Università di Pisa

Dipartimento di Informatica

Technical Report : TR-99-14

A new dual algorithm for shortest path reoptimization

S. Nguyen, S. Pallottino, M.G. Scutella'

July 13, 1999

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

A new dual algorithm for shortest path reoptimization *

Sang Nguyen†
Stefano Pallottino‡
Maria Grazia Scutellà §

Abstract

Shortest path problems are among the most studied network flow problems, with interesting applications in various fields. Mainly in large scale transportation systems, a sequence of shortest path problems needs often to be solved, where the $(k+1)^{th}$ problem differs only slightly from the k^{th} one. In that context, a promising line of research is to study efficient reoptimization approaches, able to exploit useful information from one shortest path computation to the next one.

In this work, we shall focus on the change of the origin node in the shortest path problem to be solved. After reviewing the classical (dual) algorithms described in the literature sofar, which essentially show a Dijkstra's like behavior, a new dual approach will be proposed, which could be particularly promising in practice.

1 Reoptimization approaches in shortest paths

In large scale transportation systems, a sequence of shortest path problems often needs to be solved, where the $(k+1)^{th}$ problem differs only slightly from the k^{th} one.

In the past, each problem in the sequence generally used to be solved from the beginning, independently of the previous one. The first works in the direction of exploiting reoptimization techniques were by Murchland (1970) and Dionne (1978), who faced the problem of finding the shortest cost matrix when the cost of one arc of the input graph was changed. Although of some theoretical interest, the proposed approaches proved to be impractical for the large amount

*Research grants by M.U.R.S.T. progetto cofinanziato 1997, and by C.N.R. G.N.I.M.

†Département d'Informatique et de Recherche Opérationnelle, Pavillon André-Aisenstadt, 2920 chemin de la Tour, Montréal QC H3T 1J4, Canada, e-mail: nguyens@iro.umontreal.ca

‡Dipartimento di Informatica, Università di Pisa,40, Corso Italia, 56125 Pisa, Italy, e-mail: pallo@di.unipi.it

§Dipartimento di Informatica, Università di Pisa,40, Corso Italia, 56125 Pisa, Italy, e-mail: scut@di.unipi.it

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 (1959) for a more efficient updating, is due to Fujishige (1981).

The first efficient reoptimization strategies can be found in [Gallo, 1980], where two general situations arising in transportation models are considered, and clever 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.

Based on these reoptimization strategies, an efficient algorithm is proposed in [Gallo and Pallottino, 1982] for the case of the change of the origin node, which is based on the use of the arc reduced costs instead of the original arc costs (see also [Nemhauser, 1972] and [Lawler, 1976]). Moreover, a dual simplex algorithm is suggested in [Florian, Nguyen and Pallottino, 1981] for the same kind of shortest path reoptimization.

In the following, after formally presenting the problem of finding a new shortest path tree in the case in which the origin node is changed, the most efficient algorithms from the literature, i.e. the reduced cost algorithm and the dual simplex algorithm, will be reviewed. Both algorithms will be presented in terms of dual ascent methods.[1]

Then, a new dual approach will be proposed. Unlike the dual algorithms from the literature, which follow a Dijkstra's like order to insert the nodes into the shortest path tree, the new approach may follow a different (and, hopefully, more efficient) insertion policy.

2 The shortest path reoptimization problem

2.1 The shortest path tree problem

Let $G = (N, A)$ be a 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 $c : A \to R$ be a function which assigns a cost c_{ij} to each $(i, j) \in A$. For the sake of the simplicity, we shall assume G strongly connected, i.e., a directed path exists between any pair of nodes in G.

Given an origin or *root* node $r \in N$, the *shortest path problem* (SP_r) consists of finding a directed tree T_r such that the path from r to i in T_r is one of the minimum cost paths from r to i in G. Observe that, by our previous assumption,

[1] For the sake of the completeness, the more recent auction algorithms in [Bertsekas, Pallottino and Scutellà, 1995] have to be mentioned as well. These algorithms, which belong to the family of the dual methods for standard shortest path computations, might in fact be suitable for the reoptimization problem under consideration. However, auction methods will not be described in this work, since no specialized reoptimization algorithm has been proposed sofar.

T_r is always a spanning tree. In order to guarantee a finite solution to the problem, we shall assume that no directed cycle of negative cost exists in G.

Using a linear programming notation, the problem and its dual can be formulated as follows. In the formulations, $FS(i)$ and $BS(i)$ denote the *forward star* and the *backward star* of i, respectively:

$$FS(i) = \{(u, v) \in A : u = i\}, \quad BS(i) = \{(u, v) \in A : v = i\};$$

moreover $b_r = -n + 1$ and $b_i = 1$, $\forall i \neq r$, whereas π_i denotes the *price* or *potential* of i, i.e., the dual variable associated with i.

$$\begin{array}{lll} (SP_r) \quad min & \displaystyle\sum_{(i,j)\in A} c_{ij} x_{ij} & \\ & \displaystyle\sum_{(j,i)\in BS(i)} x_{ji} - \sum_{(i,j)\in FS(i)} x_{ij} = b_i & \forall i \in N, \\ & x_{ij} \geq 0 & \forall (i,j) \in A \end{array}$$

$$\begin{array}{lll} (DSP_r) \quad max & \displaystyle (1-n)\pi_r + \sum_{j\neq r} \pi_j & \\ & \pi_j - \pi_i \leq c_{ij} & \forall (i,j) \in A \end{array}$$

Given a potential vector π, if we define the *reduced cost* $\bar{c}_{ij}$ of an arc $(i, j) \in A$ as:

$$\bar{c}_{ij} = c_{ij} + \pi_i - \pi_j,$$

then the dual feasibility can be equivalently stated as $\bar{c}_{ij} \geq 0, \forall (i, j) \in A$.

2.2 The reoptimization problem

Assume that a shortest path tree relative to the given origin r, i.e. T_r, has been computed. The reoptimization problem under consideration consists of finding the shortest path tree when the origin changes to a node $s \neq r$.

Two main algorithmic approaches have been proposed in the literature at the beginning of 80's, which will be reviewed in Sections 2.3 and 2.4. Both algorithms rely on the observation that the optimum potentials relative to T_r are feasible to (DSP_s), and so dual-like approaches seem to be promising to compute the required shortest path tree rooted at s.

These algorithms (as well as the new approach, presented in this work) are special cases of a general *reoptimization dual ascent paradigm*, which maintains:

- a (partial) shortest path tree rooted at the new origin s, hereafter denoted by $T_s = (N_s, A_s)$,

- the fragment of T_r which still has to be moved to T_s; with some abuse of the notation, we will denote it as $T_r = (N_r, A_r)$, and its nodes will be denoted as *external nodes*,
- a feasible dual vector $\pi^{(s)}$, which satisfy the complementarity slackness conditions related to (SP_s) and its dual (at the beginning, $\pi^{(s)}$ coincides with the optimum dual vector $\pi^{(r)}$ relative to the previous origin r):

$$\bar{c}_{ij} \geq 0, \quad \forall (i,j) \in A, \quad \text{(dual feasibility)};$$

$$\bar{c}_{ij} = 0, \quad \forall (i,j) \in A_s, \quad \text{(c.s. conditions)};$$

- the dual vector $\pi^{(s)}$ satisfies the complementarity slackness conditions also for the arcs of T_r if the potentials of the nodes belonging to N_r either remained unchanged, or they have been changed by the same value:

$$\bar{c}_{ij} = 0, \quad \forall (i,j) \in A_r, \quad \text{(c.s. conditions on the previous tree)}. \tag{1}$$

In order to present the typical dual ascent iteration, let us introduce some notation, which will be used throughout.

Given an external node $j \in N_r$, partition the set of the arcs entering j into the set of the arcs outgoing from the nodes of N_s, called *border arcs*, and the set of the remaining arcs, called *external arcs*, and denote these arc sets by $BS_B(j)$ and $BS_E(j)$, respectively.

$$BS_B(j) = \{(i,j) \in BS(j) : i \in N_s\};$$

$$BS_E(j) = \{(i,j) \in BS(j) : i \in N_r\}.$$

Similarly, given a subset $S \subseteq N_r$, denote by $BS_B(S)$ and $BS_E(S)$, respectively, the set of the arcs entering the nodes in S coming from N_s and those coming from the complement of S in N_r:

$$BS_B(S) = \{(i,j) \in A : i \in N_s, j \in S\};$$

$$BS_E(S) = \{(i,j) \in A : i \in N_r \backslash S, j \in S\}.$$

Based on this partitioning, define the following values for each subset $S \subseteq N_r$:

$$\delta_S = min\{\bar{c}_{ij} : (i,j) \in BS_B(S)\};$$

$$\eta_S = min\{\bar{c}_{ij} : (i,j) \in BS_E(S)\};$$

$$\gamma_S = min\{\delta_S, \eta_S\};$$

where δ_S and η_S are equal to ∞ if $BS_B(S) = \emptyset$ and $BS_E(S) = \emptyset$, respectively.

If we consider the *cut* $(\bar{S}, S)$ separating S from its complement $\bar{S} = N \backslash S$, and the cutset $A(S) = \{(i,j) \in A : i \in \bar{S}, j \in S\}$ of the arcs entering S, then γ_S, called the *dual gap* of S, is the minimum among the reduced costs of the arcs belonging to $A(S)$. Therefore, given a feasible dual solution, if one wants to modify it by only changing the potentials of the nodes in S, γ_S is the maximum increment to be added to all such potentials in such a way to maintain the dual feasibility, and guarantee that at least a new arc has a reduced cost equal to 0 after the updating.

In the particular case in which $S = \{j\}$ for some external node j, we have:

$$\delta_j = min\{\bar{c}_{ij} : (i,j) \in BS_B(j)\};$$

$$\eta_j = min\{\bar{c}_{ij} : (i,j) \in BS_E(j)\};$$

$$\gamma_j = min\{\delta_j, \eta_j\}.$$

When finite, γ_j and δ_j provide bounds for the optimal solution of (DSP_s), i.e., for the optimum potential associated with node j; in fact, it is possible to prove that $\pi_j^{(s)} - \pi_s^{(s)} + \gamma_j$ is a lower bound, while $\pi_j^{(s)} - \pi_s^{(s)} + \delta_j$ is an upper bound (Pallottino and Scutellà, 1997).

Observe that, by definition, it is:

$$\delta_S = min\{\delta_j : j \in S\}, \quad \forall S \subseteq N_r.$$

In the special case $S = N_r$, it is:

$$\delta_{N_r} = \Delta = min\{\delta_j : j \in N_r\}. \tag{2}$$

According to the previous definition, Δ is the *dual gap* associated with the whole set of external nodes T_r. Therefore, it is the maximum increment to be added to the potentials of all the external nodes in such a way to maintain the dual feasibility and guarantee that at least a new (border) arc has a reduced cost equal to 0 after the updating.

2.2.1 The typical dual ascent iteration

A dual ascent operation is performed at each iteration in order to try to augment T_s: a subset $S \subseteq N_r$ is selected, and the potential of each node in S is increased as much as possible in such a way to maintain the dual feasibility, with the aim of enlarging the set of the arcs with a zero reduced cost, i.e., the arcs which, satisfying the complementarity slackness conditions, can be used to augment T_s.

By the previously introduced definitions, the dual ascent operation thus consists of increasing the potential of each node in S by the amount γ_S:

$$\pi_i^{(s)} = \begin{cases} \pi_i^{(s)} + \gamma_S, & i \in S, \\ \pi_i^{(s)}, & otherwise. \end{cases}$$

Each arc $(i,j) \in BS_B(S)$ which has a zero reduced cost after the updating, if any, is then moved to T_s. [2]

At termination, when the potentials $\pi^{(s)}$, optimal for the root s, have been computed (i.e., when all the external nodes have been moved to T_s), the minimum path costs are given by:

$$C_i = \pi_i^{(s)} - \pi_s^{(s)}, \quad \forall i \in N.$$

A particular case happens when $|S| = 1$, i.e. $S = \{j\}$ for some external node j; in such a case, the dual ascent operation, called a *local operation at node j*, corresponds to increasing the potential of j by the amount γ_j:

$$\pi_j^{(s)} = \pi_j^{(s)} + \gamma_j. \tag{3}$$

Another special case is when $S = N_r$. In this case it is $\gamma_S = \Delta$, where Δ is the dual gap associated with N_r; the dual ascent operation, called *global operation*, becomes:

$$\pi_i^{(s)} = \pi_i^{(s)} + \Delta, \forall i \in N_r. \tag{4}$$

Observe that Δ is a threshold gap for the potentials of the external nodes, in the sense that it is the minimum increment to be added to the potentials of the external nodes to guarantee that at least one of them, i.e., $w = argmin\{\delta_j : j \in N_r\}$, called in the following a *closest node*, can be added to the partial tree T_s.

When $S = N_r$ at each iteration, we have a dual ascent interpretation of the classical Dijkstra's algorithm in the reoptimization context. In fact, the algorithm so obtained is equivalent to solving the shortest path tree problem of root s, by Dijkstra's approach, by using, as the arc costs, the optimum reduced costs obtained by the previous computation (the shortest path tree computation relative to the old root r). Observe that, in this case, the complementarity slackness conditions still hold for the arcs of A_r, as pointed out in (1).

2.2.2 An enhanced dual ascent approach

The above dual ascent operation, and so the entire reoptimization paradigm, can be specialized as follows. Let $w \in N_r$ be a closest node and $T(w) = (N(w), A(w))$ be the subtree of T_r rooted at w; moreover, let (v, w) denote a

[2]The dual ascent operation can be also performed by selecting disjoint sets $S_1, S_2, \ldots, S_h$ of external nodes, and increasing the node potentials in each subset S_i by the corresponding dual gap γ_{S_i}, $i = 1, \ldots, h$.

border arc such that $\bar{c}_{vw} = \Delta$. At the beginning, all the arcs in T_r have a zero reduced cost, and, consequently, all the arcs belonging to $A(w)$ have a zero reduced cost, too. By this property, if one would perform a global operation in the context of the previously introduced reoptimization paradigm, then both w and the entire tree $T(w)$ would be candidates for the movement to T_s (in other words, the minimum cost paths from s to the nodes in $N(w)$ include the arc (v, w)). Unfortunately, only border arcs are moved by the dual ascent operation as previously described.

This suggests to solve the shortest path problem, in an equivalent way, in the partial graph (hereafter called the *reduced graph*) obtained by removing all the external arcs (i, j) such that $i \in N_r \backslash N(w)$ and $j \in N(w)$ (as observed, a shortest path in the reduced graph is also a shortest path in the original graph):

$$A := A \backslash BS_E(N(w)). \tag{5}$$

Now, if we set $S = N(w)$, and apply the dual ascent operation in the reduced graph, we obtain a possible more efficient operation, called the *enhanced dual ascent operation*, which can move the entire subtree $T(w)$ to T_s. In fact, due to the arc reduction, it is $BS_E(N(w)) = \emptyset$, and so $\gamma_{N(w)} = \delta_{N(w)} = \Delta$, i.e., the potential of each node in $N(w)$ can increase by the threshold gap Δ:

$$\pi_i^{(s)} = \begin{cases} \pi_i^{(s)} + \Delta, & i \in N(w), \\ \pi_i^{(s)}, & otherwise. \end{cases} \tag{6}$$

Observe that, by this choice, the potentials of the nodes remaining into N_r do not change. It follows that each arc in T_r maintains a zero reduced cost as pointed out in (1), and the enhanced dual ascent operation can thus be iterated.

The following property can be easily established:

Property 1: The enhanced dual ascent paradigm moves nodes to T_s (by breaking the ties) exactly in the same order as Dijkstra's algorithm.

From an algorithmic point of view, the enhanced paradigm is thus equivalent to Dijkstra's approach in the reoptimization context. However, it can be more efficient due to the fact that only the potentials of the nodes in $N(w)$ increase by Δ at each iteration, instead of all the nodes in N_r, as in the classical approach.

The two main reoptimization algorithms from the literature ([Gallo and Pallottino, 1982] and [Florian, Nguyen and Pallottino, 1981]), described in the following, are both derived from the enhanced dual ascent paradigm.

2.3 The Reduced cost algorithm

Although presented as a primal approach, the reduced cost algorithm is indeed a dual algorithm. In fact, it performs the same operations as the enhanced dual ascent paradigm previously described, where the graph reductions are not performed explicitly (i.e., there is no physical removal of the external arcs).

For efficiency purposes, the key operation of finding the border arc (v, w) entering a closest node w is performed on the set Q of nodes $j \in N_r$ such that their border backward stars $BS_B(j)$ are not empty. So, the computation of the dual gap Δ becomes:

$$\Delta = min\{\delta_j : j \in Q\}.$$

For each $j \in Q$, only one border arc (i, j) is maintained, i.e., an arc (i, j) such that $\bar{c}_{ij} = \delta_j$; the other border arcs entering j are considered removed from the graph; therefore, the search of a closest node w in Q and the computation of Δ can be performed in $O(n)$ time.

Moreover, the computation of values $\delta_j, \forall j \in N_r$, can be performed globally in $O(m)$ time, due to the fact that when a node i moves to T_s, by examining its forward star it is possible to update the value δ_j of each node j in T_r such that an arc (i, j) exists in $FS(i)$.

Finally, in order to move the subtree $T(w)$ to T_s, the algorithm implements the set Q by means of two lists Q' and Q'', the former containing all the nodes with mimimum δ-value and the latter containing the remaining nodes. Once selected a closest node w from Q', the examination of its forward star allows one to set the δ-value of all the sons of w in T_r to a value equal to $\delta_w = \Delta$; these sons will be thus inserted to Q' and, so doing, all the nodes of $T(w)$ will be moved to Q' and therefore to T_s.

Since the reduced cost algorithm performs the same operations as the enhanced dual ascent approach, it can be classified as an efficient "ad hoc" implementation of Dijkstra's algorithm in the reoptimization context, according to Property 1.

2.4 The Dual simplex algorithm

This algorithm specializes the classical dual simplex algorithm to the reoptimization problem under investigation, based on the observation that the optimum tree T_r is a primal unfeasible but dual feasible basis when the root changes to s, and a dual simplex approach seems thus to be convenient to find the new optimum tree T_s.

Given the current dual feasible basis, at each iteration the most negative valued basic arc, which leaves the basis, is found 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 divides the tree into two arborescences that, with the same abuse of terms previously adopted, we call $T_r = (N_r, A_r)$ and $T_s = (N_s, A_s)$. The entering border arc is an arc (v, w), belonging to the cut (N_s, N_r) and oriented from N_s to N_r, having the minimum reduced cost, i.e., such that $\bar{c}_{vw} = \delta_w = \Delta$ (w is a closest node). From a practical point of view, as in the reduced cost approach, such a minimum reduced cost arc is computed by maintaining only one border arc for each node j in Q and by updating the δ-values by scanning the forward stars of the nodes entering T_s.

As in the reduced cost approach, the arc reduction (5) is (implicitly) performed, and the subtree $T(w) = (N(w), A(w))$, rooted at w, is moved to T_s. However, unlike the former approach, here the use of more sophisticated data structures describing T_r, such as the "first son", the "younger brother" and "older brother" functions, allows one to explicitly move the entire $T(w)$, and to traverse it in order to update the node labels as in (6).

Observe that, as in the reduced cost case, the potential updating is exactly the enhanced dual ascent operation. Therefore, also the dual simplex algorithm, too, can be viewed as an efficient implementation of Dijkstra's approach in the reoptimization context.

3 A new dual algorithm for shortest path reoptimization

The new reoptimization algorithm is strictly related to a recently introduced family of dual ascent shortest path algorithms, called the *hanging family* (Pallottino and Scutellà, 1997).[3]

Although the hanging family seems to be suitable for the kind of reoptimization under consideration, all the algorithms tend to collapse to the two reoptimization algorithms previously reviewed, i.e., by Property 1, to a Dijkstra's like behavior, as shown in [Pallottino and Scutellà, 1998]. "Ad hoc" dual ascent operations are thus necessary to devise an original dual ascent reoptimization approach.

In the following some basic properties will be presented, which allow one to suitably prune the graph and discover subtrees which can be moved in parallel to T_s, thus extending the enhanced dual ascent paradigm previously described. Such properties will be at the basis of the new reoptimization approach.

3.1 Basic properties

Consider the typical dual ascent iteration in the reoptimization context, where both $T_s = (N_s, A_s)$ and $T_r = (N_r, A_r)$ are described by a predecessor function p, i.e. $i = p(j)$ means that the arc (i, j) belongs either to A_s or to A_r. For each external node $j \in N_r$, $j \neq r$, let $T(j) = (N(j), A(j))$ denote the subtree of T_r rooted at j.

Given the current feasible dual vector $\pi^{(s)}$, for each external node $j \neq r$ define:

$$\alpha_j = min\{\bar{c}_{ij} : (i, j) \in BS_E(j), i \neq p(j)\}, \qquad (7)$$

[3] In the hanging family, two basic dual ascent operations are performed, which can be viewed as particular cases of the dual ascent operation introduced for the shortest path reoptimization problem (more sophisticated operations have been proposed, which mix in different ways the two basic ones): the *local operation* (3), performed at a single external node, and the *classic operation*, which corresponds to (4).

where $\alpha_j = \infty$ if $BS_E(j) \setminus \{(p(j), j)\} = \emptyset$. When finite, α_j is the minimum increment to be added to the potential of j, due to the external arcs, if we would ignore the critical arc $(p(j), j)$, which, by the complementarity slackness conditions related to the tree T_r, has a zero reduced cost. In the case of the root r, let $\alpha_r = \eta_r$.

Furthermore, for each external node $j \neq r$, introduce the values:

$$\delta_j^* = min\{\delta_j, \delta_{p(j)}^*\}; \tag{8}$$

$$\alpha_j^* = min\{\alpha_j, \alpha_{p(j)}^*\}, \tag{9}$$

where $\delta_r^* = \delta_r$ and $\alpha_r^* = \alpha_r$.

δ_j^* (when finite) is the minimum increment to be added to the potential of j due to the border arcs entering j (δ_j) and due to the border arcs entering the ancestors of j in T_r (recall that the paths from such ancestors to j have a zero reduced cost). On the other hand, α_j^* (when finite) is the minimum increment to be added to the potential of j due to the external arcs entering j and the ancestors of j in T_r.

To better explain the above introduced values, consider the graph and the shortest path tree rooted at $r = 1$ in figure 1, where the optimal potentials are associated with the nodes.

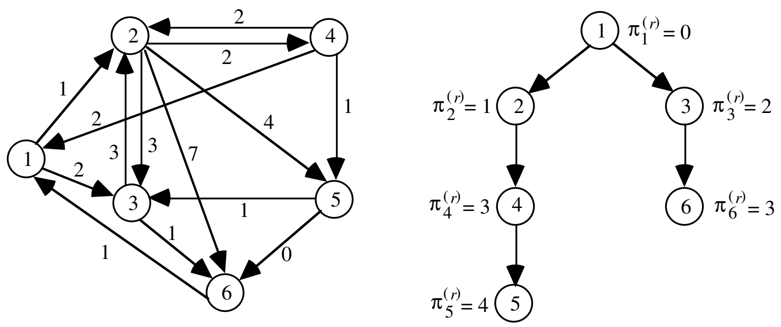

Figure 1: The graph G and the shortest path tree T_r

Let $s = 4$ be the new root; the trees T_r and T_s are depicted in figure 2, together with the border arcs, their reduced costs and the values $\delta_j, \forall j \in N_r$. It is $\Delta = min\{\delta_j : j \in N_r\} = 1$.

The values $\alpha_j, j \in N_r$, computed according to (7), are shown in figure 3. Furthermore, the values δ_j^* and $\alpha_j^*, j \in N_r$, defined according to (8) and (9), respectively, are: $\delta_1^* = 5$, $\delta_2^* = 4$, $\delta_3^* = 3$ and $\delta_6^* = 1$; $\alpha_1^* = 4$, $\alpha_2^* = 4$, $\alpha_3^* = 2$ and $\alpha_6^* = 2$.

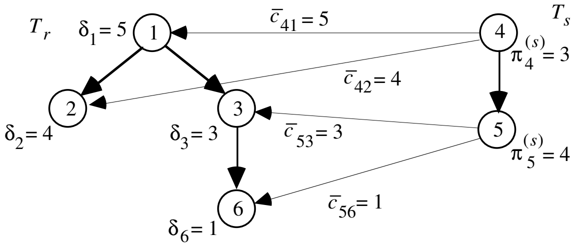

Figure 2: The trees T_r and T_s and the border arcs

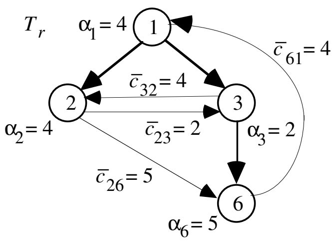

Figure 3: The values $\alpha_j, j \in N_r$

Based on the above defined values, the following properties can be proved.

Property 2: $\delta_j^* + \pi_j^{(s)} - \pi_s^{(s)}$ is an upper bound to the shortest path length from s to j; when α_j is finite, $\alpha_j + \Delta + \pi_j^{(s)} - \pi_s^{(s)}$ is a lower bound to the length of the paths from s to j which enter node j through an external arc other than $(p(j), j)$.

Such a property follows easily from the definitions of δ_j^*, α_j^* and Δ.

Property 3: If $\delta_j^* < \delta_j$ for some external node j, then all the border arcs entering j can be (implicitly) removed from G.

Proof: By Property 2, $\delta_j^* + \pi_j^{(s)} - \pi_s^{(s)}$ is an upper bound to the shortest path length from s to j; indeed, it is the length of a path of G from s to j going through an ancestor of j in T_r. If $\delta_j^* < \delta_j$, it means that the path of G, from s to j, having as its last arc the border arc providing the value δ_j, has a lenght greater than the one having lenght $\delta_j^* + \pi_j^{(s)} - \pi_s^{(s)}$. All the border arcs can thus be removed from G, since they provide a tighter upper bound for j. $\diamondsuit$

Due to Property 3, only the external nodes j such that $\delta_j = \delta_j^*$, hereafter denoted as *effective candidate nodes*, need to maintain a border arc. One consequence is that, if one needs to compute value Δ, it is sufficient to consider the set $Q^* = \{j \in N_r : \delta_j^* = \delta_j\}$ instead of Q. Clearly, such a kind of computation of value Δ can be effective from a practical point of view.

Property 4: If $\alpha_j + \Delta \geq \delta_j^*$ for some node $j \in N_r$, then all the external arcs (i, j), $i \neq p(j)$ (if $j \neq r$), can be (implicitly) removed from G.

Proof: By Property 1, when α_j is finite, $\alpha_j + \Delta + \pi_j^{(s)} - \pi_s^{(s)}$ is a lower bound to the length of the paths from s to j which enter node j through an external arc other than $(p(j), j)$; on the other hand, $\delta_j^* + \pi_j^{(s)} - \pi_s^{(s)}$ is an upper bound to the shortest path length from s to j. Condition $\alpha_j + \Delta \geq \delta_j^*$ thus implies that all the external arcs other than $(p(j), j)$ can be removed from G.$\diamondsuit$

Now, let us introduce a further graph reduction, involving the arcs of T_r.

Property 5: If $\alpha_j^* + \Delta \geq \delta_j$ for an effective candidate node $j \in Q^* \backslash \{r\}$, then also $(p(j), j)$ can be removed from G.

Proof: By definition, α_j^* is the minimum increment to be added to the potential of j due to the external arcs entering j and the other ancestors of j in T_r. Therefore, when α_j is finite, $\alpha_j^* + \Delta + \pi_j^{(s)} - \pi_s^{(s)}$ is a lower bound to the length of all the paths from s to j which enter node j through an external arc, including the tree arc $(p(j), j)$. By Property 2, $\delta_j^* + \pi_j \pi_j^{(s)} - \pi_s^{(s)}$ is an upper bound to the shortest path length from s to j; moreover, being j effective, $\delta_j + \pi_j^{(s)} - \pi_s^{(s)}$ is the length of a path of G having the border arc entering j as its last arc. By combining all these properties, $\alpha_j^* + \Delta \geq \delta_j$ implies that all the external arcs entering j can be removed from G, including $(p(j), j)$. $\diamondsuit$

Observe that, if tree arc removals are performed, T_r may become a forest.

An interesting consequence of Property 5 is that any effective candidate node j satisfying Property 5 can be hanged on T_s via the border arc (i, j), which is indeed the only arc entering j. This can happen even if $\delta_j > \Delta$. Unfortunately, at least in general, the descendants of j in T_r, i.e., the entire subtree $T(j)$, can not move to T_s, since it is possible that the minimum cost path from s to some descendants of j does not pass through node j.

As an example, consider the previous graph. There, all the external nodes are effective candidate nodes, i.e., $Q^* = \{1, 2, 3, 6\}$. Since $\alpha_3^* + \Delta = 2 + 1 = 3 = \delta_3$, then all the external arcs entering 3 can be removed. It follows that the optimal potential of node 3 is $\pi_3^{(s)} = \pi_3^{(s)} + \delta_3 = 2 + 3 = 5$. However, $\pi_6^{(s)} + \delta_6 = 3 + 1 < \pi_3^{(s)} = 5$, and consequently $T(3)$ cannot move to T_s.

Stronger conditions have thus to be discovered to allow the movement of entire subtrees.

Definition 1: An effective candidate node j is said to be *independent* if $\delta_i \geq \delta_j$, for each descendant $i \in N(j)$; in other words, defined

$$\bar{\delta}_j = min\{\delta_i : i \in N(j)\},$$

it must be $\bar{\delta}_j = \delta_j$.

Definition 2: A subtree $T(j)$ rooted at an independent node $j \in N_r$ is said to be *hangable* if j satisfies Property 5 and if $\alpha_i + \Delta \geq \delta_j$ for each proper descendant $i \in N(j)$; in other words, defined

$$\bar{\alpha}_j = min\{\alpha_i : i \in N(j)\},$$

it must be $\bar{\alpha}_j + \Delta \geq \delta_j$.

Theorem 1: Let $T(j)$ be a hangable subtree; then the optimum potential of each node $i \in N(j)$ is $\pi_i^{(s)} = \pi_i^{(s)} + \delta_j$.

Proof: Since Property 5 is satisfied, all the external arcs entering j can be removed: therefore $\pi_j^{(s)} + \delta_j$ is the optimum potential of node j.

Since j is independent, i.e., $\bar{\delta}_j = \delta_j$, the path going through the (unique) border arc entering j is the best one among the paths of G, ending with a border arc, which go to a proper descendant of j (recall that all the arcs of T_r have a zero reduced cost). All the border arcs entering proper descendants of j can thus be removed from G.

Moreover, since $\alpha_i + \Delta \geq \delta_j$ for each proper descendant $i \in N(j)$, also all the external arcs entering the proper descendants of j, with the exception of the arcs of $T(j)$, can be removed from G.

It follows that the border arc entering j is the only arc linking the entire subtree $T(j)$ to T_s. The thesis thus follows.$\diamondsuit$

Corollary 1: All the hangable subtrees $T(j)$ can simoultaneously move to T_s; the optimum potential of each node $i \in N(j)$ is obtained by increasing the current potential value $\pi_i^{(s)}$ by the amount δ_j:

$$\pi_i^{(s)} = \pi_i^{(s)} + \delta_j, \forall i \in N(j). \tag{10}$$

3.2 The algorithm

The properties stated in Section 3.1 suggest to consider the subset $\bar{Q}$ of nodes $j \in Q^*$ such that:

$$\delta_j \leq min\{\bar{\delta}_j, \alpha_j^* + \Delta, \bar{\alpha}_j + \Delta\}. \tag{11}$$

$\bar{Q}$ defines in fact the *roots of the hangable subtrees*, i.e. it specifies subtrees which can simultaneously move to T_s.

The typical iteration of the new reoptimization algorithm is based on the discovering of such a set $\bar{Q}$.

Typical iteration

- compute all the values δ_j, δ_j^*, $\bar{\delta}_j$, α_j, α_j^*, $\bar{\alpha}_j$, for each node $j \in N_r$;
- compute the set $Q^* = \{j \in N_r : \delta_j = \delta_j^*\}$ and the value $\Delta = min\{\delta_j : j \in Q^*\}$;

- for each node j satisfying conditions (11), i.e., for each root of a hangable subtree, move the subtree $T(j)$ to T_s and update the potentials as in (10).

Observe that at least a closest node w satisfies conditions (11); so, $T(w)$ is hangable by construction and always moves to T_s. In the worst case, exactly one subtree moves to T_s at each algorithm iteration, i.e. $T(w)$. In such a case, the proposed algorithm behaves like the two reoptimization algorithms previously described. In other words, it shows, too, a Dijkstra's like behavior.

However, it may happen that, during some algorithm iterations, more subtrees move in parallel to T_s. In such a case, the new reoptimization algorithm is different from the classical reoptimization algorithms from the literature. In particular, subtrees $T(j)$ requiring an increasing of their potential by an amount $\delta_j > \Delta$ can move to T_s. When this happens, the new reoptimization algorithm extends the enhanced dual ascent operation on a collection of disjoint node sets.

To illustrate the behavior of the proposed approach, consider the example discussed before, where the cost of the arc $(5, 6)$ changes from 0 to 3. The values associated with the external nodes are reported in the following table. It is $Q^* = \{1, 2, 3\}$ and $\Delta = 3$. Since $\delta_1^* = 5 > 3 = \bar{\delta}_1$, node 1 is not the root of a hangable subtree.

out of tree nodes	1	2	3	6
δ_j	5	4	3	4
α_j	4	4	2	5
δ_j^*	5	4	3	3
α_j^*	4	4	2	2
$\bar{\delta}_j$	3	4	3	4
$\bar{\alpha}_j$	2	4	2	5

On the other hand $T(3)$ is hangable, since $\delta_3 = \Delta$. In this example $T(2)$ is hangable too, since $\delta_2 \leq min\{\bar{\delta}_2, \alpha_2^* + \Delta\}$, although it is $\delta_2 = 4 > 3 = \Delta$. $T(2)$ and $T(3)$ can thus move in parallel to T_s.

The trees T_r and T_s, after the movement of $T(2)$ and $T(3)$ to T_s, are illustrated in figure 4.

At the next iteration, $(4, 1)$ and 1 enter T_s, and $\pi_1^{(s)}$ is set to value 5. The shortest path tree rooted at s is shown in figure 5, togheter with the minimum costs $C_i = \pi_i^{(s)} - \pi_s^{(s)}$ associated with each node i in T_s.

In order to complete the reoptimization algorithm presentation, let us specify now how to efficiently discover the hangable subtrees during each algorithm iteration.

Consider a typical iteration. At the beginning, compute the values δ_j and α_j, $\forall j \in N_r$. Moreover, set $\Delta = \delta_r$, $Q^* = \emptyset$, and $\bar{\alpha}_j = \alpha_j$ and $\bar{\delta}_j = \delta_j$, $\forall j \in N_r$; finally, set $\alpha_r^* = \alpha_r$ and $\delta_r^* = \delta_r$ (*initialization step*).

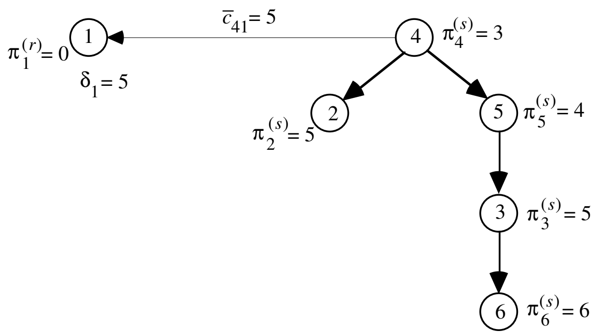

Figure 4: The new trees T_r and T_s

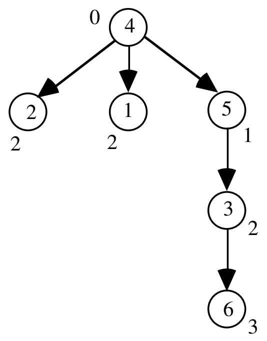

Figure 5: The shortest path tree rooted at s

After the initialization step, visit T_r in a depth-first way, starting from r, and perform the following operations for each visited node $j \neq r$:

- if j is visited in the descending phase, then set $\alpha_j^* = min\{\alpha_j, \alpha_{p(j)}^*\}$ and $\delta_j^* = min\{\delta_j, \delta_{p(j)}^*\}$;
  if $\delta_j = \delta_j^*$, then $Q^* = Q^* \cup \{j\}$ and $\Delta = min\{\Delta, \delta_j\}$;
- if j is visited in the ascending phase, then set $\bar{\alpha}_{p(j)} = min\{\bar{\alpha}_{p(j)}, \bar{\alpha}_j\}$ and $\bar{\delta}_{p(j)} = min\{\bar{\delta}_{p(j)}, \bar{\delta}_j\}$.

At the end of the visit of T_r, if $\delta_r = \Delta$ then the entire forest T_r moves to T_s, and the computation ends; otherwise, repeat the following operations until $Q^* = \emptyset$:

- select and remove a node $j \in Q^*$;
- if j still belongs to T_r and if j satisfies conditions (11), then move $T(j)$ to T_s; moreover, update the potentials as in (10).

Theorem 2: (Algorithm correctness and time complexity) The reoptimization algorithm correctly finds the shortest path tree T_s in $O(mn)$ time.

Proof: The algorithm correctness follows immediately from Theorem 1, since only hangable subtrees are moved during each algorithm iteration.

As far as the time complexity is concerned, observe that the algorithm performs at most n iterations, since at each iteration at least a closest node w moves to T_s together with the corresponding subtree $T(w)$.

Consider now a typical iteration. During all the initialization steps, the computation of values $\delta_j, \forall j \in N_r$, can be performed globally in $O(m)$ time, as shown for the reduced cost algorithm. On the other hand, the computation of values $\alpha_j, \forall j \in N_r$, requires $O(m)$ time at each algorithm iteration, since in the worst case the backward star of each node in G has to be analysed.

The other initialization operations can be performed in $O(n)$ time. The depth-first visit of T_r and the construction of Q^* require $O(n)$ time, too.

Finally, the movement of the hangable subtrees requires globally $O(n)$ time. The thesis follows.$\diamondsuit$

Due to the computational cost required by each algorithm iteration, and in particular by the computation of values $\alpha_j, \forall j \in N_r$, it is possible to think at some relaxed versions of the reoptimization approach, using lower bounds of $\alpha_j, \forall j \in N_r$, instead of the right values (observe that, if lower bounds are used, and if a node $j \in Q^*$ is discovered during an algorithm iteration which satisfies conditions (11), then j is certainly the root of a hangable subtree; in other words, the algorithm correctness is surely preserved). In particular, if values $\alpha_j, \forall j \in N_r$, are updated only in k prespecified iterations, and the old values are maintained in the other iterations, then the algorithm time complexity becomes $O(n^2 + km)$. By suitably choosing k, better time complexity figures can thus be derived.

We suggest to consider $k = 1$, i.e. to compute the α's only in the initialisation phase, and $k = \lceil n^2/m \rceil$: in both cases the algorithm time complexity reduces to $O(n^2)$; clearly, the reduced effort due to the computation of values α_j could be effective also from a practical point of view.

A final remark. To obtain a non-Dijkstra's like behavior, a necessary condition is $\alpha_j^* > 0$ for at least one external node j. Such a condition requires, in particular, that value $\alpha_r > 0$.

Observe that $\alpha_r = 0$ means that there exists an external arc entering r, say (i, r), which has a zero reduced cost. In other words, there exists a zero cost cycle in G, say C, which goes through the old root node r. When this happens, we suggest to shrink C into a new node, say R, which takes the place of the old root r. If the α-value associated with the new root R is again 0, i.e. $\alpha_R = 0$, then the shrinking operation can be iterated, until the tree to be hanged has a root with a nonzero α-value.

It is easy to show that the proposed reoptimization algorithm is still correct in the case in which shrink reduction operations are performed. Furthermore, the time complexity figures stated in Theorem 2 and in the following observations remain valid: in fact, all the shrink reduction operations can be performed in $O(n)$ time, and the inverse operations, i.e., the cycle unfolding after the algorithm termination, can be performed in $O(n)$ time, too (we refer to [Ahuja, Magnanti and Orlin, 1993] and [Tarjan, 1983] for details about cycle shrinking and their inverse operations).

4 References

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

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

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

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

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

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

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

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

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

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

Nemhauser, G. (1972). "A generalized permanent label setting algorithm for the shortest path between specified nodes", *Journal of Mathematical Analysis and Applications*, 38, 328–334.

Pallottino, S. and Scutellà, M. G. (1997). "Dual algorithms for the shortest path tree problem", *Networks*, 29, 125–133.

Pallottino, S. and Scutellà, M. G. (1998). "Shortest path algorithms in transportation models: classical and innovative aspects". In (Marcotte, P. and Nguyen, S., Eds.) *Equilibrium and advanced transportation modelling*, Kluwer, Boston, 245–281.

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