TECHNICAL TRANSACTIONS /2018
ELECTRICAL ENGINEERING
DOI: 10.4467/2353737XCT.18.028.8001
SUBMISSION OF THE FINAL VERSION: 22/1/2018

Krzysztof Schiff (kschiff@pk.edu.pl)
Department of Automatic Control and Technology Information, Faculty of Electrical and Computer Engineering, Cracow University of Technology

An ant colony optimisation algorithm for the triple matching problem

Algorytm mrówkowy dla potrójnego zagadnienia dopasowania

Abstract

In this article, ant colony optimisation algorithms for the triple matching problem are described. This is the first elaborated ant algorithm for this problem. The problem is modeled by means of a 3-dimensional array. The ant algorithm was compared with the Apx3Dmatchnig-F algorithm and tested for different values of ant algorithm parameters. The results of these tests were presented and discussed.

Keywords: triple maximum matching problem, ant colony optimization algorithm, non weighted version

Streszczenie

W artykule został przedstawiony po raz pierwszy algorytm mrówkowy dla problemu potrójnego zagadnienia dopasowania. Problem potrójnego dopasowania zaprezentowano przy pomocy tablicy trój-wymiarowej. Algorytm mrówkowy został porównany z algorytmem Apx3Dmatching-F i przetestowany przy różnych wartościach parametrów algorytmu mrówkowego, a wyniki tych testów zostały zaprezentowane i omówione.

Słowa kluczowe: potrójne maksymalne dopasowanie, algorytm mrówkowy, wersja bez wag

1. Introduction

Triple matching is a generalisation of bipartite matching. Finding the largest triple matching is a well-known NP-hard problem. It is one of Karp's 21 NP-complete problems [1] and until now, there was no polynomial time algorithm for this problem. The DNA algorithm for the triple matching problem was presented in paper [7], but this algorithm is only for molecular not for electronic computer. The approximation algorithm for the triple matching problem have been presented recently in paper [8]. The triple matching is also called the three-dimensional marriage problem [2]. An instance of the three-dimensional marriage problem consists of *n* boys, *n* girls and *n* pets. A matching M consists of *k* triples (b, g, p) such that each boy, each girl and each pet belong to exactly one triple, but since this problem is very hard to solve, a variant of this problem is studied, which is called the Cyclic Three-Dimensional Matching [3, 4], but this is not the subject of this paper. Until today, there was no ant algorithm for the maximum triple matching problem and since the maximum triple matching belongs to combinatorial problems for which ant algorithms are very suited [5], I decided to elaborate an ant algorithm for this maximal triple matching problem.

2. The triple matching problem

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

1) X, Y and Z are 3 disjoint sets, each of size n.
2) $T = \{(x, y, z): x \in X, y \in Y, z \in Z\} \in X \cdot Y \cdot Z$ and we have to find M such that:
   - $M \subseteq T$
   - $|M| = k$
   - 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 are presented in Fig. 1. An edge represents a preference. A three-dimensional clique (a triangle) represents a match between boy, girl and pet. A maximum matching at Fig. 1 is presented by three triangles: (B1, G1, P3), (B2, G2, P1) and (B3, G3, P2).

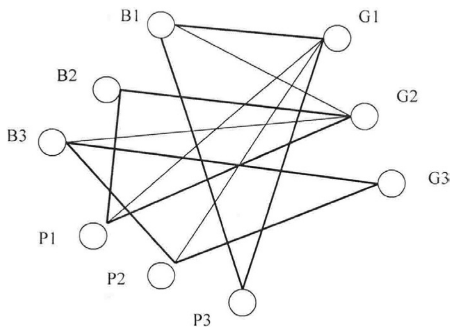

Fig. 1. The triple problem modeled as a 3-dimensional graph and its solution

This instance of the triple matching problem, which is presented in figure 1 as a three-dimensional graph, can be presented by a three-dimensional array M. Any element of this array M equal 1 means that there is a match between boy, girl and pet. If any element of this array M is equal 0, this means that there is no match. All elements of the three-dimensional array M for the instance of the maximum triple matching problem presented in figure 1 are listed below:

$M[b_1, g_1, p_1] = 0$,	$M[b_1, g_1, p_2] = 0$,	$M[b_1, g_1, p_3] = 1$,
$M[b_1, g_2, p_1] = 0$,	$M[b_1, g_2, p_2] = 0$,	$M[b_1, g_2, p_3] = 0$,
$M[b_1, g_3, p_1] = 0$,	$M[b_1, g_3, p_2] = 0$,	$M[b_1, g_3, p_3] = 0$,
$M[b_2, g_1, p_1] = 0$,	$M[b_2, g_1, p_2] = 0$,	$M[b_2, g_1, p_3] = 0$,
$M[b_2, g_2, p_1] = 1$,	$M[b_2, g_2, p_2] = 0$,	$M[b_2, g_2, p_3] = 0$,
$M[b_2, g_3, p_1] = 0$,	$M[b_2, g_3, p_2] = 0$,	$M[b_2, g_3, p_3] = 0$,
$M[b_3, g_1, p_1] = 0$,	$M[b_3, g_1, p_2] = 0$,	$M[b_3, g_1, p_3] = 0$,
$M[b_3, g_2, p_1] = 0$,	$M[b_3, g_2, p_2] = 0$,	$M[b_3, g_2, p_3] = 0$,
$M[b_3, g_3, p_1] = 0$,	$M[b_3, g_3, p_2] = 1$,	$M[b_3, g_3, p_3] = 0$.

The solution for the maximum matching problem consists of three elements of this three-dimensional array M: $M[b_1, g_1, p_3] = 1$, $M[b_2, g_2, p_1] = 1$ and $M[b_3, g_3, p_2] = 1$. Any two of these elements do not concern the same boy, same girl or same pet and thus the constraint 2 c is satisfied.

3. Structure of the ant algorithm

In ant algorithms, each ant is looking for a solution to a problem. If there are m ants, their solutions are compared and the best among them is chosen. This is repeated in each cycle of the ant algorithm, but in each cycle, also the best solution is remembered and an ant mechanism of communication is implemented.

At the beginning of an ant algorithm on all triples, which can be possibly chosen, the maximum quantity of pheromone t_{max} is assigned (line 1). Next, based on a 2-dimensional matrix of preferences between boys and girls bg[][], girls and pets gp[][] and boys and pets bp[][], a 3-dimensional array M[][][] is created. Elements of array M[][][] are equal to 1, when there is a match between boy, girl and pet, and is equal 0 in the other case. Ant algorithms consists of two main loops: one for cycle (line 4) and one for ants (line 6). Vectors allowedb[], allowedg[] and allowedp[] are used in the following way: when any ant finds a triple (b, g, p) then this boy, this girl and this pet can be included into another triple, which is part of the solution for the maximum triple matching problem (line 28), so they are excluded from the allowed boys, girls and pets, which can constitute the next triple. This exclusion is made by assigned a value 0 (line 29–31) to these vectors allowedb[], allowedg[] and allowedp[] for a particular boy, girl and pet. Thus, we can assure that any boys, any girls and any pets can be selected twice into two different triples, and thus, we assure that the received solution is correct. If elements of these vectors allowedb[], allowedg[] and allowedp[] are equal to 1, this means that these particular boys, girls and pets can be selected into another triple (line 7–9).

The *while(go)* loop is executed when there is another triple (b, g, p), which can be added into the solution (line 13). Inside this loop, each ant calculates a sum of all the pheromones deposited on triples, which can be included into solution (line 16–19), a probability of particular triple selection (line 21–22) and selects one of these triples (line 25–31) based on the rule circle method and adds this triple into the solution (line 28). We can assure that each ant finds a solution, which constitutes of *k* – triples, so the *while(go)* loop can be executed equal or less than *n* times.

Each ant checks if a better solution was found, and if so, this solution is remembered (line 32–35). In order to do this, each ant calculates a number of triples *ld*, which were included into the solution. From all solutions received in one cycle, the one, which has the greatest number of triples, is selected *ldb*. This solution *ldb* is compared to the best solution *ldg*, which was found so far by ants (line 36) and the better is remembered. These sizes of the solution are used to calculate the additional quantity of pheromone *dt* which will be deposited on triples.

Next, an ant communication system was implemented (line 37–40): an evaporation mechanism *r* was used on all existing triples. which can constitute the solution, and an additional quantity of pheromone *dt* is added on all triple,s which constitute the best solution that was found so far (line 40).

The pseudo code of the elaborated ant algorithm for the maximum triple matching problem is presented below as algorithm 1.

Algorithm 1. Ant algorithm for the maximum triple matching problem

```
1.all f[i][j][k] = tmax;
2.based on bg[][], gp[][] and bp[][] a M[][][] is created
3. ldg = 0;
4. for each cycle
5. { ldb = 0;
6. for each ant
            {
7.          for all i allowedb[] = 1;
8.          for all j allowedg[] = 1;
9.          for all k allowedo[] = 1;
10.         for all i,j,k p[][][] = 0;
11.         for all i,j,k solution[][][] = 0;
12.         go = 1;
13.         while(go)
14.     {go = 0;
15.         sumat = 0;
16.         for all i,j,k
17.           if allowedb[] = = 1 && allowedg[] = = 1 && allowedp[] = = 1 && M[][][] = = 1
                    {
18.                 go = 1;
19.                 sumat = sumat+t[][][];  }
20.         if go = = 1{
                    for all i,j,k
21.                       if allowedb[] = = 1 && allowedg[] = = 1 && allowedp[] = = 1 &&
                          M[][][] = = 1
22.                             p[][][] = t[][][]/sumat;
23.                 pr = (rand()/(double)32767);
24.                 sumap = 0;
          for all i,j,k
25.                 if allowedb[] = = 1 && allowedg[] = = 1 && allowedp[] = = 1 && M[][][] = =
1
26.                                  { sumap = sumap+p[][][];
27.                                  if sumap>pr
28.                        {         solution[][][] = 1;
29.                                            allowedb[i] = 0; i = n;
30.                                            allowedg[j] = 0; j = n;
31.                                            allowedp[k] = 0; k = n; }      }
                    }//if go = = 1
          }//while(go)
32.       ld = 0;
33.       for all i,j,k
34.                 if solution[][][] = = 1 ld = ld+1;
35.       if ld>ldb  ldb = ld;
          }//for each ant
36. if ldb>ldg lg = ldb;
37. dt = (1 / (1 - ( (ldg-ldb)/ldg)));
38. for all i,j,k
39.       if soluution[][][] = = 1
40.                 t[][][] = r*t[][][] +dt
}//for each cycle
```

4. Experiments

Since there are no other ant algorithms even there is no polynomial time exact algorithm for the maximum triple matching problem experiments, which were conducted, concern only the comparison of the elaborated ant algorithm Ant3Dmatching with the approximation Apx3Dmatching-F algorithM [8]. During the first experiment, the size of the problem was changing from *n* = 10 to *n* = 50. The average results from 10 measurements were presented in figure 2 and in table 1. We can see that when the size of the problem is rising, the Ant3Dmatching algorithm allows us to receive a bigger maximum triple matching than the Apx3Dmatching-F algorithm. During the next experiment, the number of cycles was changing from *lc* = 50 to *lc* = 250. The average results from 10 measurements were presented in figure 3 and in table 2. In addition, the Ant3Dmatching algorithm now allows us to obtain a higher maximum triple matching than the Apx3Dmatching-F algorithm.

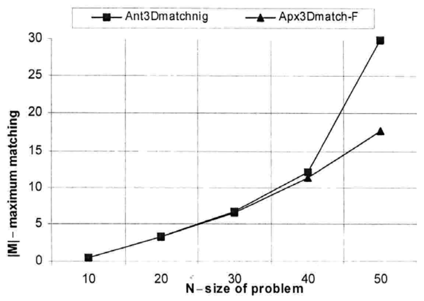

Fig. 2. An average size of maximum matching when *lc* = 100, *lm* = 30, *r* = 0.998 and *q* = 0.07

Table 1. An average size of maximum matching when *lc* = 100, *lm* = 30, *r* = 0.998 and *q* = 0.07

N	10	20	30	40	50
Ant3Dmatching	0.5	3.3	6.7	12.1	29.9
Apx3Dmatch-F	0.5	3.3	6.5	11.4	17.7

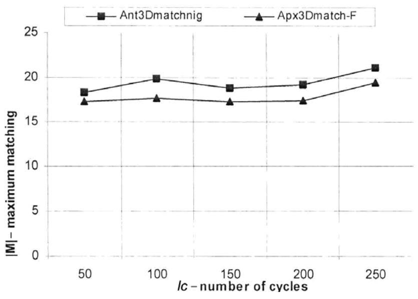

Fig. 3. An average size of maximum matching when *n* = 50, *lm* = 30, *r* = 0.998 and *q* = 0.07

Table 2. An average size of maximum matching when *n* = 50, *lm* = 30, *r* = 0.998 and *q* = 0.07

lc	50	100	150	200	250
Ant3Dmatching	18.3	19.9	18.8	19.2	21.1
Apx3Dmatch-F	17.3	17.7	17.3	17.4	19.5

The third experiment concerns the size of maximum matching when the number of ants was changing from *lm* = 10 to *lm* = 50. The average results from 10 measurements were presented in figure 4 and in table 3. There are no big differences in comparison with the case when the number of cycles was changing and the same occurred when an evaporation rate was changing from *r* = 0.990 to *r* = 0.998: the Ant3Dmatching algorithm has shown its advantage over the Apx3Dmatchnig_F algorithm. The average results from 10 measurements were presented in figure 5 and in table 4 for the case when the evaporation rate was changing.

Table 3. An average size of maximum matching when *n* = 50, *lc* = 100, *r* = 0.998 and *q* = 0.07

lm	10	20	30	40	50
Ant3Dmatching	19.8	20.8	19.9	20.3	20.2
Apx3Dmatch-F	17.6	19.4	17.7	18.4	18.4

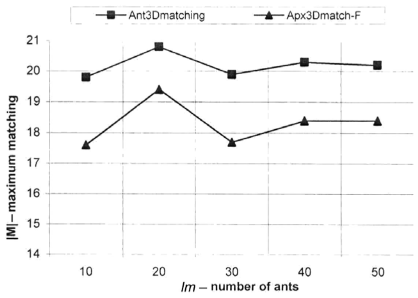

Fig. 4. An average size of maximum matching when *n* = 50, *lc* = 100, *r* = 0.998 and *q* = 0.07

Table 4. An 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	19.2	19.0	21.3	19.2	19.9
Apx3Dmatch-F	17.8	17.7	19.6	17.0	17.7

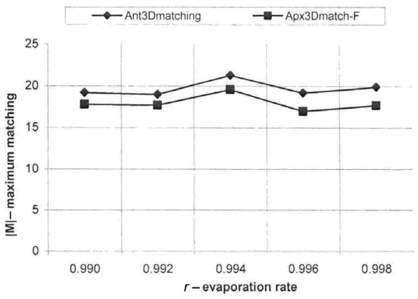

Fig. 5. An average size of maximum matching when *lc* = 100, *lm* = 30, *n* = 50 and *q* = 0.07

The last experiment concerns the size of maximum matching when the density of graph *q* was changing. The results have been shown in Table 5 and in Fig. 6. We can see that the size of maximum matching is changing when the size of the problem is constant and equal to *n* = 50, and this size of maximum matching received by the Ant3Dmatching algorithm is higher than that received by the Apx3dmatching-F algorithm for different graph density *q* = { 0.04, 0.07, 0.10, 0.13 and 0.13}. We see that the graph density *q* is very low. The graph density *q* is the probability with which the preference between boy and girl or boy and pet or girl and pet exist.

Table 5. An 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	6.6	19.9	32.6	39.1	42.8
Apx3Dmatch-F	6.3	17.7	27.9	33.2	38.6

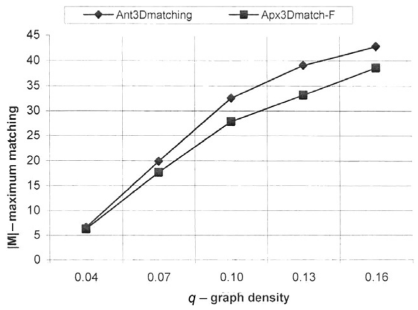

Fig. 6. An average size of maximum matching when *lc* = 100, *lm* = 30, *n* = 50 and *r* = 0.998

5. Conclusions

In this article, for the first time, an elaborated ant algorithm for the maximum triple matching problem is presented. Until now, there was no ant algorithm and no polynomial time exact algorithm for this problem. The elaborated ant algorithm, which is called the Ant3Dmatching algorithm, shows its over-performance over the approximation Apx3Dmatchnig-F algorithm. Thanks to the elaborated ant algorithm, the maximal triple matching problem can be solved very quickly and the size of the received triple matching is bigger than in the case when the Apx3Dmatching-F algorithm is used.

References

[1] Karp R.M., *Reducibility among combinatorial problems,* [in:] R. Miller, J. Thatcher, Complexity of Computer Computations, Plenum, 1972, 85–103.

[2] Knuth, D., *Stable marriage and its relation to other combinatorial problems, An introduction to the mathematical analysis of algorithms,* Amer. Math. Soc., Providence, RI, 1997.

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

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

[5] Dorigo M., Stützle T., *Ant colony optimization,* Cambridge MIT Press. Proceedings of EvoWorkshops 2002, Berlin, Heidelberg, Springer-Verlag, 2002, 61–71.

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

[7] Cao J., Chang W-L, Guo M., *Using sticker to solve the 3-dimensional matching problem in molecular supercomputers,* Int. J. High Performance Computing and Networking, Vol. 1, 2004, Nos. 1/2/3.

[8] Epifiano F.S., Ogasawara E., Soares J., Amorim M., Souza U., *O Problema de Alocacao deTutores em Aplicacoes de Provas,* XLVI Simposio Brasileiro de Pesquisa Operacional, Salvador, 16–19.09.2014 Brasil, 2014, 3007–3018.