Report n.18

A k-Shortest Path Approach to the Minimum Cost Matching Problem

by

Paolo Carraresi[1] and Claudio Sodini[2]

1 Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56100 Pisa, Italy.
2 Dipartimento di Statistica e Matematica Applicata all'Economia, Università di Pisa, via Ridolfi 10, 56100 Pisa, Italy.

Pisa, luglio 1988
This research is partially supported by Public Education Ministry.

A k-Shortest Path Approach to the Minimum Cost Matching Problem

by

Paolo Carraresi[1] and Claudio Sodini[2]

Abstract

A new algorithm for the minimum cost matching problem is presented. The proposed approach is based on computing the shortest alternating path between a pair of free nodes, w.r.t. the current minimum cost matching of cardinality u, $1 \leq u \leq \lfloor n/2 \rfloor$. If the shortest alternating path is not augmenting, the second shortest alternating path is determined, and so on until the shortest augmenting path is found. This way the cardinality of the current minimum cost matching is increased by one. The number of paths explicitly enumerated by the schema is less than or equal to the number of arcs.

1 Introduction

Let G=(N,A) be an undirected graph, where N={1,2,...,n} is the set of node and A is the set of edges with |A|=m. A *matching* M is a sub-set of edges with no two incident with a common node. A node $i \in N$ is *free* with respect to M if no edge of M is incident with i, otherwise the node is *matched.* Given a matching M, let N_F and N_M denote the set of free nodes and respectively matched nodes. A matching of G is *perfect* if no node of G is free.

Let c(i,j) be the cost of edge $(i,j) \in A$ and $c=[c(i,j): (i,j) \in A]$. The minimum cost matching problem is that of finding a perfect matching M, such that the sum of the cost of the edges in M is minimum. This problem has been widely investigated [13], and most of the approaches in the literature (see, for example, [3],[4],[5],[9]) are based on the pioneering paper of Edmonds [6]. Recently a new algorithm has been proposed [12], where the continuous relaxation of the minimum cost matching problem is solved as a network flow problem and blossom constraints violated by the current solution are added, one at a time, as equality constraints; the new problem is still solved as a network flow

1 Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56100 Pisa, Italy.

2 Dipartimento di Statistica e Matematica Applicata all'Economia, Università di Pisa, via Ridolfi 10, 56100 Pisa, Italy.

problem. The proposed approach is based on computing the shortest alternating path between a pair of free nodes, w.r.t. the current minimum cost matching of cardinality u, $1 \leq u \leq \lfloor n/2 \rfloor$. If the shortest alternating path is not augmenting, the second shortest alternating path is determined, and so on until the shortest augmenting path is found. This way the cardinality of the current minimum cost matching is increased by one. Two aspects may be pointed out: first, the number of paths explicitly enumerated by the schema is less than or equal to the number of arcs; second, the approach provides an upper bound to the optimal solution, which together with the lower bound given by the length of the current enumerated path determines an "optimality gap" of the feasible solution. The paper is organized as follows. Section 2 contains a review of basic definitions and properties of matching problems, while the description of the proposed approach is given in section 3. Finally the correctness and the worst case complexity are stated in section 4.

2 Alternating and augmenting path on the collapsed graph

A *minimum cost parametric cardinality matching problem* P(u) is that of finding a matching M_u: $|M_u| = u$, $1 \leq u \leq \lfloor n/2 \rfloor$, such that the sum of the cost of the edges in M_u is minimum.

Given a non-perfect matching M_u, an *alternating path* is a path whose edges are alternatively in and out of M_u; an *augmenting path* is a simple alternating path P_{ij} from a free node i to a free node j, i≠j. Denoted $M_u \oplus P_{ij}$, the operation of augmenting M_u yields a new matching M_{u+1} given by the edges of M_u, minus the edges of P in M_u, plus the edges of P out of M_u.

Let us define the cost C_{ij} of an augmenting path P_{ij} as the sum of the cost of the edges of P_{ij} in M_u, minus the sum of the cost of the edges of P_{ij} out of M_u. The following theorem ([13] page 115) allows the determination of the optimal solution of P(u+1), given the optimal solution of P(u).

Theorem 2.1

Let M_u be an optimal matching for P(u) and P_{ij} be a shortest augmenting path. Then $M_{u+1} = M_u \oplus P_{ij}$ is an optimal matching for P(u+1).

As stated by theorem 2.1 the problem is to finding a shortest augmenting path. To this aim, given the graph G=(N,A) and an optimal matching M_u, consider the following directed graph G'=(N',A'), where l(i,j) denote the cost of (i,j)∈ A' (see fig.1).

N'={s}∪O∪D∪W∪{t}, where

- O and D correspond to free nodes of G with respect to M_u: O={i: i is a free node} and the set D is a copy of O, i.e., D = {i+n, ∀i∈O};
- W is the set of matched nodes; the matching M_u can be described by a vector mate(.), where i=mate(j), j=mate(i), ∀i,j: (i,j)∈ M_u and mate(i)=n+i, mate(n+i)=i, ∀ i∈O.
- s is a source node not in G;
- t is a sink node not in G.

The set A' and the related cost l are given by:

- (s,i), ∀i∈O; l(s,i) =0;
- (i,j), ∀i∈O∪W,∀j∈D: (i,j-n)∈A; l(i,j)=c(i,j-n);
- (i,j), ∀i∈O∪W, ∀j∈W: (i,k)∈A and k=mate(j); l(i,j) =c(i,k)-c(k,j);
- (j,t),∀j∈D; l(j,t)=0.

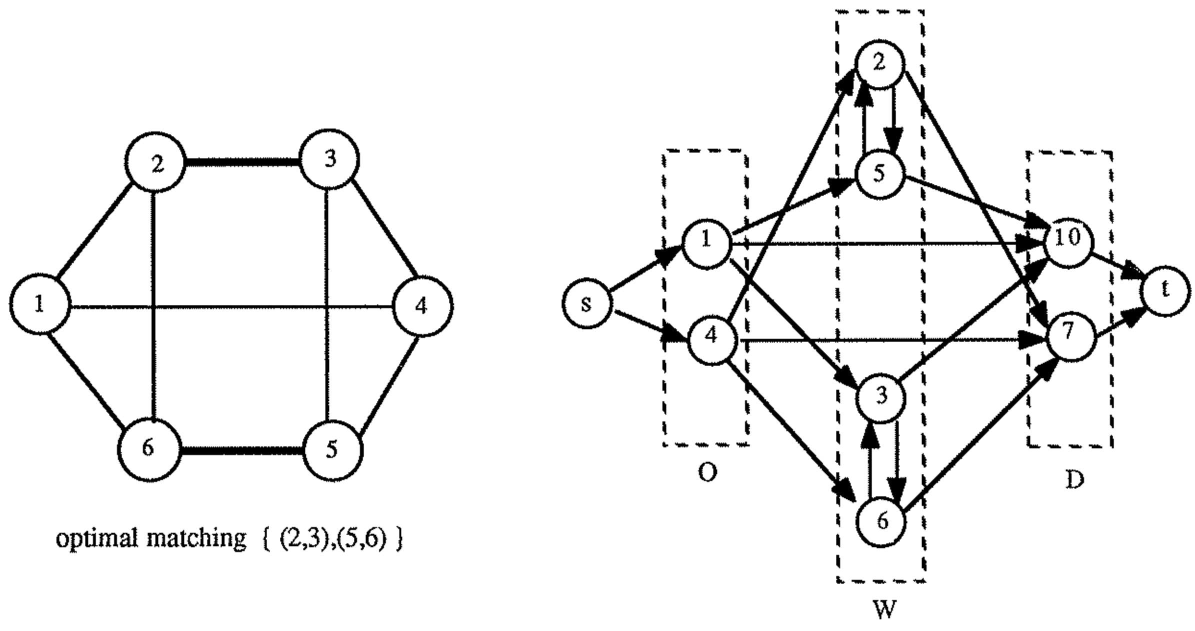


fig. 1

Note that G' is completely defined by G and by the current matching M_u. In fact, assume G is defined by the "adjacency list" AL(i), ∀ i∈N, where AL(i)={j: (i,j)∈A}; then the "forward star" FS(i)={j: (i,j)∈A'} and the "backward star" BS(i)={j: (j,i)∈A'}, i∈N' are given by:

FS(s)=O, BS(s)=ø, FS(t)=ø, BS(t)=D;

FS(i)={mate(j): j≠mate(i),j∈AL(i)}, i∈O∪W; FS(i)={t}, i∈D;

BS(i) = {j: j∈AL(mate(i)),j≠i}, i∈W∪D; BS(i)={s}, i∈O.

First observe that any s-t path in G' (i.e., a path from s to t) corresponds to a path from o∈O to d∈D which will be called o-d path and denoted by $P_{o,d}$. Note that any o-d path corresponds to an alternating path in G.

Given a path $P_{o,d}$ defined by the sequence of nodes: $P_{o,d}=\{o,w_1,w_2...,w_p,d\}$, where o∈O, d∈D and w_i∈W, i=1,...,p, the symmetric path of $P_{o,d}$, is the path $S(P_{o,d})=\{d-n,mate(w_p),...,mate(w_2),mate(w_1),o+n\}$. The symmetric path of a path in G' from w_i∈W to w_j∈W is analogously defined. A pair i,mate(i) will be called *a pair of symmetric nodes*. By definition i,i+n, ∀i∈O are pair of symmetric nodes.

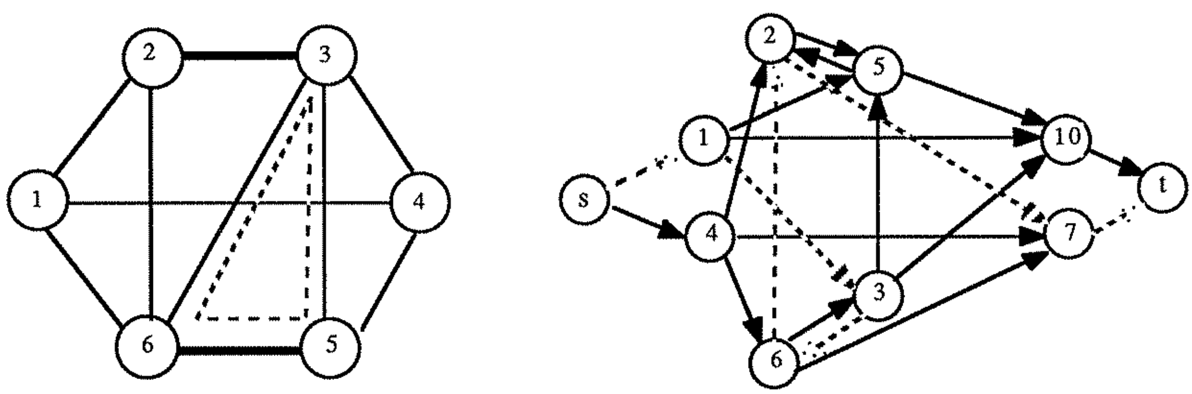
Let $T=(N',A'_T)$ be the shortest path tree on G' rooted at s, δ(i) the length of the unique path from s to i on T. T is described by the predecessor vector pred(.), where pred(s)=0. The following properties of o-d paths in T hold trivially.

Property 2.1

$P_{o,d}$ and $S(P_{o,d})$ have the same length. Δ

Property 2.2

$P_{o,d}$ does not correspond to an augmenting path iff it contains at least a pair i,j of symmetric nodes. Δ



The shortest path on G' induces an odd cycle on G.

fig. 2

Note that if $P_{o,d}$ contains at least one pair i,j of symmetric nodes then the corresponding path in G contains an odd cycle, that is, a cycle with an odd number of nodes (see fig. 2).

Now given a path $P_{o,d}$, consider a procedure which labels the nodes of this path starting from d. If the procedure labels a node j such that i=mate(j) has been labeled, then a cycle is discovered and i is called the *base* of the cycle. If such a condition is not verified and o is reached, then $P_{o,d}$ is an augmenting path. Otherwise the path is not augmenting and the set of nodes $C= S(P_{mate(i),i}) \cup P_{mate(i),i}$ will be called the nodes of the cycle. Furthermore the set of the arcs connecting nodes of the cycle, i.e., $\{(u,v) \in A': u,v \in C\}$ will be partitioned in two sub-sets: the set B of boundary arcs, $B=\{(u,v): (u,v)$ is an arc of $P_{mate(i),i}$ or of $S(P_{mate(i),i})\}$ and the set I of internal arcs, $I=\{(u,v): u,v \in C\} \setminus B$.

Note that if $P_{o,d}$ is not a simple path, then an even cycle on G corresponds to each cycle contained in the path. In the following we consider (except when explicitly stated) simple o-d paths and, for sake of simplicity, we will say that $P_{o,d}$ *contains a cycle* when the corresponding path on G contains an odd cycle.

Property 2.3

$P_{o,d}$ and $S(P_{o,d})$ are not node disjoint paths iff $P_{o,d}$ (and $S(P_{o,d})$) includes at least one pair of symmetric nodes. Δ

Property 2.4

Let T_o be the sub-tree of T rooted at o and let $P_{o,d}$ be the shortest o-d path on T_o (if any). Further assume that $P_{o,d}$ contains at least one cycle; let i be the base of the first cycle encountered when backtracking from d. Then the path $P'_{d-n,d}$ given by the three sub-paths: $S(P_{i,d})$, $P_{mate(i),i}$, $P_{i,d}$ has the same length as $P_{o,d}$ and contains exactly one cycle. Δ

A non-augmenting path with the structure of $P'_{d-n,d}$ will be called a *blocking path* (see fig. 3).

Assume that the shortest path P_{d^*-n,d^*} is a blocking path of length δ, which contains the cycle C. The following theorem characterizes augmenting paths of lengths greater than or equal to δ, which include arcs of C.

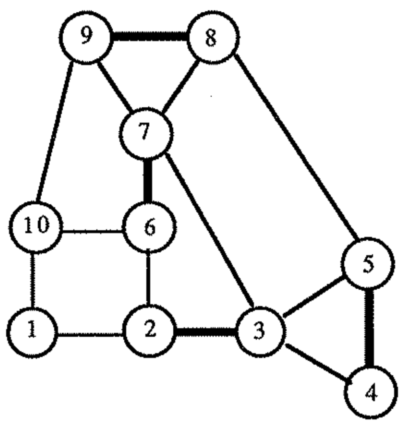
Theorem 2.2

Let $P_{o,d}$ be an augmenting path on G' which contains an internal arc (u,v) of the cycle C. If the length of $P_{o,d}$ is greater than or equal to δ, then there exists an augmenting path including only boundary arcs of C with a length less than or equal to the length of $P_{o,d}$.

Proof.

Let b be the base of C and j be the last node of $P_{o,d}$ which belongs to P_{d^*-n,d^*}. The following cases can be distinguished:

- $j \in C$, $j \neq b$; in such a case the path given by the sub-path of P_{d^*-n,d^*} (or of $S(P_{d^*-n,d^*})$ if $j \in S(P_{d^*-n,d^*})$) from d*-n to j and by the sub-path of $P_{o,d}$ from j to d satisfies the condition of the theorem.
- j belongs to the sub-path of P_{d^*-n,d^*} from b to d*; in this case the path given by the sub-path of $P_{o,d}$ from o to u, by the sub-path of P_{d^*-n,d^*} (or of $S(P_{d^*-n,d^*})$ if $j \in S(P_{d^*-n,d^*})$) from u to j and by the sub-path of $P_{o,d}$ from j to d satisfies the condition of the theorem.Δ



The shortest path on G' corresponds to the path

1,2,3,4,5,3,2,6,7,8,9,7,6,10.

The path is not augmenting and there are two blocking paths of the same length.

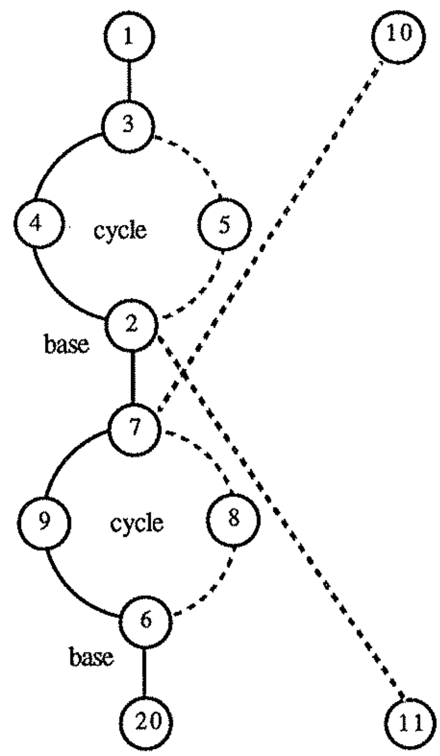


fig. 3

Observe that if G is a bipartite graph, then any o-d path in G' corresponds to an augmenting path. Consequently the problem of finding the shortest augmenting path on G, reduces to that of finding the shortest path tree on G' rooted at s. In the general case shortest o-d paths, not corresponding to augmenting path in G, must be avoided; this can be done according to an implicit enumeration schema which is described in the following.

3 The matching algorithm

Let SPT(T) be any shortest path tree routine which determines T on G'. The procedure given in table 1 yields a maximum cardinality minimum cost matching by computing a sequence of minimum cost matchings M_u: $|M_u| = u$, $u=1,2,...,p \leq \lfloor n/2 \rfloor$.

```
procedure Parametric_Matching;
begin
   InitializeM;
   u:=1;
   repeat
      u:=u+1;
      SPT(T);
      Find_Shortest_Augmenting_Path(T,P,Aug_path);
      if Aug_path then Augment(P);
   until (u = ⌊n/2⌋) or (not Aug_path)
end.
```

table 1

InitializeM determines the minimum cost matching M_1 simply by computing $c(i',j') = \min\{c(i,j): (i,j) \in A\}$ and setting : mate(i):=n+i, mate(n+i):=i $\forall i \neq i',j'$ and i':=mate(j'), j':=mate(i').

Augment(P) grows the current matching by alternating the shortest augmenting path in G returned by the procedure Find_Shortest_Augmenting_Path(T,P,Aug_path). This procedure yields a shortest augmenting path by efficiently enumerating shortest s-t paths.

The enumeration procedure starts by considering the shortest s-t path on G', that is the unique path from s to t in T. Let i, j be the first pair of symmetric nodes discovered by backtracking on T starting from t and let j be backtracked before i. If no pair of symmetric nodes belongs to the path, the path is augmenting. Otherwise the path is not augmenting, since it includes at least the cycle based at j. Moreover a blocking path of the same length is given by the sub-path from mate(j) to t and from the symmetric of the sub-path from j to t . Therefore if the path is not augmenting a blocking path has been found (Property 2.4). Furthermore the symmetric of the blocking path is a blocking path of the same length; this blocking path will be implicitly enumerated by the procedure.

In classical enumeration schemata of shortest paths ([2], [11]), successive shortest paths are found by computing "shortest deviations" w.r.t. the set of enumerated paths. A deviation w.r.t. the node v of an enumerated path, say the path h, is a path given by the sub-path from s to w on T, by any arc (w,v), where w does not belong to the path h, and by the sub-path from v to t of the path h. The arc (w,v) will be called the deviation arc. The shortest deviation w.r.t. the nodes of the enumerated paths corresponds to the next shortest path in the enumeration. If this path includes a cycle it is a non-augmenting path (not necessarily a blocking path). Note that, also in this case, the symmetric of such a path is a non-augmenting path of the same length, which will be implicitly enumerated. Since our aim is to find the shortest augmenting path, all the shortest paths which are not augmenting can be implicitly enumerated. Consequently, the deviations which include cycles previously discovered cannot be considered. This is the case when a deviation refers to a node which is after (w.r.t. backtracking) the cycle of a blocking path, i.e., the node which belongs to the sub-path from s to mate(j), where j denotes the base of the related cycle. In such a case, deviations can be computed from the nodes which belong to the sub-path from j to t and to the two subpaths from mate(j) to j of the blocking path and the symmetric path respectively, except the node mate(j).

Assume that k shortest paths have been enumerated and consider for each enumerated path the sub-paths described above. Let $G^*=(N^*,A^*)$ denote the graph spanned by these sub-paths. Note that each node of G^* is connected to t by at least one alternating path which does not include a cycle.

In order to find the path k+1, the shortest deviation w.r.t. the nodes of G^* must be computed. Then the path k+1 is backtracked, to check if the shortest augmenting path has been found. The following cases can be distinguished: i) either a pair of symmetric nodes is discovered (i.e., a cycle is found) and the path is not augmenting; ii) or the node s is reached and the path is augmenting. If i) is verified then G^* is grown by the sub-paths previously described. Otherwise if ii) is verified, that is, the node s is reached without verifying i), the shortest augmenting path has been found.

This way, an enumeration procedure which finds a shortest augmenting path has been completely defined. It is worthwhile to remark that the worst case complexity of such a procedure depends on the number and on the length (in terms of the number of arcs) of enumerated paths. In the following we want to show that a shortest augmenting path can be found in O(nm). The basic idea is to consider, for each node i of G^*, the first (i.e., the shortest) enumerated sub-path which includes i. This is suggested by the fact that the shortest deviation w.r.t. the node i refers to such a path. It should be noted that, obviously, the shortest augmenting path can be any deviation w.r.t. any path traversing a node i (not necessarily the shortest one). Therefore the procedure must also be able to compute such deviations. Following this idea, it is not necessary to explicitly store for each node i all the enumerated paths containing i but only the shortest one together with the cycle contained by the path when it is a blocking path. In fact, in this case, the cycle allows us to implicitly store the symmetric path. Let $F_t=(N_t,A_t)$ be a forest of G*, and $F_s=(N_s,A_s)$ be the symmetric forest, i.e., $(i,j)\in A_s$ iff $(mate(j),mate(i))\in A_t$. The enumeration procedure grows F_t (and implicitly F_s) by alternating paths which do not include any cycles, in such a way that $N_t \cap N_s=\emptyset$. Initially F_t is given by the sub-path of the first blocking path from the node next to mate(j) to t, where j denotes the base of the related cycle. Hence one of the tree of F_t is rooted at t. As we will see, each root $r \neq t$ of a tree of F_t refers to a given node of N_s. The graph given by the union of F_t and F_s, by the arcs addressed by the roots of F_t together with the knowledge of the cycles related to the enumerated blocking paths, allows us to efficiently store the enumerated paths and to find the shortest deviation. Consider the following notation:

for each enumerated path h, h=1,...,k

$\theta(h)$: the length of the path h,

dest(h): the destination node of the path h;

$$\text{base}(h)=\begin{cases} j_h, & \text{if the path h is blocking and } j_h \text{ is the base of the cycle,}\\ 0, & \text{otherwise;}\end{cases}$$

for each node $i\in N_t$:

$$\text{next}(i)=\begin{cases} 0, & \text{if } i\notin N_t,\\ j, & \text{if } (i,j)\in A_t \text{ and i is not a root of } F_t,\\ -j, & \text{if i is a root of } F_t, \text{ where } j\in N_s;\end{cases}$$

h(i): the enumerated path which contains i;

$$\text{cycle}(i)=\begin{cases} \text{base}(h')\neq 0, & \text{if i belongs to the cycle related to the blocking path h', where h' is the shortest blocking path whose cycle includes i, and i is not the base of the cycle;}\\ 0, & \text{otherwise.}\end{cases}$$

In order to retrieve the blocking path which corresponds to cycle(i)≠0. i.e., the first blocking path whose cycle includes i, the following notation is introduced:

$$\text{path}(i)=\begin{cases} h, & \text{if i is the base of the cycle corresponding to the blocking path h;} \\ 0, & \text{otherwise.} \end{cases}$$

As we will show later, the procedure grows F_t after having computed the shortest deviation among deviations starting from nodes of N_t and from those nodes of N_s which belong to previously discovered cycles. Therefore the following property of F_t holds true during the enumeration: each root r of F_t, r≠t , refers to a node j∈ N_s such that j belongs to a previously discovered cycle, i.e.,: if next(r)=-j, r≠t then cycle(mate(j))≠0. In fact if the shortest deviation starts from a node j∈ N_t, then the procedure grows a tree of F_t; otherwise, if the shortest deviation starts from a node j∈ N_s, then the procedure initializes a new tree of F_t rooted at r, where (r,j) is the shortest deviation arc and next(r)=-j.

Note that $N^*=N_t \cup \{l \in N_s$: cycle(mate(l))≠0}; furthermore the shortest alternating path from j to t which does not include a cycle may be found by applying the procedure depicted in table 2.

```
procedure Sh_Alt_Path(j,t);
begin
   while j≠t do
   begin
     if next(j)=0                      {j∈Ns and cycle(mate(j)≠0}
        then j:=cycle(mate(j));
     if next(j)<0                      {j is a root of Ft}
        then j:=cycle(mate(|next(j)|));
     if next(j)>0                      {j∈Nt}
        then j:=next(j)
   end
end.
```

table 2

The enumeration procedure terminates when:

- an s-t path is contained in G*; in this case the corresponding o-d path is the shortest augmenting path.

- all the shortest s-t paths have been enumerated without satisfying the previous condition. In such a case the set of augmenting paths is empty.

Assume that the forest F_t corresponding to the enumeration of k shortest path is given, and let (w,v) be the deviation arc corresponding to the shortest deviation. A backtracking from w on T checks if the path k+1 is augmenting. Let i be the current backtracked node. The following cases can be distinguished:

I_1) a new cycle is discovered;

I_2) a node $i \in N_t$ or $i \in N_s$ is reached;

I_3) the node s is reached.

If case I_1) holds, then a new blocking path has been found. Case I_2) can be partitioned as follows: - the path k+1 is not simple, since it includes an even cycle (fig. 4a); - the simple path is augmenting (fig. 4b), blocking (fig. 4c) or non-augmenting, but non-blocking, since it contains a cycle which refers to a previously enumerated blocking path (fig. 4d). Finally, if case I_3) is the first one verified, then a shortest augmenting path has been found.

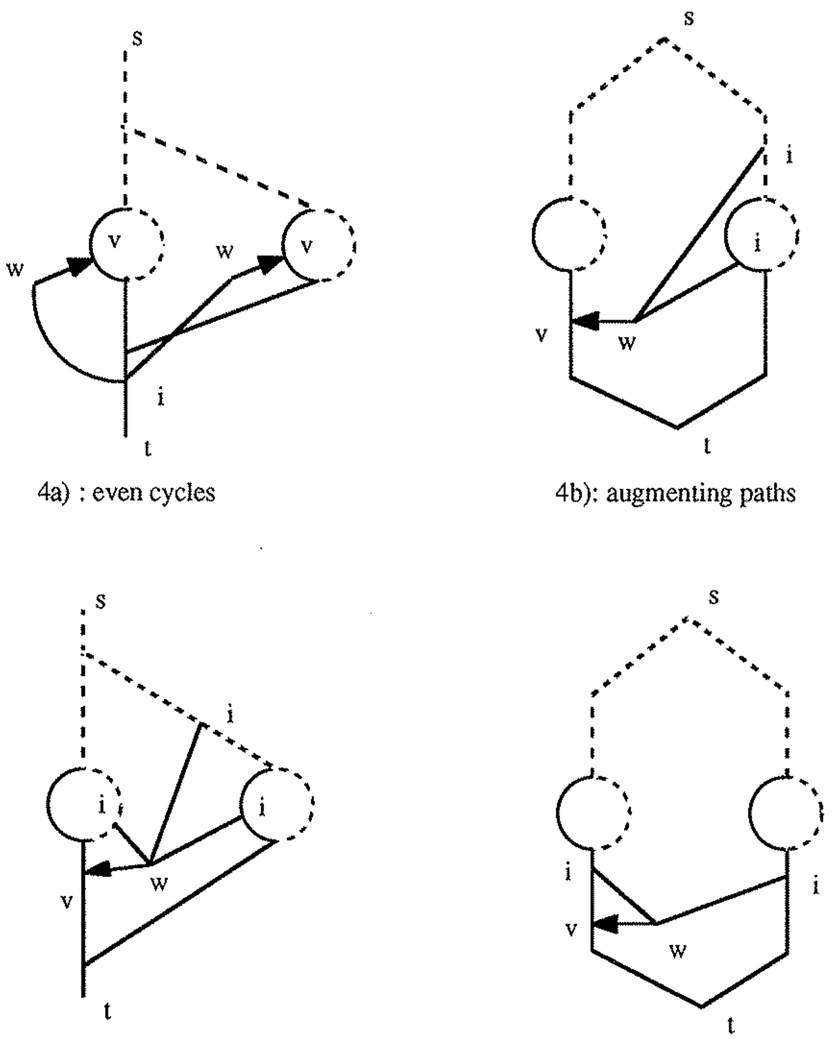
Note that when computing the shortest deviation from a node $v \in N^*$ we refer to the shortest sub-path from v to t of G^*. This way a deviation from v which refers to a different sub-path from v to t is not explicitly considered when computing shortest deviations. The following theorem shows that the enumeration based on the above definition of shortest deviation does not exclude augmenting paths. Let $P_{v,t}$ be the shortest path in G^* from v to t and let $P'_{v,t}$ be another path from v to t . Assume that the shortest deviation from v corresponds to the arc (w,v) and let $Q_{s,w}$ denote the sub-path from s to w of the path given by the shortest deviation.

Theorem 3.1

If the path P given by the sub-path $Q_{s,w}$, by the arc (w,v) and by the sub-path $P_{v,t}$ is not augmenting and the path P' given by $Q_{s,w}$, (w,v), $P'_{v,t}$ is augmenting, then there exists a node g of $P_{v,t}$ such that the shortest deviation from mate(g) is the symmetric path of P'.

Proof.

First note that P is a blocking path. In fact if P is not blocking it follows that the sub-path $Q_{s,w}$ should contain a cycle contradicting the hypothesis that P' is an augmenting path. Let (g,g') be the arc where $P'_{v,t}$ deviates from $P_{v,t}$ (see fig. 5).



4a) : even cycles

4b): augmenting paths

4c): blocking paths

4d): not augmenting paths

fig. 4

Since P' is an augmenting path, then g belongs to the cycle C related to the blocking path P and g≠b where b is the base of C. When P is enumerated it follows that mate(g) is one of the nodes which grows G^*; furthermore the shortest path from mate(g) to t in G^* is the symmetric of the sub-path of P from s to g. The shortest deviation from the node mate(g), w.r.t. such a path, corresponds to the arc (mate(g'),mate(g)) and is the symmetric of the path P'.Δ

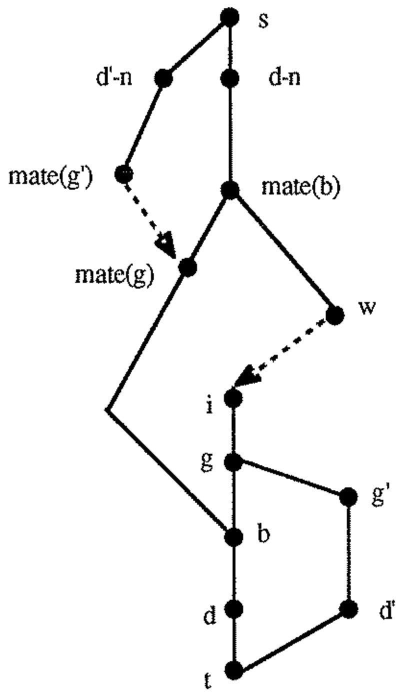


fig. 5

Let $\varphi(i)$, $i \in N$, be the length of the shortest deviation from i w.r.t. the nodes of the enumerated paths. The shortest deviation must be computed among the arcs $(i,j) \in A'$ such that $i \in N$, $j \in N^*$ and $(i,j) \notin A^*$. In fact, the arcs of A^* can be ignored, since the deviation (i,j) refers to the shortest path traversing j and considering $(i,j) \in A^*$ should produce a previously enumerated path. The length of the deviation is given by $\Delta(i,j)=l(i,j)+\delta(i)-\delta(j)+\theta(h)$, where

$$(3.1)\ h = \begin{cases} h(j), & \text{if } h(j)>0 \text{ (that is } j \in N_t\text{)}, \\ path(cycle(mate(j))), & \text{otherwise.} \end{cases}$$

Note that $l(i,j)+\delta(i)-\delta(j)$ can be interpreted as the reduced cost of (i,j) w.r.t. the optimal solution δ of the shortest path problem; consequently $l(i,j)+\delta(i)-\delta(j) \geq 0$.

Hence the length $\varphi(i)$ of the shortest deviation from i is given by:

$\varphi(i)=\Delta(i,r)=\min\{\Delta(i,j): j \in N^*, i \in N \text{ and } (i,j) \notin A^*\}$, and the corresponding deviation arc can be stored by setting r=dev(i) ($\varphi(i)=+\infty$, and dev(i)=0 if no deviation from i is known).

Let UB be the length of an augmenting path. Initially UB=$+\infty$; we will show later how the procedure can discover an augmenting path and consequently update UB.

Let Q_2 be a priority queue with respect to $\varphi(i)$, which contains each node i such that $\varphi(i)<UB$.

The deviation arc (w,v), which addresses the path k+1 is given by:

$$(3.2) \qquad \begin{aligned} &w: \varphi(w)=\min\{\varphi(i): i\in Q_2\}, \\ &v=dev(w). \end{aligned}$$

Hence to determine the path k+1 the node w is selected and removed from Q_2.

When backtracking the path k+1 starting from w, the nodes which can improve the length of the deviations stored in Q_2, are inserted in a set denoted Q_1. Such nodes are those in the sub-path of the path k+1 from the node next to mate(base(k+1)) to t, which have not been previously inserted in Q_1. Furthermore, since the symmetric of the path k+1 is implicitly enumerated, the nodes, not previously inserted in Q_1, of the sub-path of the symmetric of the path k+1 from the node next to mate(base(k+1)) to base(k+1), must be inserted in Q_1. In order to retrieve the nodes inserted in Q_1 the boolean variable $InQ_1(i)$ is defined for each node i, where $InQ_1(i)$=true if i has been inserted in Q_1, false otherwise. Note that for each node i such that $InQ_1(i)$=true, a shortest alternating path from i to t which does not include a cycle, is known.

For each node $j\in Q_1$, the "feasible" arcs of the backward star of j are scanned in order to evaluate new deviations and, possibly, update Q_2. The arc (i,j) is feasible (feas(i,j)=true) if $(i,j)\notin A^*$ and (i,j) is not an internal arc of a cycle related to a previously enumerated blocking path. An internal arc (i,j) of a cycle is not considered, since if there exists an s-t augmenting path which uses (i,j) then there exists an s-t augmenting path traversing i (not using (i,j)) with a length shorter than or equal to the length of the previous path (see Theorem 2.2). Initially feas(i,j)=true, $\forall(i,j)\in A'$; feas(i,j) is set to false when (i,j) grows G^* or when a cycle containing both i and j is discovered.

Let h be defined according to (3.1). For each feasible arc (i,j), the length $l(i,j)+\delta(i)-\delta(j)+\theta(h)$ of deviation (i,j) w.r.t. the path h, is computed and compared with $\varphi(i)$. Then if the length of the deviation (i,j) is shorter than the length $\varphi(i)$ of the current best deviation associated to the node i, then $\varphi(i)$ is updated and the new current best deviation is stored in Q_2.

The procedure which implements the updating of Q_2 starting from a node $j \in Q_1$ is shown in table 3. This procedure receives in input the node j, the path h defined by 3.1 and returns the set X which contains the improved nodes. The procedure makes use of a boolean function feasible(i,j) which returns false when feas(i,j)=false or when feas(i,j)=true but i, j are nodes of a cycle related to an enumerated path. In such a case feas(i,j) is set to false.

```
procedure Scanning(j,h,X);
  begin
    for each i∈BS(j) such that feasible(i,j) do
    begin
      if l(i,j)+δ(i)-δ(j)+θ(h)<φ(i)
      then begin
        dev(i):=j;
        φ(i):= l(i,j)+δ(i)-δ(j)+θ(h);
        Insert(X,i)
      end
    end
  end.
```

table 3

Note that the "for cycle" in the procedure Scanning(j,h,X) is the well-known updating of Bellman's equations in a shortest (first search) path tree routine [10] where:

- the cost of arc (i,j) is given by the reduced cost $l(i,j)+\delta(i)-\delta(j)$;
- $\theta(h)$ is the length of the shortest alternating path which does not include a cycle from j to t;
- $\varphi(i)$ is the length of an alternating path which does not include a cycle from i to t.

The procedure Find_Shortest_Augmenting_Path(T,P,Aug_path) is shown in table 4.

```
procedure Find_Shortest_Augmenting_Path(T,P,Aug_path);
begin
  Initialize;
  while Q2 not empty and not Aug_path do
  begin
    k:=k+1;
    w:=Select(Q2,best);
    Growing_Ft (k,w,v,φ(w),Q1,UB,Aug_path);
    if not Aug_path
    then begin
      UpdateBound(Q2,UB);
```

```
8          Empty(X);
9          while Q1 not empty and not Aug_path do
           begin
10           j:=Select( Q1,head);
11           if h(j)>0   then h:=h(j)
12                       else h:=path(cycle(mate(j)));
13           Scanning(j,h,X)
           end
        end;
14      UpdateSet(Q2,X)
      end
    end.
```

table 4

The procedure Initialize performs the following initialization:

i) k:=0; Aug_path:=false;

$cycle(i)=h(i)=next(i)=base(i)=0$, $InQ_1(i)=false$, $\varphi(i)=+\infty$, $\forall i \in N'$; $UB=+\infty$;

$feas(i,j)=true$, $\forall (i,j) \in A'$.

ii) $Q_1=\emptyset$, $Q_2=D$; $\varphi(d)=\delta(d)$, $\forall d \in D$;

$dev(d)=t$, $\forall d \in D$, $dev(i)=0$, $\forall i \notin D$.

The "while cycle" 2 terminates when a shortest augmenting path has been found or if the set of the augmenting paths is empty. The function Select (Q_2,best) returns arc (w,v) given by (3.2). This arc addresses the shortest deviation w.r.t. the nodes of N^*. Note that the procedure must evaluate the new shortest deviation from w w.r.t. the nodes which have been inserted in Q_1, that is min $\{\Delta(w,j)$: $InQ_1(j)=true$ and $(w,j) \notin A^*\}$ must be computed. In fact the node w has been removed from Q_2 and no deviation arc is related to w. Furthermore, during the enumeration schema, a cycle which includes both of the nodes of an arc inserted in Q_2 can be discovered. This implies that feasible(w,v)=false can occur and obviously such an arc must not be considered. Therefore Select (Q_2,best) must return the best arc (w,v) such that feasible(w,v)=true and for each selected arc the new shortest deviation must be computed, that is, Q_2 must be updated.

The procedure Growing_F_t(k,w,v,φ(w),Q_1,UB,Aug_path) constitutes the core of Find_Shortest_Augmenting_Path(T,P,Aug_path). This procedure performs the backtracking starting from w and updates all the parameters of the enumeration schema; in particular it grows F_t and updates Q_1. The procedure UpdateBound(Q_2,UB) removes from Q_2 the node i such that $\varphi(i) \geq UB$. The "while cycle" 9 prepares the updating of Q_2 starting from Q_1: the procedure Select (Q_1,head) returns the "head" of Q_1, the statements 11 and 12 perform (3.1). Finally UpdateSet(Q_2,X) updates Q_2 with X returned by the procedure

Scanning(j,h,X).

In the following the details of Growing_F_t(k,w,v,φ(w),Q_1,UB,Aug_path) are given in order to prove that the enumeration schema implemented by the procedure Find_Shortest_Augmenting_Path(T,P,Aug_path) is correct.

Consider case I_1). Note that this case occurs when i=mate(j), j being a previously backtracked node, i.e., j is the base of the cycle related to the blocking path k+1. Let i' be the node next to i in the backtracked path; let P(i',t) denote the sub-path from i' to t of the path k+1. Consider, in place of the path which would have been yielded by still backtracking on T, the blocking path given by the sub-path S(P(j,t)) and by the sub-path P(mate(j),t). The blocking path has the same length and, by exploiting symmetry, it can be found without further backtracking. Let U denote the following set of nodes:

U = {l∈ N': l belongs to the sub-path of P(i',t) from i' to w,
or mate(l) belongs to the sub-path of P(i',t) from i' to j and mate(l)≠mate(j)}.

The nodes of U are the only nodes which can improve deviations stored in Q_2, as stated by the following theorem.

Theorem 3.2

Assume that case I_1) is verified. If the nodes of U are inserted in Q_1 and Q_2 is updated starting from Q_1, then no augmenting path is contained in the set of shortest paths implicitly enumerated by the procedure.

Proof.

It is enough to observe that only the nodes of the path k+1 which are inserted in Q_1 can improve the deviations stored in Q_2. In fact, the deviation w.r.t. the nodes of the sub-path from v to t have been previously considered, while the deviations from the nodes in the sub-path from s to i include the cycle of the path k+1.Δ

Remark 3.2

The procedure adds to F_t the sub-path of P(i',t) from i' to v, if v∈ N_t. Otherwise, if v∈ N_s, the procedure adds to F_t the sub-path of P(i',t) from i' to w as a new tree rooted at w and sets next(w)=-v. Moreover the values h(l), l∈ Q_1 and θ(k+1), dest(k+1) are updated. The cycle related to the blocking path is stored, that is base(k+1), cycle(l), l∈ P(i',j) and path(base(k+1)) are updated. Note that for each node l: cycle(mate(l))=j a shortest augmenting path from l to t is known. Such a path is not explicitly stored in F_t, but it can be easily found by exploiting symmetry.

Consider case I_2). First of all the case when the path k+1 is not a simple path (i.e., it contains an even cycle), is considered. The following theorem, which can be easily proved, allows us to recognize a non-simple path.

Theorem 3.3

The path k+1 is not simple if and only if a node i such that i belongs to the shortest path on G* from v to t, is reached by backtracking on T.

Remark 3.3

The condition of theorem 3.3 can be easily rephrased in terms of F_t and F_s as follows.
If $v \in N_t$ then the path k+1 is not simple iff:

- $i \in N_t$ and
- dest(h(i))=dest(h(v)) and
- i belongs to the path in G* from v to t given by the procedure Sh_Alt_Path(j,t) (see table 2).

If $v \in N_s$ then the path k+1 is not simple iff:

- $i \in N_s$ and
- dest(h(mate(i)))=dest(h(mate(v))) and
- v belongs to the symmetric of the path from mate(i) to t given by the procedure Sh_Alt_Path(j,t);

or

- $i \in N_t$ and
- dest(h(i))=dest(h(mate(v))) and
- i belongs to the path in G* from cycle(mate(v)) to t given by the procedure Sh_Alt_Path(j,t).

Theorem 3.4

Assume that case I_2) is verified and the path k+1 is not simple. If the nodes of U = {$l \in N'$: l belongs to the sub-path of P(i',t) from i' to w} are inserted in Q_1 and Q_2 is updated starting from Q_1, then no augmenting path is contained in the set of shortest paths implicitly enumerated by the procedure.

Proof.

Easily derived from the proof of theorem 3.2.

Remark 3.4

The procedure adds to F_t the sub-path of P(i',t) from i' to v to F_t or from i' to w according to Remark 3.2. Then the values h(l), $l \in Q_1$ and $\theta(k+1)$, dest(k+1) are updated.

If case I_2) holds and theorem 3.3 is not verified, then the path k+1 can be an augmenting or a blocking path or a non-augmenting path (but not blocking, since it includes a cycle which is related to a previously enumerated blocking path), according to the following theorem.

Theorem 3.5

Assume that a node $i \in N_t$ or $i \in N_s$ is reached by backtracking on T and that the path k+1 is simple. Then:

- the path k+1 is augmenting iff the following conditions hold true:
  - a_1) if $h(i) \neq 0$, i.e., $i \in N_t$, then $dest(h(i)) \neq dest(h(v)+h(mate(v)))$ and $cycle(i) \neq 0$ and $h(i)=path(cycle(i))$;
  - a_2) if $h(mate(i)) \neq 0$, i.e., $i \in N_s$, then $dest(h(mate(i))) \neq dest(h(v)+h(mate(v)))$;
- the path k+1 is blocking iff the following conditions hold true:
  - b_1) if $h(i) \neq 0$, then $dest(h(i))=dest(h(v)+h(mate(v)))$ and $cycle(i) \neq 0$ and $h(i)= path(cycle(i))$;
  - b_2) if $h(mate(i)) \neq 0$, then $dest(h(mate(i)))=dest(h(v)+h(mate(v)))$;
- the path k+1 is not augmenting iff the following conditions hold true:
  - c_1) $h(i) \neq 0$ and $cycle(i) \neq 0$ and $h(i) \neq path(cycle(i))$;
  - c_2) $h(i) \neq 0$ and $cycle(i)=0$.

Proof.

(a1) and a2))

Note that either h(v)>0 ($v \in N_t$) or h(mate(v))>0 ($v \in N_s$); first consider case a1). In this case a node i of N_t which belongs to a cycle is reached starting from a node v of N_t or N_s. The condition h(i)=path(cycle(i)) states that the cycle which includes i is related to the shortest enumerated path traversing i. Consequently a sub-path from s to i which does not include a cycle is known. The path k+1 can be partitioned as follows: the sub-path from s to i, the arc (i,i'), where i' is the node next to i, the sub-path from i' to w, the arc (w,v) and the sub-path from v to t. The condition dest(h(i))≠dest(h(v)+h(mate(v))) implies that for each node l of the sub-path from s to i, mate(l) cannot belong to the sub-path from v to t; consequently, since the nodes in the sub-path from i' to w do not belong to a previously enumerated path and do not include pair of symmetric nodes it follows that the path k+1 is augmenting. The proof of case a2) is analogous.

Conversely if the path is augmenting then the sub-path from s to i is an alternating path which includes no cycles and no node l such that mate(l) is a node of the sub-path from v to t. Conditions a1) or a2) follow immediately.

(b1) and b2))

The path is not augmenting since it contains at least the pair of symmetric nodes l=dest(h(v)+h(mate(v))), mate(l). Furthermore the symmetric of the nodes of the sub-path of the path k+1 from mate(cycle(i)) to v do not belong to the path k+1. Therefore these nodes belong to a cycle which is new since the deviation arc is feasible. This implies that the path is blocking. Conversely if the path k+1 is blocking, conditions b1) or b2) trivially follow.

(c1) and c2))

It is enough to observe that conditions c1) or c2) imply that i is before (w.r.t. backtracking) the cycle related to the blocking path h(i).Δ

If conditions a_1) or a_2) are verified, a shortest augmenting path has been found and the procedure terminates. If conditions b_1) or b_2) are verified, let j=base(k+1) be the base of the cycle related to the path k+1. Consider the set U= {i: InQ_1(i)=false,i≠mate(j) and i belongs to the sub-path of the path k+1 from mate(j) to j or i belongs to the symmetric of this sub-path}. The nodes of U are the only nodes which can improve deviations stored in Q_2, as stated by the following theorem.

Theorem 3.6

Assume that the condition b_1) or b_2) of theorem 3.5 is verified. If the nodes of U are inserted in Q_1 and Q_2 is updated starting from Q_1, then no augmenting path is contained in the set of shortest paths implicitly enumerated by the procedure.

Proof.

Note that deviation arcs which have not been previously considered refer only to the nodes of U. Δ

Remark 3.6

The sub-path of P(i',t) from i' to v (or from i' to w) is added to F_t. The values h(l), $l \in Q_1$, θ(k+1), dest(k+1) and base(k+1) must be updated. Each node l of the new cycle which is not contained by any previously discovered cycle and belongs to F_t must be stored, i.e., cycle(l)=base(k+1).

Note that if the conditions c_1) or c_2) of theorem 3.5 are verified, then the results of Theorem 3.4 and Remark 3.4 can also be applied.

When the condition c_1) is verified, then the path k+1 is not augmenting and its length is given by $\theta(k+1)=l(w,v)+\delta(w)-\delta(v)+\theta(h(v)+h(mate(v)))$. Note that in this case the path k+1 allows us to determine an augmenting path (see fig. 6a) or a blocking path (see fig. 6b) of a length greater than or equal to θ(k+1), according to the following theorem.

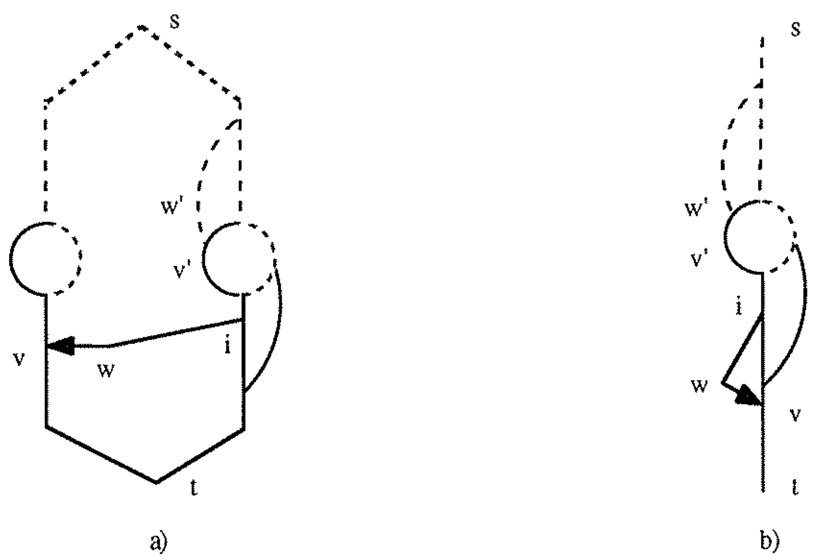


fig. 6

Theorem 3.7

Assume that the condition c_1) of theorem 3.5 holds. If dest(h(i))=dest(k+1), then a new blocking path is found. Otherwise, if dest(h(i))≠dest(k+1), then an augmenting path is found. In both cases this path is given by the sub-path from s to i of the path h=path(cycle(i)), the arc (i,i') and the sub-path of the path k+1 from i' to t. Its length is given by $l(w,v)+\delta(w)-\delta(v)+\theta(h)\geq\theta(k+1)$.

Proof.

Note that condition c1) of theorem 3.5 implies that node i is before the cycle of path h(i) but is a node of the cycle related to the path h. It follows that the sub-path from s to i of the path h is alternating with no cycle. The length of the path given by the sub-path of the path h from s to i, by the arc (i,i') and by the sub-path of the path k+1 from i' to t is given by $l(w,v)+\delta(w)-\delta(v)+\theta(h)$. If dest(h(i))=dest(k+1), then condition b1) of theorem 3.5. holds and the path is blocking; if dest(h(i))≠dest(k+1), then condition a1) holds and the path is augmenting. Δ

Remark 3.7

Let P denote the path shown by theorem 3.7. If P is augmenting, then the length of P is an upper bound to the length of the shortest augmenting path. Hence, in this case, P can be immediately enumerated by updating the upper bound UB. If P is blocking it will be enumerated by the procedure as soon as its length becomes the current shortest length.

The procedure Growing_F_t (k,w,v,φ(w),Q_1,UB,Aug_path) (see table 5) implements the backtracking according to theorems from 3.1 to 3.7, starting by the deviation arc (w,v), which refers the path k+1 of length equal to φ(w).

```
Procedure Growing_F_t (k,w,v,φ(w),Q_1,UB,Aug_path);
  begin
  θ(k):=φ(w);
  dest(k):=dest(h(v)+h(mate(v)));
  i:=w;
  j:=v;
  if v∈N_S then j=-j;                           {w is a root of F_t}
  while  mate(i) ∉ Q_1 and                      {not case I_1)}
         (h(i)+h(mate(i)) =0) and               {not case I_2)}
         i≠s  do                                {not case I_3)}
    begin
      feas(i,j):=false;
      Insert(Q_1,i);
      h(i):=k;
      next(i):=j;
      j:=i;
      i:=pred(i)
    end;
  feas(i,j);=false;
  if 3.4 does not hold                          {the path is simple}
  then  if i= s or 3.5 a_1) holds or 3.5 a_2) holds    {the path is augmenting}
        then Aug_path:=true
```

```
        else   if mate(i)∈Q1 or 3.5 b1) holds or 3.5 b2) holds   {the path is blocking}
               then FindCycle(Q1)
               else if 3.5c1) holds and dest(h(i))≠dest(k)         {the path is not augmenting}
                    then UpdateBound(UB)
    end.
```

table 5

The procedure Insert(Q_1,i) inserts node i in Q_1 and sets InQ_1(i)=true.

The procedure UpdateBound(UB) updates the upper bound UB, if case c_1) is verified and dest(h(i))≠dest(k+1) (see theorem 3.7).

Finally the procedure FindCycle(Q_1) finds the new cycle, updates the parameters related to the cycle and updates Q_1. Two cases can be distinguished.

a) mate(i) $\in Q_1$. In this case all the nodes of the cycle have been found by the procedure Growing_F_t(k,w,v,φ(w),Q_1,UB,Aug_path) and the procedure FindCycle(Q_1) sets base(k+1)=mate(i) and cycle(j)=mate(i), for each node j in the sub-path from the node next to i to mate(i), j≠mate(i). The nodes of the symmetric of the sub-path of the path k+1 from i to the node before mate(i) are inserted in Q_1.

b) Conditions b1) or b2) of theorem 3.5 hold. In this case, if i$\in N_t$ (see fig. 7), let b be the first node common to the sub-path of the path h given by (3.1), from mate(v) and to the sub-path of the path h(i) from i.

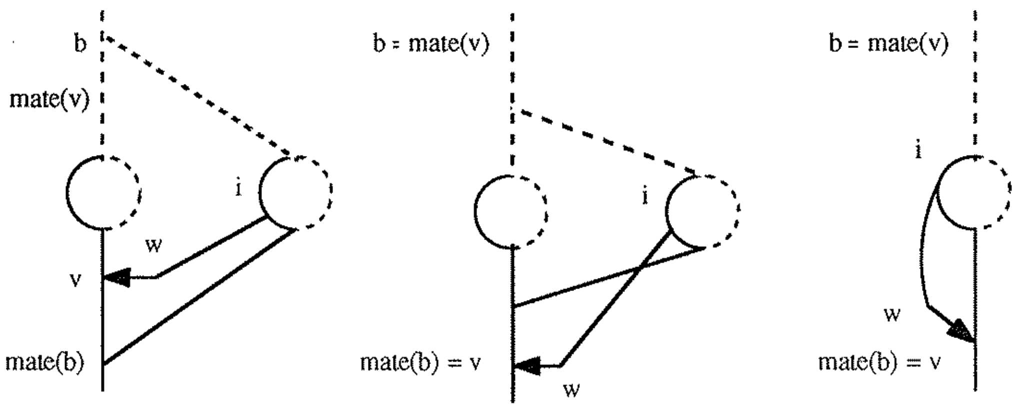


fig. 7

Otherwise, if i$\in N_s$, let b be the first node common to the sub-path of the path h given by (3.1), from mate(v) and to the sub-path of the path path(cycle(mate(i))) from i. The base of the new cycle is mate(b) and the new cycle has been completely identified.After discovering the nodes of the cycle, cycle(j)=mate(b) is assigned to each node j in the cycle such that j≠mate(b), cycle(j)=0 and h(j)≠0, that is $j \in N_t$.

4 Correctness and worst case complexity

The correctness of the procedure Parametric_Matching given in table 1 is provided by the following theorem.

Theorem 4.1

The procedure Find_Shortest_Augmenting_Path(T,P,Aug_path) determines in O(nm) the shortest augmenting path or finds that the current matching is a maximum cardinality matching.

Proof.

First note that the current matching is a maximum cardinality matching when Find_Shortest_Augmenting_Path(T,P,Aug_path) returns Aug_path=false, that is, the procedure terminates due to the condition Q_2 = empty without having found an augmenting path. Secondly the procedure halts after a finite number of steps, since it enumerates, at most, all the alternating paths; the correctness of the enumeration schema directly follows from the results of the previous section. Note that at most m selections from Q_2 can be executed, since we consider at most m deviation arcs. Therefore the while cycle 2 (see table 4) is repeated at most m times. Furthermore the complexity of Select(Q_2,best) is O(n) including the evaluation of the new shortest deviation from w. Then the overall complexity of Select(Q_2,best) is O(nm). Finally, since each node can be inserted in (and successively selected from) Q_1 at most once, the overall complexity of cycle 9 (see table 4) is O(m).Δ

The complexity of the procedure Parametric_Matching is given by $O(n(\alpha+\beta))$, where α is the complexity of the procedure SPT(T) and β is the complexity of the procedure Find_Shortest_Augmenting_Path(T,P,Aug_path). Therefore, since α=O(nm), the complexity of Parametric_Matching is $O(n^2m)$. It is worthwhile to remark that α could be reduced by using dual variables and relative costs. Improving the crude complexity analisys which provides β, is an open problem.

References

[1] Carraresi, P. and Sodini, C., "An efficient algorithm for the bipartite matching problem", **E.J.O.R., 23**, 86-93, (1986).

[2] Carraresi, P. and Sodini, C., "A binary enumeration tree to find k shortest paths", **Methods of Operations Research, 45**, 178-188, (1985).

[3] Derigs U.,: "A shortest Augmenting Path Method for Solving Minimal Perfect Matching Problem", **Networks, 11**, (1981).

[4] Derigs U.,: "A short note on matching algorithm", **Mathematical Programming Study, 26**, 200-204,(1986).

[5] Derigs U.: "Solving non-bipartite matching problems via shortest path techniques", to appear in FORTRAN codes for network optimization, **Annals of Operations Research**,(1988)

[6] Edmonds, J., "Maximum Matching and a Polyhedron with {0-1} vertices", **J. Res. NBS, 69B**, (1965).

[7] Edmonds, J. and Karp, R., "Theoretical improvements in algorithmic efficiency for network flow problems", **Journal of the ACM, 19**,248-264, (1972).

[8] Fredman, M. and Tarjan, R., " Fibonacci Heaps and Their Uses", **Journal of the ACM 34**, 596-615, (1987).

[9] Galil, Z., Micali, S. and Gabow, H.,"Priority queues with variable priority and an O(|E||V|log|V|) algorithm for finding maximal weighted matching in general graphs", **Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science**, 255-261, (1982).

[10] Gallo G. and Pallottino S.: "Shortest Path Methods: A Unifying Approach", **Math. Progr. Study, 26**, (1986).

[11] Lawler, E. L., "A procedure for computing the k best solutions to discrete optimization problems and its applications to the shortest path problem", **Management Science, 18**,(1972)

[12] Lessard, R. Minoux, M. and Rousseau, J.-M.,"A new algorithm for general matching problems using network flow subproblems", Centre de recherche sur les transports Université de Montréal, #423, (1987) (to appear in **Networks**)

[13] Lovasz, L. and Plummer, M.D., "Matching theory", **Annals of discrete mathematics, 29**, (1986)

[14] Tarjan, R., "Data Structures and Network Algorithms", **SIAM**, Philadelphia, (1983).

PUBBLICAZIONI
del
Dipartimento di Statistica e Matematica Applicata all Economia

Report n. 1 - Some Optimality Conditions in Vector Optimization.(A.Cambini- L.Martein), 1987

Report n. 2 - On Maximizing a Sum of Ratios. (A.Cambini-L.Martein-S.Schaible), 1987

Report n.3 - On the Charnes-Cooper Transformation in Linear Fractional Programming. (G.Gasparotto), 1987

Report n. 4 - Non-linear Separation Theorems, Duality and Optimality. (A.Cambini), 1987

Report n. 5 - Indicizzazione parziale: aspetti metodologici e riflessi economici. (G.Boletto), 1987

Report n. 6 - On Parametric Linear Fractional Programming. (A.Cambini-C.Sodini), 1987

Report n. 7 - Alcuni aspetti meno noti delle migrazioni in Italia. (A.Bonaguidi), 1987

Report n. 8 - On Solving a Linear Program with one Quadratic Constraint. (L.Martein-S.Schaible), 1987

Report n. 9 - Alcune osservazioni sull'equazione funzionale $\phi(x,y,z) = \phi(\phi(x,y,t),t,z)$. (E.Lari), 1988

Report n.10 - Une étude par ménage des migrations des personnes âgées: comparaison des résultats pour l'Italie et les Etats-Unis. (F.Bartiaux), 1988

Report n.11 - Metodi di scomposizione del tasso di inflazione (G.Boletto), 1988

Report n.12 - A New Algorithm for the Strictly Convex Quadratic Programming Problem. (C.Sodini), 1988

Report n.13 - On Generating the Set of all Efficient Points of a Bicriteria Fractional Problem. (L.Martein), 1988

Report n.14 - Applicazioni della programmazione frazionaria nel campo economico-finanziario. (L.Martein), 1988

Report n.15 - On the Bicriteria Maximization Problem. (L.Martein), 1988

Report n.16 - Un prototipo di sistema esperto per la consulenza finanziaria rivolta ai piccolirisparmiatori.(P.Manca), 1988

Report n.17 - Operazioni finanziarie di Soper e operazioni di puro investimento secondo Teichroew Robichek-Montalbano. (P.Manca), 1988

Report n.18 - A k-Shortest Path Approach to the Minimum Cost Matching Problem. (P.Carraresi-C.Sodini), 1988