Università di Pisa

Dipartimento di Informatica

Technical Report : TR-97-06

Shortest Path Algorithms in Transportation models: classical and innovative aspects

Stefano Pallottino, Maria Grazia Scutella'

April 14, 1997

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL: +39-50-887111. FAX: +39-50-887226

Shortest Path Algorithms in Transportation models: classical and innovative aspects[1]

Stefano Pallottino
University of Perugia, Institute of Electronics
via G. Duranti 1/A1, 06131 Perugia, Italy, pallo@di.unipi.it

Maria Grazia Scutellà
University of Pisa, Department of Computer Science
Corso Italia 40, 56125 Pisa, Italy, scut@di.unipi.it

Abstract

Shortest Path Problems are among the most studied network flow optimization problems, with interesting applications in various fields. One such field is transportation, where shortest path problems of different kinds need to be solved.

Due to the nature of the application, transportation scientists need very flexible and efficient shortest path procedures, both from the running time point of view, and also for the memory requirements. Since no "best" algorithm currently exists for every kind of transportation problem, research in this field has recently moved to the design and implementation of "ad hoc" shortest path procedures, which are able to capture the peculiarities of the problems under consideration.

The aim of this work is to present in a unifying framework both the main algorithmic approaches that have been proposed in the past years for solving the shortest path problems arising most frequently in the transportation field, and also some important implementation techniques which allow one to derive efficient procedures from the general algorithmic schema, in line with trends in current research.

More precisely, in the first part of the paper, after presenting the problem, we briefly review those classical primal and dual algorithms which seem to be the most interesting in transportation, either as a result of theoretical considerations or because of their efficiency, and also in view of their practical use in transportation models. Promising reoptimization approaches are then discussed.

The second part is devoted to dynamic shortest path problems, that arise very frequently in the transportation field. We analyze the main features of the problems and present, under suitable conditions on travel time and cost functions, a general "chronological" algorithmic paradigm, called *Chrono-SPT*. We show the potentiality of this paradigm, and, by properly performing some implementation choices, we study several, interesting special cases. Finally, we investigate some promising research avenues related to various extensions. These include "time-windows" problems, "turn penalties and prohibitions", and "multicriteria" problems, which often can be solved by extending the ideas discussed for the dynamic case. Finally, "shortest hyperpaths" problems are briefly introduced, due to their practical relevance in many transportation applications.

[1] Research grants by C.N.R. Progetto Finalizzato Trasporti II, 74.9600089 and by C.N.R. G.N.I.M.

1. Introduction

Shortest Path Problems are among the most studied network flow optimization problems. Since the end of the 1950s, more than two thousand scientific works have been published in the international literature, most of them in journals and conference proceedings concerning general combinatorial optimization on graphs, but also in numerous specialized journals.

One of the most interesting application fields is transportation.

In many transportation problems, in fact, shortest path problems of different kinds need to be solved. These include both classical problems, e.g. to determine shortest paths (under various measures, such as length, cost and so on) between some given origin/destination pairs in a certain area, and also non standard versions, e.g. to compute shortest paths either under additional constraints or on particular structured graphs.

Due to the nature of the application, transportation scientists need very flexible and efficient shortest path procedures, both from the running time point of view, and also for the memory requirements. Since no "best" algorithm exists for every kind of transportation problem, i.e. no algorithm exists which shows the same practical behavior independently of the structure of the graph, of its size and of the cost measure used for evaluating the paths, research in this field has recently moved to the design and the implementation of "ad hoc" shortest path procedures, which are able to capture the peculiarities of the problems under consideration. In particular, a great emphasis has been devoted to the choice and the implementation of efficient data structures (see, for instance, [Tarjan, 1983]).

The aim of this work is to present in a unifying framework both the main algorithmic approaches that have been proposed for solving the shortest path problems arising most frequently in the transportation field, and also some important implementation techniques which allow one to derive efficient procedures from the general algorithmic schema, in line with current trends in research. Clearly, since it is impossible to cover all the shortest path problems which can occur in this field, we have selected some of the most common situations, and shown how to make the implementation choices in those cases.

More precisely, in the first part of the paper, after presenting the problem, we briefly review those classical primal and dual algorithms which seem to be the most interesting in the transportation framework, either as a result of theoretical considerations or because of their efficiency, and also in view of their practical use in transportation models. Then, promising reoptimization approaches are discussed. This is a very important subject in large scale transportation models, where often a sequence of shortest path problems needs to be solved, where the $(k+1)^{th}$ problem differs only slightly from the k^{th} one.

The second part of the work is devoted to dynamic shortest path problems, which arise very frequently in the transportation field. We analyze the main features of the problems and present, under suitable conditions on travel time and cost functions, a "chronological" algorithmic paradigm, called *Chrono-SPT*. We show the power of this paradigm, and, by making some suitable implementation choices, we study several, interesting special cases. Finally, we investigate some promising research avenues related to various extensions, including "time-windows", "turn penalties and prohibitions" and "multicriteria" problems, which can be solved by extending ideas discussed for the dynamic case. Finally, "shortest hyperpaths" problems are also briefly discussed, due to their practical relevance in many transportation applications.

2. Shortest Path approaches in Transportation Models

Shortest Path routines are never absent from any computer code used in transportation analysis and planning. This explains why, together with the increasing role of large scale mathematical models, interest in efficient shortest path algorithms has been growing, and an enormous number of algorithms have been proposed (for surveys about Shortest Path algorithms see, among others, [Johnson, 1973a], [Denardo&Fox, 1979], [Shier&Witzgall, 1981], [Deo&Pang, 1984], [Gallo&Pallottino, 1986], [Bertsekas, 1991b], [Ahuja, Magnanti &Orlin, 1993], [Cherkassky, Goldberg&Radzik, 1996]; for surveys about Shortest Path algorithms in transportation models, see for instance [Steenbrink, 1974], [Van Vliet, 1978], [Gallo & Pallottino, 1984]).

In this section, after presenting the problem, we shall concisely describe those Shortest Path methods which seem to be the most interesting in the transportation framework, either due to theoretical considerations or because of their efficiency, and also in view of their practical use in transportation models. In the algorithm presentation, we shall describe primal and dual approaches separately. The "primal" algorithms have been the object of deep analysis, aimed at discovering similarities and relationships within this algorithm family (Gallo&Pallottino, 1986, 1988). The main result is that each primal algorithm can be seen as a particular instantiation of a unique, general, algorithmic schema, which is an adaptation of the primal network simplex approach. A similar effort to classify dual approaches for Shortest Paths, and to discover relationships among them, can be found in [Pallottino&Scutellà, 1997]. Finally, promising reoptimization approaches will be discussed.

2.1. The Shortest Path Tree Problem

Let $G = (N, A)$ be a simple directed graph, where N is the set of the nodes, of cardinality n, and A is the set of the arcs, of cardinality m. Let $c: A \rightarrow R$ be a function which assigns a *cost* c_{ij} to each $(i,j) \in A$. Given a *root* $r \in N$, the *Shortest Path Tree* problem (SPT) consists in finding a directed tree T such that the (only) path from r to i in T is one of the shortest paths from r to i in G, for each $i \in N$ which is connected to r, i.e. a directed path from r to i exists. To simplify the presentation, hereafter we shall assume that each $i \in N$ is connected to r. T thus spans onto G.

As known from the literature, where wide ranging surveys on the subject can be found (see for instance [Lawler, 1976], [Gallo&Pallottino, 1986, 1988] and [Ahuja, Magnanti&Orlin, 1993]), a finite solution exists for SPT if and only if there is no directed cycle of negative cost in G. We shall therefore assume that this property is satisfied. Using a linear programming notation, the problem and its dual can be formulated as follows. Here, $FS(i) = \{(i,j) \in A\}$ and $BS(i) = \{(j,i) \in A\}$ denote the *forward star* and the *backward star* of i, respectively, while $b_r = -n + 1$ and $b_i = 1$, $\forall\ i \neq r$.

$$(SPT) \quad \min \sum_{(i,j)\in A} c_{ij}\, x_{ij}$$

$$\sum_{(j,i)\in BS(i)} x_{ji} - \sum_{(i,j)\in FS(i)} x_{ij} = b_i\,, \qquad \forall i \in N,$$

$$x_{ij} \geq 0, \qquad \forall\ (i,j) \in A,$$

$$(DSPT) \quad \max\ (-n+1)\pi_r + \sum_{j \neq r} \pi_j$$

$$\pi_j - \pi_i \leq c_{ij}, \qquad \forall\ (i,j) \in A,$$

where π_i is the *price* or *potential* of i, i.e. the dual variable associated with i.

2.2. Primal algorithms for SPT

In order to solve (*SPT*), most of the algorithms which have been proposed in the literature follow a primal approach. In fact, at each step, they maintain a spanning arborescence T of root r, and check whether the *cost labels* C, which represent either the costs of the paths outgoing from r along the tree, or an upper bound for these costs, are a dual feasible solution, i.e. whether *Bellman conditions* (Bellman, 1958) hold:

$$C_j \leq C_i + c_{ij}, \qquad \forall\ (i,j) \in A. \tag{1}$$

A set of *candidate nodes* Q is maintained by this algorithm family, whose incident arcs might violate Bellman's conditions. At each step, a candidate node i is selected from Q, and condition (1) is checked for each arc of the forward star of i; if one of these arcs, say (i,j), violates these conditions, then the cost label C_j is properly updated, and j is inserted into Q. Although these algorithms can be interpreted as procedures for solving the *non-linear Bellman equation system*:

$$\begin{cases} \pi_r = 0, \\ \pi_j = \min\{\pi_i + c_{ij} : (i,j) \in BS(j)\}, \quad \forall\ j \in N \setminus \{r\}, \end{cases} \tag{2}$$

the dual feasibility generally holds only at termination.

Depending on the selection rule of the candidate nodes in Q, different algorithms have been proposed. The ones which always select the candidate node with the minimum distance label are known as *label-setting* or as *shortest-first search* algorithms. The first algorithm using this selection rule was proposed by Dijkstra (1959). The simplest version of this algorithm implements the set of the candidate nodes as an unordered linked-list, and it runs in $O(n^2)$ time in the case of nonnegative costs, and in $O(n2^n)$ time in the general case (Johnson, 1973b). Other versions of Dijkstra's algorithm are the so-called *S-heap* algorithm (Johnson, 1972), which implements Q as a binary heap, and the *S-bucket* algorithm, where Q is implemented by means of a buckets data structure (Dial, 1969; Van Vliet, 1978; Denardo&Fox, 1979). In the case of nonnegative costs, *S-heap* runs in $O(m\log n)$ time, while *S-bucket* works in $O(m+nc_{max})$ time when costs are integer valued (here, c_{max} denotes the maximum arc cost). Further variants of *S-bucket* and of *S-heap* have been proposed, which work in $O(m+n(\log c_{max})^{1/2})$ time and in $O(m+n\log n)$ time, respectively. The first algorithm was proposed by Ahuja, Mehlhorn, Orlin and Tarjan (1990), while the last implementation, based on Fibonacci heaps, is actually the fastest strongly polynomial algorithm in the case of nonnegative arc costs, at least from a time complexity point of view (Fredman&Tarjan, 1987).

The SPT algorithms which select a candidate node by means of different strategies are known as *label-correcting* or as *list-search* algorithms. The queue is a very natural choice when implementing the set Q. The algorithm based on the corresponding selection rule, i.e. the FIFO rule, is named *L-queue*, and it is credited to Bellman (1958), Ford (1956) and Moore (1958). It is very simple to implement, and its time complexity is $O(mn)$ whatever the sign of the arc costs. A list which combines the properties of the data structures queue and stack is the *deque*, in which addition and deletion are possible at either end of the list. A deque is used in the well-known D'Esopo-Pape algorithm, also known as *L-deque* (Pape, 1974). The deque can be viewed as a stack Q' and a queue Q'' connected in series. The first time a node is inserted into the data structure, it is inserted at the tail of Q''. When the same node, after being removed from

the set, becomes a candidate again, it is added to Q' at the head. The nodes are removed from the head of Q', if it is not empty; otherwise, the first element of Q'' is removed.

Although its worst-case complexity is $O(n2^n)$ (Kershenbaum, 1981), *L-deque* proved to be very efficient in practice mainly on sparse almost planar graphs, which is the case in most transportation applications. A similar performance in practical cases shows the algorithm *L-2queue*, which differs from *L-deque* since the nodes are inserted into Q' at its tail instead of at its head, and which runs in polynomial time (Pallottino, 1984).

Both *L-deque* and *L-2queue* partition the set of the candidate nodes into two subsets which are processed differently. A similar idea is used in *L-threshold* (Glover, Glover&Klingman, 1984; Glover, Klingman&Phillips, 1985; Glover, Klingman, Phillips&Schneider, 1985), where the set of the candidate nodes Q is partitioned into two subsets, Q' and Q'', according to a suitable threshold value, say s. Subset Q' contains only nodes with label $\leq s$, and selections are always performed at the head of Q'. When Q' is empty, the threshold s is properly updated and all the nodes in Q'' that have a label $\leq s$ are moved into Q'. The reason is to increase the probability of selecting the minimum label node, by maintaining the operational simplicity of the list-search strategies. Different versions of *L-threshold* have been proposed in the literature, some of them running in polynomial time, which differ above all in the choice of the threshold, in the policy for updating its value, in the implementation of Q' and Q'', and in the way nodes are moved from Q'' to Q'.

Another algorithm which is based on the same philosophy of *L-threshold*, i.e. which selects the candidate nodes by following a label-correcting approach, but tries to scan nodes with a small label as early as possible, was proposed by Bertsekas (1993). In fact, if the label of the candidate node to be inserted into Q is less than the one of the node at the head of Q, then the candidate node is inserted at the head; otherwise, it is inserted at the end of the data structure. The method can be combined with the threshold approach, and it has shown a good practical behavior on randomly generated graphs.

Finally, let us mention another label-correcting algorithm, i.e. the recent *topological ordering* algorithm of Goldberg and Radzik (1993). This algorithm tries to propagate as much as possible the label updating due to the nodes in Q by means of a topological visit of a proper subgraph of G, and it runs in O(mn) time.

Almost all the primal algorithms have been extensively tested. For more details about the practical efficiency of the main algorithmic approaches on different kinds of graphs, see [Johnson, 1973a], [Shier&Witzgall, 1981], [Gallo&Pallottino, 1988], [Hung&Divoky, 1988], [Mondou, 1989], [Mondou, Crainic&Nguyen, 1991], [Cherkassky, Goldberg&Radzik, 1996].

In the case of transportation models, networks generally arise which have nonnegative arc costs and structured, quasi-planar, sparse graphs. In this framework, as reported in [Gallo&Pallottino, 1984], at the current state-of-the-art *L-deque* seems to be the best choice, except for particular applications requiring specialized algorithms. *L-2queue* and *L-threshold* show a similar behavior. On the other hand, shortest-first search algorithms do not seem to be efficient in the transportation field, due to the handling of the set Q which is too expensive.

2.3. Dual algorithms for SPT

Few of the SPT algorithms that have appeared in the literature maintain dual feasibility at each step. The ancestor of this algorithm family is the classical *dual simplex algorithm*, which maintains dual feasible basic solutions. Another well-known dual algorithm is Dijkstra's, which is generally presented as a primal-like method, but can be interpreted both as a dual and a

primal-dual approach (Papadimitriou&Steiglitz, 1982). A further pioneering work in the field of dual algorithms for SPT is Bazaraa and Langley's (1974), which looks for a dual feasible solution in the case of negative arc costs (otherwise, $\pi = 0$ is a trivial dual feasible solution), while Handler and Zang (1980) propose a dual algorithm to solve a shortest path problem with the additional constraint that feasible paths must have a "weight" less than a given value λ.

More recently, pure dual ascent algorithms have been described to solve the problem under consideration. The first is an adaptation of the *Relaxation method* to SPT (Bertsekas& Tsitsiklis, 1989). This algorithm considers the Dual Lagrangean problem, and tries to maximize the dual objective function by updating the price vector π in such a way as to always maintain the complementarity slackness conditions. More precisely, only one price is changed at a time as follows:

$$\pi_j := \min\{\pi_i + c_{ij}: (i,j) \in BS(j)\}, \qquad \forall\, j \in N \setminus \{r\}.$$

If the initial price vector π is dual feasible for the Linear Dual problem (*DSPT*), i.e. it satisfies $\pi_j - \pi_i \leq c_{ij}$, $\forall\ (i,j) \in A$, then it can be proved that the algorithm converges to an optimal price vector, provided that all the directed cycles of G have a positive cost; otherwise, the algorithm may return a wrong solution. In any case, the method can be interpreted like the Gauss-Seidel technique when applied to solve the non-linear Bellman's equation system (2).

Another dual ascent algorithm was introduced by Bertsekas (1991a) by specializing the *Auction approach* for the Assignment problem to Shortest Paths. Given a destination node t, a shortest path from r to t is looked for under the assumptions that each directed cycle of G has a positive cost, and that the forward star of each node apart from t is not empty. This shortest path is searched for using the following "primal-dual" approach: at each step, a *candidate path* P outgoing from r and a feasible dual vector π are maintained, which satisfy the complementarity slackness conditions, i.e.

$$\pi_j \leq \pi_i + c_{ij},\ \forall\ (i,j) \in A \qquad \text{(dual feasibility);}$$
$$(i,j) \in P \Rightarrow \pi_j = \pi_i + c_{ij} \qquad \text{(c.s. conditions).}$$

If P reaches the destination t, i.e. a primal feasible solution has been found, P is one of the shortest paths from r to t. Otherwise, *extension* and *contraction* operations are performed at the end node of P. As proved by Bertsekas, the Auction algorithm terminates, and it runs in pseudopolynomial time in the case of integer arc costs. Under mild conditions, a weak polynomial version can be obtained using cost scaling. The algorithm can be immediately extended to solve SPT. In [Pallottino&Scutellà, 1991] and [Bertsekas, Pallottino&Scutellà, 1995], strongly polynomial versions of the Auction algorithm are proposed, which return a shortest path tree without the requirement that the cycle costs must be positive, and without assumptions on the topology of the graph, thus generalizing the Auction method for shortest paths.

Recently, a new family of dual ascent algorithms, called the *Hanging family*, has been proposed, which is strictly related to the Relaxation method, and which seems to be particularly suitable for parallel implementations and for shortest path tree reoptimization (Pallottino&Scutellà, 1997). This family maintains a partial shortest path tree rooted at r, $T = (N_T, A_T)$, and a feasible dual vector π, which satisfy the complementarity slackness conditions related to (*SPT*) and its dual:

$$(i) \quad \pi_j \leq \pi_i + c_{ij},\ \forall\ (i,j) \in A \qquad \text{(dual feasibility);}$$
$$(ii) \quad (i,j) \in A_T \Rightarrow \pi_j = \pi_i + c_{ij} \qquad \text{(c.s. conditions).}$$

At the beginning, $N_T = \{r\}$ and $A_T = \emptyset$, and π is any vector satisfying condition (i). In order to try to augment T, each $j \notin T$, called an *out of tree node*, looks for an entering arc (i,j) such that $\pi_j = \pi_i + c_{ij}$ and $i \in N_T$. If an arc (i,j) of this type is found, then, j is "hanged" to T through (i,j), since a shortest path from r to j has been found. Otherwise, local or global "Jacobi-like" dual updating operations are performed at the out of tree nodes, in order to try to extend T.

A dual ascent algorithm, called *local_dual*, is based on the following *local_reprice* operation at each out of tree node:

$$\text{for each } j \notin N_T, \qquad \textit{local_reprice}(j): \qquad \pi_j := \gamma_j,$$

where $\gamma_j = \min\{\pi_i + c_{ij}: (i,j) \in BS(j)\}$. *Local_dual* returns an optimal solution under the assumption that all cycles have a positive cost and it finds an optimal solution in pseudopolynomial time in the case of integer arc costs.

To overcome this, i.e. to guarantee that at least one node can be hanged to T at each step, the authors use some global information as follows. Firstly, partition the set of the arcs coming into each $j \notin N_T$ into the set of arcs outgoing from tree nodes, called *border arcs*, and the other ones, called *external arcs*, also denoted by $BS_B(j)$ and $BS_E(j)$, respectively. Then, compute:

$$\beta_j = \min\{\pi_i + c_{ij}: (i,j) \in BS_B(j)\};$$
$$\eta_j = \min\{\pi_i + c_{ij}: (i,j) \in BS_E(j)\};$$
$$\gamma_j = \min\{\beta_j, \eta_j\}.$$

If $\gamma_j = \beta_j$ for some out of tree node j, then j can be hanged to T after *local_reprice*; otherwise, i.e. $\beta_j > \gamma_j$ for each $j \notin N_T$, then no hanging can be performed. To avoid the second situation, the hierarchy of *global dual updating operations* illustrated in Figure 1 is introduced. All these global operations maintain the feasibility of the price vector and ensure the entry of at least one new node into T. Furthermore, the dominance relationships shown in Figure 1 can be established among the dual updating operations, where an arrow going from the node associated with the operation D_2 toward the node corresponding to D_1 means that D_1 dominates D_2[(1)].

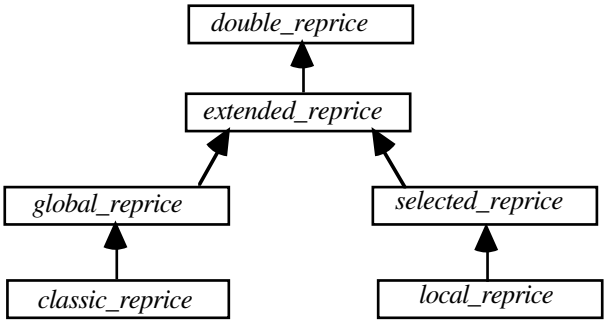


Fig. 1: the hierarchy of dual updating operations

All the global dual updating operations are based on the global information δ:

$$\delta = \min\{\beta_j - \pi_j: j \notin N_T\}.$$

δ is a threshold gap for the prices of the out of tree nodes, in the sense that it is the minimum increment to be added to the prices of the out of tree nodes to guarantee that at least

---

(1) Given two dual updating operations, say D_1 and D_2, and the dual vector π, let $\pi(D_1,\pi)$ and $\pi(D_2,\pi)$ be the dual solutions computed by D_1 and D_2, respectively, when applied to π. Then D_1 *dominates* D_2 if $\pi(D_1,\pi) \geq \pi(D_2,\pi)$.

one of them, i.e. $w = \text{argmin}\{\beta_j - \pi_j : j \notin N_T\}$, can be hanged to T. The first global dual updating operation, named *classic_reprice*, increases the price of all the out of tree nodes by δ:

for each $j \notin N_T$, *classic_reprice*(j): $\pi_j := \pi_j + \delta$.

This updating operation corresponds to the "classical" global dual updating operation along the cut $(N_T, N \setminus N_T)$. In fact, the algorithm based on *classic_reprice* is simply a dual interpretation of the well-known Dijkstra algorithm, in the case of nonnegative arc costs.

Other global dual updating operations are *global_reprice*, which is a mix between *classic_reprice* and *local_reprice*, and *selected_reprice*. More precisely, for each $j \notin N_T$, *global_reprice* performs a *local_reprice* operation at j if j can be linked to T; otherwise, the value δ is added to π_j, as performed by *classic_reprice*:

for each $j \notin N_T$, *global_reprice*(j): $\pi_j := \begin{cases} \beta_j, & \text{if } \gamma_j = \beta_j, \\ \pi(j)+\delta, & \text{otherwise.} \end{cases}$

On the other hand, *selected_reprice* can be viewed as *local_reprice* in a reduced graph obtained from G by removing all the external arcs entering w; this guarantees that at least node w can be linked to T at each step. More formally:

for each $j \notin N_T$, *selected_reprice*(j): $\pi_j := \begin{cases} \beta_w, & \text{if } j = w, \\ \gamma_j, & \text{otherwise.} \end{cases}$

Consider for example the graph in Figure 2, where the arcs in bold belong to the current partial shortest path tree, and 4, 5 and 6 are the current out of tree nodes. In the figure, the number associated with each node is the current node potential, whereas the numbers associated with the arcs are the arc costs.

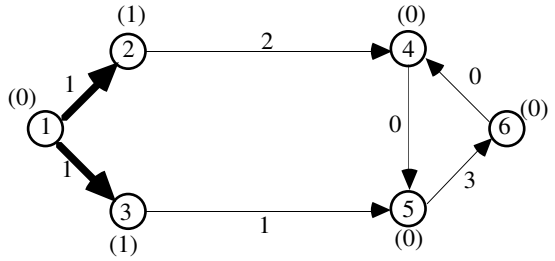


Fig.2

If *local_reprice* is performed at each out of tree node, then the new potentials are the ones illustrated in Figure 3; in this case, no out of tree node enters T. On the other hand, if *classic_reprice* is applied, since it is $\delta = 2$ and $w = 5$, then the new partial shortest path tree and the updated potentials are the ones in Figure 4 (in this example, *global_reprice* behaves as *classic_reprice*). Observe that no dominance relationship can be established between *local_reprice* and *classic_reprice* .

Fig.3: local_reprice

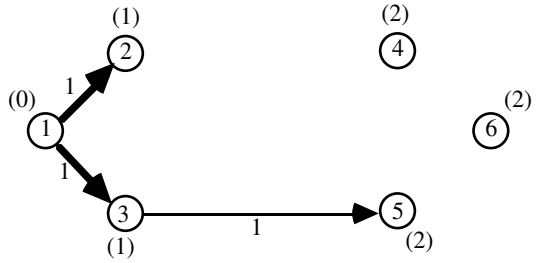


Fig.4: classic_reprice

Sharper global dual updating operations are *extended_reprice* and *double_reprice*. *Extended_reprice* merges *global_reprice* and *selected_reprice*: in fact, it modifies the price of each $j \notin N_T$ for which $\gamma_j = \beta_j$, exactly as performed by *global_reprice* (this is the case in which j can be hanged to T after *global_reprice*); however, the out of tree nodes for which this property does not hold are given the maximum between the price assigned by *global_reprice* and the one assigned by *selected_reprice*:

$$\text{for each } j \notin N_T, \quad \textit{extended_reprice}(j): \quad \pi(j) := \begin{cases} \beta_j, & \text{if } \gamma_j = \beta_j, \\ \max\{\pi(j)+\delta, \gamma_j\}, & \text{otherwise.} \end{cases}$$

The second sharper global dual updating operation is *double_reprice*, which can be interpreted as *extended_reprice* plus a second reprice operation:

$$\text{for each } j \notin N_T, \quad \textit{double_reprice}(j): \quad \pi(j) := \begin{cases} \beta_j, & \text{if } \beta_j \le \eta_j + \delta, \\ \eta_j + \delta, & \text{otherwise.} \end{cases}$$

Extended_reprice and *double_reprice* are illustrated in Figure 5 and in Figure 6, respectively. Observe that *extended_reprice* dominates both *local_reprice* and *classic_reprice*: in fact, the maximum between the potential returned by *local_reprice* and *classic_reprice* is assigned to each out of tree node. Moreover, *double_reprice* dominates *extended_reprice* and all the other dual updating operations.

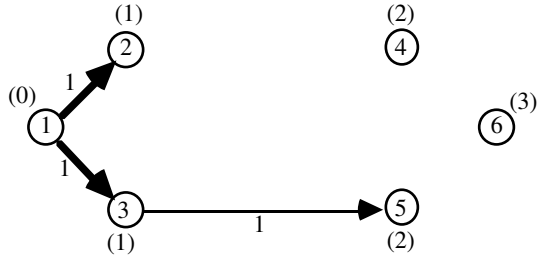


Fig.5: extended_reprice

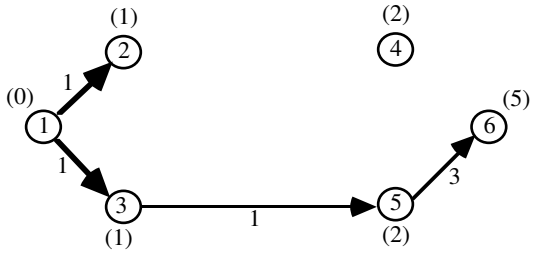


Fig.6: double_reprice

It can be proved that the general algorithmic framework based on the global dual updating operations, named *global_dual*, returns an optimum solution also in the case of zero cost cycles, and it runs in O(*mn*) time. A particular situation arises when either *classic_reprice* or a

suitable implementation of *global_reprice* is used in the above algorithmic framework. In these cases, *global_dual* runs in $O(n^2)$ time when an initial feasible dual solution is provided.

As with the Auction approach, the Hanging approach seems to be powerful mainly in the case of dense graphs, when an "ad hoc" initial dual feasible solution is provided (otherwise, the Hanging algorithms could spend a lot of time improving the initial dual solution, if it is provided in a random way). However, no indication about the practical efficiency of the approach is known at the moment. A wide computational experimentation is in progress to assess the practical performance of the algorithms, also on transportation-like networks. Our feeling, in fact, is that some dual algorithms could be a valid instrument in many transportation applications, specially for reoptimzation purposes (as will be described in the next section).

2.4. Reoptimization approaches in Shortest Paths

In large scale transportation models, a sequence of Shortest Path Problems often needs to be solved, where the $(k+1)^{\text{th}}$ problem differs only slightly from the k^{th} one. In the past years, each problem in the sequence generally used to be solved from the beginning, independently of the previous one. The first work in the direction of using reoptimization approaches was by Murchland (1970), and by Dionne (1978), who faced the problem of finding the shortest cost matrix when the cost of one arc of the input graph was changed. However, although of some theoretical interest, the proposed procedures proved to be inpractical for the large amount of memory required, above all in the framework of transportation networks. Another approach based on the updating of the shortest cost matrix, but using Dijkstra's method for a more efficient updating, can be found in [Fujishige, 1981].

The first efficient reoptimization strategies in the Shortest Path context can be found in [Gallo, 1980], where two general situations arising in transportation models are considered, and efficient reoptimization procedures are proposed for their solution. Assuming that a shortest path tree relative to a given origin r, say T_r, has been determined, the problem is considered of finding a new shortest path tree when either the origin is changed, or the origin is the same but exactly one arc is given a new cost (either higher or lower than the old one). These cases are interesting because most of the modifications occurring in practical situations can be reduced either to one of these two cases, or to a combination of them.

More in detail, when exactly one arc is given a new cost, algorithms are proposed which reoptimize the shortest path tree after the cost change, and which are extensions of Dijkstra's or of Dial's approach. These algorithms are based on rules that depend on the fact that the new cost is higher or lower than the old one.

When the origin is changed to a node $s \neq r$, Gallo's procedure recognizes that the subtree of T_r rooted at s certainly belongs to the new shortest path tree rooted at s, say T_s, and then it computes T_s by means of Dijkstra's or Dial's approach, by using the (optimum) reduced costs relative to T_r instead of the original arc costs:

$$c^*_{ij} = c_{ij} + \pi^r_i - \pi^r_j, \ \forall \ (i,j) \in A, \tag{3}$$

called the *updated costs* (π^r_i denotes the optimum potential of i relative to T_r, $\forall\ i \in N$). In fact, due to the dual feasibility of the price vector associated with T_r, $c^*_{ij} \geq 0$, $\forall\ (i,j) \in A$, and so shortest-first search procedures can also be used for original negative costs (observe that $c^*_{ij} = 0$ for each (i,j) belonging to T_r). This strategy is shown to be particularly effective when implementing Dial's approach, both in time and in memory requirements, above all when r and s are "close enough" in the input graph G. The reoptimization procedures have been tested on real large scale transportation networks, and the author's conclusion is that also the introduction

of slight improvements within the shortest path codes may result in major savings in computational time.

By extending Gallo's observations about the properties of shortest path trees when the root node is changed, a more efficient algorithm is proposed in [Gallo&Pallottino, 1982], which compares favorably with other algorithms in the literature on transportation networks. The algorithm maintains a cost label C_i^*, $\forall\, i \in N$, relative to the updated costs (3). Starting from the new root s, at each step the algorithm extends the current shortest path tree, say T^*, by inserting new nodes and new arcs until T^* spans onto G. More precisely, the algorithm considers the arcs in the cutset separating T^* from the remaining nodes, and adds to T^* a cutset arc, say (u, v), such that u belongs toT^* and v has a minimum cost label C_v^*. Moreover, since when (u, v) and v are added to T^*, all the nodes in the subtree of T_r rooted at v are reachable from v through arcs having a zero updated cost, i.e. all these nodes have the same minimum label as v, the entire subtree is moved to T^* by suitably implementing the set of the candidate nodes Q, so obtaining an efficient "ad hoc" implementation of the shortest-first search approach for reoptimization.

Although presented as a primal approach, Gallo&Pallottino's algorithm can also be interpreted as a dual method. In fact, starting from the partial shortest path tree T^*, a dual feasible solution is maintained at each step, which is updated by means of an operation which behaves exactly as *classic_reprice* (see the previous section), and T^* is enlarged at each step by inserting arcs having a zero reduced cost, i.e. for which the complementary slackness conditions hold.

As an example, consider the graph in Figure 7 and its shortest path tree T_1 rooted at node 1, where the numbers associated with the nodes are the optimum potentials.

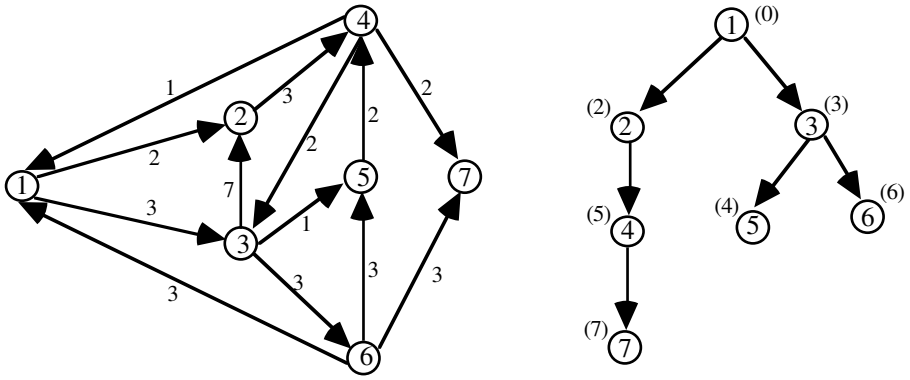


Fig. 7

The updated costs relative to T_1 are shown in Figure 8.

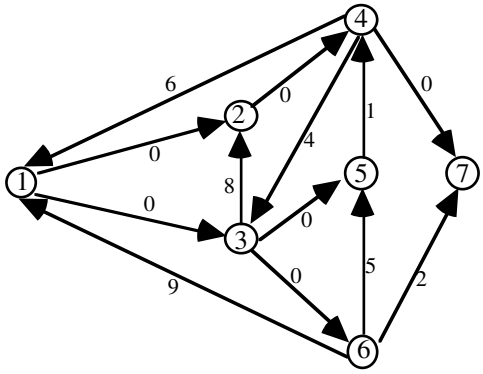


Fig. 8

Assume to change the root from 1 to 3. When this happens, the subtree of T_1 rooted at 3, i.e. T^*, surely will belong to T_3. The partition of T_1 into two arborescences and the

corresponding cutset are shown in Figure 9 on the left, where the costs are the updated costs, and the numbers associated with the nodes are the cost labels C_i^*, $\forall\ i \in N$. The node out of T^* with the minimum cost label is 4: (5,4) and 4 are added to T^* together with node 7, which has the same label as 4. The new tree and the corresponding cutset are illustrated in Figure 9, on the right. The node with the minimum cost label is now the old root 1, and the algorithm ends by moving to T^* all the subtree rooted at node 1.

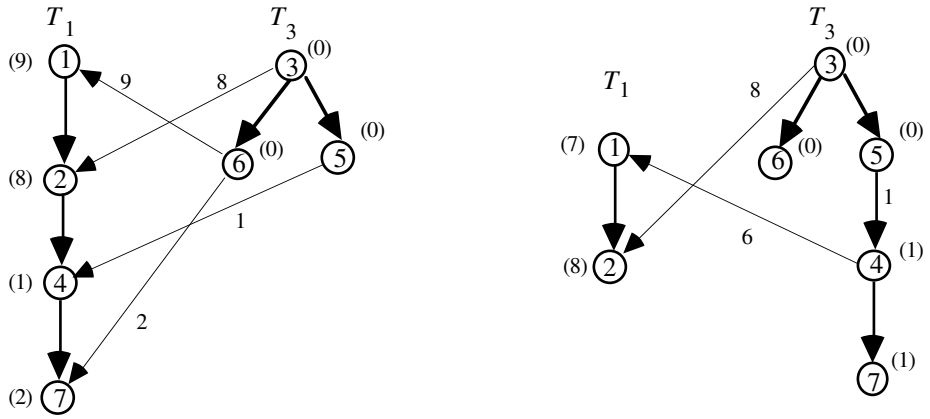


Fig. 9

The final shortest path tree T_3 is thus obtained (see Figure 10): this is the optimum tree rooted at node 3, even if the updated costs have been used for its computation. Observe that, at termination, the optimum node potentials can be easily derived from the optimum cost labels C_i^* and the optimum potentials relative to T_r. In fact, it can be proved that:

$$\pi_j^s = C_j^* + \pi_j^r - \pi_s^r, \quad \forall\ j \in N. \tag{4}$$

In our example, the optimum node potentials relative to T_3 are $\pi^s = [4, 6, 0, 3, 1, 3, 5]$.

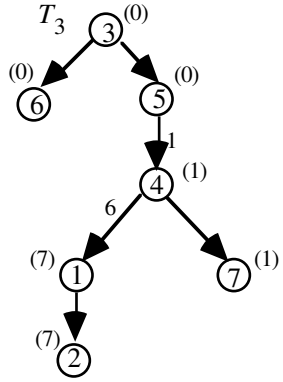


Fig. 10

In 1981 Florian, Nguyen and Pallottino proposed a variant of Gallo&Pallottino's algorithm by properly adapting the classical dual simplex approach to the root change problem. In fact, the optimum tree with respect to the given origin r is primal unfeasible but dual feasible when the root changes to a different node s, and a dual approach is appropriate in this context.
The main ideas are that, firstly, the primal unfeasibility can be efficiently recognized from the orientation of the basic arcs with respect to the new root s. In particular, the most negative valued basic arc, which leaves the basis in Florian, Nguyen&Pallottino's algorithm, can be determined without actually computing the primal basic solution: in fact, it is the first reverse arc along the basic tree path from s to r. The removal of this arc from the basis subdivides the tree into two arborescences, one of them, say T^*, containing s. Then, at each pivot step the entering arc is the arc (u, v) in the resulting cutset with u belonging to T^* and whose reduced

cost $c^*_{uv} = c_{uv} + \pi_u - \pi_v$ is the minimum one, where π_i is the potential of i, $\forall i \in N$ (at the beginning, $\pi_i = \pi^r_i$ is the optimum potential of i relative to T_r, $\forall i \in N$).

The subtree of T_r rooted at v is then automatically moved into T_s. In Figure 11, the first iteration of the algorithm when applied to the graph in Figure 7 is illustrated. There, the numbers associated with the cutset arcs are the reduced costs relative to T_r. Note that these numbers coincide with the cost labels of the nodes of T_1 in Figure 9, on the left.

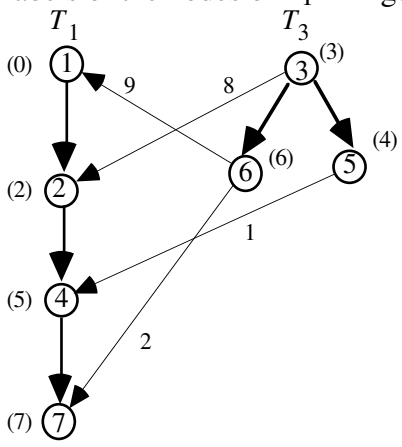


Fig. 11

Like in Gallo&Pallottino's approach, the dual variable updating is done implicitly. In fact, this updating consists in increasing the dual variables of the nodes out of T^* by an amount equal to the reduced cost of the entering arc. In practice, the main difference between the two approaches is that, in Florian, Nguyen&Pallottino's method, the old tree structure T_r is explicitly maintained, and so the algorithm can automatically move into T^* the subtree of T_r rooted at the head of the selected arc. In particular, if Florian,Nguyen&Pallottino's method is applied to the graph and the shortest path tree in Figure 7, the same operations and the same final tree described in Figures from 8 to 10 are obtained.

Finally, the Auction and the Hanging approaches, too, could be suitable for the reoptimization of the shortest path tree when the computation of shortest paths from different root nodes is required. As far as the Hanging algorithms are concerned, consider in fact the shortest path tree T_r, rooted at node r, which is returned by the Hanging approach, and assume that the root is changed to a different node s. Then consider the subtree of T_r rooted at s, i.e. T^*, and the optimal price vector, π^*. Since the complementarity slackness conditions hold for each arc of T^*, we can efficiently apply any Hanging algorithm starting from T^* and π^*, instead of starting from scratch. The algorithm iterations thus involve only those nodes that do not belong to T^*, and they can exploit all the previously computed information (i.e. tree and dual vector), with a potential saving in computing time. As previously observed, it is easy to see that the dual algorithm in the Hanging family which is based on the *classic_reprice* operation, and which is simply a dual interpretation of Dijkstra's algorithm in the case of nonnegative arc costs (as highlighted in Section 2.3), behaves exactly like Gallo&Pallottino's reoptimization procedure when applied to the computation of a shortest path tree from a different root node. Furthermore, it is possible to show that also the other hanging algorithms may collapse to the above reoptimization procedures when applied to the computation of a shortest path tree from a different root node. Our challenge in the reoptimization field is thus to propose new, “ad hoc”, dual updating operations within the Hanging approach, and to perform a wide computational investigation involving both the Hanging and the Auction approaches.

3. Time Dependent Shortest Paths

Interesting problems, that frequently arise in transportation applications, are the so-called *Dynamic Shortest Path Problems*, where the factor "time" is taken into consideration. Applications concern for example street networks, real-time intelligent vehicles and so on (see for instance [Kaufman&Smith, 1993], [De Palma,Hansen&Labbé, 1993], [Nachtigall, 1995]).

Given a directed graph $G = (N, A)$, in dynamic problems a travel time or *delay* $d_{ij}(t)$ is associated with each arc (i,j) with the following meaning: if t is the (nonnegative) *leaving time* from node i, then $t + d_{ij}(t)$ is the *arrival time* at node j. In addition to the delay, a time-dependent cost $c_{ij}(t)$ is generally associated with (i,j) as well, which is the cost of traveling from i to j through (i,j) starting at time t. Furthermore, there is the possibility of waiting at the nodes; in particular, a (unit time) *waiting cost* $w_i(t)$ can be associated with each node i, which gives the (unit time) cost of waiting at i at time t.

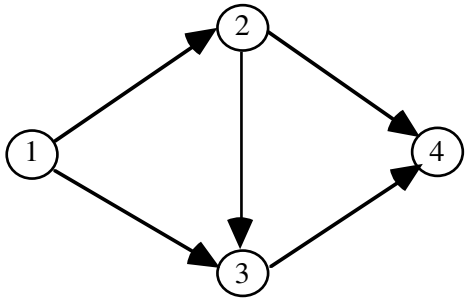
Different models have been defined and analyzed in the literature depending on the properties of the delay functions (e.g. continuous or discrete), on the possibility of waiting at the nodes (e.g. no waiting, waiting at each node, waiting only at the root node), and depending also on the choice of the leaving time from the root node (in particular, dynamic shortest paths for a fixed leaving time or for all the possible leaving times). As far as the models based on continuous delay functions are concerned, we refer to [Orda&Rom, 1990] and to [Orda&Rom, 1991]. Here, we shall limit ourselves to the case of *discrete models*, i.e. we shall assume that the time variable t can vary in the discrete set $T = \{t_1, t_2, ..., t_q\}$, and that, when applied to a time instant $t \in T$, each delay function $d_{ij}(t)$ is such that $t + d_{ij}(t) \in T$. This is not a loss of generality in the context under consideration, since, in the transportation field, time "discretisation" is generally performed.

Dynamic shortest paths in the discrete model can be efficiently studied and solved by introducing the *Space-Time Network* $R = (V, E)$ so defined:

$$V = \{i_h : i \in N, 1 \le h \le q\};$$
$$E = \{(i_h, j_k) : (i, j) \in A, t_h + d_{ij}(t_h) = t_k, 1 \le h < k \le q\}.$$

Each arc (i_h, j_k) is given the cost $c_{ij}(t_h)$. Furthermore, in order to model the situation in which waiting costs at the nodes are also allowed, "waiting arcs" are introduced in R in the following way: (i_h, i_{h+1}) represents the waiting at i from time t_h to time t_{h+1}, and it is given the unit time cost $w_i(t_h)$, $h = 1, ..., q\text{-}1$. Observe that R is a standard graph, with $|V| = nq$ and $|E| \le (m+n)q$, so its size is pseudopolynomial with respect to the size of the original graph G. Moreover, R has no cycles by construction. In particular, each chronological-type visit of the nodes of R, i.e. a visit for non-decreasing values of the time index which is associated with the nodes of R, provides a topological visit of R.

As an example, consider the graph in Figure 12, and assume that the following delays are associated with the arcs in the discrete time period $T = \{t_1, t_2, ..., t_{10}\}$, i.e. q=10.



$d_{12}(t) = [1, 1, 3, 3, 3, 3, 3, 3, 3, 3]$

$d_{13}(t) = [2, 3, 3, 3, 3, 3, 3, 3, 3, 3]$

$d_{23}(t) = [1, 1, 1, 1, 2, 2, 2, 2, 2, 2]$

$d_{24}(t) = [1, 1, 3, 3, 2, 2, 2, 2, 2, 2]$

$d_{34}(t) = [1, 1, 1, 1, 2, 2, 2, 2, 2, 2]$

Fig. 12

The associated Space-Time network R is illustrated in Figure 13, where costs are not reported and the assumption is made that waiting is possible only at node 1.

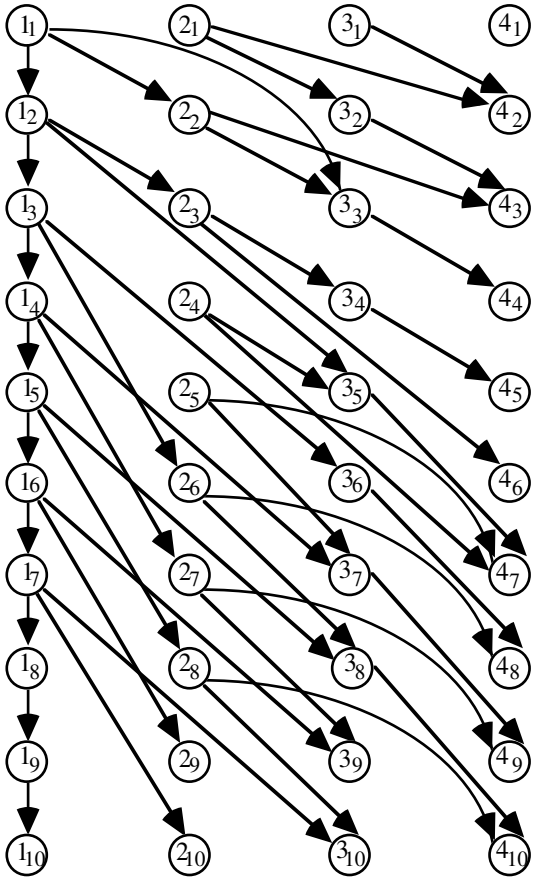


Fig. 13: the Space-Time network

A key property in the discrete case is that it is possible to solve dynamic shortest path problems working implicitly on R, by means of a topological visit being R acyclic (clearly, only the non-redundant portion of R needs to be considered and implicitly generated). So doing, the *Minimum Cost Dynamic Path Problem*, which looks for a minimum cost path from a given root r to each other node $i \neq r$, given a leaving time t, can be solved in $O(|E|) = O(mq)$ time whatever the delay, the cost and the waiting functions.

From an algorithmic point of view, we suggest selecting the nodes of (and thus implicitly generating the non-redundant portion of) R by means of a chronological order, i.e. by considering first the nodes corresponding to time t_1, then the ones for t_2 and so on. The resulting algorithmic paradigm, called *Chrono-SPT*, is formally described below. The paradigm uses a bucket-list $\boldsymbol{B} = \{B_1, B_2, ..., B_q\}$ (Dial, 1969) in order to efficiently perform the selection operations, i.e. for implementing the chronological visit of R, where B_h denotes the bucket containing the nodes to be visited at the time instant t_h, $h = 1, ..., q$ (observe that, more efficiently, a circular bucket-list could be used to save space). In our notation, $P_i(t)$ indicates the current predecessor of node i at time t, i.e. i_t, while $C_i(t)$ denotes the label associated with i_t.

We shall limit ourselves to the description of the typical algorithm iteration, where we shall denote by B_h the "current" bucket used for the node selection, i.e. the first non-empty bucket. The initial step is $B_p = \{r\}$ if the given departure time is $t = t_p$, while the other buckets are empty. The stop condition is verified when all the buckets are empty (when this happens, the minimum of the labels associated with each node $i \neq r$ gives the optimum path cost from r to i).

Procedure Chrono-SPT
* typical iteration *
select i **from** B_h; $B_h := B_h \setminus \{i\}$;
for each $(i,j) \in FS(i)$ **do**
    begin
    $t_k := t_h + d_{ij}(t_h)$;
    if $C_i(t_h) + c_{ij}(t_h) < C_j(t_k)$ **then**
        begin
        $C_j(t_k) := C_i(t_h) + c_{ij}(t_h)$;
        $P_j(t_k) := i_h$;
        if $j \notin B_k$ **then** $B_k = B_k \cup \{j\}$
        end
    end;
* if waiting at i at time t_h is allowed *
if $C_i(t_h) + w_i(t_h)\ (t_{h+1} - t_h) < C_i(t_{h+1})$ **then**
    begin
    $C_i(t_{h+1}) := C_i(t_h) + w_i(t_h)\ (t_{h+1} - t_h)$;
    $P_i(t_{h+1}) := i_h$;
    if $i \notin B_{h+1}$ **then** $B_{h+1} = B_{h+1} \cup \{i\}$
    end;

It is easy to prove that Chrono-SPT is correct and that it runs in $\Theta(q+|E^*|)$ time, where $E^* \subseteq E$ is the set of the non-redundant arcs of R implicitly generated by Chrono-SPT. Clearly, since $|E^*| \leq mq$, in the worst case the algorithm time complexity is $O(mq)$.

By considering the example in Figures 12 and 13, if the root is node 1, and the admitted leaving times are t_1, t_2 and t_3, then the non-redundant portion of R is the one in Figure 14, where costs are now associated with the arcs.

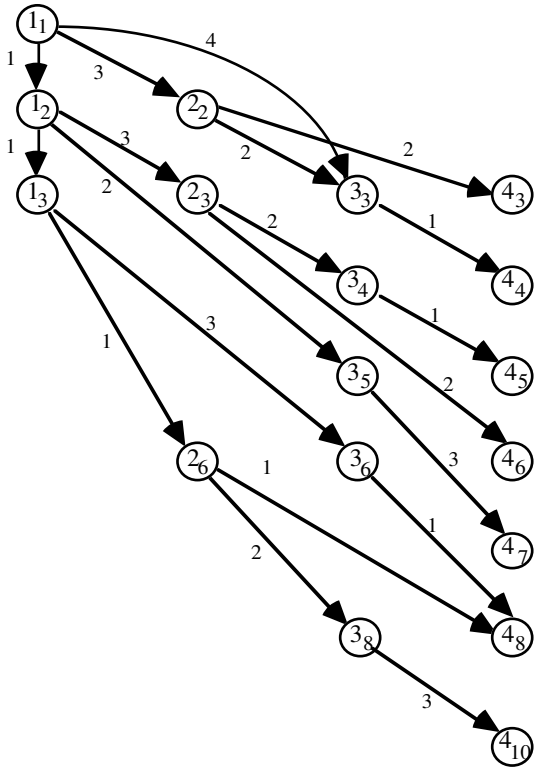


Fig. 14: the non-redundant portion of R

Assume now that a minimum cost path needs to be found on R from node 1 to node 4. In this case, the bucket-list $\boldsymbol{B} = \{B_1, B_2, ..., B_{10}\}$ constructed by Chrono-SPT for implementing the chronological visit of R is shown in Figure 15. There, each bucket B_h contains a list of

pairs of type (*node*, *label*), where *node* denotes a node to be visited at the time instant t_h, $h = 1, ..., 10$, while *label* represents the corresponding cost label.

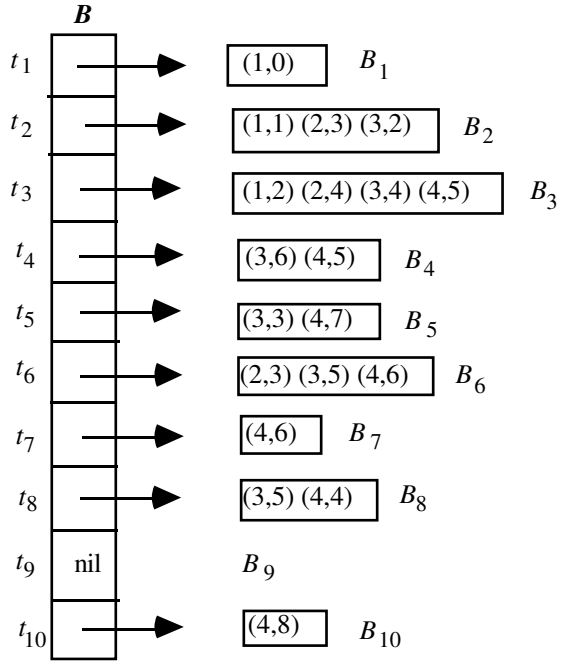
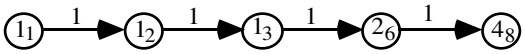


Fig. 15: the bucket-list

In our example, the minum cost label associated with node 4 is valued 4, and it is found at the time istant t_8, i.e. in the bucket B_8. The minimum cost path from 1 to 4 is then:



3.1. FIFO and Cost Consistency

When some local properties are satisfied by the dynamic model, then stronger approaches can be derived. Consider first the so-called FIFO property: an arc (i,j) is said to be a *FIFO arc* if leaving i earlier guarantees that one will arrive no later at j along (i,j). More formally:

$$t_h + d_{ij}(t_h) \leq t_k + d_{ij}(t_k), \text{ for any } t_h < t_k. \tag{5}$$

A dynamic graph G is said to be a *FIFO graph* when all its arcs are FIFO, and *non-FIFO* otherwise.

When waiting at the nodes is allowed, a similar property can be imposed on the arc costs as follows: an arc (i,j) is said to be a *Cost Consistent* (*CC*) *arc* if leaving i earlier along (i,j) does not cost more than leaving later, and G is a *CC graph* when all its arcs are CC arcs. More formally, let $t_u = t_h + d_{ij}(t_h)$ and $t_v = t_k + d_{ij}(t_k)$, for any $t_h < t_k$. Two cases are considered:

1- (i,j) is a FIFO arc ($t_u \leq t_v$); in this case it is also a CC arc if, for any $t_h < t_k$:

$$c_{ij}(t_h) + \sum_{z=u}^{v-1} w_j(t_z)\ (t_{z+1} - t_z) \leq c_{ij}(t_k). \tag{6}$$

2- (i,j) is not a FIFO arc ($t_u > t_v$); in this case it is a CC arc if, for any $t_h < t_k$:

$$c_{ij}(t_h) \leq c_{ij}(t_k) + \sum_{z=v}^{u-1} w_j(t_z)\ (t_{z+1} - t_z)\ . \tag{7}$$

The cost consistency property is illustrated in Figure 16. The above definition is different from others in the literature, for instance the one proposed by Ziliaskopoulos (1994). There, in fact, an arc (i,j) is cost consistent if $c_{ij}(t_h) \leq \sum_{z=h,k-1} w_i(t_z)\,(t_{z+1} - t_z) + c_{ij}(t_k)$, for any $t_h < t_k$.

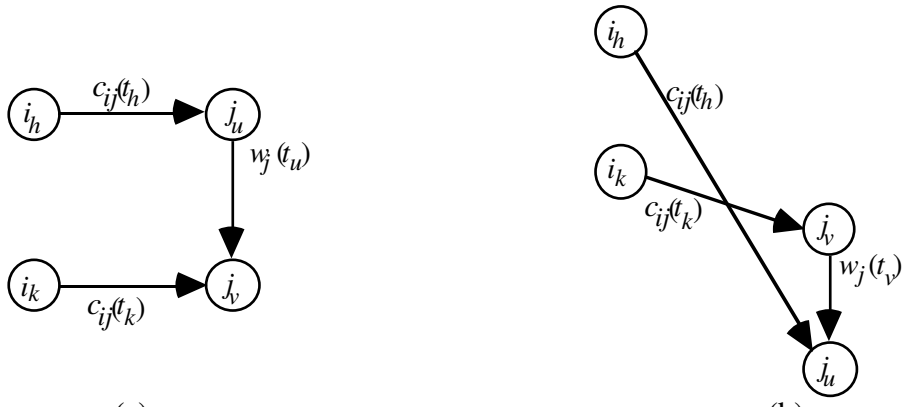


Fig. 16: CC arc (i,j): (a) (i,j) *is FIFO* $(u \leq v)$, (b) (i,j) *is not FIFO* $(u > v)$

When the graph under consideration is only FIFO or only CC, then no relevant additional properties generally hold for dynamic problems. However, if G is both FIFO and CC, i.e. when leaving earlier along any arc allows one to arrive no later and to pay no more than leaving later, then the concept of "dominated labels" can be introduced and exploited in the algorithmic paradigm Chrono-SPT. Suppose in fact we have visited two different paths from a given origin node r to a certain node i, and suppose that the two paths arrive at i at time t_h and at time t_k, respectively, with $t_h < t_k$. Assume also that the costs of the two paths, $C_i(t_h)$ and $C_i(t_k)$, are such that $C_i(t_h) \leq C_i(t_k)$. In this case, since the graph is both FIFO and CC, it is not convenient to extend the second path through any arc (i,j) outgoing from i, since this extension will not produce any minimum cost path going through node i: the cost of the second path, i.e. $C_i(t_k)$, is a so-called *dominated label* for node i, and it can be ignored. Chrono-SPT can thus be simplified in such a way to only maintain non dominated labels. Observe that, if $\boldsymbol{C}_i = \{C_i(t_{h1}), C_i(t_{h2}), ..., C_i(t_{hz})\}$ is the current set of the non dominated labels associated with node i, ordered with respect to increasing values of the time index, i.e. $t_{h1} < t_{h2} <....< t_{hz}$, then it must be:

$$C_i(t_{h1}) > C_i(t_{h2}) > ... > C_i(t_{hz}). \tag{8}$$

If the buckets implementation is adopted, as described in the previous section, since the overhead due to the maintenance of the ordered sets $\boldsymbol{C}_i$ can be time-consuming, we suggest checking dominance only when label $C_i(t_h)$ is selected from the current bucket. In this case, it is sufficient to modify Chrono-SPT in such a way to compare $C_i(t_h)$ with the last selected (and non dominated) label relative to node i, that we call *last-label*$_i$. In fact, if $C_i(t_h) \geq$ *last-label*$_i$ we can discard $C_i(t_h)$, since it is a dominated label; otherwise, we can use $C_i(t_h)$ as described in Chrono-SPT, since it is not dominated. The typical iteration of Chrono-SPT is rewritten below.

Observe that waiting arcs can be ignored, since the graph is FIFO and CC. Observe also that the maintenance overhead is charged to the selection operations, without increasing the time complexity of Chrono-SPT. In particular, during the algorithm execution, at the same time complexity we can build up in an incremental way the ordered set $\boldsymbol{C}_i$ for each node i, and thus compute the set of all the non dominated labels associated with each node.

```
* typical iteration in the case FIFO and CC*
select i from B_h; B_h := B_h \ {i};
if C_i(t_h) < last-label_i then
begin
    last-label_i :=C_i(t_h);
    for each (i,j) ∈ FS(i) do
        begin
        t_k := t_h + d_ij(t_h );
        if C_i(t_h) + c_ij(t_h) < C_j(t_k) then
            begin
            C_j(t_k) := C_i(t_h) + c_ij(t_h);
            P_j(t_k) := i_h;
            if j ∉ B_k then B_k := B_k ∪ {j}
            end
        end
end;
```

As an example, consider the dynamic graph in Figure 17 for the time period $T = \{t_1, t_2,..., t_6\}$, and assume that all the delays are equal to 1. The graph is thus FIFO.

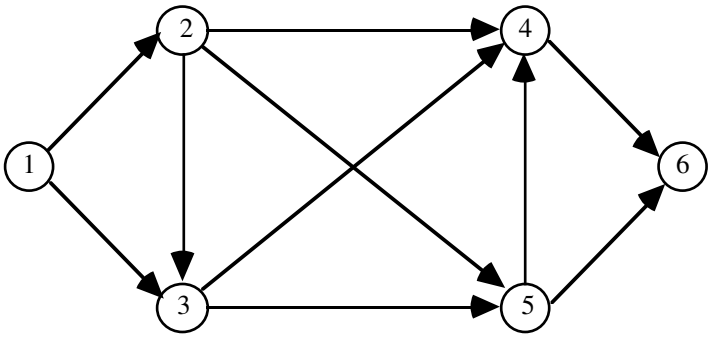


Fig. 17: a FIFO graph

If the root node is 1, the leaving time is t_1 and the destination node is 6, then the non-redundant portion of the Space-Time network R is the one in Figure 18, where costs are associated with the arcs, and waiting costs are assumed to be zero for all the waiting arcs. Observe that the CC-property is satisfied for each arc: the dynamic graph in Figure 17 is thus also a CC graph, and waiting arcs can be removed from R.

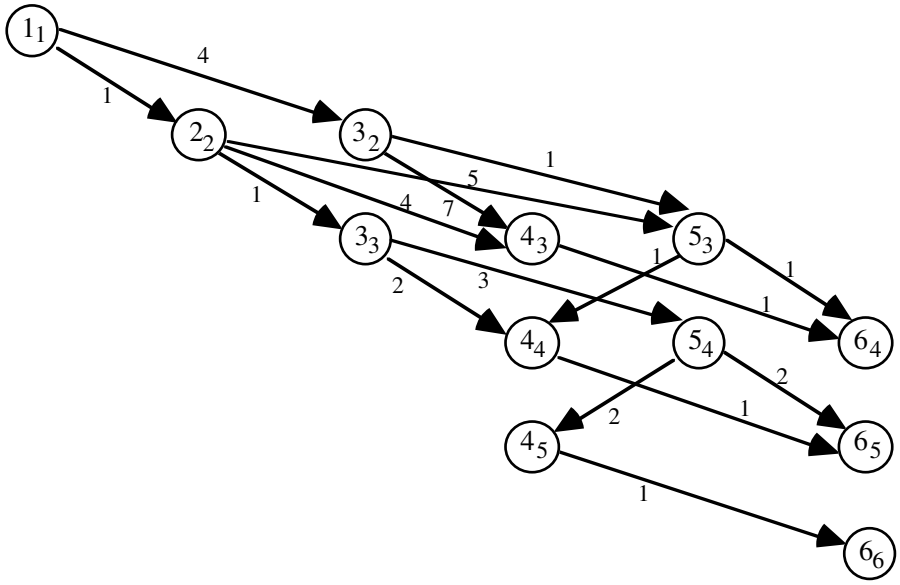


Fig. 18: the non-redundant portion of R

The bucket-list $\boldsymbol{B} = \{B_1, B_2, ..., B_6\}$ constructed by Chrono-SPT for implementing the chronological visit of R is shown in Figure 19. Since the graph is both FIFO and CC, only non dominated labels are maintained into the bucket-list.

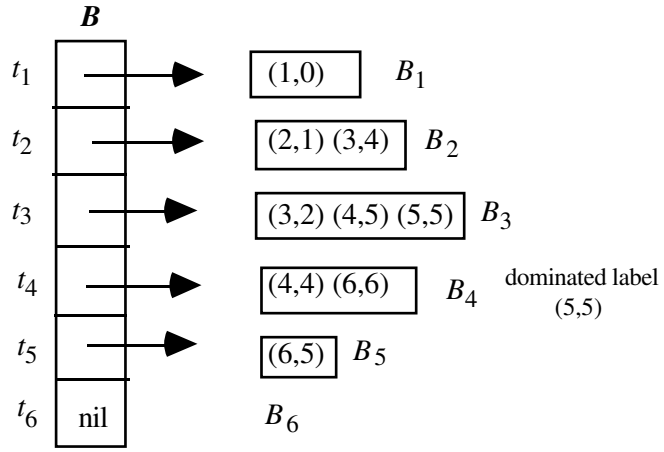


Fig. 19: the bucket-list

Observe, in particular, that since the label associated with node 5_4 is dominated, then the nodes 4_5 and 6_6 are not visited by Chrono-SPT: only a subgraph of the non-redundant portion of R is thus generally visited when the graph is FIFO and CC. The minimum cost path from 1 to 6 is then:

3.2. Minimum time for specific departure time

An important special dynamic problem is the *Minimum Time Dynamic Path Problem for specific departure time*, in which we want to find the minimum arrival time path from a given origin node r to any other node $i \neq r$, given a departure time t. In this case, if we look at the Space-Time network R, the problem can be reformulated as a "connectivity" problem on R. In fact, for any destination node i we want to find the minimum time instant $t' \in T$ for which a directed path exists in R from node r_t to node $i_{t'}$. We can efficiently solve the problem by a modification of the paradigm Chrono-SPT, in which the cost labels are ignored, and the stop condition is verified when each node $i \neq r$ is selected once from the bucket data structure. If the original graph G satisfies the FIFO property, then it is easy to show that only one non dominated label is associated with each node, i.e. the minimum arrival time at that node. Chrono-SPT is thus reduced to the classical Dial's shortest path algorithm with bucket-list. In this case, the problem time complexity is $O(m+\min\{q, n\log n\})$.

3.3. Minimum time for all departure times

A related problem is the *Minimum Time Dynamic Path Problem for all departure times*, which looks for minimum arrival time paths from any node $i \neq d$ to a given destination node d for all the possible departure times. This problem was first addressed by Cooke and Halsey (1966) and by Dreyfus (1969). Recently, it was solved by Ziliaskopoulos and Mahmassani (1993) by extending a classical shortest path approach, in $O(n^3q^2)$ time. Observe that, if Chrono-SPT is called q times, i.e. once for each $t \in T$, then an algorithm is obtained which solves the problem $\Theta(|E^*|q)$ time, i.e. in $O(mq^2)$ time. However, a more efficient approach can be devised which is based on a "reverse" chronological visit of R, that is a reverse scanning order of the bucket-

list ***B***. Let $C_i(t)$, $\forall\ i$ and $\forall\ t$, be the current minimum time of the paths from i to d leaving from i at time t. At the beginning, $C_i(t) = +\infty\ \forall\ i \neq d$ and $\forall\ t$, $C_d(t) = 0\ \forall\ t$, and (implicitly) $B_h = \{d\}$, h=1,.., q. Starting from bucket B_q, perform a "reverse" chronological visit of R: when a node is selected, its backward star is scanned, labels are properly updated and, when necessary, nodes are inserted into the bucket-list. It is easy to show that, at termination, the minimum arrival time at d is determined for each $i \neq d$ and for each possible departure time from i. Clearly, the time complexity is $\Theta(q + |E^*|)$ for this "reverse" approach as well, which is O(mq) in the worst case. An approach based on a "reverse" chronological visit of R has been independently proposed by Chabini (1997) and by Pretolani (1997). However, Chabini's algorithm visits the entire space-time network R, whereas the algorithm derived from Chrono-SPT visits only the non-redundant portion of R.

In order to illustrate the proposed approach, consider the Space-Time network R in Figure 13; assume that waiting is allowed at each node and that the only admitted arrival times are t_1, t_2, ..., t_5. If the destination node is 4, then the non-redundant portion explored by the "reverse" chronological visit of R is the one shown in Figure 20, together with the corresponding optimum time labels.

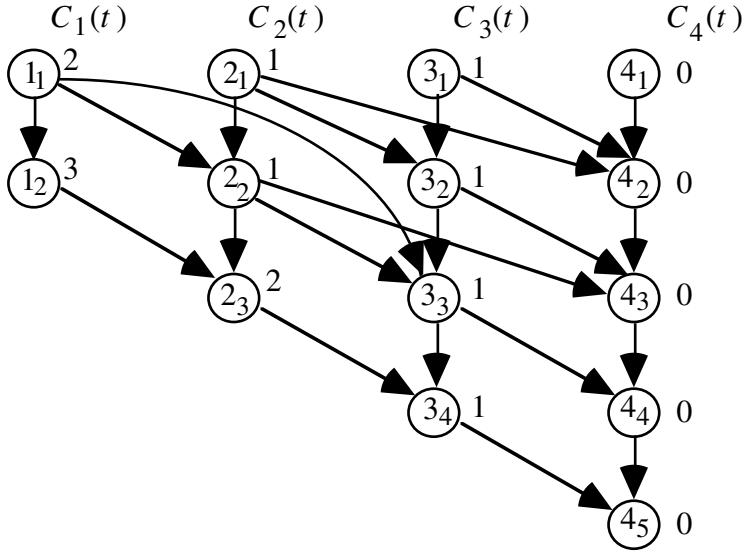


Fig. 20: all departure times and minimum time labels

3.4. Time windows

Chrono-SPT is a very powerful paradigm, which can also be applied to a related dynamic problem, the *Shortest Path Problem with Time Windows*, described by Desrosiers, Pelletier and Soumis (1983). In this problem, a constant cost c_{ij} and a constant positive delay d_{ij} are associated with each arc (i,j), and shortest paths with respect to the given costs are looked for. The crucial aspect is that a time window $[a_i, b_i]$ is associated with each node i, stating that it is possible to leave from node i only in the integer times belonging to the interval $[a_i, b_i]$ (waiting is allowed if the arrival at node i is before time a_i). In order to exploit Chrono-SPT, it is sufficient to define the Space-Time network $R = (V, E)$ as:

$$V = \{i_h: i \in N, a_i \leq h \leq b_i\};$$
$$E = \{(i_h, j_k): (i, j) \in A, a_j \leq k = \max\{h+d_{ij}, a_j\} \leq b_j\}.$$

Although defined for constant delays, the Shortest Path Problem with Time Windows can be immediately extended to the case of time dependent delays. For example, by imposing

constant costs to all the arcs of the graph in Figure 12, and associating the time windows listed in Figure 21 with its nodes, we obtain the Space-Time network partially described in Figure 21 (for the sake of simplicity, delays and costs are omitted). In that figure, the arc $(2_2,4_4)$ has been depicted in order to illustrate the possibility of waiting at 4 until time a_4.

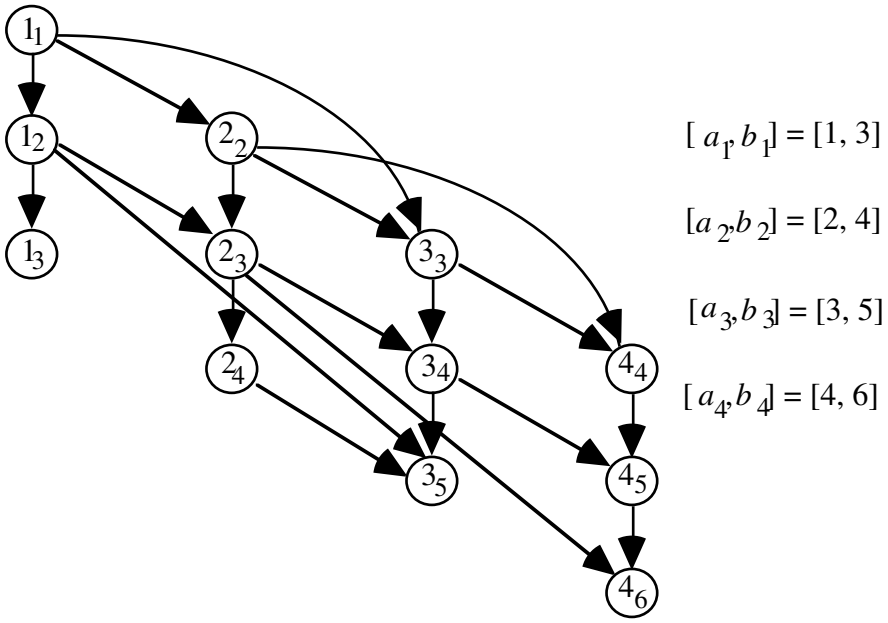


Fig. 21 the Space-Time network in the case of time windows

In the case of time windows and constant delays, the Space-Time network R is both FIFO and CC, since delays and costs are constant, and waiting costs are zero. Moreover, it is $|V| = D$, where $D = \sum (b_i - a_i + 1)$, and $|E| \leq \min\{nD, md\}$, where $d = \max\{b_i - a_i + 1\}$. Chrono-SPT thus solves the Shortest Path Problem with Time Windows in $O(\min\{nD, md\})$ time, whatever the delay function. An adaptation of Chrono-SPT in the case of time windows had already been proposed by Desrochers and Soumis (1988), following some of Pallottino's suggestions.

4. Other issues in transportation models

In many transportation problems, shortest paths are not present in their "pure" form. Often, a minimum cost path with some additional properties is looked for. The aim of this section is to present some of these special cases, which have been studied quite frequently in the literature. In particular, we shall address the minimum cost path problem with turn penalties and prohibitions, the multicriteria shortest path problem, and the shortest hyperpath problem.

4.1. Minimum cost paths with turn penalties and prohibitions

In urban private car models, turning left is often forbidden and, when allowed, traffic lights and counter flow cause an extra travel cost which has to be taken into account when finding the minimum cost itinerary between two points.

Generally speaking, given a graph $G = (N, A)$, where c_{ij} denotes the cost of the arc $(i,j) \in A$, assume that a *penalty* σ_{ab} (≥ 0) is associated with each pair of *subsequent arcs* $a = (i,j)$ and $b = (j,k)$. A finite and positive penalty represents the average cost of entering the arc b from arc a; whereas an infinite penalty represents a turn prohibition (that is, it is forbidden to enter b from a). In this framework, the path cost is the sum of the arc costs and of the pair-

of-arcs penalties relative to the path. Note that a forbidden path (i.e. a path containing a turn prohibition) has an infinite cost.

Different approaches have been followed in the literature to address the above problem (see, among the others, [Kirby&Potts, 1969] and [Ziliaskopoulos&Mahmassani, 1996]). Some approaches work on the so-called "dual network", i.e. the graph obtained by considering the original arcs as nodes and introducing one arc for each pair of subsequent arcs of the original graph, with a cost equal to the sum of the cost of the first arc and of the penalty. Other approaches build up the "expanded network", which is obtained by highlighting each movement in the intersections by means of dummy nodes and arcs, where the costs of the dummy arcs are the penalties, as shown in Figure 22. The major drawback of both approaches is that the resulting network is considerably larger than the original graph; moreover, any modification in the movement rules requires network updating.

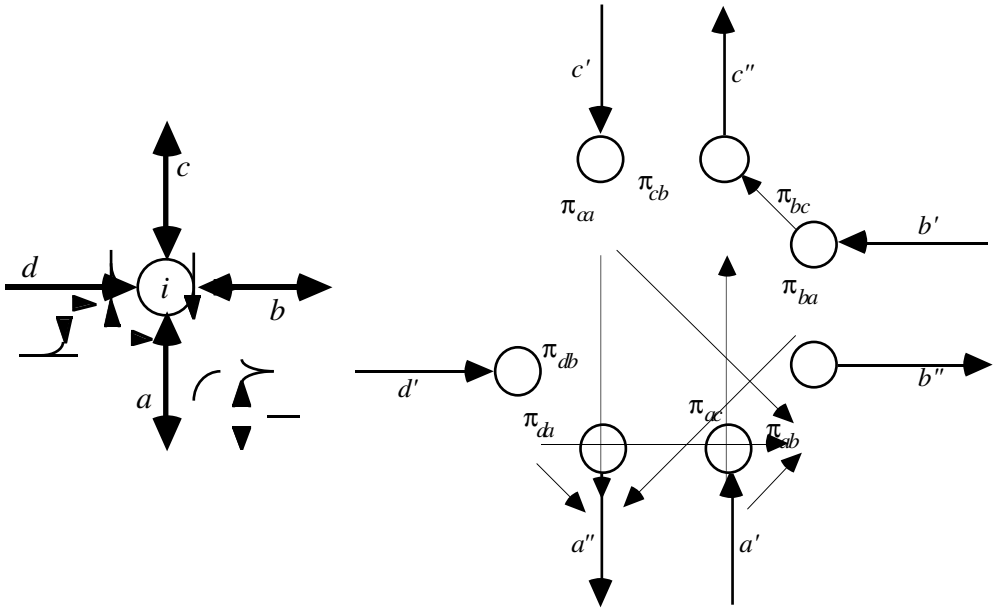


Fig. 22: an expanded intersection with penalties on movement arcs

As an alternative approach, Kirby and Potts (1969) describe a path as a sequence of arcs instead of a sequence of nodes (in this way, it is possible to define the path cost both in terms of the arc costs and of the pair-of-arcs penalties), and based on this new formulation, they generalize Bellman's classical conditions. Although Kirby and Potts suggested no algorithm, their work inspired many researchers to adapt their own shortest path algorithm to the new formulation. In the literature, in fact, there is a wide variety of so-called "arc-label" shortest path algorithms, which have been obtained by exploiting Bellman's conditions related to the problem arc-formulation within a classical shortest path algorithmic framework (for instance, see [De La Barra, 1989]). In the following, for the sake of brevity, we shall describe only the key points of this "arc-label" shortest path approach.

Let the input graph be stored in a forward adjacency list form (Gallo&Pallottino, 1988; Della Valle&Tartaro, 1994; Ziliaskopoulos&Mahmassani, 1996), where each node points to its adjacent nodes (assuming that the corresponding arc costs are stored properly) and, likewise, each arc points to its subsequent arcs (and to the corresponding pair-of-arcs penalties). The "arc-label" shortest path approach maintains a candidate list Q, a label vector C and a predecessor vector P related to the arcs: C_a is the label of the arc a, and it represents the cost of a path from a given root r to the end of a; while P_a contains the arc preceding a in the current path from r to a.

The core of the algorithm is to select and remove an arc from the list Q, say $a = (i,j)$, and to properly scan the forward star of its ending node j, i.e. $FS(j)$. Let $b = (j,k)$ denote the generic arc in $FS(j)$. If Bellman's condition for the pair of arcs a and b:

$$C_a + \sigma_{ab} + c_{jk} \geq C_b,$$

is violated, then the label and the predecessor of the arc b are updated:

$$C_b := C_a + \sigma_{ab} + c_{jk};\ P_b := a,$$

and b is inserted into Q (if not already present).

At the beginning, Q contains a dummy arc entering r. When Q is empty, P and C define respectively the set of the shortest paths (in terms of arcs), and the minimum costs from r to the end of each arc.

A label-correcting version of the "arc-label" shortest path approach has recently been proposed by Ziliaskopoulos and Mahmassani (1996). In their algorithm, as many labels as the number of the outgoing arcs are associated with each node, where label C_a represents the cost of a path from the origin r to the beginning of arc a. Let list Q be the set of the candidate nodes. When a node is selected and removed from Q, all its labels are treated as described before and, if at least one label of another node is improved on, then that node is inserted into Q. Observe that, so doing, also labels that have not been improved may be treated at each node selection operation. In our opinion, this overhead in the label management may affect the overall efficiency of the algorithm.

A hybrid algorithm was proposed by Della Valle and Tartaro (1994) in order to manage street networks where turn penalties are defined only for a portion of the street intersections. Their algorithm handles arc labels and node labels together: node labels are used when there are no turn penalties, while arc labels are adopted in the presence of turn penalties. The authors show experimentally on two Italian street networks that the proposed approach is very competitive against both the expanded network and the "pure" arc-label approach. In fact, their tests show that the hybrid algorithm dominates the first approach when at least 30% of intersections have turn penalties and prohibitions, while it dominates the arc-label algorithm regardless of the rate of intersections with turn penalties and prohibitions. When all the city intersections have turn penalties and prohibitions, the average running time of the hybrid algorithm is the same as the one of the arc-label approach (in fact, the two algorithms coincide), while it is 15% less than the average running time of approach based on the expanded network.

4.2. Multicriteria shortest paths

In many transportation problems, paths are characterized by two or more attributes: length, cost, time, risk of congestion, and so on. In some cases, the different attributes can be condensed into a new unique attribute, the "generalized cost", by combining the original attributes. In these cases, paths with a minimum generalized cost are looked for. However, in some applications the condensation of the attributes is either impossible or discouraged, and direct approaches must be followed to find optimal solutions. For example, suppose to extend the dynamic problems described in Section 3, and suppose we want to find all the non dominated paths under the two criteria "cost" and "time". It is easy to show that a unique optimal path could not exist in this situation, and that even a large set of non dominated paths might represent the "target" solution set.

For the sake of simplicity, let us consider the bicriterion shortest path problem. Given the graph $G = (N, A)$, assume that two measures c_{ij} and w_{ij} are associated with each arc $(i,j) \in A$,

which hereafter will be called the *cost* and the *weight* of (i,j), respectively. Let P be the set of the arcs in a path from a given origin to a given destination, and let $C(P)$ and $W(P)$, i.e. the cost and the weight of P, denote two functions of the costs and the weights, respectively, of the arcs belonging to P:

$$C(P) = f(\{c_{ij}: (i,j) \in P\}), \qquad W(P) = g(\{w_{ij} : (i,j) \in P\}). \tag{9}$$

Let Π denote the family of all the feasible subsets of arcs, i.e. the arcs in the paths from the given origin to the given destination. $P \in \Pi$ is a *dominated path* if there exists a path $P' \in \Pi$ such that $C(P) \geq C(P')$, $W(P) \geq W(P')$, and at least one strict inequality holds; otherwise, P is said to be *non dominated* (or *efficient*). The *bicriterion shortest path problem* is to find the sub-family $\Pi_E \subseteq \Pi$ formed by all the efficient paths from the given origin to the given destination.

The general bicriterion shortest path problem is NP-hard; Hansen (1980) studied when the problem can be solved in polynomial time, depending on the properties of functions f and g. Mote, Murthy and Olson (1991) proposed a parametric algorithm for determining all the efficient paths, which is based on linear programming relaxation. Warburton (1984, 1987) described exact and approximate algorithms for the problem solution. In particular, in (1987) he proposed some fully polynomial approximation schema for approximating all the efficient paths. Generalizations of fully polynomial approximation schema for multicriteria shortest path problems are suggested in [Safer&Orlin, 1995], where necessary and sufficient conditions are developed for the existence of such fast approximation schema. Other approaches, very efficient in some particular situations, are described in the following.

Consider the bicriterion shortest path problem from a common origin node r to every other node of the graph. Suppose that, for one of the two measures, say the path weight, the ordered list of all the possible path weights, say $\boldsymbol{W} = \{W_1, W_2, \ldots, W_k\}$, is finite and known *a priori* (this is common in several transportation applications). Suppose also that $W_j < W_{j+1}$, $j = 1, \ldots, k$-1. In these hypotheses, a pseudopolynomial algorithm can be derived by associating k different labels with each node $i \in N$, one for each possible path weight in $\boldsymbol{W}$, as follows: $C_i(W_j)$ is the current minimum cost of the paths from r to i that have a weight equal to W_j. If one keeps the set $\boldsymbol{C}_i$ of the non dominated labels related to node i, then, for each pair of non dominated labels, say $C_i(W_j)$ and $C_i(W_h)$ with $W_j < W_h$, it is easy to prove that the following inequality must hold:

$$C_i(W_j) > C_i(W_h). \tag{10}$$

Based on the list $\boldsymbol{W}$ and on the concept of dominance, two different algorithmic approaches have been proposed in the literature. The first one is a "multi-label" approach (for instance, see [Ziliaskopoulos, 1994]). For the sake of simplicity, assume that costs and weights are nonnegative, and that f and g in (9) are the "sum" function. Given a node, suppose that the non dominated labels associated with that node are listed according to the "weight" index; then, due to the non dominance property, these labels are also ordered in a decreasing way with respect to their values. When a node, say i, is selected and the leaving arc (i,j) is analyzed, each candidate label $C_i(W_h) + c_{ij}$ is compared with the label $C_j(W_h+w_{ij})$. Since w_{ij} is constant for the arc (i,j), the list of the non dominated labels of node i is simply shifted with respect to the weight index, and the label values are increased by the same value c_{ij}. It follows that the list of these candidate labels still satisfies property (10), and it can be properly "merged" with $\boldsymbol{C}_j$, that is with the current list of the non dominated labels associated with j (by keeping the lowest value when two labels with the same weight index exist). The new label list of node j can be cleaned up by removing the dominated labels during the "merging" operation.

The second approach from the literature is, in some sense, "topological", and it can be viewed as an adaptation of the version of Chrono-SPT which maintains the sets of the non dominated labels. Let us implement the sets of the non dominated labels as a bucket-list $\boldsymbol{B}$, where the h^{th} bucket B_h contains all the non dominated labels $C_i(W_h)$, for $i \in N$. At each step, the current label is selected from the lowest non-empty bucket (i.e., weights are considered according to their non-decreasing ordering), and it is processed by performing similar operations as the ones described for Chrono-SPT. Note that, here too, the physical number of buckets to be implemented can be limited to the maximum arc weight plus one (Dial, 1969). Clearly, the "topological" algorithm finds the non dominated paths according to the non-decreasing ordering of the path weights.

In general, which approach is the best to follow, i.e. the "multi-label" or the "topological" one, strongly depends on the structure of the problem under consideration. As an example, we shall describe a transportation problem for which the "topological" algorithm is very efficient. Given a transit network with arc costs, we want to find the minimum cost paths from a given origin to the other nodes subject to the constraint that the number of transfers does not exceed a maximum given number, say k. In this example, we can consider the "transfer number" as the weight associated with the arcs: transfer arcs have a weight equal to 1, while the other arcs have a weight equal to 0. The number of transfers, limited to k, is used as the index both for the labels and for the buckets. When the labels are selected from the h^{th} bucket B_h, $0 \le h \le k$, all the previous buckets have been emptied while the buckets $B_{h+2}, \ldots, B_k$ are still empty (in fact, the transfer number increases by at most one at each step). Then, the wrap-around handling of the bucket-list limits their physical number to two. As a consequence, in order to implement the bucket-list it is enough to maintain two lists, say Q_{now} and Q_{next}, where Q_{now} plays the role of B_h and B_{h+1} is represented by Q_{next}.

Based on the two lists Q_{now} and Q_{next}, a typical iteration consists in selecting a node i from Q_{now} with its corresponding label $C_{i\text{-}now} = C_i(W_h)$. As for Chrono-SPT, we suggest checking dominance at this step, and comparing $C_{i\text{-}now}$ with the last selected (and non dominated) label relative to node i, i.e. $last\text{-}label_i$ in our terminology.

If $C_{i\text{-}now} \ge last\text{-}label_i$ we can discard $C_{i\text{-}now}$, since it is a dominated label. Otherwise, for each arc $(i,j) \in FS(i)$, if (i,j) is not a transfer arc, the candidate label $C_{i\text{-}now}+c_{ij}$ is compared to $C_{j\text{-}now}$: if it is lower, then $C_{j\text{-}now}$ is suitably decreased and, if $j \notin Q_{now}$, then j is inserted into Q_{now} together with its label $C_{j\text{-}now}$; on the other hand, if (i,j) is a transfer arc, then $C_{i\text{-}now}+c_{ij}$ is compared to $C_{j\text{-}next}$ and, when necessary, j is inserted into Q_{next}. When Q_{now} is empty all the minimum cost paths with no more than h transfers have been found. When this happens, the algorithm "swaps" the two lists (i.e. Q_{next} becomes Q_{now} and vice versa), and starts to process the non dominated labels relative to paths with h+1 transfer number. When $h = k$ and Q_{now} has been emptied, then the algorithm stops. The resulting time complexity is O(mk).

As an example, consider the transfer network in Figure 23, where the transfer arcs are the bidirectional ones, having a cost equal to 1, and assume that a minimum cost path needs to be found from node o to node d subject to the constraint that the number of transfers does not exceed 2.

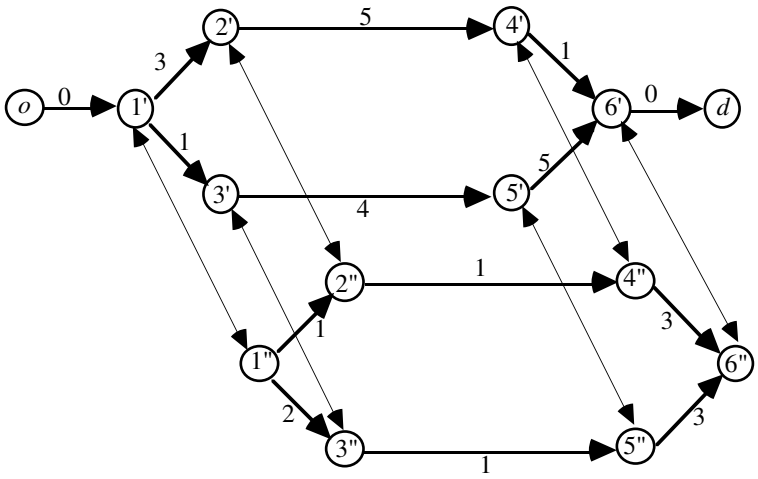


Fig.23: a transit network with transfer arcs

For this example, the set of the considered paths is depicted in Figure 24, where the pair of labels associated with each node denotes the cost and the number of transfers, respectively.

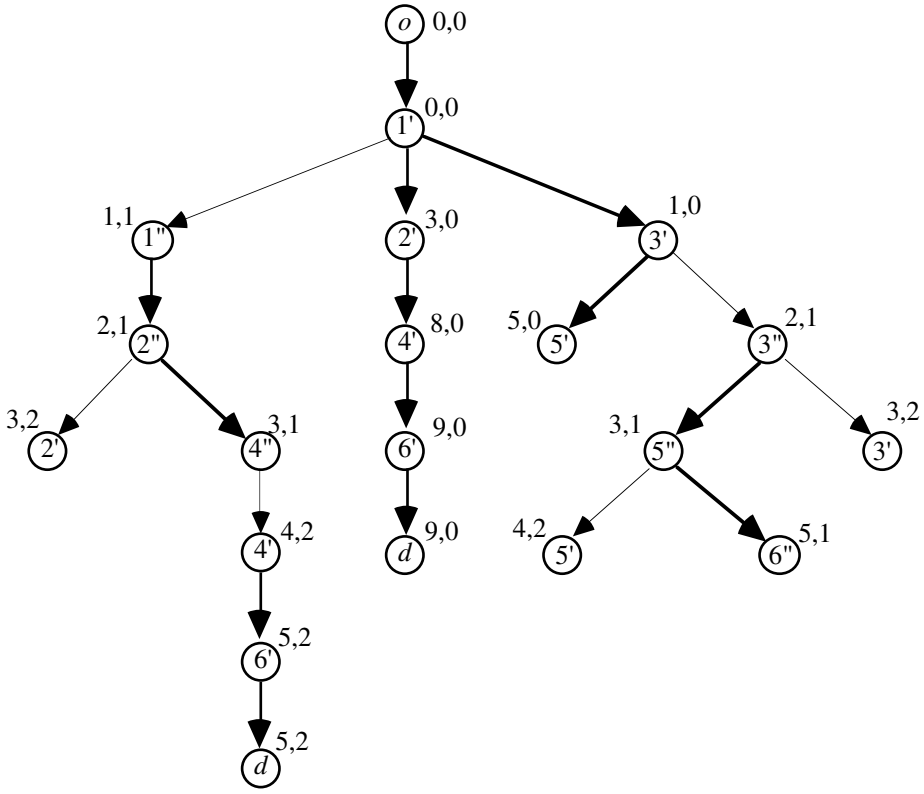
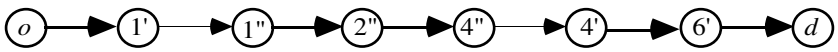


Fig.24

In this example, a minimum cost path from *o* to *d* subject to the constraint that the number of transfers does not exceed 2 is then:



The two lists Q_{now} and Q_{next} constructed by the algorithm when the number of transfers is h=1 are shown below; there, Q_{next} is the list which is obtained when Q_{now} has been emptied.

Q_{now} = {(1",1), (2",2), (3",2), (4",3), (5",3), (6",5)}
Q_{next} = {(2',3), (3',3), (4',4), (5',4), (6',5), (*d*,5)}

4.3. Shortest hyperpaths

Transit services in urban areas are frequently organized so that passengers do not know the exact arrival time of the buses at the stops, but only the frequency of each bus line. As shown in [Nguyen&Pallottino, 1988], the behavior of passengers cannot be represented by a classical path, since when passengers wait at a certain stop to start (or continue) their trip toward their destination, passengers select "a priori" a subset of lines from the lines serving that stop, which is called the *attractive set*. The meaning of the attractive set is that, when passengers wait at a stop, they do not board on buses serving a non-attractive line, but board on the first bus, serving an attractive line, which will arrive at that stop.

If we want to draw the passenger "a priori" strategy defined by the attractive sets, we should consider a set of subpaths leaving the same origin node and arriving at subsequent stops along the attractive sets; then, we should consider other subpaths outgoing from the new stops and using the attractive sets, and so on until the destination node. Each path from the origin to the destination using only attractive lines is in fact an "alternative" for the passenger's trip. The partial graph defined by all these alternatives is a "stream" of paths from the origin to the destination of the trip, called *hyperpath*. Depending on the arrivals of the buses at the stops, the passenger will travel along exactly one of the paths of the stream, with a certain probability.

The "expected cost" of the hyperpath is the mean cost of the paths belonging to the hyperpath, where the weights are the path probabilities. This hyperpath "expected cost" can be computed iteratively starting from the destination and using, at each stop and for each line considered attractive, the probability that a bus serving this line will arrive as the "first" one at that stop. For more details, see [Nguyen&Pallottino, 1986, 1988, 1989; Marcotte&Nguyen, 1997]. The transit networks introduced above can be modeled as particular directed hypergraphs, a generalization of the classical directed graphs (for more details about the theory of hypergraphs and hyperpaths, see [Gallo, Longo, Nguyen&Pallottino, 1992]). In the case of equilibrium transit assignment, it is possible to formulate the problem in terms of hypergraphs, and based on this formulation, solve the problem by iteratively computing "shortest hyperpaths". As shown in [Gallo, Longo, Nguyen&Pallottino, 1992], classical shortest path algorithms can be generalized to find certain classes of shortest hyperpaths in directed hypergraphs, with a low computational time complexity. In particular, some shortest hyperpath algorithms have been adapted to the particular hypergraph structure of the transit networks, as described in [Nguyen&Pallottino, 1986, 1989]. This is true both for shortest-first search algorithms and for list-search algorithms. For the sake of brevity, no further details about these shortest hyperpath algorithms will be reported here.

The hyperpath framework can be exploited not only for urban transit problems. In fact, there are other transportation problems which need to model a user behavior in which the exact itinerary is not completely defined "a priori", since new "information" or a new "network state" can modify the "ideal" itinerary chosen beforehand by the user. Also in these cases, hyperpaths can model quite well this kind of "adaptive user choice", in terms of a stream of alternative paths in addition to the "ideal" path.

This kind of hyperpath generalization in the context of transportation problems is a very recent topic of research, see [Marcotte&Nguyen, 1997] in this book.

Conclusions

We have described the main algorithmic approaches proposed in the literature for different kinds of shortest path problems, both on graphs and on hypergraphs, focusing on those problems which seem to be particularly interesting in the transportation framework.

In our opinion, in the future it will be difficult to discover original and innovative algorithmic approaches for the problems under consideration, both in the case of the "pure" shortest path field, and in the transportation literature. In fact, we believe that future research will focus on the production of "ad hoc" algorithms, derived from the main algorithmic paradigms appeared so far, very much depending on the choices made at the "lower" level -- such as data structures, order of node selection and so on -- which may considerably affect the algorithm performance. This "philosophy" has been illustrated in this work, where some efficient techniques for producing "ad hoc" algorithms from general algorithmic paradigms have been provided, mainly for relevant transportation applications. In particular, we have described a powerful algorithmic schema, Chrono-SPT, for time-dependent shortest path problems, and we have shown how it can be easily tailored to capture the nature of different shortest path problems, such as the bicriterion shortest path one, by producing efficient algorithms in those cases too.

We hope that our description of the algorithmic paradigms, and the techniques outlined in this work for producing efficient and specialized shortest path algorithms, may suggest and stimulate the development of other "ad hoc" procedures in the transportation field. qui

References

R. K. Ahuja, T. L. Magnanti and J. B. Orlin (1993), *Network flows: Theory*, *algorithms*, *and applications*. Prentice Hall, Englewood Cliffs, NJ.

R. K. Ahuja, K. Mehlhorn, J. B. Orlin and R. E. Tarjan (1990), "Faster algorithms for the shortest path problem", *Journal of ACM* **37**, 213-223.

M. S. Bazaraa and R. W. Langley (1974), "A dual shortest path algorithm", *SIAM Journal on Applied Mathematics* **26**, 496-501.

R. Bellman (1958), "On a routing problem", *Quarterly Applied Mathematics* **16**, 87-90.

D. P. Bertsekas (1991a), "An Auction algorithm for shortest paths", *SIAM Journal on Optimization* **1**, 425-447.

D. P. Bertsekas (1991b), *Linear network optimization: Algorithms and codes*. M.I.T. Press, Cambridge, MA.

D. P. Bertsekas (1993), "A simple and fast label correcting algorithm for shortest paths", *Networks* **23**, 703-709.

D. P. Bertsekas, S. Pallottino and M. G. Scutellà (1995), "Polynomial Auction algorithms for shortest paths", *Computational Optimization and Applications* **4**, 99-125.

D. P. Bertsekas and J. N. Tsitsiklis (1989), *Parallel and distributed computation - Numerical methods*. Prentice Hall, Englewood Cliffs, NJ.

I. Chabini (1997), "A new algorithm for shortest paths in discrete dynamic networks", 8th IFAC Symposium on Transportation Systems, Chania, Greece.

B. V. Cherkassky, A. V. Goldberg and T. Radzik (1996), "Shortest paths algorithms: Theory and experimental evaluation", *Mathematical Programming* **73**, 129-174.

L. L. Cooke and E. Halsey (1966), "The shortest route through a network with time-dependent internodal transit times", *Journal of Mathematical Analysis and Applications* **14**, 492-498.

T. De La Barra (1989), *Integrated land use and transport modelling*. Cambridge Univ. Press.

A. De Palma, P. Hansen and M. Labbé (1993), "Commuters' paths with penalties for early or late arrival times", *Transportation Science* **24** (4), 276-286.

G. Della Valle and D. Tartaro (1994), "Ricerca dei percorsi di minimo costo mediante un algoritmo ibrido in presenza di penalità di svolta", *Ricerca Operativa* **24**, 5-38.

E. V. Denardo and B. L. Fox (1979), "Shortest-route methods: 1. Reaching, pruning, and buckets", *Operations Research* **27** (1), 161-186.

E. V. Denardo and B. L. Fox (1979), "Shortest-route methods: 2. Group knapsacks, expanded networks, and branch-and-bound", *Operations Research* **27** (3), 548-566.

N. Deo and C. Pang (1984), "Shortest path algorithms: Taxonomy and annotation", *Networks* **14** , 275-323.

M. Desrochers and F. Soumis (1988), "A generalized permanent labelling algorithm for the shortest path problem with time windows", *INFOR* **26** (3), 191-212.

J. Desrosiers, P. Pelletier and F. Soumis (1983), "Plus court chemin avec contraintes d'horaires", *R.A.I.R.O.* **17** (4), 357-377.

R. B. Dial (1969), "Algorithm 360: Shortest path forest with topological ordering", *Communications of the A.C.M.* **12**, 632-633.

E. W. Dijkstra (1959), "A note on two problems in connexion with graphs", *Numerishe Matematik* **1**, 269-271.

R. Dionne (1978), "Étude et extension d'un algorithme de Murchland", *INFOR* **16**, 132-146.

S. E. Dreyfus (1969), "An appraisal of some shortest-path algorithms", *Operations Research* **17**, 395-412.

M. Florian, S. Nguyen and S. Pallottino (1981), "A dual simplex algorithm for finding all shortest paths", *Networks* **11**, 367-378.

L. R. Ford Jr. (1956), *Network flow theory*, Rand Co., P-293.

M. L. Fredman and R. E. Tarjan (1987), "Fibonacci heaps and their uses in improved network optimization algorithms", *Journal of the ACM* **34**, 596-615.

S. Fujishige (1981), "A note on the problem of updating shortest paths", *Networks* **11**, 317-319.

G. Gallo (1980), "Reoptimization procedures in shortest path problems", *Rivista di Matematica per le Scienze Economiche e Sociali* **3**, 3-13.

G. Gallo, G. Longo, S. Nguyen and S. Pallottino (1992), "Directed hypergraphs and applications", *Discrete Applied Mathematics* **40**, 177-201.

G. Gallo and S. Pallottino (1982), "A new algorithm to find the shortest paths between all pairs of nodes", *Discrete Applied Mathematics* **4**, 23-35.

G. Gallo and S. Pallottino (1984), "Shortest path methods in transportation models", in (M. Florian, ed.) *Transportation planning models*, North-Holland, 227-256.

G. Gallo and S. Pallottino (1986), "Shortest path methods: A unifying approach", *Mathematical Programming Study* **26**, 38-64.

G. Gallo and S. Pallottino (1988), "Shortest path algorithms", *Annals of Operations Research* **13**, 3-79.

F. Glover, R. Glover and D. Klingman (1984), "Computational study of an improved shortest path algorithm", *Networks* **14**, 25-36.

F. Glover, D. Klingman and N. Phillips (1985), "A new polynomially bounded shortest path algorithm", *Operations Research* **33**, 65-73.

F. Glover, D. Klingman, N. Phillips and R. F. Schneider (1985), "New polynomial shortest path algorithms and their computational attributes", *Management Science* **31** (9), 1106-1128.

A. V. Goldberg and T. Radzik (1993), "A heuristic improvement of the Bellman-Ford algorithm", *Applied Mathematics Letters* **6**, no. 3, 3-6.

G. Y. Handler and I. Zang (1980), "A dual algorithm for the constrained shortest path problem", *Networks* **10**, 293-310.

P. Hansen (1980), "Bicriterion path problems", in (G. Fandel and T. Gal, eds.) *Multicriteria decision making*: *theory and applications*, Lecture Notes in Economics and Mathematical Systems **177**, Springer, Heidelberg, 109-127.

M. S. Hung and J. J. Divoky (1988), "A computational study of efficient shortest path algorithms", *Computers and Operations Research* **15**, 567-576.

Y. Israeli and A. Ceder (1996), "Multi-objective approach for designing transit routes with frequencies", in (L. Bianco and P. Toth, eds.) *Advanced methods in transportation analysis*, Springer, Berlin, 157-182.

D. B. Johnson (1973a), *Algorithms for shortest paths*, Ph.D. Thesis, Cornell Univ., tr-73-169.

D. B. Johnson (1973b), "A note on Dijkstra's shortest path algorithm" *Journal of the ACM* **20** (3), 385-388.

E. L. Johnson (1972), "On shortest paths and sorting", *Proc. 25th ACM Annual Conference*, 510-517.

D. E. Kaufman and R. L. Smith (1993), "Fastest paths in time-dependent networks for intelligent vehicle-highway systems applications", *IVHS Journal* **1** (1), 1-11.

A. Kershenbaum (1981), "A note on finding shortest path trees", *Networks* **11**, 399-400.

R. F. Kirby and R. B. Potts (1969), "The minimum route problem for networks with turn penalties and prohibitions", *Transportation Research* **3**, 397-408.

E. L. Lawler (1976), *Combinatorial optimization: Networks and matroids*. Holt, Rinehart and Winston, New York, NY.

P. Marcotte and S. Nguyen (1997), "A hyperpath formulation of capacitated traffic assignment", *Proc. Equilibrium and advanced transportation modelling colloquium*, this book.

P. Mateti and N. Deo (1982), "Parallel algorithms for the single source shortest path problem", *Computing* **29**, 31-49.

J.-F. Mondou (1989), *Mise au point d'un système interactif-graphique pour la planification tactique du transport des marchandises*, Mémoire de maîtrise, Université de Montréal, CRT-677.

J.-F. Mondou, G. T. Crainic and S. Nguyen (1991), "Shortest path algorithms: A computational study with the C programming language", *Computers and Operations Research* **18** (8), 767-786.

E. F. Moore (1959), "The shortest path through a maze", *Proc. International Symposium on Theory of Switching, part 2*, Harvard Univ. Press, 285-292.

J. Mote, I. Murthy and D. Olson (1991), " A parametric approach to solving bicriterion shortest path problems", *European Journal of Operational Research* **53**, 81-92.

J. D. Murchland (1970), *A fixed matrix method for all shortest distances in a directed graph and for the inverse problem*, Ph.D. Thesis, Univ. of Karlsruhe.

K. Nachtigall (1995), "Time depending shortest-path problems with applications to railway networks", *European Journal of Operational Research* **83** (1), 154-166.

S. Nguyen and S. Pallottino (1986), "Assegnamento dei passeggeri ad un sistema di linee urbane: determinazione degli ipercammini minimi", *Ricerca Operativa* **38**, 29-74.

S. Nguyen and S. Pallottino (1988), "Equilibrium traffic assignment for large scale transit networks", *European Journal of Operational Research* **37**, 176-186.

S. Nguyen and S. Pallottino (1989), "Hyperpaths and shortest hyperpaths", in (B. Simeone, ed.) *Combinatorial optimization*, Lecture Notes in Mathematics **1403**, Springer-Verlag, Berlin, 258-271.

A. Orda and R. Rom (1990), "Shortest-path and minimum-delay algorithms in network with time-dependent edge length", *Journal of the ACM* **37** (3), 607-625.

A. Orda and R. Rom (1991), "Minimum weight paths in time-dependent network", *Networks* **21** (3), 295-320.

S. Pallottino (1984), "Shortest path methods: Complexity, interrelations and new propositions", *Networks* **14**, 257-267.

S. Pallottino and M. G. Scutellà (1991), "Strongly polynomial algorithms for shortest paths", *Ricerca Operativa* **60**, 33-53.

S. Pallottino and M. G. Scutellà (1997), "Dual algorithms for the shortest path tree problem", forthcoming in *Networks*.

C. H. Papadimitriou and K. Steiglitz (1982), *Combinatorial optimization*: *Algorithms and complexity*. Prentice Hall, Englewood Cliffs, NJ.

U. Pape (1974), "Implementation and efficiency of Moore-algorithms for the shortest route problem", *Mathematical Programming* **7**, 212-222.

D. Pretolani (1997), private communication.

H. M. Safer and J. B. Orlin (1995), "Fast approximation schemes for multi-criteria combinatorial optimization", Sloan School of Management, Massachusetts Institute of Technology, Working Paper 3756-95.

D. R. Shier and C. Witzgall (1981), "Properties of labeling methods for determining shortest path trees", *Journal of Research of the National Bureau of Standards* **86**, 317-330.

P. A. Steenbrink (1974), *Optimization of transport networks*, J. Wiley, London.

R. E. Tarjan (1983), *Data structures and network algorithms*, SIAM, Philadelphia, PA.

D. Van Vliet (1978), "Improved shortest path algorithms for transport networks", *Transportation Research* **12**, 7-20.

A. R. Warburton (1984), "Bicriterion shortest path problems", University of Ottawa, Working Paper 84-27.

A. R. Warburton (1987), "Approximation of Pareto optima in multiple-objective shortest-path problems", *Operations Research* **35** (1), 70-79.

A. Ziliaskopoulos (1994), *Optimum path algorithms on multidimensional networks*: *Analysis, design, implementation and computational experience*, Ph.D. Thesis, University of Texas at Austin.

A. K. Ziliaskopoulos and H. S. Mahmassani (1993), "Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications", *Transportation Research Record* **1408**, 94-100.

A. K. Ziliaskopoulos and H. S. Mahmassani (1996), "A note on least time path computation considering delays and prohibitions for intersection movements", *Transportation Research B* **30B** (5), 359-367.