EPiC Series in Computing

Volume 75, 2021, Pages 98–107

CAINE 2020. The 33rd International Conference on Computer Applications in Industry and Engineering

Improved Algorithm for the Incremental Assignment Problem

Pinzhi Wang[1*], Youssou Gningue[1*] and Haibin Zhu[2]

[1]Laurentian University, Ontario, Canada

[2]Nipissing University, Ontario, Canada

* pwang@laurentian.ca, ygningue@cs.laurentian.ca

Abstract

The Assignment Problem is a basic combinatorial optimization problem. In a weighted bipartite graph, the Assignment Problem is to find the largest sum of weights matching. The Hungarian method is a well-known algorithm, which is combinatorial optimization. Adding a new row and a new column to a weighted bipartite graph is called the Incremental Assignment Problem (IAP). The algorithm of the Incremental Assignment Problem utilizes the given optimal solution (the maximum weighted matching) and the dual variables to solve the matrix after extending the bipartite graph. This paper proposes an improvement of the Incremental Assignment Algorithm (IAA), named the Improved Incremental Assignment Algorithm (IIAA). The improved algorithm will save the operation time and operation space to find the optimal solution (the maximum weighted matching) of the bipartite graph.

Key words: Assignment problem, weighted bipartite graph, Hungarian algorithm, incremental assignment problem.

1 Introduction

A matching or independent edge set in a graph is a set of edges without common vertices. There

are several algorithms of the matching problem, such as matching in weighted bipartite graphs, matching in unweighted graphs, matching in general graphs. In unweighted graphs maximum cardinality matching is sought [1]. In matching in weighted bipartite graphs, each edge has an associated value. A maximum weighted bipartite matching is defined as a matching where the sum of the values of the edges in the matching has a maximal value. If the graph is not complete bipartite, missing edges are inserted with value zero. Finding such a matching is known as the assignment problem. A common variant consists of finding a minimum-weighted perfect matching [2]. In this research, we are interested in the maximum weighted matching problem in bipartite graphs. The well-known algorithm for the assignment problem is Kuhn-Munkres algorithm or the Hungarian algorithm, originally proposed by H.W.Kuhn in 1955 [3] and refined by J. Munkres in 1957 [4]. This algorithm has $O(n^3)$ complexity when it is implemented with proper data structures.

The incremental assignment problem is described as: 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 [5]. It can be solved with the Hungarian Algorithm as the ordinary assignment problem. But in Toroslu's work, they propose an algorithm which 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. The complexity of the algorithm is $O(n^2)$.

Considering there will be thousands or million weights. It is costly to calculate the extended feasible labels by using incremental assignment problem algorithm. The goal of this paper is to present an algorithm to improve the incremental assignment problem algorithm, which reduces the complexity to $O(n)$ in four situations.

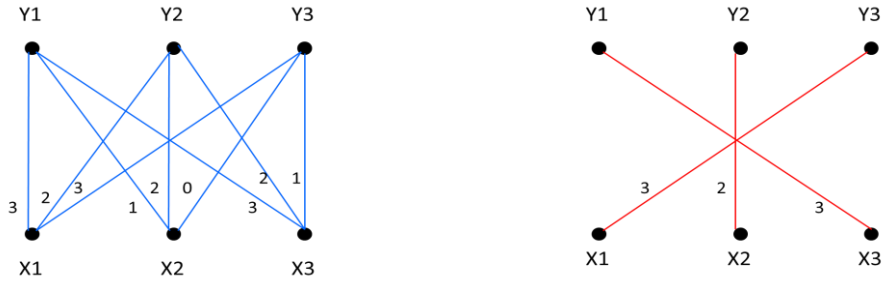
2 Background

2.1 Terminology and Notation

A graph $G = (V, E)$ is bipartite if there exist two disjoint partitions X and Y ($V = X \cup Y$, $X \cap Y = \emptyset$) and no edge connects vertices in the same partition ($E \subseteq X \times Y$). A matching M is a subset of the edges E ($M \subseteq E$), such that $\forall v \in V$ at most one edge in M is incident upon v. The size of a matching is $|M|$, the number of edges in M. A maximum matching is a matching of maximum size (maximum number of edges). In a maximum matching, if any edge is added to it, it is no longer a matching.

A weighted bipartite graph $G = (X \cup Y, X \times Y)$ is the graphs in which rows correspond to the X

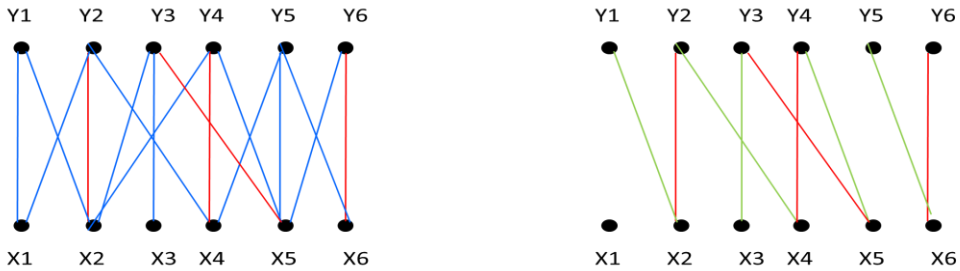
partition and columns correspond to the Y partition of vertices. Each entry W_{ij} represents the weight of the edge between the vertices X_i and Y_j. The weight of matching M is the sum of the weights of edges in M.



bipartite graph bipartite weighted graph

Figure 1: Bipartite graph & bipartite weighted graph

Given a matching M, an alternating path is a path that begins with an unmatched vertex and whose edges belong alternately to the matching and not to the matching. An augmenting path is an alternating path that starts from and ends on free (unmatched) vertices. All alternating paths originating from a given unmatched node form a Hungarian tree.



bipartite graph alternating/ augmenting path

Figure 2: Bipartite graph & alternating/ augmenting graph

In Figure 2, let M be a matching of G. Vertex v is matched if it is endpoint of edge in M; otherwise v is free. $Y_2, Y_3, Y_4, Y_6, X_2, X_4, X_5, X_6$ are matched, other vertices are free. Y_5, X_6, Y_6 is an alternating path. $Y_1, X_2, Y_2, X_4, Y_4, X_5, Y_3, X_3$ is an augmenting path.

2.2 Incremental Assignment Algorithm

The algorithm assigns dual variables α_i to each node U and dual variables β_j to each node V. The assignment problem is feasible when $\alpha_i + \beta_j \geq W_{ij}$. The Hungarian algorithm maintains feasible values for all the α_i and β_j from initialization through termination. An edge in the bipartite graph is called admissible when $\alpha_i + \beta_j = W_{ij}$.

The algorithm first determines the feasible values for the two new dual variables α_{n+1} and β_{n+1}

(the other dual variables are still feasible from the solution of the $n \times n$ problem). It then essentially performs a single stage of the Hungarian algorithm to find an augmenting path between the two new vertices U_{n+1} and V_{n+1}, adjusting dual variables to add new admissible edges to the equality subgraph as needed. The matched and unmatched edges are flipped along the discovered augmenting path increases the cardinality of the matching by one, thus resulting in a complete matching. Note that the two new nodes u_{n+1} and v_{n+1} may not need to be matched to each other in the extended matching. Because the algorithm involves executing only one stage of the Hungarian algorithm after performing an $O(n)$ initialization step, the computational complexity of the incremental assignment algorithm is $O(n^2)$. The incremental assignment algorithm is illustrated as follows.

Incremental Assignment Algorithm:

Input:

- An assignment problem comprising a bipartite graph, $\{U, V; E\}$ (where $|U| = |V| = n + 1$) and an $(n + 1) \times (n + 1)$ matrix of edge weights W_{ij}
- An optimal solution to the $n \times n$ sub-problem of the above assignment problem, comprising a matching M^* of the first n nodes of U to the first n nodes of V, and the final values of the dual variables α_i and β_j for $i \in 1 \ldots n$ and $j \in i \ldots n$

Output: An optimal matching M, for the $(n + 1) \times (n + 1)$ problem.

1. Perform initialization:
   (a) Begin with the given matching, $M_0 = M^*$.
   (b) Assign feasible values to the dual variables α_{n+1} and β_{n+1} as follows:

$$\beta_{n+1} = \max\left(\max_{1 \le i \le n}\left(W_{i(n+1)} - \alpha_i\right), W_{(n+1)(n+1)}\right)$$

$$\alpha_{n+1} = \max_{1 \le j \le n+1}\left(W_{(n+1)j} - \beta_j\right)$$

2. Perform the routine **Stage** below.
3. Output the resulting matching M.

Stage:

1. Designate each exposed (unmatched) node in U as the root of a Hungarian tree.
2. Grow the Hungarian tree rooted at the exposed nodes in the equality subgraph.

Designate the indices i of nodes u_i encountered in the Hungarian tree by the set I^*, and the indices j of nodes v_j encountered in the Hungarian tree by the set J^*. If an augmenting path is found, go to Step 4. If not, and the Hungarian trees cannot be grown further, proceed to Step 3.

3. Modify the dual variables α_i and β_j as follows to add new edges to the equality subgraph. Then go to Step 2 to continue the search for the augmenting path.

$$\theta = \min_{i \in I^*, j \notin J^*}(\alpha_i + \beta_j - W_{ij})$$

$$\alpha_i \leftarrow \begin{cases} \alpha_i - \theta & i \in I^* \\ \alpha_i & i \notin I^* \end{cases}$$

$$\beta_j \leftarrow \begin{cases} \beta_j + \theta & j \in J^* \\ \beta_j & j \notin J^* \end{cases}$$

4. Augment the current matching by flipping matched and unmatched edges along the selected augmenting path. That is, M_k (the new matching at stage k) is given by $(M_{k-1} - P) \cup (P - M_{k-1})$, where M_{k-1} is the matching from the previous stage and P is the set of edges on the selected augmenting path.

3 Improved Incremental Assignment Algorithm

The original matrix adds a new pair of vertices to the maximum-weighted-matched bipartite graph, in which feasible vertices and the maximum weighted matching are given. Any feasible label to the newly added pair of vertices is assigned. By using these labels, we can determine the maximum weighted matching of the whole extended bipartite graph. This algorithm is adopted form the Hungarian algorithm, which uses the feasible labels of the vertices together with the maximum weighted matching.

Improved Incremental Assignment Algorithm:

Input:

- An assignment problem comprising a bipartite graph, $\{V, E\}$ (where $V = X \cup Y, X \cap Y = \emptyset$, $|X| = |Y| = n + 1$) and an $(n + 1) \times (n + 1)$ matrix of edge weights W_{ij}
- An optimal solution to the $n \times n$ sub-problem of the above assignment problem, comprising a matching M^* of the first n nodes of X to the first n nodes of Y, and the final values of the dual variables α_i and β_j for $i \in 1 \ldots n$ and $j \in i \ldots n$

Output: An optimal matching M, for the $(n + 1) \times (n + 1)$ problem.

1. Perform initialization:

   (a) Find the difference of each role and each column as follows:

$$DC_k = \max_{1 \le i \le n}(W_{i(n+1)} - \alpha_i) = W_{k,n+1} - \alpha_k$$

$$LC = \mathrm{Card}(\mathrm{Argmax}_{1 \le i \le n}(W_{i(n+1)} - \alpha_i))$$

$$DR_t = \max_{1 \le j \le n}(W_{(n+1)j} - \beta_j) = W_{n+1,t} - \beta_t$$

$$LR = Card(Argmax_{1\le j\le n}(W_{(n+1),j} - \beta_j))$$

$$I = DC_k + DR_t$$

(b) If $I \le W_{(n+1)(n+1)}$, then

$X_{(n+1)(n+1)} = 1$, then go to step 3

Else

If $LC = LR = 1$ then

If $X_{kt} = 1$,

Set $X_{k(n+1)} = 1$ and $X_{(n+1)t} = 1$,

Then go to step 3.

Else ($X_{kt} \neq 1$)

Find the complementary column s of the basic variable on row k

Find the complementary row r of the basic variable on column t

If X_{rs} satisfies $C_{rs} = \alpha_r + \beta_s$ (Equality Graph) then

Set $X_{rs} = 1$ then $X_{k(n+1)} = 1$ and $X_{(n+1)t} = 1$

Then go to step 3

Else go to step c.

Else if $\max(LC, LR) > 1$

If $\max(LC, LR) = LC$ then

For $j = 1, \dots, LR$ do

Find the basis variable $X_{rj} = 1$

If $DC_k = (W_{r(n+1)} - \alpha_r)$ then set

$X_{rj} = 0$, $X_{j(n+1)} = 1$ and $X_{(n+1)r} = 1$

Go to step 3

End For

Else if $\max(LC, LR) > 1$

If $\max(LC, LR) = LR$ then

For $i = 1, \dots, LC$ do

Find the basis variable $X_{is} = 1$

If $DR_t = (W_{(n+1)s} - \beta_s)$ then set

$X_{is} = 0$, $X_{(n+1)i} = 1$ and $X_{s(n+1)} = 1$

Go to step 3

End For

End If

(c) Incremental Assignment Algorithm: Assign feasible values to the dual variables α_{n+1} and β_{n+1} as follows:

$$\beta_{n+1} = \max\left(DC_k, W_{(n+1)(n+1)}\right)$$

$$\alpha_{n+1} = DR_t$$

2. Perform the **Stage** from the basic Hungarian algorithm detailed in the Hungarian Algorithm.
3. Output the resulting matching M.

4 Implementation and Performance Experiments

When properly implemented, the Incremental Assignment Algorithm (IAA) can operate with the computational complexity of $O(n^2)$ [5]. To verify the performance of the Improved Incremental Assignment Algorithm (IIAA), a program is implemented based on that of the IAA.

Five cases are designed for the Improved Incremental Assignment Algorithm (IIAA) in 100 dimensions (a matrix with 100×100 elements). Case I shows the operation time of the circumstances that $X_{kt} = 1$ and $I \leq W_{(n+1)(n+1)}$. Case II shows the running time of the matrix that $X_{kt} = 1$ and $I > W_{(n+1)(n+1)}$. Case III displays the operation time when $X_{kt} \neq 1$, find the complementary column s of the basic variable on row k and the complementary row r of the basic variable on column t, if X_{rs} satisfies $C_{rs} = \alpha_r + \beta_s$. Case IV reveals the running time when we have more than one largest difference of the row and more than one largest difference of the column. Case V is a general case, which expresses that the random matrix can be solved by the IIAA.

All the experiments are workable in both algorithms. Matrix is formed by randomly creating weights. The test takes 100 random new pairs of vertices in the 100 dimension matrix.

$$\text{Chances} = \frac{\text{Numbers of each case}}{\text{Total number of the cases}} \times 100\%$$

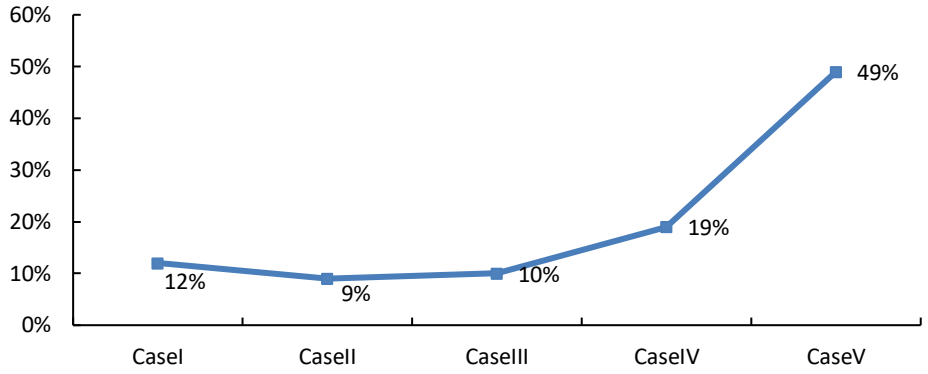


Figure 1: Trend lines for the chances of five cases happen.

Figure 3 provides the trend line of chances that a random matrix solved by Improved Incremental Assignment Algorithm (IIAA).

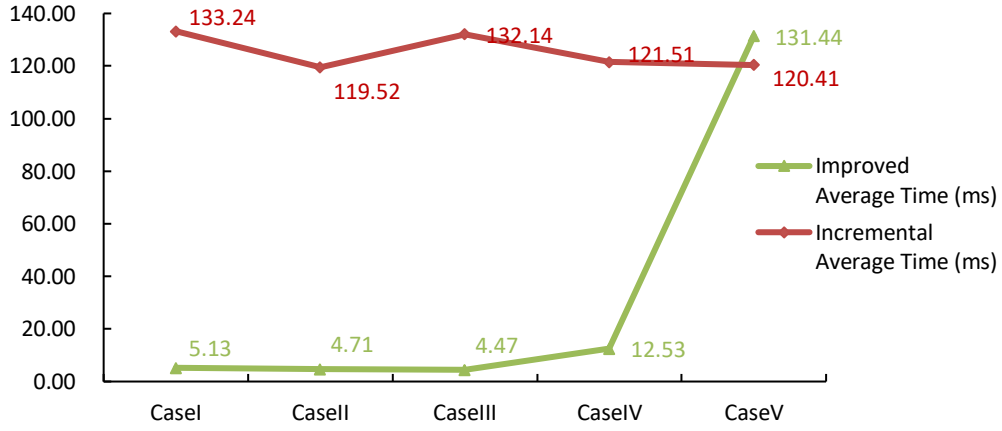


Figure 4. Trend lines for average process time for different cases

Figure 4 presents the average time of different cases, which have been solved by the Improved Incremental Assignment Algorithm and Incremental Assignment Algorithm. The time unit is millisecond.

Time / Dimension	Average IIAA (ms)	Average IAA (ms)	Percentage
Case I	5.13	133.24	-96.15%
Case II	4.71	119.52	-96.06%
Case III	4.47	132.14	-96.62%
Case IV	12.53	121.51	-89.69%
Case V	131.44	120.41	9.16%

Table 1: The Process Time for Each Case

Table 1 presents the average operation time of different pairs of vertices which has been solved by Improved Incremental Assignment Algorithm and Incremental Assignment Algorithm for each case. The time unit is millisecond.

$$\text{Percentage} = \frac{\text{Improved IAA average time} - \text{IAA average time}}{\text{IAA average time}}$$

5 Performance Analysis

Table 1 shows the typical data collected from the experiments stated previously. The average operation time of Incremental Assignment Algorithm (IAA) required by random matrix is linearly increased. It is possible for some particular smaller dimension matrix to take more time than a larger dimension matrix because the value distributions significantly affect the time needed in the relevant algorithm. The average running time of IIAA is stable regardless of the increase of the dimension. The difference average running time between IIAA and IAA becomes higher because of the different numbers of iterations. With a higher dimension, the number of iterations of IAA will go higher. But IIAA still has only one iteration no matter how high the dimension is.

In Case I, Case II and Case III, IIAA saves up to 96% of operation time in 100 dimensions matrix. In Case IV, IIAA saves about 90% of operation time. The data express that Improved Incremental Assignment Algorithm will save much time than the Incremental Assignment Algorithm.

For all the 100 random matrices, 51% matrices meet the conditions of IIAA, the rest of the matrices will go to IAA that is the Case V (general case).

If the matrix is produced randomly, the general expectation of improvement is 47% compared with the operation time using the Incremental Assignment Algorithm.

Overall, when the matrix is solved by IIAA, the process time will be much faster than IAA.

6 Conclusion

In this paper, an improved algorithm has been proposed to solve Incremental Assignment Problem. From the view of the overall program, the algorithm improves a lot with the running time of operation. From finding the largest difference of the row and the column, with the use of exchange with its maximum weighted matching, the problem has been reduced. Consequently, the step of iteration can be reduced largely.

From the perspective of functions, our solution directly provides an improved way to solving the Incremental Assignment Problem. Not only the operation time will be saved, but also the occupied space will be reduced.

The computational complexity of our algorithm is $O(n^2)$, because the most complicated case for our algorithm will go to Incremental Assignment Algorithm. As the complexity of Incremental Assignment Algorithm is $O(n^2)$, our algorithms are also at the same level.

Our solution has many advantages, but it still necessary to point out the disadvantages. About 51% of the 100 random matrices meet the conditions of Improved Incremental Assignment Algorithm, the rest of the matrix will go to Incremental Assignment Algorithm. The chance is limited and needs to be improved in the future.

References

[1] Y. Xie. An o(n2.5) Algorithm: For Maximum Matchings in General Graphs. Journal of Applied Mathematics and Physics 06, no. 09: 1773–82, 2018.
[2] https://en.wikipedia.org/wiki/Assignment_problem
[3] H. W. Kuhn. The Hungarian Method for the Assignment Problem. Naval Research Logistics Quarterly 2, no. 1–2: 83–97, March 1955.
[4] J. Munkres. Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics 5, 1, 32-38, March 1957
[5] I. H. Toroslu, and G. Üçoluk. Incremental Assignment Problem. Information Sciences 177, no. 6 (March 15, 2007): 1523–29.