technical transactions

No. 2020/005

electrical engineering

sciendo

An improved ant algorithm for the triple matching problem

Krzysztof Schiff

kschiff@pk.edu.pl | Orcid 0000-0002-2651-6799

Department of Automatic Control and Technology Information, Faculty of Electrical and Computer Engineering, Cracow University of Technology

Scientific Editor: Piotr Drozdowski, Cracow University of Technology
Technical Editor: Aleksandra Urzędowska, Cracow University of Technology Press
Language Editor: Tim Churcher, Big Picture
Typesetting: Małgorzata Murat-Drożyńska, Cracow University of Technology Press

Received: November 11, 2018
Accepted: April 3, 2020

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Competing interests: The authors have declared that no competing interests exist.

Citation: Schiff, K. (2020). An improved ant algorithm for the triple matching problem. *Technical Transactions, e2020005*. https://doi.org/ 10.37705/TechTrans/e2020005

Abstract

In this article a new version of the ant colony optimisation algorithm with a desirability function for the triple matching problem is described. The problem is modelled by means of two 2-dimensional arrays. The new version of the ant algorithm was compared with the previous version of the ant algorithm and tested for different values of ant algorithm parameters; the results of these tests are presented and discussed.

Keywords: triple maximum matching problem, ant colony optimisation

1. Introduction

Triple matching is a generalisation of bipartite matching. Finding the largest triple matching problem is a well-known NP-hard problem. It is one of Karp's 21 NP-complete problems (Karp, 1972; Knuth, 1997) and until thus far now, there has been no exact polynomial time algorithm for this problem. There are no exact polynomial time algorithms for the maximum triple-matches problem, so there have been many heuristics algorithms elaborated for them (Biro, McDermid, 2010; Eriksson, Sjostrand, Strimling, 2006; Chen, 2012). Amongst heuristic algorithms there are ant algorithms and they are very well suited for obtaining solutions for combinatorial problems (Dorigo, Stützle, 2002). The previous version of the elaborated ant algorithm for the triple matching problem is the algorithm, which has recently been presented in a previous paper by the author (Schiff, 2018). An instance of the three-dimensional triple maximum matching problem consists of three sets of persons: the first of b persons, the second of g persons and the third of p persons and in these sets, we are looking for maximum matching M, which consists of k triples (b, g, p) such that each b, each g and each p belongs to exactly one triple. Today, there exists an ant algorithm for the maximum triple matching problem (Schiff, 2018) and this previous version of the ant algorithm has been improved and the new version of this algorithm is presented in this paper; results of the comparison of both of these two ant algorithms are also shown in this paper.

2. The triple matching problem

In triple matching, we are given X, Y, Z, and T where:

1) X, Y and Z are 3 disjoint sets, each of size n.
2) $T = \{(x,y,z): x \hat{I} X, y \hat{I} Y, z \hat{I} Z\} \hat{I} X \times Y \times Z$ and we have to find M such that:
   a) $M \acute{I} T$
   b) $|M| = k$
   c) for any 2 distinct elements of M, (x,y,z) and (x',y',z'), $x \neq x', y \neq y'$ and $z \neq z'$

The maximum triple matching problem relies on finding max $|M|$.

The triple matching problem and a solution for this problem is presented in Fig. 1. An edge represents a preference. Each three-dimensional clique (a triangle) represents a match between b person, g person and p person. A maximum matching in Fig. 1 is presented by three triangles: (B1, G1, P3), (B2, G2, P1) and (B3, G3, P2).

This instance of the triple matching problem which is presented in Fig. 1 as a three-dimensional graph can be presented by three 2-dimensional arrays, which have been shown in Table 1: the first array shows the relations between a set of b persons and a set of g persons, the second shows the relations between a set of b persons and a set of p persons and finally, the third shows the relations between a set of g persons and a set of p persons.

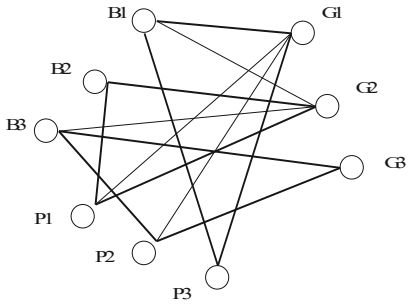

Fig. 1. The triple problem modelled as a 3-dimensional graph and its solution

Table 1. Three two-dimensional arrays for the case presented in Fig. 1

gp[][]	P1	P2	P3		bp[][]	P1	P2	P3		bg[][]	G1	G2	G3
G1	1	1	1		B1	0	0	1		B1	1	1	0
G2	1	0	0		B2	1	0	0		B2	0	1	0
G3	0	1	0		B3	0	1	0		B3	0	1	1

3. Structure of the ant algorithm

Each ant in the ant algorithm creates a solution to a problem. In each cycle, solutions received by m ants are compared and the best one is remembered, which is compared with the best solutions from previous cycles. The ant

https://doi.org/ 10.37705/TechTrans/e2020005

algorithm which is presented in this article, differs from the ant algorithm which is presented in the previous paper (Schiff, 2018). The difference lies in the way of modelling; the algorithm which is presented in the previous paper (Schiff, 2018) uses a three-dimensional array, while the algorithm which is presented in this paper uses three two-dimensional arrays, which are shown in Table 1. These arrays are used to model a problem. This change of data structure results in a new ant algorithm and to a new algorithm action.

At the beginning, on all edges (which represent preferences between b persons, g persons and p persons) the maximum quantities of pheromone are given t_{max} ($tgb[][] = t_{max}$, $tgbp[][] = t_{max}$). Then for each pair (g,b) the number of preferences between each pair (g,b) and each p person were computed and the largest number of such preferences among all pairs (g,b) and p person was determined (lines 1–8). The desirability functions fn for all pairs (g,b) were computed (lines 9–11).

The ant algorithm consist of two loops: the first for cycles and the second for ants (line 12). Arrays allowed_b[] and allowed_g[] are used in such a way that when any ant finds a relation of preferences between b person and g person, then this b person and this g person could not be chosen for any other relation of preference. This exclusion is made by the assignment of a value of 0 to elements of arrays allowed_b[] and allowed_g[], which represent this b person and this g person (lines 41–42). Thus, we assure that any b person and any g person are not chosen twice for two different matches and thus, we assure that the solution will be correct. If elements of arrays allowed_b[] and allowed_g[] has values of 1 assigned, this means that these boys and these girls could be chosen by ants in order to make a new match. The selection of pairs of b persons and g persons is made by each ant by means of a roulette (lines 13–47) and here, the desirability function is used (lines 21 and 29). Next, for our intermediate solution sol[][], this means for the received maximum number of pairs of g persons and b persons, ant constructs a table gbp[][] and assigns a number 1 when a pair (b, g) has the same preference with p persons (lines 48–53) and now we have triples (b, g, p) and between them, we look for the maximum number and thus constraint 2 c is fulfilled.

The loop *while(yes)* is repeated as long as there is a match (b,g) which could be added to the solution (line 13). Thus, each ant finds a solution, which comprised of k – matches. The loop *while(yes)* can be repeated n times. All matches between b person and g person are in the array sol[b][g], so based on this array, an array bgp[b or g][p] is created (lines 48–53). When we have a match between a g person and a b person, an ant checks if they have the same mutual preference with a p person and if so, a value of 1 is assigned to an element of array bgp[b or g][p] = 1. If there is no such preference, a value of 0 will be assigned to element of array bgp[b or g][p] = 0. It does not matter if we choose the index of a b or g person, but we have to be cinsistent in our selection.

Lines 54–84 of the algorithm work in the same way as in lines 13–47 if the algorithm), but there is difference in that the action is made on the array bgp[][] instead of the array bg[][]; there is not a desirability function in the roulette method and the solution is kept in the array sol2[][].

Lines 83 and 84 are known steps of the ant algorithm and why these steps are not described here in details. If someone wants to know these details, they can be found in paper (Schiff, 2018). Each ant checks whether a solution is better than previously found solutions and if so, this solution is remembered (line 83). To achieve this, each ant computes a number of matches ld, which a solution is constituted of. From all solutions received in one cycle, only one with the largest number of matches ldb is selected. This number ldb is compared with the number ldg and the larger number is kept, this means the better solution is kept. These numbers are used to compute quantities of the pheromone, which should be put on all matches which constitute a solution, this means they should be put on preferences in table gb[][] and gbp[][]. Next, an ant communication system is implemented (line 84): the evaporation rate of pheromone r and additional quantities of pheromone dt are calculated and deposited on all

matches form the best solution. The additional quantity of the pheromone is calculated according to the pattern $dt = (1/(1-((ldg-ldb)/ldg)))$, where ldg is the highest number of triples which have been found during the entire action of the algorithm until the moment of update and ldb is the highest number of triples which have been found during the last cycle.

The partial, but main code of the elaborated new version of the ant algorithm for the three-dimensional maximum matching problem is presented as Algorithm 1.

Algorithm 1

The elaborated new version of the ant algorithm for the maximum triple--matching problem

```
1. fnmax = 0;
2. for(ii = 0; ii<n; ii++)
3.         for(jj = 0; jj<n; jj++)
4.           { if (km[ii][jj] = = 1)
5.                       for(zz = 0; zz<n; zz++)
6.                                 if ((gp[ii][zz] = = 1) && (bp[jj][zz] = = 1))
7.                                           fn[ii][jj] = fn[ii][jj]+1;
8.           if (fn[ii][jj]>fnmax) fnmax = fn[ii][jj];                    }
9. for(ii = 0; ii<n; ii++)
10.      for(jj = 0; jj<n; jj++)
11.        if (fn[ii][jj]>0) (fn[ii][jj] = ( 1/ (1 - ( (float) (fnmax-fn[ii][jj])/fnmax))));
12. for all cycles and ants, each ants looks for the solution
13. yes = 1; while(yes)
14. {    yes = 0;
15.      sumat = 0;
16. for(ii = 0; ii<n; ii++)
17.      { if (avalable_g[ii] = = 1)
18.        {sumap = 0;
19.       for(jj = 0; jj<n; jj++)
20.                  if (((gb[ii][jj] = = 1) && (avalable_b[jj] = = 1)) && (fn[ii][jj]>0))
21.                  {          sumat = sumat+ (tgb[ii][jj] * fn[ii][jj]);
22.                             yes = 1;                                    }
23.        }//jj       }//ii
24. if (yes = = 1)
25.      { for(ii = 0; ii<n; ii++)
26.      { if (avalable_g[ii] = = 1)
27.      { for(jj = 0; jj<n; jj++)
28.                  if ((gb[ii][jj] = = 1) && (avalable_b[jj] = = 1))
29.                  {          prob[ii][jj] = (tgb[ii][jj]*fn[ii][jj]);
30.                             prob[ii][jj] = (prob[ii][jj]/ sumat); } }//jj }//ii
31. pr = (rand()/(double)32767);
32.      sumap = 0;
33.      for(ii = 0; ii<n; ii++)
34.      {  if (avalable_g[ii] = = 1)
35.        { for(jj = 0; jj<n; jj++)
36.                  if ((gb[ii][jj] = = 1) && (avalable_b[jj] = = 1))
37.                  { sumap = sumap+prob[ii][jj];
38.                             ppjj = jj;
39.                   if (sumap> = pr)
40.                  { sol[ii][jj] = 1;
41.                   avalable_g[ii] = 0;
42.                   avalable_b[jj] = 0;
43.                   ppjj = jj;
44.                   jj = n;
45.                   ii = n; }}}//j         }//ii
46. }//if (yes = = 1)
```


https://doi.org/ 10.37705/TechTrans/e2020005

```
47. }//while(yes) we have a maximum matching between g persons and b persons
        in sol[][]
48. for(ii = 0; ii<n; ii++)
49. for(jj = 0; jj<n; jj++)
50.      if (sol[ii][jj] = = 1)
51.      { for(k = 0; k<n; k++)
52.        if ((gp[ii][k] = = 1) && (bp[jj][k] = = 1))
53.                                  gbp[ii][k] = 1; }
54. yes = 1;
55. while(yes)
56. { yes = 0; sumat = 0;
57. for(ii = 0; ii<n; ii++)
58. {    if (avalable_gb[ii] = = 1)
59.      {  for(jj = 0; jj<n; jj++)
60.                  if ((gbp[ii][jj] = = 1) && (avalable_p[jj] = = 1))
61.                  {          yes = 1; sumat = sumat+tgbp[ii][jj]; } } }//ii
62.     if (yes = = 1)
63.     {for(ii = 0; ii<n; ii++)
64.     { if (moznax[ii] = = 1)
65.      { for(jj = 0; jj<n; jj++)
66.                  if ((kmz[ii][jj] = = 1) && (moznay[jj] = = 1))
67.                              prob[ii][jj] = (tkmz[ii][jj]/sumat); }//jj }//ii
68. pr = (rand()/(double)32767);
69       sumap = 0;
70.     for(ii = 0; ii<n; ii++)
71.     { if (moznax[ii] = = 1)
72.       { for(jj = 0; jj<n; jj++)
73.       {          if ((kmz[ii][jj] = = 1) && (moznay[jj] = = 1))
74.                  { sumap = sumap+prob[ii][jj];
75.                  if (sumap> = pr)
76.                  { avalable_p[jj] = 0;
77.                  sol2[ii][jj] = 1;
78.                  avalable_g[ii] = 0;
79.                  avalable_b[jj] = 0;
80.                  jj = n; ii = n; } }}//jj } }//ii
81.     }//if (yes = = 1)
82. }//while(yes) our solution is in sol2[][]
83. the best solution of all ants }//ants
84. the mechanism of evaporation communication }//cycles
```

4. Experiments

The conducted experiments focussed on the comparison of the elaborated new version of the ant algorithm in which a desirability function was used (the IM_Ant3Dmatching algorithm), with the previous algorithm (the Ant3Dmatching algorithm) (Schiff, 2018). During all experiments, the IM-Ant3Dmatching algorithm performed better than the Ant3Dmatching algorithm with regards to the maximum number of triple-matches. Results of comparison are shown in Tables 2–6. The values in all the tables are the average results from ten measurements which were performed under the same parameters.

Table 2. Average size of maximum matching when lc = 100, lm = 30, r = 0.998 and q = 0.07

N	20	30	40	50	60
Ant3Dmatching	1.1	6.3	11.2	17.9	29.5
IM_Ant3Dmatching	1.1	6.7	12.2	19.9	32.2

Table 3. Average size of maximum matching when n = 50, lm = 30, r = 0.998 and q = 0.07

Lc	50	100	150	200	250
Ant3Dmatching	17.1	17.9	18.9	18.4	19.4
IM_Ant3Dmatching	18.6	19.9	20.5	20.5	21.0

Table 4. Average size of maximum matching when n = 50, lc = 100, r = 0.998 and q = 0.07

lm	10	20	30	40	50
Ant3Dmatching	19.0	20.4	17.9	18.5	18.8
IM_Ant3Dmatching	20.8	22.8	19.9	20.7	20.0

Table 5. Average size of maximum matching when lc = 100, lm = 30, n = 50 and q = 0.07

R	0.990	0.992	0.994	0.996	0.998
Ant3Dmatching	17.8	19.8	18.5	19.6	17.9
IM_Ant3Dmatching	20.0	21.7	20.8	21.0	19.9

Table 6. Average size of maximum matching when lc = 100, lm = 30, n = 50 and r = 0.998

q	0.04	0.07	0.10	0.13	0.16
Ant3Dmatching	6.5	17.9	32.9	38.8	42.2
IM_Ant3Dmatching	6.8	19.9	37.2	44.3	47.5

5. Conclusions

In this article, a new version of the ant algorithm with a desirability function is presented, which is called IM_Ant3Dmatching. This new algorithm has been compared with a previous version of the ant algorithm called Ant3Dmatching and the new ant algorithm showed its superiority in all of the conducted experiments, which were conducting and yielded a higher number of triple mtaches.

The new version of the ant algorithm uses three two-dimensional arrays, the previous uses only one three-dimensional array. The new version has a lower time complexity ($lc*lm*n^4$) than the previous ant algorithm ($lc*lm*n^5$). The new version of ant algorithm uses also a desirability function when the previous ant algorithm uses any desirability function.

References

Biro, P., McDermid, E. (2010). Three-sided stable matching with cyclic preferences. *Algorithmica*, *58*(1), 5–18.

Chen, J. (2012). Iterative Expansion and Color Coding: an Improved Algorithm for 3D-Matching. *ACM Transactions on Algorithms*, 6.1–6.22.

Dorigo, M., Stützle, T. (2002). Ant colony optimization. In *Proceedings of EvoWorkshops 2002* (pp. 61–71). Berlin: Heidelberg: Springer-Verlag.

Eriksson, K., Sjostrand, J., Strimling, P. (2006). Three-dimensional stable matching with cyclic preferences. *Math. Soc. Sci.*, *52*(1), 77–87.

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. Miller, J. Thatcher (Eds.), *Complexity of Computer Computations* (pp. 85–103). New York: *Plenum*.

Knuth, D. (1997). *Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of algorithms*. Rhode Island: American Mathematical Society.

Schiff, K. (2018). An ant algorithm for the triple matching problem. *Technical Transaction*, *2*(115), 179–186.

https://doi.org/ 10.37705/TechTrans/e2020005

Udoskonalony algorytm mrówkowy dla potrójnego zagadnienia dopasowania

Streszczenie

W artykule został przedstawiony w nowszej wersji algorytm mrówkowy wraz z funkcją pożądania dla problemu potrójnego zagadnienia dopasowania. Problem potrójnego dopasowania zaprezentowano przy pomocy tablic dwuwymiarowych. Algorytm mrówkowy został porównany ze starszą wersją algorytmu mrówkowego i przetestowany przy różnych wartościach parametrów algorytmu mrówkowego, a wyniki tych testów pokazano i omówiono.

Słowa kluczowe: zagadnienie potrójnego maksymalnego dopasowania, algorytm mrówkowy