Technical Report: TR-96-03, March 01, 1996

Dual Algorithms for the Shortest Path Tree Problem

Pallottino,Stefano; Scutella',M.Grazia

Dipartimento di Informatica, Università di Pisa
corso Italia 40, 56125 Pisa, Italy

Abstract

We consider dual approaches for the Shortest Path Tree Problem. After a brief introduction to the problem, we review the most important dual algorithms which have been described in the literature for its solution, and propose a new family of dual ascent algorithms. In these algorithms, "local" and "global" dual updating operations are performed at the nodes in order to enlarge a partial shortest path tree, which at the beginning contains only the root node, until a shortest path tree is found. Several kinds of dual updating operations are proposed, which allow one to derive different dual algorithms from a general schema. One of them, in particular, which is based only on global operations, can be viewed as a dual interpretation of Dijkstra's classical algorithm. Due to their structure, all the proposed approaches are suitable for parallel implementations. They are also suitable for reoptimization approaches, when the computation of shortest paths from different root nodes is required.

ACM–CR Subject Classification: G.2.2 Network problems; F.2 Analysis of algorithms and problem complexity

Keyword and phrases: Shortest paths, dual algorithms

Introduction

The Shortest Path Tree problem (SPT) is the easiest and one of the most studied network flow problems, for which an enormous number of algorithms have been proposed (some are important for their computational time complexity; others for their practical efficiency). Yet most of these algorithms can be seen as a particular instantiation of a unique, general, algorithmic schema, which is an adaptation of the primal network simplex approach (Gallo&Pallottino, 1986). On the other hand, very little effort has been dedicated to the study of dual-like algorithms for SPT. For instance, it is well known that Dijkstra's algorithm (1959), which is in general presented as a primal-like algorithm, can be interpreted both as a dual and a primal-dual approach (Papadimitriou&Steiglitz, 1982). However, after the pioneering paper by Bazaraa and Langley (1974), a limited number of dual algorithms have been proposed. In particular, a dual-simplex algorithm to solve a sequence of SPT problems differing in the root node can be found in [Florian, Nguyen&Pallottino, 1981]. More recently, Bertsekas specialised his Auction algorithm, which really is dual ascent, to solve SPT (Bertsekas, 1991a, 1991b). In [Pallottino&Scutellà, 1991; Bertsekas, Pallottino&Scutellà, 1995], it has been proved that these Auction algorithms can be implemented so as to be strongly polynomial.

Here, we propose a new family of dual algorithms for SPT. Given a dual feasible solution π and a partial shortest path tree T (which, at the beginning, coincides with the root node), the algorithms augment T, step by step, by looking for arcs, going into the nodes out of T, which satisfy the complementarity slackness conditions with respect to π, and whose tail belongs to T. For any node j which is out of T, if an entering arc is found which satisfies the above properties, then j is added to T through the discovered arc; otherwise, dual updating operations are proposed which allow one to connect at least one new node to T. Most of the algorithms in the family run in $O(mn)$ time. If the outgoing arcs of each node are properly pre-sorted, an algorithm running in $O(n^2)$ time can be derived. All the proposed algorithms are suitable for parallel implementations. They are also suitable for the problem of the shortest path tree reoptimization, i.e. when the computation of shortest paths from different root nodes is required.

Before presenting the algorithm family, we would like to point out that this paper is essentially methodological. In particular, the computational investigation of the proposed algorithms will be the subject of a future work, where the dual ascent algorithms will be used in a shortest path tree reoptimization framework.

The paper is organised as follows. In Section 2, the classical dual algorithms are reviewed, and their main properties are analysed. In Section 3, the new family of dual algorithms is presented, together with the correctness proofs, while the relationship among the algorithms is described in Section 4. Finally, some implementation issues and the time complexity analysis are reported in Section 5.

2. Classical dual algorithms for SPT

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 $l: A \rightarrow R$ be a function which assigns a *length* $l(i,j)$ to each $(i,j) \in A$. Given a *root* $r \in N$, the *Shortest Path Tree* problem 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 (Lawler, 1976; Gallo&Pallottino, 1986, 1988; Ahuja, Magnanti&Orlin, 1993), a finite solution exists for SPT if and only if there is no directed cycle of negative length 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, $FS(i) = \{(h,j) \in A: h=i\}$ and $BS(i) = \{(j,h) \in A: h=i\}$ denote the forward star and the backward star of i, respectively.

$$(SPT) \quad \min \sum_{(i,j)\in A} l(i,j)x(i,j)$$

$$\sum_{(j,i)\in BS(i)} x(j,i) \; - \sum_{(i,j)\in FS(i)} x(i,j) = b(i), \quad \forall\, i \in N,$$

$$x(i,j) \geq 0, \qquad \forall\, (i,j) \in A,$$

where $b(r)= -n + 1$ and $b(i) = 1$, $\forall\, i \neq r$;

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

$$\pi(j) - \pi(i) \leq l(i,j), \qquad \forall\, (i,j) \in A,$$

where $\pi(i)$ is the *price* or *potential* of i, i.e. the dual variable which is associated with i.

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 vector d, which represents either the distances of the nodes from r along the tree, or an upper bound for these distances, is a dual feasible solution, i.e.

$$d(j) \leq d(i) + l(i,j), \qquad \forall\, (i,j) \in A.$$

Although these algorithms can be interpreted as procedures for solving the *non-linear Bellman equation system*:

$$\pi(r) = 0,$$
$$\pi(j) = \min\{\pi(i) + l(i,j): (i,j) \in BS(j)\}, \;\; \forall\, j \in N \setminus \{r\},$$

the dual feasibility generally holds only at termination (Gondran&Minoux, 1984). On the other hand, few of the algorithms that have appeared so far maintain the dual feasibility at each step. The ancestor of this algorithm family is the classical *dual simplex algorithm*, which works on dual feasible basic solutions. Efficient adaptations of this algorithm have been proposed, which compute the shortest paths on G from different root nodes. The dual simplex method is in fact adapted to the reoptimization of the shortest path tree as follows. Given the optimum tree T, rooted at node r, the change of the root to a different node s maintains T dual feasible but renders it primal unfeasible for the new problem. By exploiting the dual feasibility of T, the dual simplex method is then easily and efficiently tailored to solve the new problem starting from T (Florian, Nguyen&Pallottino, 1981).

Another pioneering work in the field of dual algorithms for SPT is the one by Bazaraa and Langley (1974), which looks for a dual feasible solution in the case of negative arc lengths (otherwise, $\pi = 0$ is a dual feasible solution). Starting from the price vector $\pi = 0$, Bazaraa&Langley's procedure updates π in an iterative way, node by node, as follows: if $l(i,j) + \pi(i) - \pi(j) < 0$, then $\pi(j) := \pi(j) + \min\{l(i,j) + \pi(i) - \pi(j): (i,j)\in BS(j)\}$, $j = 1, 2,..., n$.

This step is repeated at most n times, until either the dual feasibility is satisfied or a negative cycle is discovered.

More recently, a dual ascent algorithm has been described, which is an adaptation of Bertsekas's *Relaxation method* (Bertsekas&Tsitsiklis, 1989) to SPT. This algorithm considers the Dual Lagrangean problem, and tries to maximise 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:

$$\pi(j) := \min\{\pi(i) + l(i,j): (i,j) \in BS(j)\}, \qquad \forall\, j \in N \setminus \{r\}.$$

If the initial price vector π is dual feasible for the Linear Dual problem (*DSPT*), i.e. it satisfies $\pi(j) - \pi(i) \leq l(i,j)$, $\forall\ (i,j) \in A$, 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 length; otherwise, the algorithm may return a wrong solution. In any case, the method can be interpreted as the Gauss-Seidel technique when applied to solve the non-linear Bellman equation system.

Another dual ascent algorithm was introduced by Bertsekas (1991a), by specialising to the Shortest Path Problem the *Auction approach* for the Assignment problem (Bertsekas, 1979). 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 length, 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.

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

If P reaches the destination t, i.e. a primal feasible solution has been found, P is thus one of the shortest paths from r to t.

At each step, P and π are updated by means of the following *extension* and *contraction* operations on the end node of P, say node i:

a) if there is an edge (i,j) such that $\pi(j) = \pi(i) + l(i,j)$, then P is *extended* by inserting (i,j), i.e. $P := P \cup \{(i,j)\}$;

b) otherwise, $\pi(i)$ is decreased to the value $\max\{\pi(h) - l(i,h): (i,h) \in FS(i)\}$ and, if $i \neq r$, P is *contracted* by removing its last edge.

As proved in [Bertsekas, 1991a], the Auction algorithm terminates, and it runs in pseudopolynomial time in the case of integer arc lengths. Under mild conditions, a weak polynomial version can be obtained using length scaling. The algorithm can be immediately extended to solve SPT. In fact, in this case, Bertsekas suggests continuing the iterations of the algorithm until each node has become the end node of P at least once. In [Pallottino&Scutellà, 1991; Bertsekas, Pallottino&Scutellà, 1995], strongly polynomial versions of the Auction algorithm were proposed, which return a shortest path tree without the requirement that the cycle lengths must be positive, and without assumptions on the topology of the graph, thus generalising the Auction method for shortest paths. These algorithms "prune" the graph by means of *reduction* and *blocking* operations, until it "collapses" into a shortest path tree. A *reduction* operation removes one or more arcs from the graph G, thus obtaining a partial graph G' such that any shortest path tree in G' is a shortest path tree in G as well. A *blocking* operation prevents a node i

from becoming the end node of the candidate path P when no new nodes can be added to the shortest path tree through a path containing i. The resulting algorithms run in $O(m^2)$, $O(mn)$, $O(n\min\{m, n\log n\})$ or in $O(n^2)$ time, depending on the implementation of the *extension*, *contraction*, *reduction* and *blocking* operations.

Backward versions of the Auction algorithm have been proposed as well (Bertsekas, 1991a, 1991b). By considering the single destination problem, a candidate path P' ending at t is maintained, which is gradually extended backwards. When r becomes the origin node of P', then a shortest path from r to t is found. Also, combined forward/backward algorithms have been proposed, which maintain both a candidate path P outgoing from r and a candidate path P' entering t: when the two paths meet, i.e. the ending node of P is the origin node of P', then a shortest path is found.

3. A new family of dual ascent algorithms

Here, a new family of dual ascent algorithms is proposed, which is strictly related to the adaptation of the Relaxation method proposed by Bertsekas. The idea is to maintain a *partial shortest path tree* rooted at node r, $T = (N_T, A_T)$, and a feasible dual vector π, which satisfy the complementarity slackness conditions related to (*SPT*) and its dual, which, for the sake of presentation, are rewritten below:

(i) $\quad \pi(j) \le \pi(i) + l(i,j), \ \forall\ (i,j) \in A$ $\quad$ (dual feasibility);

(ii) $\quad (i,j) \in A_T \Rightarrow \pi(j) = l(i,j) + \pi(i)$ $\quad$ (c.s. conditions).

At the beginning, T contains only the root r, i.e. $N_T = \{r\}$ and $A_T = \emptyset$, and π is any vector satisfying condition (i). In order to try to augment T, each node $j \notin T$, called an *out of tree node*, looks for an entering arc (i,j) such that $\pi(j) = l(i,j) + \pi(i)$ and $i \in N_T$.

Firstly, assume that an arc (i,j) of this type is found; then, j is inserted to T through (i,j).

More formally, the following *link* operation is performed:

$$link(j,T): \qquad N_T := N_T \cup \{j\}, \ A_T := A_T \cup \{(i,j)\}.$$

In the above case, the following property holds true:

Property 3.1- *If $j \notin N_T$ becomes connected to the partial shortest path tree T, then a shortest path from r to j is found.*

Proof:

Let (i,j) be the arc entering j which satisfies $\pi(j) = l(i,j) + \pi(i)$, with $i \in N_T$. Consider the path P_i which connects r to i in T, and the path $P_j = P_i \cup \{(i,j)\}$ which links r to j after the insertion of j and (i,j) into T. Since condition (ii) is maintained, then $\pi(v) = l(w,v) + \pi(w)$, $\forall\ (w,v)$ along P_j. Hence, since the dual feasibility is maintained as well, P_j is shortest for j. ♦

Consider now the case in which no arc (i,j), with $j \notin N_T$ and $i \in N_T$, verifies $\pi(j) = l(i,j) + \pi(i)$. In this case, the following *local dual updating operation* along the cut $(N \setminus \{j\}, \{j\})$ is performed at node j, in order to try to link j to T: $\pi(j) := \min\{\pi(i) + l(i,j): (i,j) \in BS(j)\}$.

Property 3.2- *The local dual updating operation at node j maintains* π *dual feasible*; *moreover, at least one arc* (i,j) *entering j satisfies* $\pi(j) = l(i,j) + \pi(i)$.

Proof:

The updated dual variable $\pi(j)$ satisfies:

$$\pi(j) \le \pi(i) + l(i,j), \qquad \forall\ (i,j) \in BS(j),$$
$$\pi(k) \le \pi(j) + l(j,k), \qquad \forall\ (j,k) \in FS(j).$$

Thus, π remains feasible. Moreover, if $(i,j) = \text{argmin}\{\pi(i) + l(i,j)\colon (i,j) \in BS(j)\}$, then $\pi(j) = l(i,j) + \pi(i)$ holds for arc (i,j) after the local dual updating. ♦

More formally, the following operations are performed at each out of tree node j in order to try to link it to T:

findmin(j, γ_j): $\quad \gamma_j = \min\{\pi(i) + l(i,j)\colon (i,j) \in BS(j)\}$;
local_reprice(j, γ_j): $\quad \pi(j) := \gamma_j$,

until a *link* operation, connecting j to T, can be performed.

Hereafter, *local phase* on an out of tree node j shall denote a sequence of *findmin*, *local_reprice* and, if possible, *link* operations at j. Clearly, the following property holds:

Property 3.3- *Each price vector computed during a local phase is feasible*, *and the prices can only increase*.

A first dual ascent algorithm can now be suggested, based on local phases performed at each out of tree node. This algorithm is well suited to parallel implementations, since each local phase can be performed in parallel at each node out of the tree. The algorithm, called *Local_Dual*, performs the following typical iteration:

Typical iteration of Local _Dual

for each $j \notin N_T$ **do**
  begin
    findmin(j, γ_j); *local_reprice*(j, γ_j);
    if $\exists\ i \in N_T$ **such that** $\pi(i) + l(i,j) = \pi(j)$ **then** *link*(j,T)
  end.

Local _Dual is essentially an implementation of the Relaxation method for the Shortest Path problem (Bertsekas&Tsitsiklis, 1989; section 5.2.1). In particular, *Local _Dual* always starts with a dual feasible solution, and it returns an optimal solution under the assumption that all cycles have a positive length, as in the general method. The cost of each iteration is O(m); in fact, in the worst case, the backward star of each node $j \notin N_T$ must be scanned to perform *findmin*(j, γ_j). However, the algorithm might find an optimal solution in pseudopolynomial time in the case of integer arc lengths, as shown in Fig. 1. If we start from the zero price vector, each local phase increases the price of each $j \neq r$ by 1; after M iterations, the price of each $j \neq r$ thus reaches the value M, and then j enters T (here M denotes a very large positive number).

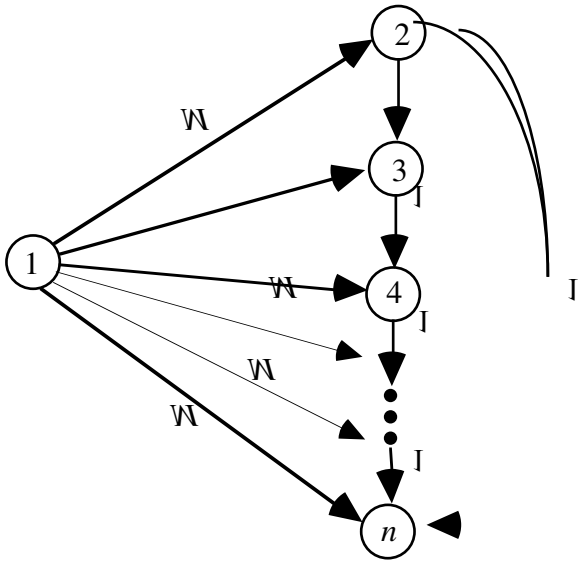

Fig. 1

In order to compute a shortest path tree without any assumption on the cycle length, and to be able to derive strongly polynomial algorithms, let us 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*, and denote these sets by $BS_B(j)$ and $BS_E(j)$, respectively:

$$BS_B(j) = \{(i,j) \in BS(j): i \in N_T\};$$
$$BS_E(j) = BS(j) \setminus BS_B(j).$$

Now, the *findmin* operation can be rewritten in the following, equivalent, form:

$$\textit{bordermin}(j, \beta_j): \quad \beta_j = \min\{\pi(i) + l(i,j): (i,j) \in BS_B(j)\};$$
$$\textit{externalmin}(j, \eta_j): \quad \eta_j = \min\{\pi(i) + l(i,j): (i,j) \in BS_E(j)\};$$
$$\textit{findmin}(j, \gamma_j): \quad \gamma_j = \min\{\beta_j , \eta_j\}.$$

In the above operations, β_j and η_j are assumed to be $+\infty$ in the case of empty sets. If we assume $\pi(r) = 0$, then we have:

Property 3.4- *γ_j and β_j are a lower bound and an upper bound to the shortest distance of j from r, respectively.*

Proof:

The feasibility of the price vector implies that γ_j does not exceed the length of any path connecting r to j; as a consequence, γ_j is a lower bound to the shortest distance from r to j. β_j denotes the length of a path from r to j; more precisely, it denotes the shortest distance among the paths consisting of tree arcs, except for the last one. Consequently, β_j is an upper bound to the shortest distance from r to j.♦

By Property 3.4, if $\gamma_j = \beta_j$ then *link*(j,T) can be performed after *local_reprice*. On the other hand, if $\beta_j > \gamma_j$ for each $j \notin N_T$, then no *link* operation can be performed by *Local _Dual*. To avoid this, and to guarantee that a link operation is always possible, several *global dual updating operations* are introduced. With this in mind, let us consider the following *globalmin* operation:

$$\textit{globalmin}(\delta): \quad \delta := \min\{\beta_j - \pi(j): j \notin N_T\}.$$

Let $w = \text{argmin}\{\beta_j - \pi(j): j \notin N_T\}$. Note that w is the out of the tree node which is the "closest" to the root, and that δ is a kind of threshold gap for the prices of the out of tree nodes: in fact, if $\gamma_j - \pi(j)$ is less than δ, $\forall\, j \notin N_T$, then no node can be linked to T.

A hierarchy of global dual updating operations can now be defined, based on *globalmin*, which will be used by our algorithm family instead of *local_reprice*.

The first, straightforward, kind of global updating, named *classic_reprice*, increases the price of all the out of tree nodes by δ:

$$\text{for each } j \notin N_T, \qquad \textit{classic_reprice}(j): \qquad \pi(j) := \pi(j) + \delta.$$

This updating operation corresponds to the "classical" global dual updating operation along the cut $(N_T, N \setminus N_T)$, and the threshold gap δ is the dual gap along the corresponding cutset. In fact, an algorithm based on *classic_reprice* is just a dual interpretation of the well-known Dijkstra's algorithm, in the case of non-negative arc lengths.

A second dual updating operation, named *global_reprice*, can be defined as follows:

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

Starting from a feasible price vector π and a tree T, rooted at r, in which condition (*ii*) holds true, we can perform the *global_reprice* operations instead of the *local_reprice* ones, thus maintaining the feasibility of the price vector, and ensuring the entry of at least one new node into T, as stated by the following Lemma:

Lemma 3.1- *Global_reprice maintains conditions* (*i*) *and* (*ii*); *moreover*, *at least node w can be linked to T.*

Proof:

Let us denote by $\pi'(j)$ the updated price of node j. By construction, condition (*i*) is satisfied. Consider in fact a node $j \notin N_T$ such that $\gamma_j = \beta_j \le \eta_j$. Since $\pi'(i) \ge \pi(i)$ for each node i, as shown by Property 3.3, then we have $\pi'(j) = \beta_j \le \pi(i) + l(i,j) \le \pi'(i) + l(i,j)$ for each $(i,j) \in BS(j)$. The same property holds for each $j \notin N_T$ such that $\gamma_j = \eta_j < \beta_j$. In this case, since $\pi'(j) = \pi(j) + \delta$ and $\delta \le \beta_j - \pi(j)$, then $\pi'(j) \le \beta_j \le \pi(i) + l(i,j) \le \pi'(i) + l(i,j)$, $\forall\, (i,j) \in BS_B(j)$; the same relationship holds for any $(i,j) \in BS_E(j)$, since both $\pi(i)$ and $\pi(j)$ increase by the same value δ. Consider now condition (*ii*). It is still valid for each arc belonging to T, since no change is performed by *global_reprice* along the tree.

After the dual updating operation, for each $j \notin N_T$ such that $\pi'(j) = \beta_j$, if any exists, there is at least one entering arc (i,j) with $i \in N_T$ such that $\pi'(j) = \pi(i) + l(i,j)$; as a consequence, each such node j can be linked to the tree through a *link* operation. Anyway, at least the "closest" node w, for which $\delta = \beta_w - \pi(w)$, can be linked to T, since after the price updating $\pi(w) = \beta_w$ holds. ♦

Consider now the following dual updating operation, called *selected_reprice*:

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

This updating at each out of tree node j can be interpreted as a particular *local_reprice* operation at j. This is obvious for $j \ne w$; on the other hand, as far as node w is concerned, setting

$\pi(w) = \beta_w$ can be viewed as a *local_reprice* operation in a *reduced graph*, $G' = (N, A')$, obtained from G by removing all the external arcs entering w, i.e. $A' = A \setminus BS_E(w)$. The last observation will be made clearer below.

Consider now the price vector obtained by applying the *selected_reprice* operations instead of the *local_reprice* ones. In this case, too, node w can be linked to T after the updating. However, the dual feasibility may be lost. In fact, since $\pi(w)$ is updated by considering only the border arcs going into w, condition (*i*) may be violated for the entering external arcs. Then, in order to guarantee the dual feasibility, each arc of $BS_E(w)$ is removed from G, i.e. a *reduction operation* is performed. As will be shown, this reduction guarantees that a shortest path tree will be found, i.e. the shortest path tree found in the reduced graph G' is also one of the shortest path trees in the original graph.

Lemma 3.2- *Selected_reprice maintains conditions* (*i*) *and* (*ii*) *in the reduced graph G', and any shortest path tree in G' is a shortest one in the original graph*; *moreover*, *at least w can be linked to T.*

Proof:

Recall that $w = \text{argmin}\{\beta_j - \pi(j): j \notin N_T\}$, and there exists a border arc (i,w), with $i \in N_T$, such that $\beta_w = \pi(i) + l(i,w)$.

If $\beta_w \leq \eta_w$, i.e. $\gamma_w = \beta_w$, then *selected_reprice* is equivalent to a *local_reprice* operation for each $j \notin N_T$, and the path connecting r to w through (i,w) is a shortest one. As a consequence, conditions (*i*) and (*ii*) still hold, w can be linked to T, and the removal of the external arcs entering w transforms the problem into an equivalent one.

However, the case in which $\gamma_w = \eta_w < \beta_w$ is different. In this case, conditions (*i*) and (*ii*) are valid in the reduced graph G', and w can be linked to T since the path, say P_w, which is the union between the only path P_i from r to i in T and the arc (i,w) is a shortest path for w in G'. What we have to show is that P_w is one of the shortest paths in the original graph G as well. Suppose that there exists a path P'_w in G whose length $l(P'_w)$ is strictly less than β_w, which is the length of P_w: $l(P'_w) < \beta_w$. Let (u, v) be the unique border arc in P'_w, i.e. since $u \in N_T$ and $v \notin N_T$, and P'_v and P'_{vw} denote the two subpaths of P'_w, respectively.

If we apply condition (*i*) to the arcs belonging to the subpath P'_{vw}, then we get:

$$l(P'_{vw}) \geq \pi(w) - \pi(v).$$

As a consequence, since $l(P'_v) = \pi(u) + l(u,v)$, then we obtain:

$$l(P'_w) \geq \pi(u) + l(u,v) + \pi(w) - \pi(v).$$

By the definition of β_v, it is $\beta_v \leq \pi(u) + l(u,v)$, and so $l(P'_w) \geq \beta_v + \pi(w) - \pi(v)$.

By definition of *globalmin* it is $\beta_w \leq \beta_v + \pi(w) - \pi(v)$, and so $l(P'_w) \geq \beta_w$, which contradicts the assumption that P'_w is strictly shorter than P_w. P_w is thus a shortest path in the original graph G as well. ♦

Sharper dual updating operations can now be introduced. The first one, named *extended_reprice*, merges *global_reprice* and *selected_reprice* as follows:

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

Note that *extended_reprice* modifies the price of each $j \notin N_T$ for which $\gamma_j = \beta_j$ exactly as performed by *global_reprice*; however, the other out of tree nodes are given the maximum between the price assigned by *global_reprice* and the one assigned by *selected_reprice*.

Lemma 3.3- *Extended_reprice maintains conditions* (*i*) *and* (*ii*); *moreover*, *at least w can be linked to T.*

Proof:

The maintenance of conditions (*i*) and (*ii*) follows immediately from Lemmas 3.1 and 3.2.

As far as the link of a new node to T is concerned, each node $j \notin N_T$ such that $\gamma_j = \beta_j$, if any exists, can be linked to the tree. In any case, at least node w, for which $\delta = \beta_w - \pi(w)$, can be linked to T, since $\pi(w) = \beta_w$ after *extended_reprice*. ♦

Finally, consider the following dual updating operation, that will be referred to as *double_reprice*:

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

Lemma 3.4- *Double_reprice maintains conditions* (*i*) *and* (*ii*); *moreover*, *at least w can be linked to T.*

Proof:

Let $\pi'(j)$ be the updated price of $j \notin N_T$. Firstly, let us prove that condition (*i*) is satisfied for each $j \notin N_T$ such that $\beta_j \leq \eta_j + \delta$ (**case *a***).

For each $(i,j) \in BS_B(j)$, we have $\pi'(j) = \beta_j \leq \pi(i) + l(i,j) \leq \pi'(i) + l(i,j)$. Consider now an arc $(i,j) \in BS_E(j)$. It is $\pi'(j) = \beta_j \leq \eta_j + \delta \leq \pi(i) + l(i,j) + \delta \leq \pi'(i) + \delta + l(i,j)$. Since $i \notin N_T$, either $\pi'(i) = \beta_i$, or $\pi'(i) = \eta_i + \delta$. In both cases condition (*i*) is still true.

Consider now the nodes $j \notin N_T$ such that $\pi'(j) = \eta_j + \delta < \beta_j$ (**case *b***). Condition (*i*) clearly holds for each $(i,j) \in BS_B(j)$, while $\pi'(j) = \eta_j + \delta \leq \pi(i) + l(i,j) + \delta \leq \pi'(i) + \delta + l(i,j)$ for each $(i,j) \in BS_E(j)$. The statement then follows from the corresponding proof of *case a.*

As far as condition (*ii*) is concerned, as observed for the other dual updating operations, it is still valid for each arc belonging to T, since no change is made along the tree.

Finally, for each $j \notin N_T$ such that $\pi'(j) = \beta_j$, if some node of this type exists, there is at least one entering arc (i,j) with $i \in N_T$ such that $\pi'(j) = \pi(i) + l(i,j)$, and, as a consequence, j can be linked to the tree. In any case, at least node w, for which $\delta = \beta_w - \pi(w)$, can be linked to T; in fact it is $\eta_w + \delta = \eta_w + \beta_w - \pi(w) \geq \beta_w$, and so a *link* operation can always be performed at w. ♦

In the following section, the relationships among the proposed dual updating operations will be better investigated. Then, several strongly polynomial dual ascent algorithms will be described, and their time complexity will be analysed.

4. Dominance relationship among the dual updating operations

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.

We say that D_1 *dominates* D_2 if $\pi(D_1,\pi) \geq \pi(D_2,\pi)$. This dominance relationship is significant from a theoretical point of view. The aim of the dual ascent algorithms is in fact to increase the dual variables until their optimum value, and the use of D_1 instead of D_2 guarantees a not greater (and possibly lower) number of dual updating operations. Note that the relationship is transitive.

By considering the dual updating operations introduced in Section 3, the following dominance relationships hold true, 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:

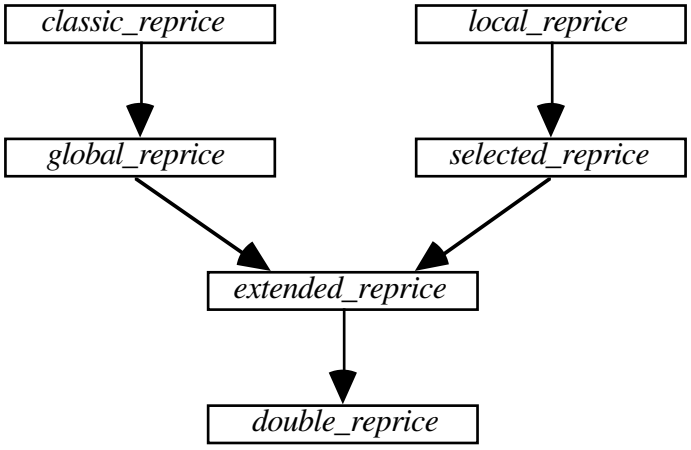

Fig. 2

The dual operation *global_reprice* clearly dominates *classic_reprice*. Similarly, *selected_reprice* dominates *local_reprice*. In fact, *selected_reprice* updates the prices of the out of tree nodes as performed by *local_reprice* (i.e. $\pi(j):=\gamma_j$, for each $j \notin N_T$), with the exception of node w, whose potential is set to $\beta_w \geq \gamma_w$.

Like *global_reprice*, *extended_reprice* sets to β_j the prices of the nodes satisfying $\gamma_j = \beta_j$; on the other hand, the price of every other out of tree node j is set to $\max\{\pi(j) + \delta; \gamma_j\}$. But it is $\max\{\pi(j) + \delta; \gamma_j\} \geq \pi(j) + \delta$, which is the value assigned to j by *global_reprice*. *Extended_reprice* thus dominates *global_reprice*.

Consider now $j \notin N_T$. If $j = w$, then *selected_reprice* sets $\pi(w) = \beta_w$. By considering *extended_reprice*, either $\gamma_w = \beta_w$, and then $\pi(w) = \beta_w$ too, or $\pi(w) = \max\{\pi(j) + \delta, \gamma_j\}$; also in the last case it is $\max\{\pi(j) + \delta, \gamma_j\} = \beta_w$, since $\delta = \beta_w - \pi(w)$. Consider now $j \neq w$. In this case, *selected_reprice* sets $\pi(j) = \gamma_j$. If $\gamma_j = \beta_j$, then *extended_reprice* sets $\pi(j) = \gamma_j$ too, otherwise (i.e. $\gamma_j < \beta_j$), it is $\max\{\pi(j) + \delta, \gamma_j\} \geq \gamma_j$. *Extended_reprice* thus dominates *selected_reprice* too.

Finally, let us consider the last dominance relationship. Let j be a node such that $\beta_j \leq \eta_j + \delta$. In this case, $\pi(j) = \beta_j$ by *double_reprice*, which is clearly greater than or equal to the value assigned by *extended_reprice*, being β_j the shortest distance relative to node j. On the other hand, let j such that $\beta_j > \eta_j + \delta$. In this case *double_reprice* sets $\pi(j) = \eta_j + \delta$, while *extended_reprice* gives $\pi(j) = \max\{\pi(j) + \delta, \gamma_j\}$, due to the fact that $\gamma_j < \eta_j + \delta < \beta_j$. But it is $\max\{\pi(j)+\delta, \gamma_j\} \leq \eta_j + \delta$. It follows that *double_reprice* dominates *extended_reprice*.

5. New strongly polynomial dual ascent algorithms

First of all, let us show how to implement the previously defined operations, and analyse their consequent running times.

Bordermin can be implemented as follows: let β_j be the value associated with the out of tree node j at a certain stage of the algorithm (initially, $\beta_j = +\infty$). Each time a new node i enters T, its forward star is scanned and, for each arc $(i,j) \in FS(i)$, β_j is updated as follows:

$$\beta_j := \min\{\beta_j, \pi(i) + l(i,j)\}.$$

The overall cost of the *bordermin* operations is thus O(m), because each arc will contribute to these operations no more than once.

In order to implement *externalmin*, we can scan the backward star of each out of tree node j, construct the set $BS_E(j)$ by removing the border arcs, and compute η_j in the following way:

$$\eta_j := \min\{\pi(i) + l(i,j): (i,j) \in BS_E(j)\}.$$

It follows that *externalmin* runs costs O(m) per iteration. Clearly, together with β_j and η_j, the arcs for which the minimum is obtained can be suitably stored.

Let us consider now *globalmin*. Value δ can be computed by considering the cutset separating the current tree nodes from the other nodes:

$$\delta := \min\{\beta_j - \pi(j): j \notin N_T\}.$$

Here too, the out of tree node w, for which value δ is obtained, can be suitably stored. Clearly, *globalmin* costs O(n) per iteration.

The new family of dual ascent algorithms can now be described, where each algorithm differs in the dual updating operation used. The general algorithmic paradigm, called *Global_Dual*, is based on the following typical iteration, where *Global_update*(j, β_j, η_j, δ) denotes one of the global dual updating operations described in Section 3:

Typical iteration of Global_Dual

```
δ := +∞;
for each j ∉ N_T do δ := min{δ, β_j - π(j)};
w := argmin{β_j - π(j): j ∉ N_T};
for each j ∉ N_T do
    begin
        externalmin(j, η_j);
        Global_update(j, β_j, η_j, δ);
        if π(j) = β_j then
            begin
                link(j,T);
                for each (j,h) ∈ FS(j) such that h ∉ N_T do β_h := min{β_h, π(j) + l(j,h)}
            end
    end.
```

Global_Dual returns an optimum solution also in the case of zero length cycles (as follows from Lemmas 3.1 to 3.4), and it runs in O(mn) time. In fact, no more than n iterations are performed, and each iteration costs O(m), as follows from the time complexity analysis of the

operations *bordermin*, *externalmin* and *globalmin*. In particular, if *classic_reprice* is used in *Global_Dual*, then the *externalmin* operations are no longer necessary, and the resulting algorithm, Dijkstra's, runs in $O(n^2)$ time. Another particular situation arises when *global_reprice* is used in the above algorithmic framework. In this case, at the end of each iteration, the price of each out of tree node increases by the same value δ. So, if the backward stars of the nodes are pre-sorted into a non decreasing order with respect to the initial reduced costs, then the external arcs entering the out of tree nodes remain ordered during the algorithm iterations. As a consequence, the output of *externalmin*, when applied at node *j*, always corresponds to the first external arc in the ordered backward star of *j*. *Externalmin* at *j* can thus be implemented by scanning the backward star of *j* to the first external arc. By using this implementation, the overall cost of the *externalmin* operations is $O(m)$, hence:

Theorem 5.1- *Global_Dual runs in* $O(n^2)$ *time when global_reprice is used and the backward stars of the nodes are pre-sorted in a non decreasing way with respect to the initial reduced costs.*

Observe that *Global_Dual* can be implemented in a parallel way, though some implementations may require an interruption of the parallelism, at least during the computation of the threshold gap δ. Note that the dual approach proposed here is also suitable for the reoptimization of the shortest path tree, i.e. when the computation of shortest paths from different root nodes is required. In fact, consider the shortest path tree *T*, rooted at node *r*, returned by the proposed approach, and assume that the root is changed to a different node, say *s*. Consider the subtree of *T* rooted at *s*, let it be T_s, and the optimal price vector, let π^*. Since the complementarity slackness conditions hold for each arc of T_s, we can efficiently apply the dual ascent algorithms starting from T_s and π^*, instead of starting from scratch. The algorithm iterations thus involve only those nodes that do not belong to T_s, and they can exploit all the previously computed information (i.e. tree and dual vector), with a potential saving of computing time.

References

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

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

Bertsekas, D. P. (1979), "A Distributed Algorithm for the Assignment Problem", Lab. for Information and Decision Systems, M.I.T., Working Paper.

Bertsekas, D. P. (1991a), "An Auction Algorithm for Shortest Paths", *SIAM Journal on Optimization*, **1**, 425-447.

Bertsekas, D. P. (1991b), *Linear Network Optimization: Algorithms and Codes*, M.I.T. Press, Cambridge, MA.

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

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

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

Florian, M., S. Nguyen and S. Pallottino (1981), "A Dual Simplex Algorithm for Finding all Shortest Paths", *Networks*, **11**, 367-378.

Gallo, G. and S. Pallottino (1986), "Shortest Path Methods: A Unifying Approach", *Mathematical Programming Study*, **26**, 38-64.

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

Gondran, M. and M. Minoux (1984), *Graphs and Algorithms*, Wiley-Interscience, New York, NY.

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

Pallottino, S. and M. G. Scutellà (1991), "Strongly Polynomial Algorithms for Shortest Paths", *Ricerca Operativa*, **60**, 33-53.

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