Solving large-scale assignment problems by Kuhn-Munkres algorithm

Hong Cui[1,a], Jingjing Zhang[2,b], Chunfeng Cui[3,c], Qinyu Chen[4,d]

[1]Fiberhome Telecommunication Technologies Co.,Ltd, Wuhan, 430074, China
[2]College of Computer, National University of Defense Technology, Changsha, 410073, China
[3]Chinese Academy of Sciences, Beijing, 100000, China
[4]Netease.com, Beijing, 100000, China

[a]email:cui_hong1971@hotmail.com, [b]email:zhangjj_506@163.com,
[c]email:cuichf@lsec.cc.ac.cn, [d]email:chenqinyu@corp.netease.com

.

Keywords: Assignment problem; Kuhn-Munkres algorithm; sparse-KM; parallel-KM

Abstract. Kuhn-Munkres algorithm is one of the most popular polynomial time algorithms for solving classical assignment problem. The assignment problem is to find an assignment of the jobs to the workers that has minimum cost, given a cost matrix $X \in \mathbb{R}^{m\times n}$, where the element in the i-th row and j-th column represents the cost of assigning the i-th job to the j-th worker. the time complexity of Kuhn-Munkres algorithm is $O(mn^2)$, which brings prohibitive computational burden on large scale matrices, limiting the further usage of these methods in real applications. Motivated by this observation, a series of acceleration skills and parallel techniques have been studied on special structure. In this paper, we improve the original Kuhn-Munkres algorithm by utilizing the sparsity structure of the cost matrix, and propose two algorithms, sparsity based KM(`sKM`) and parallel KM(`pKM`). Furthermore, numerical experiments are given to show the efficiency of our algorithm. We empirically evaluate the proposed algorithm `sKM`) and (`pKM`) on random generated largescale datasets. Results have shown that `sKM`) greatly improves the computational performance. At the same time, (`pKM`) provides a parallel way to solve assignment problem with considerable accuracy loss.

Introduction

The assignment problem is one of the fundamental combinatorial optimization problems. It consists of finding a maximum weighted matching in a weighted bipartite graph. It is important both theoretically and practically. On one hand, it is a special case of more complex problem, such as the generalized assignment problem, matching problem in graph, minimum cost flow problem. On the other hand, many real world problems can be categorized in assignment problem, such as worker assignment problem.

The study of the assignment problem is quite mature. One of the most popular polynomial time algorithms is the Kuhn-Munkres algorithm, which was developed and published in 1955 by Kuhn [1]. As the assignment problem is linear programming in nature, the primal simplex method, and primal-dual simplex method for linear programming are also adapted to the assignment problem [2]. Kuhn-Munkres algorithm is also called "Hungarian method" because it was largely based on the earlier works of two Hungarian mathematicians in 1916 [3] and 1931 [4], even before the definition of linear programming.

Theoretically, the Kuhn-Munkres algorithm is guaranteed to reach the global optimal. There are a lot of study and modifications of Kuhn-Munkres algorithm. Bourgeois extend Kuhn-Munkres algorithm to rectangular arrays [5] in 1971. The Kuhn-Munkres algorithm is the first proved polynomial algorithm. Munkres observed that it is (strongly) polynomial [6] in 1957. Further, Edmonds and Karp [7], and independently Tomizawa noticed that it can be modified to achieve an $O(n^3)$ running time. For more details, the interested readers are referenced to [8, 9, 10, 11, 12, 13].

Formally speaking, given a cost matrix $X \in \mathbb{R}^{m\times n}$, where the element in the i-th row and j-th column represents the cost of assigning the i-th job to the j-th worker, the time complexity of newest Kuhn-Munkres algorithm is $O(mn^2)$, which brings prohibitive computational burden on large scale matrices, limiting the further usage of these methods in real applications. Algorithms suitable for large scale computing are more and more important. In history, a serial of work take advantage of special data structure. In 1987, Jonker et. al. [14] introduced three sets `TODO`, `READY` and `SCAN`, and their experiments show sound speed up for sparse problems. To decrease compuational time cost, Bertsekas et. al [15] discussed three possible synchornous and asynchronous implementations of the Kuhn-Munkres algorithm. They mainly focused on generate the

augmenting path in parallel, and differ in the work to be done by a processor before their latest computation been dealt. However, as the Kuhn-Munkres algorithm steps are highly related to each other, their algorithms have to discard any path conflicts with previous paths. And their numerical experiments also show very low parallel ratio.

In this paper, we focus on utilizing special data structure in large scale problems. With the assumption that the data is sparse, we designed two algorithms `sKM` and `pKM` based on the original Kuhn-Munkres algorithm. The outline of the following paper is organised as follows. We describe the formulation of the assignment problem and the original Kuhn-Munkres algorithm in §2. Our sparse and parallel algorithms are depicted in §3. At last, we give the numerical experiments to shown the efficiency of out algorithm.

The Assignment Problem

Consider that there are n jobs to be done with m workers available. Without loss of generality, we assume $n \leq m$. Any worker could be assigned to perform one of these jobs, each pair (job, worker) has a time cost for him to perform the job. Our goal is to complete all these jobs while minimizing total time cost, while assigning each worker to exactly one job and vice versa. Specifically, the assignment problem can be formulated to the following optimization problem:

$$\begin{array}{ll} \min & \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} * a_{ij}, \\ \text{s.t.} & \sum_{i=1}^{n} a_{ij} = 1, \quad 0 \leq \sum_{j=1}^{m} a_{ij} \leq 1, \\ & a_{ij} \in \{0,1\} \end{array} \tag{1}$$

where $C \in \mathbb{R}^{n\times m}$ is the cost matrix, c_{ij} is the cost of worker i to perform job j, $A \in \mathbb{R}^{n\times m}$ is the resulting binary matrix, $a_{ij} = 1$ if and only if the i-th worker is assigned to the j-th job, otherwise $a_{ij} = 0$.

The problem (1) could be considered as a mixed integer programming, which is NP-hard in general [16]. However, noting that the constraint matrix is totally unimodular, we can ignore the 0-1 constraints $a_{ij} \in \{0, 1\}$. As a result, the problem (1) is reduced to a linear programming, which is polynomial solvable.

From a different perspective, the assignment problem can also be considered as the maximum matching problem in graph theory, or bipartite graph in this case [17]. Let each edge between the i-th worker and j-th job has weight of c_{ij}. Then our task is to find minimum-weight matching in the graph, and the matching should consist of n edges, as the bipartite graph needs to be complete.

Kuhn-Munkres Algorithm

There are many implentation versions of Kuhn-Munkres algorithm with time complexity $O(n^3)$: graph [18], matrix [19]. In the following, without loss of generality, we shall discuss maximum assignment problem in a graph perspective. Furthermore, it is easy to transform the maximum problem to the minimum one, for example, by applying the following operation on the cost matrix $\overline{C}_{ij} = \text{maxCost} - C_{ij}$, where maxCost $= \max_{i,j} C_{ij}$.

Considering a complete bipartite graph $G = (V, E)$ where $V = X \bigcup Y, E \in X \times Y$, and $X \bigcap Y = \emptyset$. Note the weight of edge (x, y) as $w(x, y)$. A vertex v's neighborhood is the set $J_G(v)$ with all the vertices that share an edge with v, and a set S's neighborhood is the set $J_G(S)$ with all vertices that share an edge with a vertex in S. Define the labels $l(x)$ for each vertex in graph, which can also be seen as dual variables of the problem (1), with each label of a vertices corresponding to its only matching constraint. A feasible labelling is a function $l : V \to R$ satisfies the following condition $l(x) + l(y) \geq w(x, y), \forall x \in X, \forall y \in Y$. Considering a matching $M (M \subseteq E)$, vertex v is called matched if it is a vertex in M, otherwise it is called exposed (free, unmatched). Here we denote by G_l the subgraph of G which contains those edges where $l(x) + l(y) = w(x, y)$. G_l is a spanning subgraph of G, and includes all vertices from G. G_l only includes those edges from the bipartite matching which allow the vertices to be perfectly feasible.

Therotically, if M^* is a perfect matching in the equality subgraph G_l, then M^* is a maximum-weighted matching in G. We now show that there are no perfect matching with greater weight than M^*. Suppose M is

Algorithm 1 Original Kuhn-Munkres algorithm

Input: An bipartite graph $G = (V, E)$, and weight $w(x, y)$.

Output: Optimal perfect matching M.

Step 1: Generate initial labelling l and matching M in G_l.

Step 2: If M perfect, stop. Otherwise pick free vertex $u \in X$. Set $S = u$, $T = \emptyset$.

Step 3: If $J_l(S) = T$, update labels (forcing $J_l(S) \neq T$)

$$\alpha_l = \min_{s \in S, y \notin T} \{l(x) + l(y) - w(x, y)\} \tag{2}$$

$$\hat{l}(v) = \begin{cases} l(v) - \alpha_l, & v \in S \\ l(v) + \alpha_l, & v \in T \\ l(v), & \text{otherwise} \end{cases}$$

Step 4: If $J_l(S) \neq T$, chose $y \in J_l(S) - T$:

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

any given perfect mathcing, then

$$\begin{aligned} w(M) &= \sum_{x,y \in M} w(x, y) \leq \sum_{x,y \in M} \{l(x) + l(y)\} \\ &= \sum_{x} l(x) + \sum_{y} l(y) = \sum_{x,y \in M^*} \{l(x) + l(y)\} \\ &= \sum_{x,y \in M^*} w(x, y) \\ &= w(M^*) \end{aligned}$$

Therefore, M^* is a maximal perfect matching, and the Kuhn-Munkres algorithm is guaranteed to reach the global optimal.

Numerically, the time complexity of Algorithm 1 is $O(mn^2)$, where m is the number of elements in X, and n is the number of elements in Y.

Sparse and parallel Kuhn-Munkres algorithms

In this section, firstly we shall present two algorithms: sparse Kuhn-Munkres algorithm `sKM` and parallel Kuhn-Munkres algorithm `pKM`, then we will combine these two algorithms into `spKM`.

The sparse Kuhn-Munkres algorith `sKM`

We make the following assumption on the above formulation throughout two algorithms in the paper, sparse Kuhn-Munkres algorithm `sKM` and parallel Kuhn-Munkres algorithm `pKM`, then we will combine these two algorithms into `spKM`. Specifically, each node $x \in X$ is linked to partial nodes $y \in Y$.

Assumption 1:The original graph G is sparse.

In practice, the assumption is always hold true, especially for large scale problems. For instance, there are 10,000 jobs and 15000 worker and some workers are only capable of doing his most skilled jobs, so one job could be potentially assigned to at most 200 workers, then assignment cost matrix is relatively sparse, about 5‰in practice. It should be noted that ***operations in KM algorithm are mostly (or could be equally transformed to) row-wise and column-wise***, there are seldom random access operations. As a result, the sparsity and the memory access mode could make the sparse KM algorithm greatly reduce complexity of the KM algorithm.

At first, we implement an $O(mn^2)$ version of KM algorithm for dense matrix, and analyze the memory access operations in the algorithm. We find that these operations can be equally transformed to a row-wise and column-wise manner, which would take great advantage of efficient through a crosslinker data structure. The crosslinker is used here to store the data, and the data is manipulated along the links among the nodes.

Algorithm 2 The parallel Kuhn-Munkres algorithm pKM with t threads

Input: The weight matrix $W \in \mathbb{R}^{m\times n}$ with $m \leq n$

Output: The maximum perfect assignment for m workers.

Step 0: Reformulate the distance W into its band structure style matrix A.

Step 1: Split all the columns of A into t blocks uniformly.

Step 2: Distribute all rows into t blocks corresponding to the rows.

Step 3: Solve the subproblems in parallel by Algorithm 1.

Step 4: Check for uniqueness of each job, if more than two workers share the same job, chose a worker that has the maximal profit; output the results.

The crosslinker is relatively efficient to manipulate data with sparsity property, especially the operations are in column-wise or row-wise way.

The complexity analysis of sKM is analysed as follows. Define the number of nun-zero elements as

$$\tau = \#\text{ of non-zero elements.}$$

With the sparse assumption, we show the time complexity of Algorithm sKM is $O(m\tau)$. Denote a iteration as finding an augmenting path. Since at least one more match can be found each iteration, the total iteration number is $O(m)$. A complete iteration begins at **Step 2**, ends at first item of **Step 4**, and repeats between **Step 3** and **Step 4**, herein the main computation complexity lies in updating Equation (2). Firstly, the computation complexity to initialize α_l is $O(\tau)$, as there are no more than τ components. Noting that S and T is monotone increasing, the total changes of elements are no more than τ. Hence, the complexity each iterate to update α_l is $O(\tau)$. Therefore, the total complexity is $O(m\tau)$.

The parallel Kuhn-Munkres algorith pKM

Suppose there are m jobs in X, and n workers in Y. Further, without loss of generality, we assume that $m \leq n$.

Definition 4.1 The **breadth** bd of a matrix $A \in R^{m\times n}$ is defined as

$$bd := \max_{i,j} |j - i|, \quad \text{s.t.} \quad a_{i,j} \neq 0. \tag{3}$$

Theoretically, for any sparse matrix, if we arrange the row with the most zeros at the bottom, and rearrange its zeros at the left part; repeat this procedure for the rest of matrix while keep the arranged zeros unchanged. At last, the left-bottom of the matrix will be zeros. Similarly, we can rearrange zeros to the right-up part of the matrix.

For the assignment problem, the essence of band structure is to put the **similar jobs** together. Here, we claim two jobs are similar if they share many common workers.

Two notices should be claimed before we proceed. Firstly, for each subproblem in **Step 3** of Algorithm 2, it remains that jobs are less than workers. Secondly, if the reformulated data has very strong band structure, or even block diagonal, then we can split the original problem in k subproblems, and then solve each subproblem by Algorithm 2 with t computation processing unit. In this case, we can solve the original problem by as many core as possible.

The time complexity of Algorithm 2 is $O(m(n/\mathtt{t})^2)$. This can be followed from the fact that there are in total t subproblems computed in parallel, for each subproblem, there are m jobs, $n/\mathtt{t}$ workers, hence each subproblem costs $O(m(n/\mathtt{t})^2)$.

Algorithm `spKM`

Considering a sort of cases with great many jobs and relatively few workers, or formally speaking, $m << n$, where m is the number of workers and n is the number of jobs. There is high parallel potential to solve this kind of assignment problems.

We propose sparsity based parallel KM(`spKM`), which is achieved by replacing Algorithm 1 by `sKM` in **Step 3** of Algorithm 2. This algorithm utilizes the sparse and band structure of the assignment problem at most. The paralleled sparse KM consists of three stages: divide, subproblem solving by sparse KM and results merge. It should be noted that the merge process will not degrade the optimization of any subproblem.

Experiment of Sparse KM

We estimate the performance of the implemented sparse KM algorithm in this subsection, and compare it with the original KM algorithm under the sparse and dense conditions. We randomly generate the sparse job-worker cost matrix and use it to estimate the efficiency of the sparse KM on data with different scale. And our code will be provide to public after publish.

Table 1: The speed up of sparse KM on sparse matrix

scale	sparse rate	time of sparse KM	time of KM	speed up
1000	0.004994	0.08	11.48	143.50
1100	0.0045405	0.09	15.3	170.00
1200	0.0041625	0.11	19.93	181.18
1300	0.0038426	0.13	25.79	198.38
1400	0.00356837	0.16	32.28	201.75
1500	0.00333067	0.19	39.02	205.37
1600	0.00312266	0.21	47.01	223.86
1700	0.0029391	0.23	57.22	248.78
1800	0.00277593	0.25	67.64	270.56
1900	0.00262992	0.28	78.55	280.54

Table 2: The effective time rate of `spKM` on sparse cost matrix with the processing unit is fixed

scale	sparse rate	time solving	time total	effective time rate
6000	0.00192219	0.42	0.61	0.6885
7000	0.00161228	0.42	0.62	0.6774
8000	0.00138843	0.42	0.66	0.6364
9000	0.00121916	0.43	0.67	0.6667
10000	0.00108667	0.43	0.68	0.6324

In Table 1, we randomly generate cost matrix with scale of 500×500 to 1900×1900 with sparse rate changes from 2 ‰to 4%, and the speed up is the time of original KM divide the time of sparse KM, as expected the speed up is positively correlated to the reciprocal of the sparse rate. We further estimate the effective time rate of sparsity based parallel KM `spKM` on sparse cost matrix with scale of 6000×6000 to 10000×10000, we fix the processing unit upto 6 and the task size at least of 4000.

It should be noted that the performance of sparse KM is highly related to the sparse rate of the cost matrix. When the cost matrix is dense the sparse KM algorithm will perform worse than original KM algorithm as the sparse KM is built on crosslinker which will perform badly in dense cases.

Discussion

Kuhn-Munkres algorithm is one of the most popular polynomial time algorithms for solving classical assignment problem. The assignment problem is to find an assignment of the jobs to the workers that has minimum cost, given a cost matrix $X \in \mathbb{R}^{m \times n}$, where the element in the i-th row and j-th column represents the cost of assigning the i-th job to the j-th worker. the time complexity of Kuhn-Munkres algorithm is $O(mn^2)$, which

brings prohibitive computational burden on large scale matrices, limiting the further usage of these methods in real applications. Motivated by this observation, a series of acceleration skills and parallel techniques have been studied on special structure. In this paper, we improve the original Kuhn-Munkres algorithm by utilizing the sparsity structure of the cost matrix, and propose two algorithms, sparsity based KM(sKM) and parallel KM(pKM). Furthermore, numerical experiments are given to show the efficiency of our algorithm. We empirically evaluate the proposed algorithm sKM) and (pKM) on random generated largescale datasets. Results have shown that sKM) greatly improves the computational performance. At the same time, (pKM) provides a parallel way to solve assignment problem with considerable accuracy loss.

Reference

[1] Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly 2(2) (1955) 8397

[2] Bertsekas, D.P.: A new algorithm for the assignment problem. Mathematical Programming 21(1) (1981) 152171

[3] Konig, D.: Uber graphen und ihreanwendung aufdeterminantentheorie und mengenlehre. Mathematische Annalen 77(4) (1916) 453465

[4] Egervary, J.: Matrixok kombinatorius tulajdonsagairol [on combinatorial properties of matrices]. Matematikais Fizikai Lapok (in Hungarian) 38 (1931) 1628

[5] Bourgeois, F., Lassalle, J.C.: An extension of the munkres algorithm for the assignment problem to rectangular matrices. Communications of the Acm 14 (1971) 802804

[6] Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics 5(1) (1957) 3238

[7] Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. (1972)

[8] Kuhn, H.W.: Variants of the hungarian method for assignment problems. Naval Research Logistics Quarterly, 3(253-258) (1956)

[9] Kuhn, H.W.: On the origin of the hungarian method, in: History of mathe- matical programming a collection of personal reminescences. CWI Amsterdam and North-Holland (1991) 7781

[10] Frank, A.: On kuhns hungarian method a tribute from hungary. Egrervary Technical Report 14 (2004)

[11] Bertsekas, D.P.: The auction algorithm for assignment and other network flow problems. Birkhauser Boston (1991) 105112

[12] Zhu, H., Zhou, M.C., Alkins, R.: Group role assignment via a kuhnmunkres algorithm-based solution. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICSPART A: SYSTEMS AND HUMANS 42(3) (2012) 739750

[13] Mills-Tettey, A., Stent, A., Dias, M.B., Mills-Tettey, A., Stent, A.: The dynamic hungarian algo- rithm for the assignment problem with changing costs. Carnegie Mellon University (2007)

[14] Jonker, R., Volgenant, A.: A shortest augmenting path algorithm for dense and sparse linear as- signment problems. Computing (1987)

[15] Bertsekas, D.P., Castanon, D.A.: Parallel asynchronous hungarian methods for the assignment problem. Orsa Journal on Computing 5 (2003) 212230

[16] Wolsey, L.A.: Integer Programming. Wiley Publication (1998)

[17] Glover, F.: Maximum matching in a convex bipartite graph. Naval Research Logistics Quarterly 14(3) (1967) 313316

[18] Andree, M.: o(n3) implementation of the hungarian algorithm (2011)