Discrete Applied Mathematics 18 (1987) 155-165
North-Holland

ALGORITHMS FOR FINDING *K*-BEST PERFECT MATCHINGS*

Chandra R. CHEGIREDDY and Horst W. HAMACHER

Department of Industrial & Systems Engineering and Center for Optimization and Combinatorics, University of Florida, Gainesville, FL 32611, USA

Received 6 March 1987

In the K-best perfect matching problem (KM) one wants to find K pairwise different, perfect matchings $M_1, \ldots, M_k$ such that $w(M_1) \geq w(M_2) \geq \cdots \geq w(M_k) \geq w(M)$, $\forall M \neq M_1, M_2, \ldots, M_k$. The procedure discussed in this paper is based on a binary partitioning of the matching solution space. We survey different algorithms to perform this partitioning. The best complexity bound of the resulting algorithms discussed is $O(Kn^3)$, where n is the number of nodes in the graph.

1. Introduction

Let $G=(V, E)$ represent an undirected graph with vertex set V and edge set E. A matching $M \subseteq E$ has the property that no two edges in M share the same node. Let $|S|$ denote the cardinality of any set S. A matching is said to be perfect if $|M| = |V|/2$. We assume without loss of generality that $|V|$ is even and that a perfect matching always exists. Let $w: e \to \mathbb{R}_+ \cup \{0\}$ be a weight function assigning nonnegative weights for all $e \in E$. (If some of the weights $w(e)$ are negative, we can compute $W = \min\{w(e) \mid e \in E\}$ and update all weights by $w(e) = w(e) - W$ to make them nonnegative.) The weight of any matching M is denoted by $\sum_{e \in M} w(e)$. The problem of finding the *K-best perfect matchings* (KM) can now be stated as finding all distinct perfect matchings $M_1, \ldots, M_k$ such that $w(M_1) \geq w(M_2) \geq \cdots \geq w(M_k) \geq w(M)$, $\forall M \neq M_1, M_2, \ldots, M_k$. We discuss algorithms for finding such K-best perfect matchings given G and positive integer K.

In Section 2 we introduce a general procedure to solve KM which is formulated in terms of finding second best solutions in some restricted solution spaces. We present the complexity of the general procedure and propose alternative implementations for finding such second best solutions in Section 3. For each of these alternatives, we present the worst case complexity of solving KM. In Section 4, the special case of bipartite matchings is considered.

* This research was partially supported by NSF grant No. ECS 8412926

2. General algorithm for K-best perfect matchings

The general algorithm for finding the K-best perfect matchings can easily be obtained from the algorithm for any K-best combinatorial optimization problem [12, 13]. Lawler [13] presented a generalization of Murty's algorithm [15] for ranking the solutions of an assignment problem, and their partitioning strategy is slightly different from the way the solution space is partitioned in the procedure presented by Hamacher and Queyranne [12].

Let Ω denote the whole solution space and let sets I and O be subsets of E. We define a restricted solution space $\Omega_{I,O}=\{M: M$ a perfect matching, $I\subseteq M$, $O\cap M=\emptyset\}$. The algorithm builds the sets I and O iteratively and finds the second best matchings in each of the two branches created in every iteration, as shown in Fig. 1.

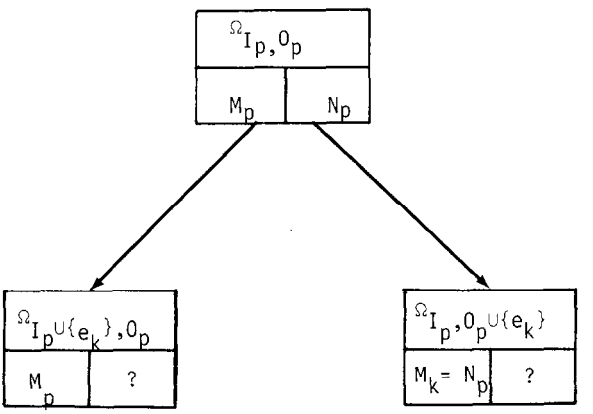

Fig. 1. Partitioning of solution space.

Let M_p and N_p be the best and second best matchings at any node selected in iteration k for further partitioning. We choose an element in $M_p - N_p$, say e_k, and set $I_{k+1}=I_p$, $I_p=I_p\cup\{e_k\}$ and $O_{k+1}=O_p\cup\{e_k\}$ (see Fig. 1). By iteratively applying such a branching procedure we obtain the following algorithm for KM.

K-Best Matching Algorithm (AKM)

Input. $G=(V, E)$ undirected graph; $w(e)\geq 0$, $e\in E$ edge weights;
K total number of ranked solutions desired

Output. K-best matchings $M_1, \ldots, M_k$

Start. $I_1=0_1=\emptyset$; $k=1$

Step 1. Find best matching M_1 and second best matching N_1 in G.

Step 2. Let $w(N_p)=\min\{w(N_i)\,|\,i=1, \ldots, k\}$
If $w(N_p)=-\infty$
Then stop {Only $k-1$ matchings exist}

Else $M_k = N_p$
If $k = K$
Then stop
Else Choose $e_k \in M_p - N_p$

Step 3. Update sets I and O
$I_{k+1} = I_p$, $I_p = I_p \cup \{e_k\}$, $O_{k+1} = O_p \cup \{e_k\}$
Set $k = k+1$

Step 4. Compute the second best matchings N_p and N_k in $\Omega_{(I_k, O_k)}$ and $\Omega_{(I_p, O_p)}$ resp.
Goto Step 2

The fact that this algorithm indeed gives us the K-best perfect matchings can be verified very easily. We can observe that at each iteration k, $\{\Omega_{I_i, O_i} \mid i = 1, \ldots, k\}$ is a partition of the solution space Ω. We are choosing the best solution among all these restricted solution spaces excluding $M_1, \ldots, M_{k-1}$, yielding the desired result, namely M_k.

We see that the algorithm AKM involves finding the second best matchings in some restricted solution spaces as specified in Step 4. If the complexity of solving one such second best matching problem is A_n, the overall complexity of KM is of the order $O(n^3 + (K-1)A_n)$, since at most $(K-1)$ many such problems need to be solved and since the complexity of solving the best perfect matching is $O(n^3)$ [8, 14, 5]. We propose three different approaches to solve such a problem in the next section. For the special case of bipartite matchings, one of these approaches, the shortest alternating cycle approach is detailed in Section 4.

3. Algorithms for second best perfect matchings

In this section we outline different approaches to obtain a second best perfect matching in a solution subspace $\Omega_{I,O}$ given the best matching in that subspace. For this purpose, we transform this problem into finding a second best matching in a graph without any restrictions by restructuring the graph as follows. We delete all the end nodes of $e \in I$ and their incident edges. For the edges $e \in O$, we set $w(e) = -\infty$. It can be easily seen that the second best solution of this subproblem together with the edges in I, provided its weight is larger than $-\infty$, will yield the second best solution in $\Omega_{I,O}$. From now on, we assume that the graph $G = (V, E)$ has been transformed as above and we will deal with the problem of finding a second best perfect matching without any restrictions. The general algorithm can be thought of as one partitioning step in the algorithm of Lawler and Murty [13, 15].

Second Best Matching Algorithm (ASBM)

Input. G graph; $M_1 = \{e_1, \ldots, e_n\}$ best matching
Output. M_2 second best matching

Step 1. $i=1$
Step 2. Delete $\{e_j \mid j<i\}$ and all incident nodes and adjacent edges from G.
Step 3. Set $w(e_i)=-\infty$
Step 4. Find best matching N_i in the new graph
Step 5. Let $N_i=N_i \cup \{e_j \mid j<i\}$
Step 6. **If** $i<n$
Then $i=i+1$; **goto** Step 2
Else $M_2=N_p$ where $w(N_p)=\max\{w(N_j) \mid j=1,\ldots,n\}$.
Comment. If $w(N_p)=-\infty$, then G has only a single perfect matching.

We can see that the essential idea of this algorithm is to partition the solution space by forbidding one by one edges in the best matching and forcing in all the edges considered before into the solution. We can derive different algorithms from this basic procedure as follows.

Solving from Scratch. Step 4 of the above algorithm is a matching problem. An obvious implementation is to solve each problem from scratch, making each iteration an $O(n^3)$ procedure. Since we have at most $n/2$ many such iterations, the complexity of ASBM is $O(n^4)$. From the analysis in Section 2, we obtain the overall complexity of AKM to be $O(n^3+(K-1)n^4)$, which is $O(Kn^4)$. This bound can be improved.

Sensitivity Analysis Approach. We describe the application of sensitivity analysis to obtain a second best matching in the situation where one edge weight is set to $-\infty$ and that edge is in the matching. If we denote with Γ the family of all odd node sets with cardinality not less than three and if $\delta(T)=\{(i,j)\in E \mid i\in T, j\notin T\}$, $\alpha(T)=\{(i,j)\in E \mid i \text{ and } j\in T\}$, $\forall T\in\Gamma$, then the linear programming (LP) formulation of the matching problem [8] can be stated as

$$\text{Maximize} \quad \sum_{(i,j)\in E} w_{ij}X_{ij}$$

$$\text{subject to} \quad \sum_{j:(i,j)\in E} X_{ij}=1 \qquad \forall i,$$

$$\sum_{(i,j)\in\delta(T)} X_{ij}\geq 1 \qquad \forall T\in\Gamma,$$

$$X_{ij}\geq 0 \qquad \forall (i,j)\in E.$$

It is well known that this LP has a 0–1 optimal solution M^* and $X^*_{ij}=1$ if and only if $(i,j)\in M^*$.

The dual of this linear program is

$$\text{minimize} \quad \sum_{i\in V}\mu_i+\sum_{T\in\Gamma}\mu_T$$

$$\text{subject to} \quad \mu_i+\mu_j+\sum_{(i,j)\in\delta(T),\, T\in\Gamma}\mu_T\geq w_{ij} \qquad \forall (i,j)\in E,$$

$$\mu_T\leq 0 \qquad \forall T\in\Gamma.$$

The complementary slackness conditions become

$$X_{ij}=1 \rightarrow w_{ij}-\mu_i-\mu_j-\sum_{(i,j)\in\delta(T),\ T\in\Gamma}\mu_T=0 \quad \forall (i,j)\in E,$$

$$\mu_T=0 \quad \text{or} \quad \sum_{(i,j)\in\delta(T)} X_{ij}=1 \qquad \forall T\in\Gamma.$$

We define new weights $w'_{ij}=W-w_{ij}$, where $W>w_{ij}$, for all $(i,j)\in E$. With this weights, we can use the Shortest Augmenting Path procedure (SAP) [5] to solve the above problem and obtain the optimal solution, which is primally and dually feasible and satisfies the complementary slackness conditions. Let us denote by (r, t), the edge whose weight is being changed to $-\infty$. We set $w'_{rt}=W-w_{rt}=\infty$ and adopt the sensitivity analysis procedure of Derigs [6], to find the next best matching as follows.

Step 1. **If** the vertex r is shrunk in a pseudonode
Then Expand the blossoms containing r
Modify the dual variables accordingly.

Step 2. Set $w'_{rt}=\infty$, $\delta=w'_{rt}-\mu_r-\mu_t-\sum_{(r,t)\in\delta(T)}\mu_T=\infty$, $\mu_r=\mu_r+\delta=\infty$

Step 3. Make vertex r the new root and grow an alternating tree, using the SAP procedure.

In Step 1, the expansion of all the blossoms containing node r, can be done as follows [6]. We add two nodes a and b and two edges (a, r) and (b, t) with the dual variables and the edge weights as shown in Fig. 2 and find an alternating path between the two nodes a and b using the SAP procedure.

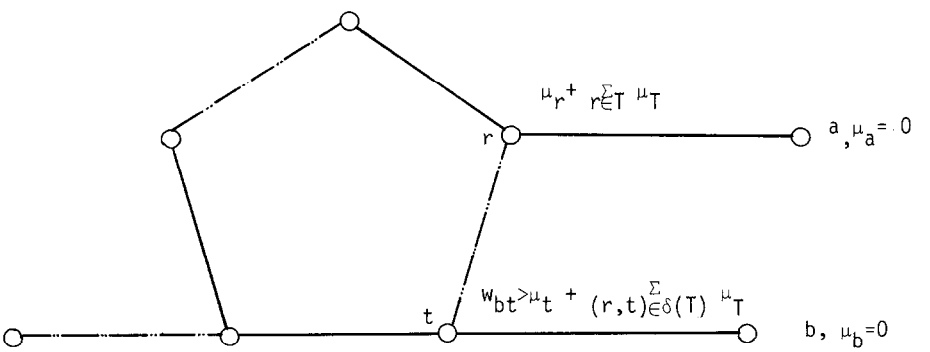

Fig. 2. Auxiliary graph to expand blossoms containing r.

The edge weight w_{bt} is set such that when edge (b, t) becomes part of the alternating path, all the blossoms containing r are expanded. A detailed discussion of how the edge weights affect the expansion of blossoms is given in Weber [17].

The advantage of this procedure as opposed to the solution from scratch lies in

the fact that finding the second best matching involves at most two shortest augmenting path iterations, which are $O(n^2)$ operations. The complexity of ASBM reduces to $O(n^3)$ making the overall complexity of algorithm AKM $O(Kn^3)$.

An alternative implementation of the sensitivity analysis is to use the primal algorithm of Cunningham and Marsh [2]. We make use of an equivalent blossom inequality in the LP formulation, which can be stated as

$$\sum_{(i,j)\in\alpha(T)} X_{ij} \leq \tfrac{1}{2}(|T|-1) \quad \forall T \in \Gamma.$$

This will result in a dual constraint $\mu_i + \mu_j + \sum_{T\in\Gamma,(i,j)\in\alpha(T)} \mu_T \geq w_{ij}$, and the complementary slackness conditions become

$$X_{ij} = 1 \rightarrow w_{ij} = \mu_i + \mu_j + \sum_{T\in\Gamma,(i,j)\in\alpha(T)} \mu_T - w_{ij} = 0,$$

$$\mu_T > 0 \rightarrow \sum_{(i,j)\in\alpha(T)} X_{ij} = \tfrac{1}{2}(|T|-1).$$

When we set $w_{rt} = -M$ (M a large integer) the complementary slackness condition $\bar{w}_{rt} = -w_{rt} + \mu_r + \mu_t + \sum_{(r,t)\in\alpha(T)} \mu_T = 0$ is obviously violated. We update μ_r by $\mu_r = -(\mu_t + \sum_{(r,t)\in\alpha(T)} \mu_T + M)$ to reestablish it. Therefore we get a pair of primal and dual solutions of the matching LP which is primally feasible, satisfies the complementary slackness conditions and violates the dual constraint only for the edges with r as an end point. A single iteration of Cunningham and Marsh's primal algorithm will therefore yield an updated optimal solution in $O(n^2)$. The resulting implementation of AKM is therefore of order $O(Kn^3)$. Both the implementations of the sensitivity analysis approach are an order of magnitude better than solving the second best matching problems from scratch for small values of K (i.e., polynomial in n). If K itself is exponential, which is not uncommon in some applications, the effect of this reduction will be substantial in terms of actual computational time.

Shortest Alternating Cycles. A path from node i to node j is the sequence of edges from i to j. A path is said to be a cycle if i and j happen to be the same node. A path is said to be *alternating* with respect to a given matching M, if the edges in the path alternately are in the matching M and not. The symmetric difference of two sets is indicated by the symbol $\ominus$ and is the set of all elements contained in either one of them but not in both. We can obtain the second best matching by interchanging the matched and unmatched edges along the shortest weight alternating cycle with respect to the best matching as illustrated by the following theorem.

Given a matching M and an alternating cycle C with respect to M, define the incremental weight $\delta(C)$ as

$$\delta(C) = w(C \cap M) - w(C - M)$$

where $\delta(C)$ indicates the increase or decrease in the weight of the resulting matching when the edges along C are interchanged in M. The shortest alternating cycle with

respect to M is the alternating cycle whose $\delta(C)$ is minimal. The following theorem shows the use of shortest alternating cycles in finding second best matchings.

Theorem 1. *Let M_1 be the best matching in G and let C_1 be the shortest alternating cycle. Then the symmetric difference $M_1 \ominus C_1$ is a second best matching in G.*

Proof. Let M be any matching different from M_1 in G. Since both M and M_1 are perfect matchings the $M_1 \ominus M$ decomposes into pairwise disjoint cycles $D_1, \ldots, D_q$ which alternately contain edges in M and M_1. Now the weight of M can be written as $w(M) = w(M_1) - \delta(D_1) - \cdots - \delta(D_q)$. Since C_1 is the shortest alternating cycle with respect to M_1 we have $\delta(C_1) \leq \delta(D_i)$ for $i = 1, \ldots, q$. Hence $w(M_1 \ominus C_1) = w(M_1) - \delta(C_1) \geq w(M)$, making $M_1 \ominus C_1$ a second best matching. Notice that the shortest alternating cycle C_1 of M_1 satisfies $\delta(C_1) \geq 0$. Otherwise a contradiction to the optimality of M_1 would follow. □

An immediate consequence of the preceding theorem is to reduce the problem of finding the second best matching to the problem of finding the shortest alternating cycle. There are some algorithms to find the shortest alternating cycles in general graphs developed by Brown [3] but they are very tedious computationally as they employ a Branch and Bound strategy and have a worst case bound which is exponential. We can use the sensitivity analysis approach by forbidding each of the edges in the matching in turn, and combine it with the shortest alternating path obtained in that iteration, to find the shortest alternating cycle involving that particular edge. The minimum over all such cycles yields the shortest alternating cycle with respect to the best matching. But this process will not improve the worst case bound or the number of computations required. For the special case of bipartite matchings, we can reduce the problem of finding a shortest alternating cycle to that of finding a shortest cycle in a transformed graph. This is demonstrated in the next section.

4. *K*-best bipartite matchings

The graph G is called a bipartite graph if the nodes can be partitioned into two sets S and T, such that no two nodes in S and T are adjacent, i.e., all edges extend between S and T. We denote such a bipartite graph as $G = (S, T, E)$ where $V = S \cup T$.

We make use of the matching M to construct a directed graph $G' = (S, T, E')$ from G. The node sets remain the same and the arc $e' \in E'$ is oriented from S to T if edge e is not in M and the cost of e', denoted by $c(e') = -w(e)$. The arcs e' in E' corresponding to $e \in M$ get an orientation from T to S and a cost $c(e') = w(e)$.

Theorem 2. *A graph G' constructed as above with respect to the best matching M_1, does not contain any negative cycles.*

Proof. Suppose it does. Notice that any cycle in G' corresponds to an alternating cycle in G, because we can leave S only on unmatched edges and get back to S on matched edges which are the only arcs with the reverse orientation and the weight of this cycle is equal to the incremental weight $\delta(C)$ of Theorem 1. We also know that we can augment along this cycle and obtain a different matching M_2 with $w(M_2) > w(M_1)$ which contradicts the assumption that M_1 is the best matching. □

Finding Shortest Cycles in G'. For bipartite graphs we have reduced the problem of finding a shortest alternating cycle in G to the problem of finding a shortest cycle in G' where some edge weights are negative but no negative cycles exist. We can therefore apply the Floyd–Warshall algorithm to find shortest paths between all pairs of nodes [10]. The cycle corresponding to the minimal weight diagonal entry after the final iteration will yield the shortest cycle. The computational complexity of the Floyd–Warshall algorithm is $O(n^3)$ and it is well known that this bound is tight. It also has a storage requirement of $O(n^2)$.

A second approach involves finding the shortest distance between nodes p and q such that $(p, q) \in M$. If $d(p, q)$ denotes the distance of the node p to node q, $d(p, q) + c_{qp}$ will give us the length of the shortest cycle including the nodes p and q. We can find the shortest distance for each such pair of nodes $(p, q) \in M$ and the minimum over all the $d(p, q) + c_{qp}$ will yield the shortest cycle. This approach can also be classified as a sequential all pairs problem [11]. This involves one iteration of a label correcting algorithm [16], which allows the edge costs to be non-positive, to compute the shortest path tree rooted at an arbitrary node. The node labels correspond to the dual variables in the following linear programming formulation of the shortest path problem from the node p to node q:

$$\text{minimize} \quad \sum_{ij} c_{ij} X_{ij}$$

$$\text{subject to} \quad \sum_j X_{ij} - \sum_k X_{ki} = \begin{cases} 1 & \text{if } i = p, \\ 0 & \text{if } i \neq p \text{ and } i \neq q, \\ -1 & \text{if } i = q, \end{cases}$$

$$X_{ij} \geq 0 \quad \forall i, j.$$

The dual of this problem is

$$\text{maximize} \quad \pi_p - \pi_q$$

$$\text{subject to} \quad \pi_i - \pi_j \leq c_{ij} \quad \forall i, j,$$

$$\pi_i \text{ unrestricted.}$$

After the first iteration we make use of the node labels or dual variables to modify the costs to be non-negative using the transformation $c'_{ij} = c_{ij} + \pi_i - \pi_j$ [1, 9], to make all the costs non-negative. This transformation leaves the weight of the cycles unchanged but enables us to apply a label setting algorithm (e.g. [7]) to the remain-

ing pairs of nodes. The formal algorithm can be stated as follows:

Label Correcting – Label Setting Algorithm (ALCLS)

Input. $G'=(V, E)$ graph; $c: E \rightarrow R$ cost function

Step 1. Compute shortest path tree $T(p)$ rooted at any node $p \in V$ using a Label Correcting algorithm.
Let $d(p, i)$ be the distance of node i from p.
Let π_i be the optimal dual variables.
The length of the minimum cycle including p is given by $d(p, q)+c_{qp}$ such that $(p, q) \in M$.
Set $c'_{ij}=c_{ij}+\pi_i-\pi_j$ to transform the costs such that $c'_{ij} \geq 0$.

Step 2. For $i \in S$, $i \neq p$ **repeat**
Apply the label setting algorithm with c'_{ij} as the new costs to find the shortest distances between i and j, $\forall\{(i, j) \mid (i, j) \in M\}$.
Obtain the length of each minimum cycle.

Step 3. Find the minimum weight of all the cycles and the cycle corresponding to that will be the shortest cycle.

This algorithm will perform better than the Floyd-Warshall algorithm, since computationally label correcting and label setting algorithms are faster when compared to the Floyd-Warshall algorithm [4]. The worst case complexity is the summation of the bound for one label correcting iteration which is $O(n^3)$ and at most $n/2$ iterations of a label setting algorithm each of order $O(n^2)$, resulting in an complexity of $O(n^3)$ for obtaining a second best matching.

In the last approach we avoid the label correcting algorithm by using the linear programming formulation of the weighted bipartite matching problem which is

$$\begin{aligned} \text{maximize} \quad & \sum_{i,j} w_{ij} X_{ij} \\ \text{subject to} \quad & \sum_{j} X_{ij}=1 \quad \forall i, \\ & \sum_{i} X_{ij}=1 \quad \forall j, \\ & X_{ij} \geq 0 \qquad \forall (i, j) \in E. \end{aligned}$$

The dual of this linear program is

$$\begin{aligned} \text{minimize} \quad & \sum_{i} \mu_i + \sum_{j} \sigma_j \\ \text{subject to} \quad & \mu_i + \sigma_j \geq w_{ij} \quad \forall (i, j) \in E, \\ & \mu_i,\ \sigma_j \text{ unrestricted.} \end{aligned} \tag{1}$$

The complementary slackness condition is

$$X_{ij}=1 \rightarrow \mu_i+\sigma_j=w_{ij} \quad \forall (i, j) \in E. \tag{2}$$

Since M_1 is the best matching in $G=(S, T, E)$ and μ_i and σ_j are the optimum dual matching variables for $i \in S$, $j \in T$, we have

$$\mu_i + \sigma_j \geq w_{ij} \quad \text{or} \quad -w_{ij} + \mu_i + \sigma_j \geq 0 \tag{3}$$

from the dual feasibility of equation (1).

Moreover $(i, j) \in M_1$ implies $X_{ij} = 1$ and therefore

$$w_{ij} - \mu_i - \sigma_j = 0 \tag{4}$$

from the complementary slackness condition. Recall that in the construction of the G' for transforming the alternating cycle problem to the shortest cycle problem we defined

$$c_{ij} = \begin{cases} w_{ij} & \text{if } (i, j) \in M_1, \\ -w_{ij} & \text{if } (i, j) \notin M_1 \end{cases} \tag{5}$$

and we want to find a feasible dual vector π_i, π_j for $i \in S$, $j \in T$ such that $c_{ij} - \pi_i + \pi_j \geq 0$ for the shortest path problem. Now we define

$$\begin{aligned} \pi_i &= -\mu_i \quad \text{if } i \in S, \\ \pi_j &= \sigma_j \quad \text{if } j \in T. \end{aligned} \tag{6}$$

There are two cases possible here.

Case 1: $i \in S$, $j \in T$. From (5) and (6) we have

$$c_{ij} - \pi_i + \pi_j = -w_{ij} + \mu_i + \sigma_j \geq 0,$$

Case 2: $i \in T$, $j \in S$. From (4), (5) and (6) we have

$$c_{ij} - \pi_i + \pi_j = w_{ij} - \sigma_i - \mu_j = 0.$$

Hence the π_i defined as in (6) are dually feasible for the shortest path formulation. We can now revise the c_{ij} to $c'_{ij} = c_{ij} - \pi_i + \pi_j \geq 0$, using the optimum matching dual variables, and avoid the application of label correcting algorithm in ALCLS. Since the complexity of each application of label setting algorithm is $O(n^2)$, the complexity of obtaining the second best matching is $O(n^3)$.

All the approaches presented above for bipartite graphs have a worst case complexity of $O(n^3)$ for finding the second best matching. From the analysis of Section 2, the overall complexity of finding the K best bipartite matchings will be $O(n^3 + (K-1)n^3)$, which is $O(Kn^3)$, the best bound that we could obtain so far. Note that this is the same as the bound obtained using the sensititivity analysis approach for the non-bipartite matchings.

Acknowledgement

We thank Ulrich Derigs for several comments clarifying the sensitivity analysis for matching problems.

References

[1] M.S. Bazaraa and R.W. Langley, A dual shortest path algorithm, J. SIAM 26 (3) (1974) 496–501.
[2] W.H. Cunningham and A.B. Marsh III, A primal method for solving minimal perfect matching problems, Math. Programming Study 8 (1978) 50–72.
[3] J.R. Brown, Shortest alternating path algorithms, Networks 4 (1974) 314–334.
[4] C.R. Chegireddy and H.W. Hamacher, Algorithms and programs for K-th best weighted bipartite matching problem, Technical Report, Dept. of Industrial & Systems Engineering, University of Florida, 1985, to appear.
[5] U. Derigs, A shortest augmenting path method for solving minimal perfect matching problems, Networks 11 (1981) 379–390.
[6] U. Derigs, Shortest augmenting paths and sensititivity analysis for optimal matchings, Report No 82222-OR, Universität Bonn, April 1982.
[7] E.W. Dijkstra, A note on two problems in connection with graphs, Numer. Math. 1 (1959) 269–271.
[8] J. Edmonds, Paths, trees and flowers, Canad. J. Math. 17 (1965) 449–467.
[9] J. Edmonds and R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, J. ACM 19 (2) (1972) 248–264.
[10] R.W. Floyd, Algorithm 97: shortest path, Comm. ACM 5 (6) (1962) 345.
[11] G. Gallo and S. Pallotino, Shortest path methods: A unifying approach, Netflow 83 (1983).
[12] H.W. Hamacher and M. Queyranne, K-best solutions to combinatorial optimization problems, Research Report No 83-5 Industrial and Systems Engineering Department, University of Florida, 1983 Ann. Oper. Res. 4 (1985/6) 123–143.
[13] E.L. Lawler, A procedure for computing the K-th best solutions to discrete optimization problems and its application to the shortest path problem, Management Sci. 18 (7) (1972) 401–405.
[14] E.L. Lawler, Combinatorial Optimization: Networks and Matroids (Holt, Rinehart and Winston, New York, 1976).
[15] K.G. Murty, An algorithm for ranking all the assignments in order of increasing cost, Operations Research 16 (1968) 682–687.
[16] U. Pape, Algorithm 562: shortest path lengths, ACM Trans. Math. Software 5 (1980) 450–455.
[17] G.M. Weber, Sensitivity analysis of optimal matchings, Networks 11 (1981) 41–56.