival time functions $\tau_j + t_{jk}(\tau_j)$ at the extremity vertices of the arcs are also depicted in Figure 4.

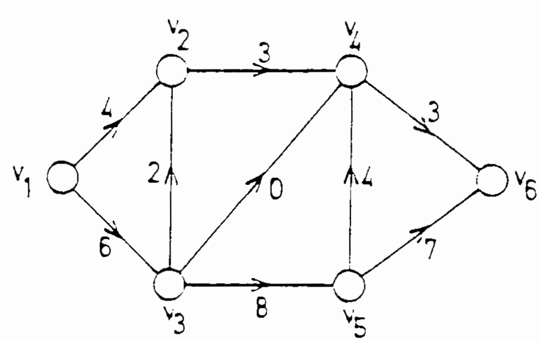

FIGURE 3 : THE NETWORK OF THE EXAMPLE.

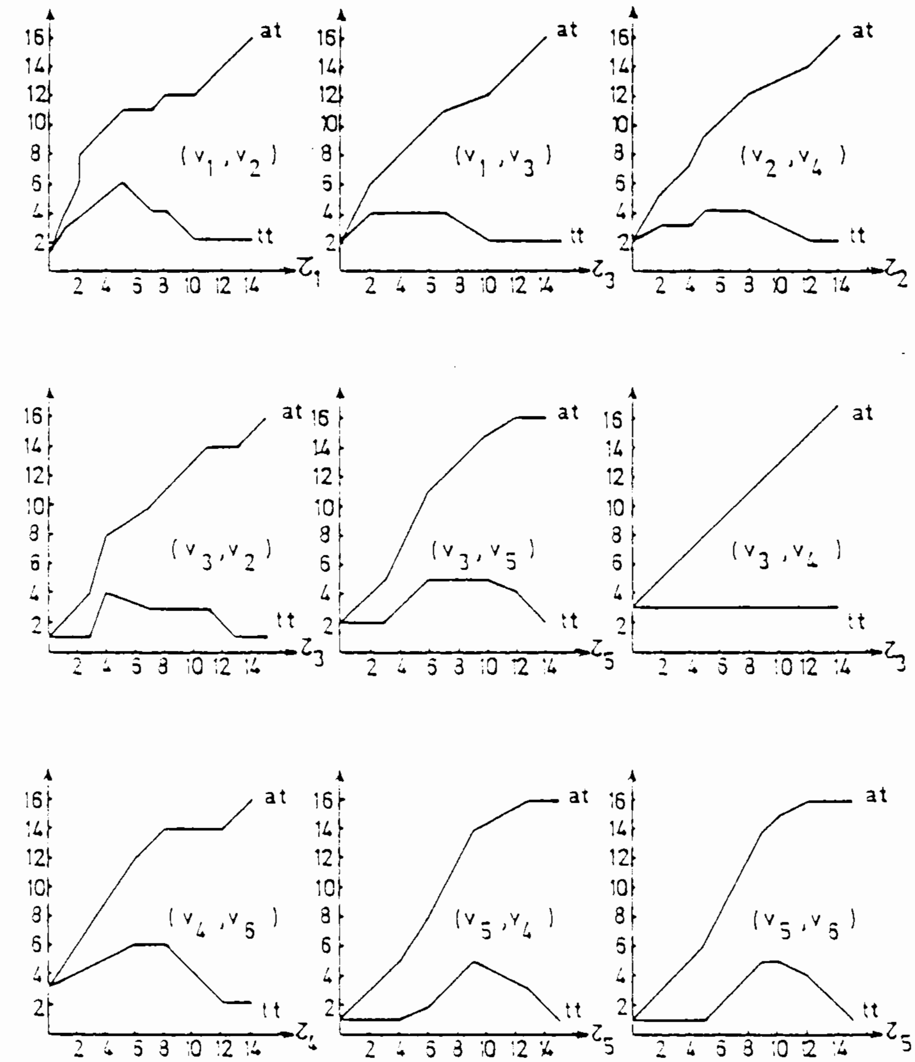

FIGURE 4 : TRAVEL TIMES (tt) AND ARRIVAL TIMES (at) FOR ALL ARCS OF THE NETWORK OF THE EXAMPLE.

More precisely, the lists L_{jk} of break-points of the travel time functions $t_{jk}(\tau_j)$ for $(v_j, v_k) \in A$ are the following :

L_{12} = < (0,1);(1,3);(2,6);(5,6);(7,4);(8,4);(10,2);(14,2) > ,

L_{13} = < (0,2);(2,4);(7,4);(10,2);(15,2) > ,

L_{24} = < (0,2);(2,3);(4,3);(5,4);(8,4);(12,2);(14,2) > ,

L_{32} = < (0,1);(3,1);(4,4);(7,3);(11,3);(13,1);(15,1) > ,

L_{35} = < (0,2);(3,2);(6,5);(10,5);(12,4);(14,2) > ,

$L_{34} = \langle (0,3);(14,3) \rangle$,

$L_{46} = \langle (0,3);(6,6);(8,6);(12,2);(14,2) \rangle$,

$L_{54} = \langle (0,1);(4,1);(6,2);(9,5);(13,3);(15,1) \rangle$, and

$L_{56} = \langle (0,1);(5,1);(9,5);(10,5);(12,4);(15,1) \rangle$.

Let $\tau_1 = 0$, $\tau^* = 7$, $\Delta = 0$, $\alpha = 2$ and $\beta = \gamma = 1$. At step 2, we obtain the following lower bounds on the costs : $\underline{c}_1 = 9$, $\underline{c}_2 = 6$, $\underline{c}_3 = 3$, $\underline{c}_4 = 3$, $\underline{c}_5 = 7$ and $\underline{c}_6 = 0$.

We have applied step 3 for $\tau^i = 5,6,\ldots,15$. For each τ^i , the so-obtained departure times τ_k^i , $k = 1,\ldots,6$, are listed in Table 2.

τ^i \ τ_k^i	τ_1^i	τ_2^i	τ_3^i	τ_4^i	τ_5^i	τ_6^i
5	0	–	2	1.333	4	5
6	0.500	0	3	2	5	6
7	0.625	0.444	3.250	2.667	5.500	7
8	0.750	0.889	3.500	3.333	6	8
9	0.875	1.333	3.750	4	6.500	9
10	1	1.778	4	4.667	7	10
11	1.125	2.333	4.250	5.333	7.500	11
12	1.250	3	4.500	6	8	12
13	1.380	4	4.750	7	8.500	13
14	5	8	9	12	9	14
15	6	10	10	13	10	15

Table 2 : Departure times τ_k^i , $k = 1,\ldots,6$ for $\tau^i = 5,6,\ldots,15$.

The details of the application of steps 5 and 6 are given in Table 3. The list P is given for each iteration. The 6-tuples without star contitute the set R. We have chosen $\rho = 1.1$ and $\sigma = 1.7$. For clarity, the lower bounds $\underline{c}_k$ and $\underline{t}_k(\tau_1 + \mu(i))$ for $k = j(i)$ are also indicated in Table 3.

We finally obtain $z_{opt} = 27.75$ and there is one optimal path : $(v_1, v_2) \cup (v_2, v_4) \cup (v_5, v_6)$.

i	$j(i)$	$p(i)$	$l(i)$	$m(i)$	$\underline{c}_k$	$\underline{t}_k(\tau_k)$	$b(i)$	$e(i)$	
1	1	0	0	0	9	5	21	28.4*	
2	2	1	4	1	6	7	27	42.3	
3	3	1	6	2	3	3	21	23.6*	
1	1	0	0	0	9	5	21	28.4*	
2	2	1	4	1	6	7	27	42.3	
3	3	1	6	2	3	3	21	23.6*	
	2	3	8	3	6	9	43	62.5	
4	4	3	6	5	3	5	32	42.8	
5	5	3	14	4	7	1	33	34.4*	
1	1	0	0	0	9	5	21	28.4*	
2	2	1	4	1	6	7	27	42.3	
3	3	1	6	2	3	3	21	23.6*	
4	4	3	6	5	3	5	32	42.8	
5	5	3	14	4	7	1	33	34.4*	
	4	5	18	5	3	5	44	54.8	
6	6	5	21	5	0	0	33	33*	= z_{opt}
1	1	0	0	0	9	5	21	28.4	
2	2	1	4	1	6	7	27	42.3*	
3	3	1	6	2	3	3	21	23.6*	
4	4	3	6	5	3	5	32	42.8	
5	5	3	14	4	7	1	33	34.4*	
6	6	5	21	5	0	0	33	33*	= z_{opt}
1	1	0	0	0	9	5	21	28.4*	
2	2	1	4	1	6	7	27	42.3*	
3	3	1	6	2	3	3	21	23.6*	
4	4	3	6	5	3	5	32	42.8	
5	5	3	14	4	7	1	33	34.4*	
6	6	5	21	5	0	0	33	33*	= z_{opt}
7	4	2	7	3.5	3	4.5	27	36.75*	
1	1	0	0	0	9	5	21	28.4*	
2	2	1	4	1	6	7	27	42.3*	
3	3	1	6	2	3	3	21	23.6*	
4	4	3	6	5	3	5	32	42.8	
5	5	3	14	4	7	1	33	34.4*	
6	6	5	21	5	0	0	33	33*	
7	4	2	7	3.5	3	4.5	27	36.75*	
8	6	7	10	8.25	0	0	27.75	27.75*	= z_{opt}
1	1	0	0	0	9	5	21	28.4*	
2	2	1	4	1	6	7	27	42.3*	
3	3	1	6	2	3	3	21	23.6*	
5	5	3	14	4	7	1	33	34.4*	
7	4	2	7	3.5	3	4.5	27	36.75*	
8	6	7	10	8.25	0	0	27.75	27.75*	= z_{opt}

Table 3. Details of Steps 5 and 6.

REFERENCES

N. Deo and C.-Y. Pang, 1984, Shortest-Path Algorithms : Taxonomy and Annotation, **Networks**, 14, 275-323.

M. Ben-Akiva and A. de Palma, 1986, Some Circumstances in which Vehicles Will Reach Their Destinations Earlier by Starting Later : Revisited, **Transportation Science** 20, 52-55.

E.W. Dijkstra, 1959, A Note on Two Problems in Connection with Graphs, **Numerische Mathematik** 1, 269-271.

M. Gondran and M. Minoux, 1979, **Graphes et Algorithmes**, Eyrolles, France.

G.Y. Handler and Y. Zang, 1980, A Dual Algorithm for the Constrained Shortest Path Problem, **Networks** 10, 293-310.

S. Pearl, **Heuristics**, 1984, Addison-Wesley, Mass.

K. Small, 1982, The Scheduling of Consumer Activities : Work Trips, **American Economic Review** 72, 467-479.