Incremental Assignment Problem

Ismail H. Toroslu and Göktürk Üçoluk

Department of Computer Engineering
Middle East Technical University, Ankara, Turkey

Abstract

In this paper we introduce the incremental assignment problem. In this problem, a new pair of vertices and their incident edges are added to a weighted bipartite graph whose maximum weighted matching is already known, and the maximum weighted matching of the extended graph is sought. We propose an $\mathcal{O}(|V|^2)$ algorithm for the problem.

Key words: Assignment problem, weighted bipartite graph, Hungarian algorithm

1 Introduction

Matching is a well-studied algorithmic problem. There are several variations of the matching problem, such as matching in general graphs, matching in bipartite graphs, and matching in weighted/unweighted graphs. In unweighted graphs maximum cardinality matching is sought (see [9,17] for unweighted bipartite graphs and [8,22] for unweighted general graphs). On the other hand, in weighted graphs maximum weighted matching is also explored (see [11,15] for weighted bipartite graphs and [7,10,13,21] for weighted general graphs). In this work we are interested in the maximum weighted matching problem in bipartite graphs, also known as the assignment problem. Surveys on matching problems and their algorithms can be found in [14,25]. New forms of the assignment problem have been studied in recent works such as [1,5,16,18,20,26,27]. Also, [4] contains a survey of several other forms of assignment problems.

The most well-known algorithm for the assignment problem is the classical Kuhn-Munkres algorithm [19,23], also known as the Hungarian algorithm. This algorithm has $\mathcal{O}(|V|^3)$ complexity [1] when it is implemented with proper data structures [21]. There are some newer algorithms with different complexities, but they utilize the maximum weights on the edges [2,12,24]. These

[1] $|V|$ is the size of a partition in a bipartite graph.

newer algorithms are not suitable for an incremental version of the problem. The goal of this paper is to present an algorithm for the incremental assignment problem without imposing any restriction on the weights.

To the best of our knowledge there exists no reference work on the incremental assignment problem, which can be described as follows:

> *Given a weighted bipartite graph and its maximum weighted matching, determine the maximum weighted matching of the graph extended with a new pair of vertices, one on each partition, and weighted edges connecting these new vertices to all the vertices on their opposite partitions.*

The incremental assignment problem can be solved with an algorithm of the ordinary assignment problem. However, such an approach does not use already known information about the maximum weighted matching of the subgraph with only one less pair of vertices of the given bipartite graph (i.e., the maximum-weighted-matched part of the bipartite graph). In this paper, we propose a novel algorithm, based on the Kuhn-Munkres algorithm, that utilizes the given maximum weighted matching of the maximum-weighted-matched part of the bipartite graph in order to determine the maximum weighted matching of the whole (extended) bipartite graph with $\mathcal{O}(|V|^2)$ complexity. We will use the descriptions and terminology of [3] throughout the paper.

2 Preliminaries

A *bipartite graph* $G = (\mathcal{V}, \mathcal{E})$ is an undirected graph whose vertices are divided into two disjoint sets (partitions) X and Y (i.e. $\mathcal{V} = X \cup Y$ and $X \cap Y = \emptyset$) and no edge connects vertices in the same partition (i.e. $\mathcal{E} \subseteq X \times Y$). A *matching* M is a subset of the edges $\mathcal{E}$ ($M \subseteq \mathcal{E}$), such that every vertex can be incident to only one edge in the matching. A matching *saturates* a vertex a if some edge of M is incident with a; otherwise vertex a is *unsaturated*. If every vertex of G is saturated, then the matching is *perfect*.

In this paper, we are interested in matching in weighted bipartite graphs with same-sized partitions ($|X| = |Y|$). Unless otherwise stated, we will use the term bipartite graph to represent this kind of weighted bipartite graph. The weight of the matching is the total weight of the edges in the matching. In the assignment problem, maximum-weighted perfect matching is sought in bipartite graphs.

An *augmenting path* A in a graph with matching is a path that starts and ends with unmatched edges and alternates between matched and unmatched edges.

The matching edges form the set $M(A)$. If a path starts with an unmatched edge, but ends with a matched edge, then it is called an *alternating path*. All alternating paths originating from the same unmatched vertex form a tree, which is grown from its matched leaves by a pair of unmatched and matched edges, respectively, when it is executed. Such trees are known as *Hungarian trees*.

A weighted bipartite graph $G = (X \cup Y, X \times Y)$ having partitions with size V can be represented by a weight matrix W of size $V \times V$. In the weight matrix, rows correspond to the X partition and columns correspond to the Y partition of vertices. Each entry W_{ij} represents the weight of the edge between the vertices X_i and Y_j.

Feasible vertex labeling l is defined on the vertices of both partitions of the bipartite graph as follows:

$$l(X_i) + l(Y_j) \geq W_{ij} \quad \forall X_i \in X, \quad \forall Y_j \in Y$$

The subgraph corresponding to the feasible vertex labeling l defined by the edges satisfying the following equality is called the *equality subgraph* G_l:

$$l(X_i) + l(Y_j) = W_{ij}$$

This means that the edges whose weights are equal to the sum of the labels of their end vertices and of their incident vertices form the equality subgraph G_l.

3 The Algorithm

Our algorithm for the incremental assignment problem adds a new pair of vertices to the maximum-weighted-matched bipartite graph whose feasible vertex labeling is also given, together with the maximum weighted matching. Then, it assigns any feasible labeling to the newly added pair of vertices and, by using this labeling, determines the maximum weighted matching of the whole extended bipartite graph. Our algorithm is adopted from the Kuhn-Munkres algorithm, which produces the feasible vertex labeling of the vertices together with the maximum weighted matching. Similarly, our algorithm also generates the new feasible vertex labeling of the extended graph while producing the maximum weighted matching.

When the maximum-weighted-matched graph is extended with a new pair of vertices, this extension is reflected in the form of a new weight matrix by adding a new row and a new column to the previous weight matrix as the

$(V+1)$st row and column corresponding to the edges incident to the new vertices.

A detailed description of our algorithm is as follows:

Input: The extended bipartite graph G, its $(V+1) \times (V+1)$ weight-matrix W, a feasible vertex labeling l of the first V vertices for both partitions such that it corresponds to the equality subgraph that contains the maximum weighted matching among the first V vertices, and this maximum weighted matching M_V^* among the first V vertices of the partitions.

Output: Maximum weighted matching M_{V+1}^* and the updated labels l of the vertices of the extended bipartite graph.

Incremental Assignment Algorithm:

(1) Determine a feasible vertex labeling of the extended weight matrix by using the feasible vertex labelings for the maximum-weighted-matched subgraph, and by choosing labels for the new vertices, such that the feasible vertex labeling constraints are satisfied for the new row and the new column. This is done by using the following equations:

$$l(Y_{V+1}) = \max\left\{ \max_{1 \le i \le V} \{W_{i(V+1)} - l(X_i)\}, W_{(V+1)(V+1)} \right\}$$

$$l(X_{V+1}) = \max_{1 \le i \le (V+1)} \left\{ W_{(V+1)i} - l(Y_i) \right\}$$

The equality subgraph that corresponds to this labeling contains all matching edges of the maximum weighted matching of the first V pair of vertices. Thus, if an edge between the two new vertices is in the equality subgraph, then add this edge to the current matching, and stop. Otherwise, two new vertices, one on each partition, are the only unsaturated vertices of the equality subgraph.

(2) On the equality subgraph G_l, using the matching M_V^* grow a Hungarian tree rooted at the new (unsaturated) vertex, of the partition X (called U). While growing the Hungarian tree, include all the vertices encountered in X (including U) into S and all the vertices encountered in partition Y into T.

(3) If an augmenting path A is found, interchange matched and unmatched edges in the augmenting path. We call this interchanged path A'. Then determine the new matching by increasing the cardinality by one as follows:

$$M_{V+1}^* = (M_V^* - M(A)) + M(A')$$

M_{V+1}^* represents the maximum weighted matching of the extended bipartite graph. The algorithm stops.

(4) If a Hungarian tree is obtained and no augmenting path is found, revise the labeling l in order to preserve all the matching edges in G_l, while adding new edges from S to $Y - T$. As in the Kuhn-Munkres algorithm, labeling is revised by decreasing the total summation of the labels, which is also equivalent to the maximum weight matching when perfect matching exists in the graph. During the revision of the labels, the labels of the nodes in S are decreased by the smallest possible amount which will add at least one edge between S and $Y - T$. At the same time, the labels of the nodes in T are increased by the same amount in order to preserve all the current matchings between S and T. Adding such edges will potentially increase the chance of producing an augmenting path. This is achieved as follows:

(a) Determine the smallest possible change in labels:

$$\lambda_l = \min_{\substack{X_i \in S \\ Y_j \in Y - T}} \{l(X_i) + l(Y_j) - W_{ij}\}$$

Direct implementation of the calculation of λ_l requires $\mathcal{O}(|V|^2)$ time. However, this calculation can be done more efficiently if for every vertex in $Y - T$, an edge with the smallest slack is kept as follows:

$$slack[Y_j] = \min_{X_i \in S} \{l(X_i) + l(Y_j) - W_{ij}\}$$

Then, λ_l can be calculated as:

$$\lambda_l = \min_{Y_j \in Y - T} \{slack[Y_j]\}$$

(b) Revise the feasible labelings using the change:

$$l'(v) = \begin{cases} l(v) - \lambda_l \text{ if } v \in S \\ l(v) + \lambda_l \text{ if } v \in T \\ l(v) \qquad \text{otherwise} \end{cases}$$

(5) Go to step 2 to search for an augmenting path with the new equality subgraph defined by the new labelings.

Theorem 1 (Correctness of the algorithm) *The Incremental Assignment Algorithm determines the maximum weighted matching of the extended bipartite graph with size* $(V + 1) \times (V + 1)$.

Proof: When the maximum-weighted-matched bipartite graph is extended with a new pair of vertices, a feasible vertex labeling for the extended bipartite graph can be determined by using the labelings on the maximum-weighted-matched part of the graph and by choosing labels for the new vertices such

that the feasible labeling constraints are satisfied. With these labelings all the matched edges will be in the equality subgraph. Also, if there is no edge between the new pair of vertices, then these two vertices will be the only unmatched vertices in the equality subgraph. Since there is only a single unmatched vertex pair, one on each partition respectively, it is possible to find only a single augmenting path in the equality subgraph. Therefore, the cardinality of the matching must only be incremented by one by inverting the matchings/unmatchings of the edges on the augmenting path when it is discovered. As a result, finding and inverting the augmenting path yields a perfect matching. If the equality subgraph has a perfect matching, then it is the maximum weighted matching (see the related theorem [2] in [3], which refers to [6,19,23]). If the augmenting path is not discovered with the current feasible vertex labelings, then labels must be modified in order to grow the Hungarian tree (note that there is only a single Hungarian tree rooted at the unmatched vertex of one partition). Since each time the labels are modified as described in the algorithm, alternating paths of the Hungarian tree are extended at least by one edge (see the description of Hungarian method in [3]) and eventually an augmenting path that starts with the only unmatched vertex in one partition and ends with the only unmatched vertex in the other partition will be discovered. That equality subgraph will include the perfect matching and, thus, it is the maximum weighted matching of the bipartite graph. □

Theorem 2 (Complexity of the algorithm) *The Incremental Assignment Algorithm has* $\mathcal{O}(|V|^2)$ *complexity.*

Proof: In the first step of the algorithm the feasible vertex labeling of the new vertices are determined. As is seen in the algorithm, choosing the feasible vertex labeling for the new row and the new column can be done simply by performing a linear scan, so it has $\mathcal{O}(|V|)$ complexity. In the proof of the theorem given above, it was shown that discovering only one augmenting path is sufficient for completing the execution of the algorithm. Therefore, the main iteration is the relabeling of the vertices in order to extend the Hungarian tree to seek an augmenting path. The feasible vertex labeling modifications always grow the Hungarian tree by at least one vertex from each partition. Thus, in the worst case, after V iterations it will be possible to obtain the augmenting path. In each iteration, finding the minimum slack and growing the Hungarian tree with the newly introduced edges on the equality subgraph requires $\mathcal{O}(|V|)$ operations by using the proper data structure described in the algorithm (proposed by [21]). The computation of these slack values for the first time requires $\mathcal{O}(|V|^2)$ complexity. Whenever a vertex is added into S, the slacks for the vertices in $Y - T$ must be recomputed, requiring only linear time. Since there are V vertices that can be added into S, the total

[2] **Theorem:** Let l be a feasible vertex labeling of G. If G_l contains a perfect matching M^*, then M^* is an optimal matching of G.

time required for all the computations of the slack values is $\mathcal{O}(|V|^2)$. Thus, the total complexity of the algorithm is $\mathcal{O}(|V|^2)$. □

4 Example

Consider the 4×4 weighted bipartite graph described by its weight matrix as follows:

	Y_1	Y_2	Y_3	Y_4	$x_i \Downarrow$
X_1	5	1	1	1	0
X_2	4	3	1	3	−1
X_3	5	4	3	4	0
X_4	1	6	2	5	2
$y_i \Rightarrow$	5	4	3	5	

Assume that we are given the assignment among the first **3** vertices, which corresponds to the diagonal elements of the weight matrix. The last row and column correspond to the newly added vertex pairs. First, feasible labels are assigned to X_4 and Y_4 as shown above. Then, an augmenting path will be determined that yields an assignment on the weighted bipartite graph that was extended by the vertex pair.

In Figure 1, the weight matrix, the feasible vertex labelings of its vertices, the S and T sets, corresponding to the Hungarian tree rooted at vertex X_4, are shown. The equality subgraph obtained from the weight matrix and the feasible vertex labeling is shown in Figure 2.

As seen in Figures 1 and 2, X_4 and Y_4 are the only unsaturated vertices and there is no augmenting path in the equality subgraph. Figure 3 depicts the new labelings on the weight matrix after labels are revised using λ_l which is obtained from the previous labeling. Also, Figure 4 shows the equality subgraph corresponding to the new labeling, which includes an augmenting path. The augmenting path is shown in bold. It is inverted to increase the cardinality of the matching by one and this produces a perfect matching, which also corresponds to the maximum-weighted matching (i.e., the assignment) of the extended bipartite graph.

	Y_1	Y_2	Y_3	Y_4	$x_i \Downarrow$	
X_1	**(5)**	1	1	1	0	S (−1)
X_2	(4)	**(3)**	1	3	−1	S (−1)
X_3	(5)	(4)	**(3)**	4	0	
X_4	1	**(6)**	2	5	2	S (−1)
$y_i \Rightarrow$	5	4	3	5		
	T (+1)	T (+1)				

- Circled edges represent the equality subgraph.
- Matching edges are marked in bold.
- S and T mark the corresponding set of vertices.
- λ_l increment/decrement values are indicated by (+1)/(−1)

Fig. 1. Situation before the first iteration of the algorithm: Weight Matrix

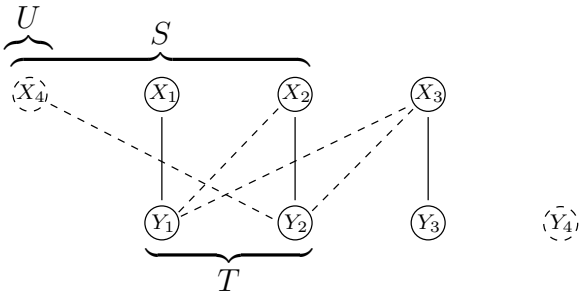

Fig. 2. Situation before the first iteration of the algorithm: Equality Subgraph

	Y_1	Y_2	Y_3	Y_4	$x_i \Downarrow$
X_1	**(5)**	1	1	1	−1
X_2	(4)	(3)	(1)	**(3)**	−2
X_3	5	4	**(3)**	4	0
X_4	1	**(6)**	2	5	1
$y_i \Rightarrow$	6	5	3	5	

Fig. 3. Situation after the first iteration of the algorithm: Weight Matrix

5 Conclusion

In this paper, we proposed an $\mathcal{O}(|V|^2)$ algorithm for the incremental assignment problem. By using the same technique, the decremental assignment problem can also be solved. In the decremental assignment problem a pair of vertices, not necessarily connected with a matching edge, and their incident edges are removed from a weighted bipartite graph with a given assignment. Also, when an already maximum-weighted-matched bipartite graph with size V is extended with k new pairs of vertices, where k is much smaller than V

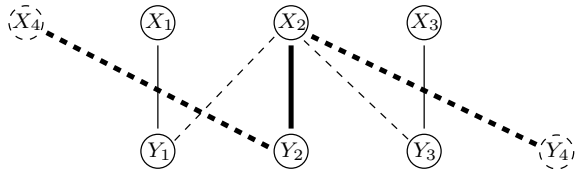

Fig. 4. Situation after the first iteration of the algorithm: Equality Subgraph

($k << V$), the use of this incremental algorithm will be more efficient (which will take $\mathcal{O}(k|V|^2)$ time) compared with determining the full assignment from scratch by using a standard assignment algorithm (which will take $\mathcal{O}(|V|^3)$ time).

References

[1] J. Aguilar and E. Gelenbe, Task assignment and transaction clustering heuristics for distributed systems *Information Sciences* **97** 1-2 (1997) 199–219

[2] R. K. Ahuja, J. B. Orlin and T. L. Magnanti, Survey of assignment algorithms and network flows, in: G. L. Nemhauser, A. H. G. Rinnooy Kan and M. J. Todd eds. *Handbooks in Operations Research and Management Science. Vol. 1: Optimization* (North Holland, Amsterdam, 1989) 211–369.

[3] J.A. Bondy and U. S. R. Murty, *Graph Theory with Applications* (6th ed., North-Holland, Amsterdam, 1984).

[4] R. E. Burkard, Selected topics on assignment problems, *Discrete Applied Mathematics*, **123** (2002) 257–302.

[5] W. Cook and A. Rohe, Computing minimum-weight perfect matchings, *Informs J. on Computing* **11** (1999) 138–148.

[6] J. Edmonds, Paths, trees and flowers, *Canadian J. Math.* **17** (1965) 449–467.

[7] J. Edmonds, Maximum matching and polyhedron with 0,1-vertices, *J. Res. Nat. Bur. Standards* **29B** (1965) 125–130.

[8] S. Even and O. Kariv, An $O(n^{2.5})$ algorithm for maximum matching in general graphs, *Proceedings of IEEE FOCS* (1975) 100–112.

[9] S. Even and R.E. Tarjan, Network flow and testing graph connectivity, *SIAM J. Comput.* **4** (1975) 507–518.

[10] H. N. Gabow, An efficient implementation of Edmond's algorithm for maximum matching on graphs, *J. ACM* **23** (1975) 221–234.

[11] H. N. Gabow, Z. Galil and T. H. Spencer, Efficient implementation of graph algorithms using contraction, *J. ACM* **36** 3 (1989) 540–572.

[12] H. N. Gabow and R.E. Tarjan, Faster scaling algorithms for general graph matching problems, *Tech. Report. CU-CS-432-89* (Dept. Comp. Sci., U. Colorado- Boulder, 1989).

[13] H. N. Gabow, Data structures for weighted matching and nearest common ancestors with linking, *Proc. 1st ACM-SIAM Symp. Disc. Algs., SIAM*, (San Francisco 1990) 434–443.

[14] Z. Galil, Efficient algorithms for finding maximum matching in graphs, *ACM Computing Surveys* **18** (1986) 23–38.

[15] D. Goldfarb, Efficient dual simplex algorithms for the assignment problem, *Math. Program.* **33** (1985) 187–203

[16] A. Holland and B. O'Sullivan, Fast Vickrey-Pricing for the assignment problem, *Proc. of ERCIM/CologNet International Workshop on Constraint Solving and Constraint Logic Programming* (Budapest 2003)

[17] J. E. Hopcroft and R. M. Karp: An $N^{5/2}$ algorithm for maximum matchings in bipartite graphs, *SIAM J. Comput.* **2** (1973) 225–231.

[18] M. Y. Kao, T. W. Lam, W. K. Sung and H.F. Ting, A decomposition theorem for maximum weight bipartite matchings, *SIAM J. Comput.* **31** (2001) 18–26.

[19] H. W. Kuhn, The Hungarian method for the assignment problem, *Naval Res. Logist. Quart.* **2** (1955) 83–97.

[20] J. D. Lamb, A note on the weighted matching with penalty problem, *Pat. Recog. Let.* **19** (1998) 261–263

[21] E. L. Lawler, Combinatorial Optimization: Networks and Matroids, (Holt, Rinehart, Winston, Newyork, 1976).

[22] S. Micali and V. V. Vazirani, An $O(\sqrt{|V|} \cdot |E|)$ Algorithm for finding maximum matching in general graphs, *Proceedings of IEEE FOCS* (1980) 17–27.

[23] J. Munkres, Algorithms for the assignment and transportation problems, *J. Soc. Indust. Appl. Math.* **5** (1957) 32–38.

[24] J. B. Orlin and R. K. Ahuja, New scaling algorithm for the assignment and minimum cycle mean problems, *Mathematical Programming* **54** (1992) 41–56.

[25] W. R. Pulleyblank, Matching and Extensions in Handbook of Combinatorics, ed. R. Graham, M. Grötschel, L. Lovasz, (Elsevier, 1995) 179–232.

[26] I. H. Toroslu, Personnel assignment problem with hierarchical ordering constraints, *Computers and Industrial Engineering*, **45**, (2003) 493-510.

[27] W. C. K. Yen and C. Y. Tang, An optimal algorithm for solving the searchlight guarding problem on weighted interval graphs, *Information Sciences* **100** 1-4 (1997) 1–25