Advanced Studies in Biology, Vol. 4, 2012, no. 1, 27 – 35

The Hungarian Algorithm with a Single Input Set

Fatemeh Rajabi Alni

Department of Computer Engineering, Islamic Azad University
North Tehran Branch, Tehran, Iran
fatemehrajabialni@yahoo.com

Alireza Bagheri

Department of Computer Engineering & IT
Amirkabir University of Technology, Tehran, Iran
ar_bagheri@aut.ac.ir

Abstract

Let A and B be two finite sets of points with total cardinality n. The maximum-cost many-to-many matching is a matching that matches each point in A to at least one point in B and each point in B to at least one point in A, such that sum of the matching costs is maximized.
The maximum-cost many-to-many matching can be reduced to the maximum-weight bipartite edge cover problem, and be solved in $O(n^3)$ time using the Hungarian method. In the Hungarian algorithm it is assumed that the two sets A and B are disjoint. In this paper, we change the basic Hungarian algorithm, such that it can find the maximum-cost many-to-many matching between the points of a single set. We also present an algorithm for computing the degree satisfier maximum-cost many-to-many matching between A and B sets, where the degree demands of A and B are satisfied. That is the points of A and B must be matched with at least D_A and D_B points of the other set, respectively.

Keywords: many-to-many point matching, Hungarian method, maximum-cost matching, single set, degree demand.

1. Introduction

Let A and B be two finite sets of points with total cardinality n. We can define a relationship between the points in A and the points in B using the concept of matching. This concept has various applications in computational biology [1], operations research [2], pattern recognition [3], computer vision [6], music information retrieval [8] and computational music theory [9].

A matching between two sets is a function that pairs individual points in one set with individual points in the other set. There are different types of matching. A one-to-one matching between A and B is a perfect matching between the two sets [7]. A many-to-one matching maps each point of A to exactly one point of B and each point of B to at least one point of A [5]. A one-to-many matching maps each point of A to at least one point of B and each point of B to exactly one point of A [5]. A many-to-many matching between the two sets maps each point of A to at least one point of B and vice-versa [2].

Eiter and Mannila [10] have been originally studied the many-to-many matching problem as link distance. The link distance between two sets, is the minimum-cost many-to-many matching between the two sets. They reduced the problem of the computation of the link distance to the minimum-weight perfect matching problem in a bipartite graph, and solved it in $O(n^3)$ time.

In this paper, we change the basic Hungarian algorithm, such that it can find the maximum-cost many-to-many matching between the points of a single set. We also present an algorithm for computing the degree satisfier maximum-cost many-to-many matching between A and B sets, where the degree demands of A and B are satisfied that is the points of A and B are matched with at least D_A and D_B points of the other set, respectively.

2. Preliminaries

A graph $G = (V, E)$ is bipartite if there exists partition $V = X \cup Y$ with $X \cap Y = \emptyset$ and $E \subseteq X \times Y$. A matching M is a subset $M \subseteq E$ such that $\forall v \in V$ at most one edge in M is incident upon v. Size of a matching, $|M|$, is the number of the edges in M. The quality of a matching is measured by a cost function δ that assigns a cost $\delta(a_i, b_j)$ to each pairing (a_i, b_j). The cost of a matching M is sum of the weights of the all edges $(a_i, b_j) \in M$ for $1 \leq i \leq s$ and $1 \leq j \leq t$. A Maximum matching is matching M such that the weight of every other matching $w(M')$ satisfies $w(M') \leq w(M)$. A vertex v is matched if it is endpoint of an edge in M, otherwise v is free. A path is alternating if its edges alternate between M and $E - M$. An alternating path is augmenting if both endpoints are free. An augmenting path has one less edge in M than in $E - M$. Replacing the M edges by the $E - M$ ones increments size of the matching. An alternating tree is a tree rooted at some free vertex v in which every path is an alternating path. A vetex labeling is a function $\ell : V \to R$. A feasible labeling is one such that $\ell(x) + \ell(y) \geq w(x, y), \forall x \in X, y \in Y$. The equality graph (with respect to ℓ) is $G = (V, E_\ell)$ where $E_\ell = \{(x,y) : \ell(x) + \ell(y) = w(x,y)\}$.

Neighbor of $u \in V$ and set $S \subseteq V$ is defined as $N_\ell(u) = \{v : (u,v) \in E_\ell\}$ and

$N_\ell(s) = \cup_{u \in S} N_\ell(u)$, respectively.

Kuhn-Munkres lemma. Let $S \subseteq A$ and $T = N_\ell(S) \neq B$. Set $\alpha_\ell = min_{a_i \in S, b_j \notin T} \{\ell(a_i) + \ell(b_j) - w(a_i, b_j)\}$

$$\ell'(v) = \begin{cases} \ell(v) - \alpha_\ell & if\ v \in S \\ \ell(v) + \alpha_\ell & if\ v \in T \\ \alpha_\ell & otherwise \end{cases}$$

Then ℓ' is a feasible labeling. [11]

Proof.

- If $(a_i, b_j) \in E_\ell$ for $a_i \in S, b_j \in T$ then $(a_i, b_j) \in E_{\ell'}$.
- If $(a_i, b_j) \in E_\ell$ for $a_i \notin S, b_j \notin T$ then $(a_i, b_j) \in E_{\ell'}$.
- There is some edge $(a_i, b_j) \in E_{\ell'}$ for $a_i \in S, b_j \notin T$

Kuhn-Munkres theorem. If ℓ is a feasible labeling and M is a perfect matching in E_ℓ, then M is a max-weight matching[11].

3. The basic Hungarian algorithm

In this section, we briefly explain the basic Hungarian algorithm. Let $A = \{a_1, a_2, .., a_s\}$, and $B = \{b_1, b_2, .., b_t\}$ be two sets of points with total cardinality n, a maximum-cost many-to-many matching, is the matching that matches each point $a_i \in A$ to at least 1 and at most t points of B, and each point $b_j \in B$ to at least 1 and at most s points of A, such that sum of the matching costs is

maximized. The quality of a matching is measured by a cost function δ that assigns a cost $\delta(a_i, b_j)$ to each pairing (a_i, b_j). The cost of a matching is sum of the costs of the all pairings (a_i, b_j) for $1 \leq i \leq s$ and $1 \leq j \leq t$.
The basic Hungarian algorithm finds the maximum weight perfect matching in a complete bipartite graph G, which has two parts, A and B. There are s nodes in A part and t nodes in B part. The nodes of the part A are indexed by a_1 to a_s, and the nodes of the part B are indexed by b_1 to b_t. The weight of each edge (a_i, b_j) of the complete bipartite graph G is equal to $\delta(a_i, b_j)$. Without loss of generality, by adding the edges of weight 0, we may assume that G is a complete weighted graph.The basic Hungarian algorithm has 4 steps that are explained in the following.

1. **Finding an initial feasible labeling.**

   Generate initial labeling ℓ and matching M in E_ℓ. The initial matching can be an empty matching $M = \emptyset$. Finding an initial feasible labeling is simple. Just use:

   $\forall b_j \in B, \ell(b_j) = 0, \forall a_i \in A, \ell(a_i) = max_{b_j \in B} \{w(a_i, b_j)\}$

   With this labeling it is obvious that $\forall a_i \in A, b_j \in B, w(a_i, b_j) \leq \ell(a_i) + \ell(b_j)$.
2. If M is perfect, stop otherwise pick free vertex $a_i \in A$ and set $S = \{a_i\}, T = \emptyset$.
3. **Improving labels.**

   If $N_\ell(S) = T$, update labels (forcing $N_\ell(S) \neq T$)

$$\alpha_\ell = min_{a_i \in S, b_j \notin T} \left\{\ell(a_i) + \ell(b_j) - w(a_i, b_j)\right\}$$

$$\ell'(v) = \begin{cases} \ell(v) - \alpha_\ell & if\ v \in S \\ \ell(v) + \alpha_\ell & if\ v \in T \\ \alpha_\ell & otherwise \end{cases}$$

4. If $N_\ell(S) \neq T$, pick $b_j \in N_\ell(S) - T$.

- If b_j is free, $a_i - b_j$ is an augmenting path.
  Augment M and go to 2.
- If b_j is matched, say to z, extend alternating tree:
  $S = S \cup \{z\}, T = T \cup \{b_j\}$. Go to 3.

Lemma1. In an augmenting path two end vertices of the path is free and the edges of it is alternating between M and $E-M$. If we show the free node by f and the matched node by nf, then the augmenting path is a 4-vertex path (f, nf, nf, f).

Proof. Suppose that the lemma is false and therefore the length of the augmenting path is greater than 3, then since the third edge is in $E-M$ the fourth edge must be in M, and it is not possible because the fourth node is free. ☐

In each phase of the Hungarian algorithm, $|M|$ increases by 1, so there are at most V phases. In implementation, $\forall \mathrm{b_j} \notin T$ keeps track of $slack_{\mathrm{b_j}} = min_{\mathrm{a_i} \in S}\{\ell(\mathrm{a_i}) + \ell(\mathrm{b_j}) - w(\mathrm{a_i}, \mathrm{b_j})\}$. Initializing all slacks at beginning of phase takes $\mathrm{O}(|\mathrm{V}|)$ time. In step 4, we must update all slacks when vertex moves from $\overline{\mathrm{S}}$ to S. This takes $\mathrm{O}(|\mathrm{V}|)$ time and only $|\mathrm{V}|$ vertices can be moved from $\overline{\mathrm{S}}$ to S, giving $\mathrm{O}(|\mathrm{V}|^2)$ time per phase. There are $|\mathrm{V}|$ phases and $\mathrm{O}(|\mathrm{V}|^2)$ work per phase so the total running time is $\mathrm{O}(|\mathrm{V}|^3)$.

Lemma2. In a maximum-cost many-to-many matching M, sum of the degrees of the points of two matched sets A and B are the same, that is $\mathrm{D_A} = \mathrm{D_B}$.

Proof. Each maximum-cost many-to-many matching M contains $|M|$ numbers of the edges that are between a point of A and a point of B, so sum of the degrees of the points of both A and B sets are the same and equal to $|M|$. ☐

4. The Hungarian algorithm with a single input point set

In this section, we explain our changed Hungarian algorithm. Let $\mathrm{A} = \{\mathrm{a_1}, \mathrm{a_2}, .., \mathrm{a_n}\}$ be a point set , a maximum-cost many-to-many matching with a single input set A is a matching that matches each point $\mathrm{a_i} \in \mathrm{A}$ to at least 1 and at most n points of A, such that sum of the matching costs is maximized. In this version of the matching problem, we apply the changed Hungarian algorithm on a complete bipartite graph G, which has two parts A and A'. There are n nodes in A part and n nodes in A' part. The nodes of the part A are indexed by $\mathrm{a_1}$ to $\mathrm{a_n}$, and the nodes of the part A' are indexed by $\mathrm{a'_1}$ to $\mathrm{a'_n}$. The weight of each edge $(\mathrm{a_i}, \mathrm{a'_j})$ of the complete bipartite graph G is equal to $\delta(\mathrm{a_i}, \mathrm{a_j})$ for $i \neq j$ and 0 for $i = j$.

By lemma 1, when we augment a path we remove one edge and insert two edges to the matching M. If we consider the augmenting path as $(\mathrm{a_i}, \mathrm{a'_j}, \mathrm{a_k}, \mathrm{a'_l})$, we remove $(\mathrm{a'_j}, \mathrm{a_k})$ edge and insert the edges $(\mathrm{a_i}, \mathrm{a'_j})$ and $(\mathrm{a_k}, \mathrm{a'_l})$ to augment M. But in our changed Hungarian algorithm when we insert the edge $(\mathrm{a_i}, \mathrm{a'_j})$ to the matching, we also insert the edge (a_j, a'_i) to the matching and when we remove the edge (a_i, a'_j) from the matching we also remove the edge (a_j, a'_i) from it. In the following we explain our changed Hungarian algorithm. We consider another set M' and when augmenting the main matching, that is removing (a'_j, a_k) edge from M and inserting the edges (a_i, a'_j) and (a_k, a'_l) to M, we also remove (a_j, a'_k) edge from M' and insert the edges (a_j, a'_i) and (a_l, a'_k) to M'. In the changed Hungarian algorithm a vertex is matched if it is in M' or M.

1. **Finding an initial feasible labeling.**
   This step is as the first step of the basic Hungarian algorithm.
2. If $M \cup M'$ is perfect, stop otherwise pick free vertex $a_i \in A$ and set $S = \{a_i\}, T = \emptyset$.
3. **Improving labels.**
   If $N_\ell(S) = T$, update labels (forcing $N_\ell(S) \neq T$)

$$\alpha_\ell = min_{a_i \in S, a'_j \notin T} \left\{\ell(a_i) + \ell(a'_j) - w(a_i, a'_j)\right\}$$

$$\ell'(v) = \begin{cases} \ell(v) - \alpha_\ell & if\ v \in S \\ \ell(v) + \alpha_\ell & if\ v \in T \\ \alpha_\ell & therwise \end{cases} \text{ then:}$$

1. If $N_\ell(S) \neq T$, pick $a'_j \in N_\ell(S) - T$.

- If a'_j is free, $a_i - a'_j$ is an augmenting path.
  Augment M and M'. That is remove (a'_j, a_k) edge from M and insert the edges (a_i, a'_j) and (a_k, a'_l) to M, remove (a'_k, a_j) edge from M' and insert the edges (a_j, a'_i) and (a_l, a'_k) to M'. Go to 2.
- If a'_j is matched, say to z, extend alternating tree:
  $S = S \cup \{z\}, T = T \cup \{a'_j\}$. Go to 3.

Theorem 1. The changed Hungarian algorithm computes the maximum-cost many-to-many matching between the points of a point set in $O(n^3)$ time.

Proof. The changed Hungarian algorithm stops when $M \cup M'$ is perfect. We show that when $M \cup M'$ is perfect the labels of G is a feasible labeling, so by Kuhn-Munkres theorem [11], M is a max-weight matching.

After stopping the changed Hungarian algorithm, for each edge (a_i, a'_j) in M', one by one, we update the labels of the nodes of G. For each edge $(a_i, a'_j) \in M'$, we compute

$\alpha_\ell = \ell(a_i) + \ell(a'_j) - w(a_i, a'_j)$, and then update the labels of the nodes as follows:

$$\ell'(v) = \begin{cases} \ell(v) - \alpha_\ell & if\ v = a_i \\ \ell(v) & otherwise \end{cases}$$

Then, $\ell(a_i) + \ell(a'_j) = \ell(a_i) - \ell(a_i) - \ell(a'_j) + w(a_i, a'_j) + \ell(a'_j) = w(a_i, a'_j)$, and so $(a_i, a'_j) \in E_{\ell'}$.

The time complexity of the changed Hungarian algorithm does not change, since the bipartite graph G is not changed and the changed Hungarian algorithm stops when $M \cup M'$ is perfect.☐

5. A special case

Let $A = \{a_1, a_2, .., a_s\}$ and $B = \{b_1, b_2, .., b_t\}$ be two sets of points with total cardinality n, and D_A and D_B be their degree demands. A degree satisfier maximum-cost many-to-many matching, called DSM, is a matching between A and B, where the degree demands of A and B are satisfied. That is the points of A and B must be matched with at least D_A and D_B points of the other set, respectively. In this section, we present an $O(n^2 \log n)$ time algorithm for computing the degree satisfier maximum-cost many-to-many matching between A and B sets.

$DSM - matching\ algorithm(A, B, D_A, D_B)$

1. Let G be the complete bipartite graph between A and B sets. Sort the edges of G with increasing edge-costs, such that $e_i\ for\ 1 \leq i \leq st$ denotes the ith edge of the sorted edges.
2. Select the last $\max(D_A, D_B)$ edges of the sorted edges and insert it to DSM.

Theorem 2. The degree satisfier maximum-cost many-to-many mathcing DSM between two sets A and B can be computed in $O(n^2 \log n)$ time using DSM-matching algorithm.

Proof. If DSM is not the degree satisfier maximum-cost many-to-many matching, there is a matching DSM' such that $w(DSM) \leq w(DSM')$, so sum of the edges in DSM' is greater than in DSM and it is not possible, because DSM is a set of the edges that has the greatest value. Ordering the edges takes $O(n^2 \log n)$ time, since the number of the edges in the complete bipartite graph G is $O(n^2)$. ☐

6. Conclusion

In this paper, we change the basic Hungarian algorithm, such that it can find the maximum weight matching between the points of a single set. The time complexity of the changed Hungarian algorithm with a single input point set is as the basic Hungarian algorithm. It is expected that the complexity of this n-dimensional matching problem will be reduced by exploiting the geometric information, so one and two dimensional versions of this problem remains open.

References

[1] A. Ben-Dor, R. M. Karp, B. Schwikowski, and R. Shamir, The restriction scaffold problem, Journal of Computational Biology, 3(2003), 385-398.

[2] R. E. Burkard and E. Cela, Linear assignment problems and extensions, Handbook of Combinatorial Optimization, Dordrecht, 4(1999), Kluwer Academic Publishers, 5-149.

[3] S. R. Buss and P. N. Yianilos, A bipartite matching approach to approximate string comparison and search, Technical report, NEC Research Institute, Princeton, New Jersey, 1995.

[4] T. C. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, MIT Press, Cambridge Mass, 2001.

[5] J. Colannino and G. T, An algorithm for computing the restriction scaffold assignment problem in computational biology, Information Processing Letters, 95(2005), 466-471.

[6] M. Fatih Demirci, Ali Shokoufandeh, Yakov Keselman, Lars Bretzner, and Sven Dickinson, Object recognition as many-to-many feature matching, International Journal of Computer Vision, 69(2006), 203-222.

[7] R. M. Karp and S. -Y. R. Li, Two special cases of the assignment problem, Discrete Mathematics, 13 (1975), 129-142.

[8] G. T. Toussaint, A comparison of rhythmic similarity measures, Proceedings of the 5th International Conference on Music Information Retrieval, Barcelona, Spain, October 10-14 2004, Universitat Pompeu Fabra, 242-245.

[9] G. T. Toussaint, The geometry of musical rhythm, In Proceedings of the Japan Conference on Discrete and Computational Geometry, 3742(2005), Springer-Verlag, Berlin-Heidelberg, 198-212.

[10] T. Eiter and H. Mannila, Distance measures for point sets and their computation, Acta Informatica, 34(1997), 109–133.

[11] L. R. Foulds. Combinatorial Optimization for Undergraduates, Springer Verlag, New York, 1984.

Received: October, 2011