RESEARCH IN LOGISTICS & PRODUCTION

# LOGISTIC STAFF MANAGMENT – ANT ALGORITHM FOR THE OPTIMAL TEAM CREATION

Krzysztof Schiff

Faculty of Electrical and Computer Engineering, Cracow University of Technology, Warszawska 24, 31-155, Kraków, Poland, E-mail: kschiff@pk.edu.pl

**Abstract:** During daily logistic management managers often met difficult decision optimization problem concerning company staff. Such problems often rely on matching workers between them and team leader or matching worker to a place and a time slot and often they, menagers want to optimise their resources in order to receive such matches as maximum as possible as regars a cardinality number of a such set. These optimisation problem are often NP-difficults and to solve them menagers need special software tools. To aid managers in such situation artificial methods are used. Between artificials methods is a one called the ant colony optimisation algorithm and why in this article an ant colony optimization algorithm for the maximum cardinality 3-dimensional matching problem is described. The problem is modeled by means of 2-dimensional arrays. The elaborated ant algorithm was compared with another existing ant algorithm and tested for different values of ant algorithm parameters. Results of these tests were presented and discussed. The elaborated algorithm shows its superiority.

**Paper type: Research Paper**

**Published online:** 31 January 2019
Vol. 9, No. 1, pp. 49–59
DOI: 10.21008/j.2083-4950.2019.9.1.5

ISSN 2083-4942 (Print)
ISSN 2083-4950 (Online)


***Keywords:*** *logistic staff management, three dimensional matching problem, ant colony*

## 1. INTRODUCTION

Logistic management of company staff is difficult and complex problem in daily life of each manager. Optimal logistic decisions often should be taken very quickly, but these decisions are very difficult in their nature. Often these problems which should be solved belongs to NP-difficult problems, this means to such problems, which solutions can be received by algorithms with nonpolynominal time complexity and which solution cannot be obtained in reasonable time. Artificial methods are used to solve such difficult problems, which allow us to find optimal solution quickly (Król, 2015); (Leśniak, 2015); (Kacprzak, 2015). Among NP-difficult problems there is the problem of staff assignment to jobs. Very good review of literature on subject, which concerns the assignment of workers to jobs is paper (Niknafs, 2013). A one of NP-difficult problems, which can be met in area of logistic staff management, is a problem of matching workers to them together and to team leader. Managers often want to receive such triple-matches as much as possible, so they want to optimise their resources and they want to find the maximum number of such triple-matches. This problem of finding maximum number of triple-matching is theme of this paper. Amongst methods of artificial intelligence which are used to support logistic staff management is an ant colony optimisation method and such an example of artificial intelligence methods usage in order to find solutions to staff assignment problem are papers (Devi, 2014); (Aftab, 2012), the another is usage of genetic selection in papers (Stylianou, 2013); (Bergh, 2012); (Younas, 2014); (Kotwal, 2015); (Fira, 2012). A good review on staff scheduling is paper (Bergh, 2012). Amongst staff management problems is a one, which consist on matching workers to them together and to team leader. This problem can be modelled by means of three separable sets: one for team leader, one for workers and one for workers too. In such construction, which consists of three separable sets, we are looking for the optimal solution of our problem. The maximum number of triple-matches, which is find in construction of three separable sets, is our optimal solution. This problem is already known as the 3-dimensional maximum matching problem. This problem is a generalisation of the maximum matching problem in bipartite graph, for which many polynomial-time algorithms already exists and the solution can be obtained very fast. Problem of finding the maximum number of the three-dimensional matching problem is one of NP-difficult problems and it was a subject of research in such papers as (Balas, 1989); (Crama, 1992); (Frieze, 1974); (Hansen, 1973); (Aiex, 2005); (Pasiliao, 2003). This problem is one of the 21 NP-difficult Karp problems (Karp, 1972) and also it is known as three-dimensional marriage problem (Knuth, 1997). Three-dimensional marriage problem concerns $n$ boys, $n$ girls and $n$ pets. Matching $M$ it is sum of k triple $(b, g, p)$ such that each boy, each girl and each pet belongs to exactly one triple. There is a need to find the maximal number k of such triple $(b, g, p)$. In the area of staff management there is a mutual triple-matches of $p$ workers to $p$ workers and $p$ team

leaders. There are no exact polynomial time algorithms for maximum triple-matches problem, so there are elaborated many heuristics algorithms for them (Biro, 2010); (Erikson, 2006); (Clemons, 2003); (Chen, 2012); (Liu, 2006) or meta-heuristics (Huang, 2006); (Clemons, 2004); (Gutin, 2009); (Jiang, 2008); (Gutin, 2011). Amongst such algorithms there are ants algorithms and they are very good suited for obtaining solutions of combinatorial problems (Dorigo, 2002). I have just belaboured an ant algorithm for three-dimensional maximum matching problem, which is presented in paper (Schiff, 2018) and which has been shown to be better than already kwon algorithms for this problem when we take into consideration the maximum number of triple-matches (Crama, 1992); (Epifiano, 2014). In this paper I presents a new ant algorithm, which allow to faster and for bigger dimensional problem as regard the sizes of three separable sets find the solution in comparison with this ant algorithm, which I have presented in paper (Schiff, 2018).

## 2. MAXIMUM CARDINALITY OF TRIPLE METCHING SET

Triple matching problem consists of *X, Y, Z* such, that:

1. *X, Y* and *Z* are three disjunctive sets, each of them consists of n elements

2. $T=\{(x, y, z): x \in X, y \in Y, z \in Z\} \in X \times Y \times Z$ and we look to find such a set *M*, so:

a) $M \subseteq T$

b) $|M| = k$

c) for any pair of triple from a set M, $(x, y, z)$ and $(x', y', z')$, $x \neq x'$, $y \neq y'$ and $z \neq z'$

Maximum cardinality of triple matching set consists on finding maximum $|M|$. This problem with solution has been shown in Figure 1. Each edge represents a relation of preferential between boy and girl. The triangle which is shown by means of bold line represents a matching between boy, girl and pet or if you like between a worker, a time slot and a place or between two workers and a team leader. The maximum cardinality number of matchings – three triangles (*B1, G1, P3*), (*B2, G2, P1*) and (*B3, G3, P2*) has been shown in Figure 1.

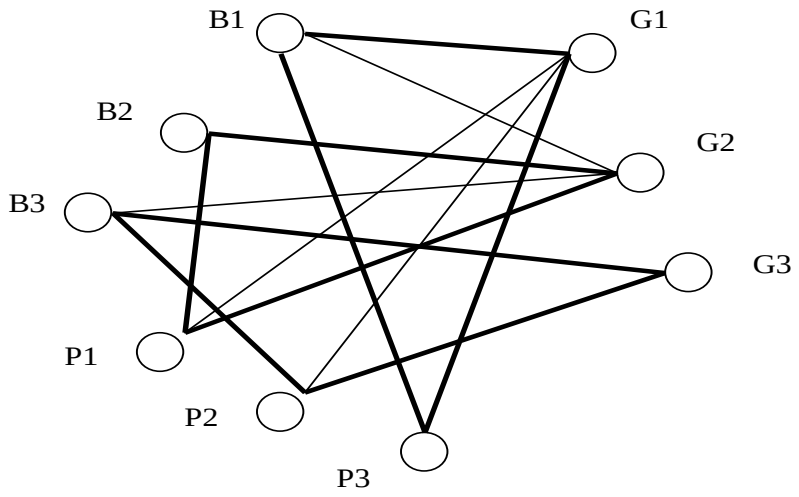


**Fig. 1.** Maximum cardinality of the triple-matching set

The problem of maximum triple-matching can be modelled by means of three-dimensional array *M*. Any element of the table *M* is equals *1*, this means that a matching exists between worker *1*, worker *2* and a team leader. When it is equals *0*, this means that such a matching does not exist. All elements of the three-dimensional *M* array are listed below for the case, which is shown in Figure 1.

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

The solution for our problem of finding the maximum number of triple-matching consist of three elements of array *M*. According with our case, which is presented in Figure 1, they are *M[b1 ,g1 ,p3 ]=1*, M[b2 ,g2 ,p1 ]=1 and *M[b3 ,g3, p2 ]=1*. Any two elements of this solution does not concerns the same boy, the same girl or the same pat and thus the constraint *2c* is fulfilled.

The solution of our problem can be shown by means of three two-dimensional arrays as it can be seen in Table 1. The solution of our problem is shown in Figure 1.

**Table 1.** Three two-dimensional arrays for the case

| Mgp[][] | P1 | P2 | P3 | Mbp[][] | P1 | P2 | P3 | Mbg[][] | 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. STRUCTUTRE OF THE ANT ALGORITHM

Each ant in the ant algorithm create 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 other best solutions from precedent cycles.

The ant algorithm which is presented in this article, different from the ant algorithm, which is presented in paper (Schiff, 2018). Difference rely on a way of modelling, this algorithm, which is presented in paper (Schiff, 2018) uses a three-dimensional array, when this algorithm, which is presented in this paper uses 3 two-dimensional arrays, which have been shown in Table 1, These arrays are used to model a problem. This structure change conduct to a new ant algorithm and to a new algorithm action.

During the algorithm description a boy, a girl and a pet are uses instead of worker 1, worker 2 and a team leader, so thus the algorithm description is more understandable.

At the beginning on all edges, which represent preferential between boys, girls and pets, maximum quantities of pheromone are given. *tmax* (line 1). The ant algorithm consisted of two loops: the first one for cycles (line 4) and the second one for ants (line 6). Arrays *allowedb[]* and *allowedg[]* are used in such a way, that: when any ant find a relation of preference between boy and girl, then this boy and this girl could not be chosen to any other relation of preference and thus the constraint *2 c* is fulfilled. This exclusion is made by the assignment of value 0 to elements of arrays *allowedb[]* and *allowedg[]*, which represent this boy and this girl (line 29–30). Thus we assure that any boy and any girl do not be chosen twice to two different matches and thus we assure that the solution will be correct. If elements of arrays *allowedb[]* and *allowedg[]* has assigned values 1, this means that these boys and these girls could be chosen by ants in order to make a new match (line 7–8).

The loop while(go) is repeated so long as long there is a match (*b, g*), which could be added to solution (line 13). Inside this loop the sum of the pheromone deposited so far on preferences is calculated, which could be chosen to solution (line 16–19), probability of such selection (line 21–22) and also this selection is done by means of roulette method (line 25–30) and afterword's the selected match is added to a solution (line 28). Thus each ant find a solution, which is constructed of k – matches. The loop while(go) can be repeated n times.

All matches between boys and girls are in the array *solution[b][g]*, so base on this array an array *Mbgp[b lub g ][p]* is created (line 31–38). When we have a match between worker 1 and worker 2 ant checks if they have the same mutual preference to pet and if so a value 1 is assigned to an element of array *Mbgp[b* or *g][p]=1*. If there is no such a preference a value 0 is assigned to element of array *Mbgp[b or g ][p]=0*. It does not matter if we choose index of a boy or a girl, but we have to be consequent in our selection.

Now this part of algorithm (line 39–61) is performing in the same way as the part (line 6–30), but there is difference such that the action is made on the array *Mbgp[][]* instead of the array *Mbg[][]* and the solution is kept in the array solution2[][].

Each ant check if a solution is better than already founded solutions and if so this solution is remembered (line 62–65). To do this each ant compute a number of matches *ld*, which a solution is constructed of. From all solutions received in one cycle only one with the biggest number of matches *ldb* is selected. This number *ldb* is compared with the number *ldg* (line 66) and the bigger number is kept, this means the better solution is kept. These numbers are used to compute a quantities of the pheromone, which should be put on all matches, which constitute a solution.

Next an ant communication system is implemented (line 67–69): the evaporation rate of pheromone r and additional quantities of pheromone *dt* is calculated and deposited on all matches form the best solution (line 68–69).

Pseudo–code of elaborated ant algorithm for three-dimensional maximum matching problem is presented as algorithm.

The elaborated ant algorithm for the maximum triple-matching problem.

```
1. all tbg[i][j]=tmax;
2. all tbgp[i][j]=tmax;
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;
10.                         for all i,j p[][]=0;
11.                         for all i,j solution[][]=0;
12.                         go=1;
13.                         while(go)
14.                         { go=0;
15.                         sumat=0;
16.                         for all i,j
17.                            if allowedb[]==1 && allowedg[]==1 && Mbg[][]==1
18.                           { go=1;
19.                           sumat=sumat+tbg[][];   }
20.                         if go==1
```

```
{ for all i,j   if allowedb[]==1 && allowedg[]==1 && Mbg[][]==1
22.                                    p[][]=tbg[][]/sumat;
23.                                  pr=(rand()/(double)32767);
24.                                  sumap=0;
                                    for all i,j
25.                                    if allowedb[]==1 && allowedg[]==1 && Mbg[][]==1
26.                                        { sumap=sumap+p[][];
27.                                        if sumap>pr
28.                                        {    solution[][]=1;
29.                                            allowedx[i]=0; i=n;
30.                                            allowedy[j]=0; j=n;  }    }
}//if go==1
}//while(go)
31. licz=0;
32. for all i,j
33.   Mbgp[i][j]=0;
34.    if solution[i][j]==1
35.       for all k
36.            if bp[i][k]==1 && wp[j][k]==1
37.                                    { Mbgp[i][k]=1;
38.                        Licz=licz+1;}
39.                                    for all i allowedb[]=1;
40.                                    for all j allowedp[]=1;
41.                                    for all i,j p[][]=0;
42.                                    for all i,j solution2[][]=0;
43.                                    go=1;
44.                                    while(go)
45.                                    { go=0;
46.                                    sumat=0;
47.                                    for all i,j
48.                                      if allowedb[]==1 && allowedp[]==1 && Mbgp[][]==1
49.                                    { go=1;
50.                                    sumat=sumat+tbgp[][];    }
51.                                    if go==1
{ for all i,j
52.                          if allowedb[]==1 && allowedp[]==1 && Mbgp[][]==1
53.                                        p[][]=tbgp[][]/sumat;
54.                                    pr=(rand()/(double)32767);
55.                                    sumap=0;
                                    for all i,j
56.                                        if allowedb[]==1 && allowedp[]==1 && Mbgp[][]==1
57.                                            { sumap=sumap+p[][];
58.                                            if sumap>pr
59.                                            {    solution2[][]=1;
60.                                                allowedb[i]=0; i=n;
61.                                                allowedp[j]=0; j=n;   }   }
}//if go==1
```

```
}//while(go)
62.                         ld=0;
63.                         for all i,j
64.                           if solution2[][]==1 ld=ld+1;
65.                         if ld>ldb   ldb=ld;
}//for each ant
66. if ldb>ldg    lg=ldb;
67. dt = (1 / (1 - ( (ldg-ldb)/ldg)));
68. for all i,j     if solution2[][]==1 tbgp[][]=r*tbgp[][] +dt
69. for all i,j     if solution[][]==1     tbg[][]=r*tbg[][] +dt
}//for each cycle
```

## 4. EXPERIMENTAL RESULTS

There are two algorithms for the maximum triple-matching problem and one of them is described in paper (Schiff, 2018) and the another algorithm, which is presented in this paper. The first is called *N_ANT3DMATCHING* and the second is called *ANT3DMATCHING*. They were compared with each other. During experiments, which were conducted, the maximum number of triple-matches was observed.

During the first experiment the dimension of problem was changing from *n=10* to *n=50*. Averages maximum numbers of triple-matches from 10 measurements are presented in Table 2. It can be see that elaborated algorithm *N_ANT3DMATCHING* allows to receive bigger numbers of triple-matches in comparison with the algorithm *ANT3DMATCHING*, which is presented in paper (Schiff, 2018).

Next the number of cycles was changing from *lc=50* to *lc=250*. Averages maximum numbers of triple-matches from 10 measurements are presented in Table 3. The algorithm, which is called *N_ANT3DMATCHING*, shows his superiority over algorithm, which is called *ANT3DMATCHING* in regards the number of triple-matches, which has been received.

**Table 2.** Average numbers of triple-matching when lc=100, lm=30, r=0.998, q=0.07

| N | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|
| N_ANT3DMATCHING | 0,4 | 2,1 | 7,1 | 13,2 | 23,2 |
| ANT3DMATCHING | 0,3 | 1,9 | 6,6 | 12,3 | 20,4 |

**Table 3.** Average numbers of triple-matching when n=50, lm=30, r=0.998 i q=0.07

| Lc | 50 | 100 | 150 | 200 | 250 |
|---|---|---|---|---|---|
| N_ANT3DMATCHING | 21,8 | 23,2 | 21,8 | 21,4 | 21,2 |
| ANT3DMATCHING | 19,9 | 20,4 | 20,0 | 19,6 | 19,2 |

During the next experiment, which was conducted, the number of ants was chancing from *lm=10* to *lm=50*. Averages maximum numbers of triple-matches from 10 measurements are presented in Table 4. There is no a big difference in comparison with the precedent experiment, in which the number of cycles was changing and it is likewise when the evaporation rate is changing from *r=0.990* to *r=0.998.* Averages maximum numbers of triple-matches from 10 measurements in case of changing evaporation rate are presented in Table 5.

**Table 4.** Average numbers of triple-matching when n=50, lc=100, r=0.998 i q=0.07

| Lm | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|
| N_ANT3DMATCHING | 21,4 | 22,4 | 23,2 | 21,5 | 22,7 |
| ANT3DMATCHING | 19,4 | 19,8 | 20,4 | 18,8 | 20,4 |

**Table 5.** Average numbers of triple-matching when lc=100, lm=30, n=50 i q=0.07

| R | 0.990 | 0.992 | 0.994 | 0.996 | 0.998 |
|---|---|---|---|---|---|
| N_ANT3DMATCHING | 20,4 | 22,8 | 22,6 | 21,8 | 23,2 |
| ANT3DMATCHING | 19,1 | 21,1 | 20,3 | 20,0 | 20,4 |

**Table 6.** Average numbers of triple-matching when lc=100, lm=30, n=50 i r=0.998

| Q | 0.04 | 0.07 | 0.10 | 0.13 | 0.16 |
|---|---|---|---|---|---|
| N_ANT3DMATCHING | 6,1 | 23,2 | 38,3 | 44,8 | 47,8 |
| ANT3DMATCHING | 6,1 | 20,4 | 33,8 | 39,6 | 42,3 |

During the last experiment the density of preference *q* was changing Results have been shown in Table 6. We can see that the number of triple-matches is changing when the size of problem is constant and equals *n=50*. We see that density of preference is low. The density of preference is the probability with which preference between worker 1 and worker 2 exists or between worker 1 and a team leader or between worker 2 and a team leader. All time new elaborated algorithm, which is called *N_ANT3DMATCHING* allow to receive a better solution than the algorithm *ANT3DMATCHING* with regards the maximum number of triple-matches.

## 5. CONCLUSION

In this article a new ant algorithm, called *N_ANT3DMATCHING*, for the maximum triple-matching problem is presented. All experiments shown that the algorithm, called *N_ANT3DMATCHING*, shown its superiority over the algorithm called *ANT3DMATCHING*, since it let us to receive a bigger number of triple-

matches and in faster way. Time complexity of algorithm, called *ANT3DMATCHING* is smaller than the algorithm, called *ANT3DMATCHING*. Time complexity of the new elaborated algorithm is (*lc*lm*n4*), when the early algorithm has time complexity (*lc*lm*n5*), so a new algorithm can perform n times faster than the old one and this is also an important achievements.

## REFERENCES

Aftab M.T., Umer M. & Ahmad R. (2012), Jobs scheduling and worker assignment problem to minimize makespan using ant colony optimization metaheuristic, International Journal of Infustrial and Manufacturing Engineering, vol. 6, no. 12, pp. 2823–2826.

Aiex R, Resende M, Pardalos P.M. & Toraldo G. (2005), GRASP with path relinking for three-index assignment problem INFORMS Journal on Computing, 17, pp. 224–247.

Balas E. & Saltzman M.J. (1989), Facets of the three–index assignment polytope, Discrete Appl. Math. 23, pp. 201–229.

Bergh J., Belien J., Bruecker P., Demeulemeester E. & Boeck L. (2012), Personnel scheduling: a literature review, dostępna: *https://core.ac.uk/download/pdf/34570361.pdf*

Biro P. & McDermid E. (2010), Three-sided stable matchings with cyclic preferences, Algorithmica, vol. 58, no. 1, pp. 5–18.

Chen J., Liu Y., Lu S., Sze S.H. & Zhang F. (2012), Iterative expansion and colour coding: An improved algorithm for 3D–matching, ACM Trans. Algorith., vol. 8, no. 1, pp. 1–6.

Clemons W, Grundel D. & Jeffcoat D. (2004), Applying simulated annealing on the multidimensional assignment, Proc. 2nd Operat. Control and Optimis. Conf., pp. 45–61.

Clemons W.K., Grundel D.A. & Jeffcost D.E. (2004), Applying simulated annealing to the multidimensional assignment problem, Word Scientific, pp. 45–61.

Crama Y. & Spieksma F.C.R. (1992), Approximation ALG for three-dimensional assignment problem with triangle inequalities, Europ. J.Oper. Res., vol. 60, pp. 273–279.

Devi R., Monica D., Devi S. & Subashini D. (2014), Scheduling and resource allocation for employee in software industry, Int.J. Adv. Comp. Eng. & Net., vol. 2, no.5, pp. 26–29.

Dorigo M. & Stützle T. (2002), Ant colony optimisation, Cambridge MIT Press. Proceedings of EvoWorkshops 2002, Heidelberg: Springer-Verlag, Berlin, pp. 61–71.

Epifiano F.S., Ogasawara E., Soares J., Amorim M. & Souza U. (2014), O Problema de Alocacao deTutores em Aplicacoes de Provas, XLVI Simposio Brasileiro de Pesquisa Operacional, Salvador, 16–19 sepembre 2014 Brasil, pp. 3007–3018.

Eriksson K., Sjostrand J. & Strimling P. (2006), Three-dimensional stable matching with cyclic preferences, Math. Soc. Sci., vol. 52, no. 1, pp. 77–87.

Fira L. & Brezulianu A., (2012), A genetic algorithm for scheduling of resources in well-services company, Int.J. of Advanc. Research in Artif. Intell., vol. 5, no. 1, pp. 1–6.

Frieze A.M. (1974), A bilinear programming formulation of the 3-dimensional assignment problem, Math. Programming, vol. 7, pp. 376–379.

Gutin G. & Karapetyan D. (2009), A memetic Alg for the multidimensional assignment problem in Engineering Stochastic Local Search Alg,, SLS 2009, Lecture Notes in Computer Ścience, Springer, Berlin, Heidelberg, vol 5752.

Gutin G. & Karapetyan D. (2011), A new approach to population sizing for memetic algorithm: a case for the multidimensional assignment prob., Evol. Comput. 19 345–371.

Hansen P. & Kaufman L. (1973), A primal-dual alg. for the three-dimensional assignment problem, Cahiers Centre Etudes Rech. Opr., vol. 15, pp. 327–336.
Huang G. & Lim A. (2006), A hybrid genetic algorithm for the three-index assignment problem, European J. of Operational Research, vol, 172, no. 10, pp. 249–257.
Jiang H, Xuan J. & Zhang X (2008), An approximate muscle guided global optimisation algorithm for the three-index assignment problem, Evol. Comput. pp. 2404–2410.
Kacprzak L., Rudy J. & Żelazny D. (2015), Multi-criteria 3-dimensional bin packing problem, Research in Logistics and Production, no.1, pp. 85–94.
Karp R.M. (1972), Reducibility among combinatorial problems, R.Miller (Ed.), Complexity of Computer Computations, Plenum, pp. 85–103.
Knuth D. (1972), Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of Algorithms, Amer. Math. Soc., Providence.
Kotwal J.G. & Dhope T.S. (2015), Solving Task Allocation to the Worker Using Genetic Algorithm, Inter. J. of Comput. Sc. and Inform. Technol., vol 6. no.4, pp. 3736–3741.
Król A. (2015), Algorithm for the transportation network nodes aggregation using fuzzy logic, Logistics and Transport, vol. 26, no. 2, pp. 67–74.
Leśniak P. & Boharski P. (2015), Application of genetic algorithms in design of public transport network, Logistics and Transport, vol. 26, no. 2, pp. 75–82.
Liu Y., Lu S. & Chen J., Sze S.H. (2006), Greedy localisation and colour-coding: improved matching and packing, Alg In International Workshop on Parameterised and Exact Computation IWPEC, Springer, vol. 4169 of LNCS, pages 84–95.
Niknafs A., Denzinger J. & Ruhe G. (2013), A systematic literature review of the personnel assignment problem, Proceedings of International Multi-Conference of Engineers and Computer Ścientist 2013, vol II, IMECS 2013, 2013, Hong Kong.
Pasiliao E.L. (2003), Alg. for Multidimensional Assignment Problems, Ph.D. thesis, Department of Industrial and Systems Engineering, University of Florida.
Schiff K. (2018), Ant ant colony optimisation for the triple matching problem, Technical Transactions, vol. 115, iss. 2 , pp. 179–186.
Stylianou C. & Andreou A.S. (2013), A multi-objective genetic algorithm for Inteligent Software Project Scheduling and Team Staffing, Intell. D. Tech., vol 1, no. 7, pp. 59–80.
Stylianou C., Gerasimous S. & Andreou A.S. (2012), A prototype tool for intelligent software project scheduling and team staffing, Int. Conf. on tools with Art. Intel., 2012, 277–284.
Younas I. (2015), Using Genetic Algorithms for Large Scale Optimization of Assignment, Planning and Rescheduling Problems, Doct. Th. in Comp. Systems, Stockholm.

## BIOGRAPHICAL NOTES

Krzysztof Schiff received his BSc in Electrical Engeeniring from Cracow University of Science (AGH) in Cracow in 1992 and his BSc in Management and Marketing in 1995 from Cracow University of Science (AGH) in Cracow in 1995, the PhD in Automatic Control and Robotics from Cracow University of Science (AGH) in Cracow in 1998. He is currently an scientist at Cracow University of Technology in Cracow. His professional interest include computers algorithms for combinatorial optimisation problems, many of them are in logistics and transport.