Algorithmic Operations Research

The Computational Efficiency of the Ji-Lee-Li Algorithm for the Assignment Problem

Boris Goldengorin et Gerold Jäger

Volume 3, numéro 1, winter 2008

URI : https://id.erudit.org/iderudit/aor3_1sc01

Aller au sommaire du numéro

Éditeur(s)

Preeminent Academic Facets Inc.

ISSN

1718-3235 (numérique)

Découvrir la revue

Citer cet article

Goldengorin, B. & Jäger, G. (2008). The Computational Efficiency of the Ji-Lee-Li Algorithm for the Assignment Problem. *Algorithmic Operations Research*, *3*(1), 79–81.

Résumé de l'article

Ji et al. have conjectured that using the matrix form (to represent a basic solution) instead of the Simplex tableau in the dual Simplex method will lead to an algorithm with the time complexity comparable to the Hungarian algorithm for solving the Assignment Problem. In this note we show that both the time complexity and the CPU times of the Ji et al. algorithm are far away from being competitive to the Hungarian algorithm.

Algorithmic Operations Research Vol.3 (2008) 79–81

The Computational Efficiency of the Ji-Lee-Li Algorithm for the Assignment Problem

Boris Goldengorin

Faculty of Economics and Business, University of Groningen, 9700 AV Groningen, The Netherlands, and Department of Mathematics and Informatics, Khmelnitsky University of Economics and Business, Ukraine

Gerold Jäger

Computer Science Institute, University of Halle-Wittenberg, D-06099 Halle (Saale), Germany

Abstract

Ji et al. have conjectured that using the matrix form (to represent a basic solution) instead of the Simplex tableau in the dual Simplex method will lead to an algorithm with the time complexity comparable to the Hungarian algorithm for solving the Assignment Problem. In this note we show that both the time complexity and the CPU times of the Ji et al. algorithm are far away from being competitive to the Hungarian algorithm.

Key words: Assignment Problem, Hungarian Algorithm, Dual Simplex Algorithm.

1. Introduction

For a matrix $C = (c_{ij})_{1 \le i,j \le n} \in \mathbb{Z}^{n,n}$ the Assignment Problem (AP) is defined as follows:

$$\min \Big\{ \sum_{i=1}^{n} c_{i,\pi(i)} : \pi \in \Pi_n \Big\}$$

where Π_n is the set of all permutations of $\{1, \ldots, n\}$.

The approaches proposed for the solution of the Assignment Problem can be classified into three classes: primal-dual (shortest path) algorithms, pure primal algorithms, and pure dual algorithms. An experimental evaluation of a best representative algorithm from each of these classes shows that the best algorithm is the Hungarian algorithm based on the primal-dual (shortest path) approach and König-Egervary's theorem [2]. The time complexity of the Hungarian algorithm is $O(n^3)$ (see e.g., [2]).

Ji et al. [5] have suggested a reduction of a $n \times n$ AP instance to an equivalent $2n \times 2n$ instance for solving the AP. Their algorithm (abbreviated by JLL-algorithm) is a pure dual algorithm following to the dual Simplex method. In conclusions, Ji et al. [5] conjecture that since the JLL-algorithm adopts a compact matrix form instead of the Simplex tableau in the dual Simplex algorithm and since it is not based on König-Egervary's theorem, it can be an alternative to the Hungarian algorithm. In [2], eight codes are selected and compared on a wide set of dense instances containing both randomly generated and benchmark instances. These codes represent the most popular and efficient methods for solving the Assignment Problem (see [1,3,7]) as follows: "four algorithms are pure shortest path methods, one is a mixture of auction and shortest path technique, two are implementations of a pure auction technique and the last one is a pseudoflow-based algorithm" (see [2]). Since the Jonker and Volgenant's code (abbreviated by JV-algorithm [7]) has a good and stable average performance for all tested instances [2] and it is based on the Hungarian algorithm, we have compared the JLL-algorithm against the JV-algorithm.

In this note we show that the transformation of an $n \times n$ AP instance into a $2n \times 2n$ AP instance and the exclusion of an application of König-Egervary's theorem are fatal for the JLL-algorithm.

Our note is organized as follows. We consider a trivial (all zeros) AP instance in Section 2.. In Section 3. we present a comprehensive experimental evaluation of the JLL-algorithm in comparison to the JV-Algorithm on a wide set of small instances, because the average size instances are intractable in reasonable PC times by the JLL-algorithm. Our conclusions appear in Section 4..

Email: Boris Goldengorin [B.Goldengorin@rug.nl], Gerold Jäger [jaegerg@informatik.uni-halle.de].

2. An Illustrative Example for the JLL-Algorithm

We consider the JLL-algorithm on an example of a matrix with all zero entries. Since all entries are zeros, any well-known algorithm, for example the Hungarian algorithm, will skip the reduction steps and straightforward by application of the König-Egervary's algorithm will output an optimal matching, but – as a simple induction type reasoning shows – the JLL-algorithm needs $\frac{(n-1)n}{2}$ loops for finding such a matching. Thus the time complexity of JLL-algorithm is at least $O(n^4)$ and worse than the time complexity $O(n^3)$ of the Hungarian algorithm.

3. An Experimental Evaluation of the JLL-algorithm

We have implemented the JLL-algorithm in C under Linux on a GenuineIntel Intel® Xeon™ 2.66 machine with 2 GB RAM and have compared it with the C implementation of the JV-algorithm by Jonker and Volgenant [8].

We have conducted our experiments on the following 8 classes of matrices, the first 6 of which are of dimensions 50, 100, 150, 200.

- **Class 1.** Zero matrix, i.e. each matrix element equals zero.
- **Class 2.** Machol-Wien-Class [9,10]: the matrix element (i, j) equals $(i-1)\cdot(j-1)$.
- **Class 3.** Gutin-Yeo-Zverovich Class [4]: the matrix element (i, j) equals

$$\begin{cases} n^3 & \text{for } i = n, j = 1; \\ in & \text{for } j = i+1, i = 1, 2, \ldots, n-1; \\ n^2 - 1 & \text{for } i = 3, 4, \ldots, n-1; j = 1; \\ n\min\{i, j\} + 1 & \text{otherwise.} \end{cases}$$

- **Class 4.** Each matrix element equals 1 with probability 0.25 and 10^6 otherwise (sparsity 0.25).
- **Class 5.** Each matrix element equals 1 with probability 0.5 and 10^6 otherwise (sparsity 0.5).
- **Class 6.** Each matrix element equals 1 with probability 0.75 and 10^6 otherwise (sparsity 0.75).
- **Class 7.** All 26 asymmetric instances from TSPLIB [11].
- **Class 8.** 10 asymmetric problem generators from Johnson et al. [6], called *amat*, *coin*, *crane disc*, *rect*, *rtilt*, *shop*, *stilt*, *super*, *tmat*, are considered as a subclass of our class 8. In [6], 10 instances of dimensions 100, and 10 of dimension 316 are chosen for each of these generators.

The execution times in seconds of the algorithms for all classes can be found in Tables 1-7. The results show that for *all* instances the JV-algorithm is much faster than the JLL-algorithm.

4. Conclusions

Ji et al. [5] have conjectured that using the matrix form (to represent a basic solution) instead of the Simplex tableau in the dual Simplex algorithm will lead to a competitor to the Hungarian algorithm for solving the AP. They have not indicated the time complexity of their JLL-algorithm for the AP. In section 2. we have observed that even for a trivial zero matrix the JLL-algorithm needs $\frac{(n-1)n}{2}$ complete loops. Thus the time complexity of JLL-algorithm is at least $O(n^4)$ and worse than the time complexity $O(n^3)$ of the Hungarian algorithm.

Our computational results, obtained for the JLL-algorithm, show that the primal-dual (shortest path) algorithm based on the König-Egervary's theorem and implemented by Jonker and Volgenant (JV-algorithm) [7] outperforms the JLL-algorithm for a wide set of classes with dimension from 17 to 443. For the largest instances including the Machol-Wien instances [9,10] (which are the most difficult instances for the Hungarian algorithm), the JV-Algorithm is faster by a factor of 1,000 for almost all cases.

Acknowledgement: The research of both authors was supported by a DFG grant MO645/7-3, Germany.

References

[1] D.P. Bertsekas. A New Algorithm for the Assignment Problem. Math. Program. 1981; 21: 152-171.

[2] M. Dell'Amico, P. Toth. Algorithms and Codes for Dense Assignment Problems: the State of the Art. Discrete Appl. Math. 2000; 100(1-2): 17-48.

[3] A.V. Goldberg, R. Kennedy. An Efficient Cost Scaling Algorithm for the Assignment Problem. Math. Program. 1995; 71: 153-177.

[4] G. Gutin, A. Yeo, A. Zverovich. Traveling Salesman Should Not Be Greedy: Domination Analysis of Greedy Type Heuristics for the TSP. Discrete Appl. Math. 2002; 117: 81-86.

[5] P. Ji, W.B. Lee, H. Li. A New Algorithm for the Assignment Problem: an Alternative to the Hungarian Method. Computers Ops. Res. 1997; 24(11): 1017-1023.

Table 1

n	Class 1		Class 2		Class 3		Class 4		Class 5		Class 6	
	JV	JLL	JV	JLL	JV	JLL	JV	JLL	JV	JLL	JV	JLL
50	0.000	0.456	0.003	0.447	0.000	0.055	0.000	0.107	0.000	0.218	0.000	0.321
100	0.001	9.627	0.013	9.463	0.003	0.607	0.000	2.641	0.001	4.808	0.000	6.221
150	0.002	59.226	0.028	58.193	0.008	2.542	0.001	20.381	0.001	29.239	0.002	39.316
200	0.003	203.971	0.064	200.259	0.018	6.445	0.003	95.001	0.002	100.483	0.003	141.136

Comparison of JV-algorithm and JLL-algorithm for classes 1 to 6

Table 2

Name	JV	JLL	Name	JV	JLL	Name	JV	JLL	Name	JV	JLL
br17	0.000	0.001	p43	0.000	0.015	ry48p	0.000	0.015	ft53	0.000	0.036
ft70	0.001	0.152	ftv33	0.000	0.010	ftv35	0.000	0.013	ftv38	0.000	0.018
ftv44	0.000	0.023	ftv47	0.000	0.025	ftv55	0.001	0.031	ftv64	0.000	0.056
ftv70	0.001	0.060	ftv100	0.001	0.262	ftv110	0.001	0.333	ftv120	0.001	0.480
ftv130	0.001	0.601	ftv140	0.001	0.856	ftv150	0.001	1.049	ftv160	0.001	1.213
ftv170	0.002	1.605	kro124p	0.001	0.259	rbg323	0.016	105.273	rbg358	0.019	338.208
rbg403	0.021	975.603	rbg443	0.026	1652.237						

Comparison of JV-algorithm and JLL-algorithm for class 7

Table 3

n	amat		coin		crane		disk		rect	
	JV	JLL	JV	JLL	JV	JLL	JV	JLL	JV	JLL
100	0.001	0.381	0.000	0.208	0.002	0.278	0.001	0.388	0.001	0.205
316	0.009	29.323	0.004	9.977	0.015	13.685	0.014	33.652	0.004	9.514

Table 4

n	rtilt		shop		stilt		super		tmat	
	JV	JLL	JV	JLL	JV	JLL	JV	JLL	JV	JLL
100	0.002	0.637	0.004	1.363	0.003	0.325	0.000	0.488	0.000	0.472
316	0.025	39.793	0.059	116.715	0.042	16.753	0.007	44.043	0.009	34.335

Comparison of JV-algorithm and JLL-algorithm for class 8, average over 10 instances for each subclass and each dimension

[6] D.S. Johnson, G. Gutin, L.A. McGeoch, A. Yeo, W. Zhang, A. Zverovich. Experimental Analysis of Heuristics for the ATSP. Chapter 10 in: The Traveling Salesman Problem and Its Variations. G. Gutin, A.P. Punnen (eds.). Kluwer, Dordrecht, 445-489, 2002.

[7] R. Jonker, A. Volgenant. A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems. Computing 1987; 38: 325-340.

[8] R. Jonker, A. Volgenant. Source code of the JV-algorithm. Available at http://www.magiclogic.com/assignment.html.

[9] R.E. Machol, M. Wien. A 'hard' Assignment Problem. Oper. Res. 1976; 24: 190-192.

[10] R.E. Machol, M. Wien. Errata to [9]. Oper. Res. 1977; 25(2): 364.

[11] G. Reinelt. TSPLIB – a Traveling Salesman Problem Library. ORSA J. Comput. 1991; 3: 376-384.

Received 29 March, 2007; revised 23 May 2007; accepted 29 May 2007