Proceedings of the Federated Conference on Computer Science and Information Systems pp. 415–419 DOI: 10.15439/2017F63 ISSN 2300-5963 ACSIS, Vol. 11

Ant Colony Optimization Algorithm for Workforce Planning

Stefka Fidanova
IICT, BAS
Sofia, Bulgaria
E-mail: stefka@parallel.bas.bg

Gabriel Luque
DLCS University of Mlaga
29071 Mlaga, Spain
E-mail: gabriel@lcc.uma.es

Olympia Roeva
IBPhBME, BAS
Sofia, Bulgaria
E-mail: olympia@biomed.bas.bg

Marcin Paprzycki
SRI, PAS
Warsaw, Poland
E-mail: marcin.paprzycki@ibspan.waw.pl

Pawel Gepner
Intel Corporation
Swindon, UK
E-mail: pawel.gepner@intel.com

***Abstract*—The workforce planning helps organizations to optimize the production process with aim to minimize the assigning costs. A workforce planning problem is very complex and needs special algorithms to be solved. The problem is to select set of employers from a set of available workers and to assign this staff to the jobs to be performed. Each job requires a time to be completed. For efficiency, a worker must performs a minimum number of hours of any assigned job. There is a maximum number of jobs that can be assigned and a maximum number of workers that can be assigned. There is a set of jobs that shows the jobs on which the worker is qualified. The objective is to minimize the costs associated to the human resources needed to fulfill the work requirements. On this work we propose a variant of Ant Colony Optimization (ACO) algorithm to solve workforce optimization problem. The algorithm is tested on a set of 20 test problems. Achieved solutions are compared with other methods, as scatter search and genetic algorithm. Obtained results show that ACO algorithm performs better than other two algorithms.**

***Index Terms*—Workforce Planning, Ant Colony Optimization, Metaheuristics**

I. Introduction

THE workforce planning is an important industrial decision making problem. It is a hard optimization problem, which includes multiple level of complexity. This problem contains two decision sets: selection and assignment. The first set is selected employees from the larger set of available workers. The second set is assignment the employees to the jobs to be performed. The aim is minimal assignment cost while the work requirements are fulfil. The workforce planing is an essential question of the human resource management.

The problem is very complex with strong constraints and it is impossible to apply exact methods for instances with realistic size. A deterministic workforce planing problem is studied in [9], [14]. In the work [9] workforce planning models that contain non-linear models of human learning are reformulated as mixed integer programs. The authors show that the mixed integer program is much easier to solve than the non-linear program. In [14] a model of workforce planning is considered. The model includes workers differences, as well as the possibility of workers training and upgrading. A variant of the problem with random demands is proposed in [3], [15]. In [3] a two-stage stochastic program for scheduling and allocating cross-trained workers is proposed considering a multi-department service environment with random demands. In to some problems uncertainty has been employed [10], [12], [13], [17], [18]. In this case the corresponding objective function and given constraints is converted into crisp equivalents and then the model is solved by traditional methods [13] or the considered uncertain model is transformed into an equivalent deterministic form as it is shown in [17]. Most of them simplifies the problem by omitting some of the constraints. Some conventional methods can be applied on workforce planning problem as mixed linear programming [5], decomposition method [15]. However, for the more complex non-linear workforce planning problems, the convex methods are not applicable. On this case is applied some heuristic method including genetic algorithm [1], [11], memetic algorithm [16], scatter search [1]. In this work we propose an Ant Colony Optimization (ACO) algorithm for workforce planning problem. So far the ACO algorithm is proved to be very effecting solving various complex optimization problems [6], [8].

We consider the variant of the workforce planning problem proposed in [1]. Our algorithm performance is compared with genetic algorithm and scatter search.

The rest of the paper is organized as follows. In Section 2 the mathematical description of the problem is presented. In Section 3 the ACO algorithm for workforce planing problem is proposed. Section 4 show computational results and comparison with other methods. In Section 5 some conclusions and directions for future works are done.

II. The Workforce Planning Problem

On this paper we use the description of workforce planing problem given by Glover et al. [7]. There is a set of jobs $J = \{1, \dots, m\}$, which must be completed during a fixed period (week for example). Each job j requires d_j hours to be completed. The set of available workers is $I = \{1, \ldots, n\}$. For efficiency reason every worker must perform every of assigned to him job minimum h_{min} hours. The worker i is available s_i hours. The maximal number of assigned jobs to a same worker is j_{max}. The workers have different skills and the set A_i shows the jobs that the worker i is qualified to perform. The maximal number of workers which can be assigned during the planed period is t or at most t workers may be selected from the set I of workers and the selected workers can be capable to complete all the jobs. The aim is to find feasible solution that optimizes the objective function.

Every worker i and job j are related with cost c_{ij} of assigning the worker to the job. The mathematical model of the workforce planing problem is as follows:

$$x_{ij} = \begin{cases} 1 & \text{if the worker } i \text{ is assigned to job } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if worker } i \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ij} = \text{number of hours that worker } i$$
$$\text{is assigned to perform job } j$$

$$Q_j = \text{set of workers qualified to perform job } j$$

$$\text{Minimize} \sum_{i \in I} \sum_{j \in A_i} c_{ij}.x_{ij} \qquad (1)$$

Subject to

$$\sum_{j \in A_i} z_{ij} \le s_i.y_i \qquad i \in I \qquad (2)$$

$$\sum_{i \in Q_j} z_{ij} \ge d_j \qquad j \in J \qquad (3)$$

$$\sum_{j \in A_i} x_{ij} \le j_{max}.y_j \qquad i \in I \qquad (4)$$

$$h_{min}.x_{ij} \le z_{ij} \le s_i.x_{ij} \qquad i \in I, j \in A_i \qquad (5)$$

$$\sum_{i \in I} y_i \le t \qquad (6)$$

$$\begin{array}{ll} x_{ij} \in \{0,1\} & i \in I, j \in A_i \\ y_i \in \{0,1\} & i \in I \\ z_{ij} \ge 0 & i \in I, j \in A_i \end{array}$$

The objective function of this problem minimizes the total assignment cost. The number of hours for each selected worker is limited (inequality 2). The work must be done in full (inequality 3). The number of the jobs, that every worker can perform is limited (inequality 4). There is minimal number of hours that every job must be performed by every assigned worker to can work efficiently (inequality 5). The number of assigned workers is limited (inequality 6).

Different objective functions can be optimized with the same model. In this paper our aim is to minimize the total assignment cost. If $\tilde{c}_{ij}$ is the cost the worker i to performs the job j for one hour, than the objective function can minimize the cost of the hall jobs to be finished (on hour basis).

$$f(x) = \text{Min} \sum_{i \in I} \sum_{j \in A_i} \tilde{c}_{ij}.x_{ij} \qquad (7)$$

Some worker can have preference to perform part of the jobs he is qualified and the objective function can be to maximize the satisfaction of the workers preferences or to maximize the minimum preference value for the set of selected workers.

As we mentioned above in this paper the assignment cost is minimized (equation 1). This problem is similar to the Capacitated Facility Location Problem (CFLP). The workforce planning problem is difficult to be solved because of very restrictive constraints especially the relation between the parameters h_{min} and d_j. When the problem is structured (d_j is a multiple of h_{min}), it is more easier to find feasible solution, than for unstructured problems (d_j and h_{min} are not related).

III. Ant Colony Optimization

The ACO is a metaheuristic methodology which follows the real ant colonies behavior when they look for a food and return back to the nest. Real ants use chemical substance, called pheromone, to mark their path ant to can return back. An isolated ant moves randomly, but when an ant detects a previously laid pheromone it can decide to follow the trail and to reinforce it with additional quantity of pheromone. The repetition of the above mechanism represents the auto-catalytic behavior of a real ant colony, where the more ants follow a given trail, the more attractive that trail becomes. Thus the ants collectively can find a shorter path between the nest and source of the food. The main idea of the ACO algorithms comes from this natural behavior.

A. Main ACO algorithm

Metaheuristic methods are applied on difficult in computational point of view problems, when it is not practical to use traditional numerical methods. A lot of problems coming from real life, especially from the industry. These problems need exponential number of calculations and the only option, when the problem is large, is to be applied some metaheuristic methods in order to obtain a good solution for a reasonable time [4].

ACO algorithm is proposed by Marco Dorigo [2]. Later some modification are proposed mainly in pheromone updating rules [4]. The artificial ants in ACO algorithms simulates the ants behavior. The problem is represented by graph. The solutions are represented by paths in a graph and we look for shorter path corresponding to given constraints. The requirements of ACO algorithm are as follows:

- Suitable representation of the problem by a graph;
- Suitable pheromone placement on the nodes or on the arcs of the graph;

- Appropriate problem-dependent heuristic function, which manage the ants to improve solutions;
- Pheromone updating rules;
- Transition probability rule, which specifies how to include new nodes in the partial solution.

The structure of the ACO algorithm is shown on Figure 1.

```
Ant Colony Optimization
Initialize number of ants;
Initialize the ACO parameters;
while not end condition do
    for $k = 0$ to number of ants
        ant $k$ choses start node;
        while solution is not constructed do
            ant $k$ selects higher probability node;
        end while
    end for
    Update pheromone trails;
end while
```

Fig. 1: Pseudo-code of ACO algorithm

The transition probability $p_{i,j}$, to choose the node j, when the current node is i, is a product of the heuristic information $\eta_{i,j}$ and the pheromone trail level $\tau_{i,j}$ related with this move, where $i, j = 1, \dots, n$.

$$p_{i,j} = \frac{\tau_{i,j}^a \eta_{i,j}^b}{\sum\limits_{k \in Unused} \tau_{i,k}^a \eta_{i,k}^b}, \tag{8}$$

where $Unused$ is the set of unused nodes of the graph.

A node becomes more profitable if the value of the heuristic information and/or the related pheromone is higher. At the beginning, the initial pheromone level is the same for all elements of the graph and is set to a small positive constant value τ_0, $0 < \tau_0 < 1$. At the end of every iteration the ants update the pheromone values. Different ACO algorithms adopt different criteria to update the pheromone level [4].

The main pheromone trail update rule is:

$$\tau_{i,j} \leftarrow \rho \tau_{i,j} + \Delta \tau_{i,j}, \tag{9}$$

where ρ decreases the value of the pheromone, like the evaporation in a nature. $\Delta \tau_{i,j}$ is a new added pheromone, which is proportional to the quality of the solution. The quality of the solution is measured by the value of the objective function of the solution constructed by the ant.

An ant start to construct their solution from a random node of the graph of the problem. The random start is a diversification of the search. Because the random start a relatively few number of ants can be used, comparing with other population based metaheuristics. The heuristic information represents the prior knowledge of the problem, which we use to better manage the ants. The pheromone is a global experience of the ants to find optimal solution. The pheromone is a tool for concentration of the search around best so far solutions.

B. ACO algorithm for Workforce Planning

One of the essential point of the ant algorithm is the proper representation of the problem by graph. In our case the graph of the problem is 3 dimensional and the node (i, j, z) corresponds worker i to be assigned to the job j for time z. At the beginning of every iteration every ant starts to construct their solution, from random node of the graph of the problem. For every ant are generated three random numbers. The first random number is in the interval $[0, \dots, n]$ and corresponds to the worker we assign. The second random number is in the interval $[0, \dots, m]$ and corresponds to the job which this worker will perform. The third random number is in the interval $[h_{min}, \dots, \min\{d_j, s_i\}]$ and corresponds to the number of hours worker i is assigned to performs the job j. After, the ant applies the transition probability rule to include next nodes in the partial solution, till the solution is completed.

We propose the following heuristic information:

$$\eta_{ijl} = \begin{cases} l/c_{ij} & l = z_{ij} \\ 0 & otherwise \end{cases} \tag{10}$$

This heuristic information stimulates to assign the most cheapest worker as longer as possible. The ant chooses the node with the highest probability. When an ant has several possibilities for next node (several candidates have the same probability to be chosen), then the next node is chosen randomly between them.

When a new node is included we take in to account how many workers are assigned till now, how many time slots every worker is assigned till now and how many time slots are assigned per job till now. When some move of the ant do not meets the problem constraints, then the probability of this move is set to be 0. If it is impossible to include new nodes from the graph of the problem (for all nodes the value of the transition probability is 0), the construction of the solution stops. When the constructed solution is feasible the value of the objective function is the sum of the assignment cost of the assigned workers. If the constructed solution is not feasible, the value of the objective function is set to be equal to -1.

Only the ants, which constructed feasible solution are allowed to add new pheromone to the elements of their solutions. The new added pheromone is equal to the reciprocal value of the objective function.

$$\Delta \tau_{i,j} = \frac{\rho - 1}{f(x)} \tag{11}$$

Thus the nodes of the graph of the problem, which belong to better solutions (with less value of the objective function) receive more pheromone than others and become more desirable in the next iteration.

At the end of every iteration we compare the iteration best solution with the best so far solution. If the best solution from the current iteration is better than the best so far solution (global best solution), we update the global best solution with the current iteration best solution.

The end condition used in our algorithm is the number of iterations.

TABLE I: Test instances characteristics

Parameters	Value
n	20
m	20
t	10
s_i	[50,70]
j_{max}	[3,5]
h_{min}	[10,15]

TABLE II: ACO parameter settings

Parameters	Value
Number of iterations	100
ρ	0.5
τ_0	0.5
Number of ants	20
a	1
b	1

IV. Computational Results

In this section we report test results and compare them with results achieved by other methods. We analyse the algorithm performance and the quality of the achieved solutions. The software, which realizes the algorithm is written in C and is run on Pentium desktop computer at 2.8 GHz with 4 GB of memory.

We use the artificially generated problem instances considered in [1]. The test instances characteristics are shown in Table I.

The set of test problems consists of ten structured and ten unstructured problems. The structured problems are enumerated from $S01$ to $S10$ and unstructured problems are enumerated from $U01$ to $U10$. The problem is structured when d_j is proportional to h_{min}.

As a stopping criteria for our ACO algorithm we use the number of iterations. The number of iterations is fixed to be 100. The parameter settings of our ACO algorithm is shown in Table II. This values are fixed experimentally.

The algorithm is stochastic and from a statistical point of view it needs to be run minimum 30 times to guarantee the robustness of the average results. We perform 30 independent runs of the algorithm. After we did statistical analysis of the results applying ANOVA test to guarantee the significance of the difference between the results achieved by different methods.

Lets compare the computational results achieved by our ACO algorithm and those achieved by genetic algorithm (GA) and scatter search (SS) presented in [1]. Table III shows the achieved results for structured instances while Table IV shows the achieved results for unstructured instances. We observe that ACO algorithm outperforms the other two algorithms. The ACO is a constructive method and when the graph of the problem and heuristic information are appropriate and they represent the problem in a good way, they can help a lot of for better algorithm performance and achieving good solutions. Our graph of the problem has a star shape. Each worker and job are linked with several nodes, corresponding to the time, for which the worker is assigned to perform this job. The proposed heuristic information stimulates the cheapest workers to be assigned for longer time. It is a greedy strategy. After the first iteration the pheromone level reflects the experience of the ants during the searching process thus affects the strategy. The elements of good solutions accumulate more pheromone, during the algorithm performance, than others and become more desirable in the next iterations.

Now we will compare the execution time of the proposed ACO algorithm with the execution time of the other two algorithms. The algorithms are run on similar computers. In Tables V and VI is reported average execution time over 30 runs of every of the algorithms. It is seen that the ACO algorithm finds the solution faster than GA and SS. Considering the execution time the GA and SS algorithms have similar performance. By the Tables III, IV, V and VI we can conclude that ACO algorithm gives very encouraging results. It achieves better solutions in shorter time than the other two algorithms, SS and GA. If we compare memory use, the ACO algorithm uses less memory than GA (GA population size is 400 individuals [1]) and similar memory to SS (initial population size is 15 and reference set is 8 individuals [1]).

TABLE III: Average results for structured problems

Test	Objective function value		
problem	SS	GA	ACO
S01	936	963	807
S02	952	994	818
S03	1095	1152	882
S04	1043	1201	849
S05	1099	1098	940
S06	1076	1193	869
S07	987	1086	812
S08	1293	1287	872
S09	1086	1107	793
S10	945	1086	825

TABLE IV: Average results for unstructured problems

Test	Objective function value		
problem	SS	GA	ACO
U01	1586	1631	814
U02	1276	1264	845
U03	1502	1539	906
U04	1653	1603	869
U05	1287	1356	851
U06	1193	1205	873
U07	1328	1301	828
U08	1141	1106	801
U09	1055	1173	768
U10	1178	1214	818

TABLE V: Average time for structured problems

Test	Execution time, s		
problem	SS	GA	ACO
S01	72	61	26
S02	49	32	21
S03	114	111	22
S04	86	87	25
S05	43	40	21
S06	121	110	23
S07	52	49	23
S08	46	42	24
S09	70	67	20
S10	105	102	22

TABLE VI: Average time for unstructured problems

Test	Execution time, s		
problem	SS	GA	ACO
U01	102	95	22
U02	94	87	20
U03	58	51	20
U04	83	79	20
U05	62	57	23
U06	111	75	22
U07	80	79	21
U08	123	89	20
U09	75	72	26
U10	99	95	20

V. Conclusion

In this article we propose ACO algorithm for solving workforce planning problem. We compare the performance of our algorithm with other two metahuristic methods, genetic algorithm and scatter search. The comparison is done by various criteria. We observed that ACO algorithm achieves better solutions than the other two algorithms. Regarding the execution time the ACO algorithm is faster. The ACO population consists 20 individuals and the used by the algorithm memory is similar to one used by the SS and less than the memory used by the GA. We achieved very encouraging results. As a future work we will combine our ACO algorithm with appropriate local search procedure for eventual further improvement of the algorithm performance and solutions quality.

Acknowledgment

Work presented here is partially supported by the National Scientific Fund of Bulgaria under grants DFNI-DN02/10 "New Instruments for Knowledge Discovery from Data, and their Modelling" and DFNI I02/20 "Efficient Parallel Algorithms for Large-Scale Computational Problems", and by the Polish-Bulgarian collaborative grant "Parallel and Distributed Computing Practices".

References

[1] Alba E., Luque G., Luna F., *Parallel Metaheuristics for Workforce Planning*, J. Mathematical Modelling and Algorithms, Vol. 6(3), Springer, 2007, 509-528.

[2] Bonabeau E., Dorigo M. and Theraulaz G., *Swarm Intelligence: From Natural to Artificial Systems*, New York,Oxford University Press, 1999.

[3] Campbell G., *A two-stage stochastic program for scheduling and allocating cross-trained workers*, J. Operational Research Society 62(6), 2011, 10381047.

[4] Dorigo M, Stutzle T., *Ant Colony Optimization*, MIT Press, 2004.

[5] Easton F., *Service completion estimates for cross-trained workforce schedules under uncertain attendance and demand*, Production and Operational Management 23(4), 2014, 660675.

[6] Fidanova S., Roeva O., Paprzycki M., Gepner P., *InterCriteria Analysis of ACO Start Startegies*, Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, 2016, 547-550.

[7] Glover F., Kochenberger G., Laguna M., Wubbena, T. *Selection and assignment of a skilled workforce to meet job requirements in a fixed planning period.* In:MAEB04, 2004, 636641.

[8] Grzybowska K., Kovcs, G., *Sustainable Supply Chain - Supporting Tools*, Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, Vol. 2, 2014, 13211329.

[9] Hewitt M., Chacosky A., Grasman S., Thomas B., *Integer programming techniques for solving non-linear workforce planning models with learning*, European J of Operational Research 242(3),2015, 942950.

[10] Hu K., Zhang X., Gen M., Jo J., *A new model for single machine scheduling with uncertain processing time*, J Intelligent Manufacturing, Vol 28(3), Springer, 2015, 717-725.

[11] Li G., Jiang H., He T., *A genetic algorithm-based decomposition approach to solve an integrated equipment-workforce-service planning problem*, Omega, Vol. 50, Elsevier, 2015, 117.

[12] Li R., Liu G., *An uncertain goal programming model for machine scheduling problem.* J. Inteligent Manufacturing, Vol. 28(3), Springer, 2014, 689-694.

[13] Ning Y., Liu J., Yan L., *Uncertain aggregate production planning*, Soft Computing, Vol. 17(4), Springer, 2013, 617624.

[14] Othman M., Bhuiyan N., Gouw G., *Integrating workers' differences into workforce planning*, Computers and Industrial Engineering, Vol. 63(4), 2012, 10961106.

[15] Parisio A, Jones CN., *A two-stage stochastic programming approach to employee scheduling in retail outlets with uncertain demand*, Omega, Vol. 53, Elsevier, 2015, 97-103.

[16] Soukour A., Devendeville L., Lucet C., Moukrim A., *A Memetic algorithm for staff scheduling problem in airport security service*, Expert Systems with Applications, Vol. 40(18), 2013, 75047512.

[17] Yang G., Tang W., Zhao R., *An uncertain workforce planning problem with job satisfaction*, Int. J. Machine Learning and Cybernetics, Springer, 2016. doi:10.1007/s13042-016-0539-6 http://rd.springer.com/article/10.1007/s13042-016-0539-6

[18] Zhou C., Tang W., Zhao R., *An uncertain search model for recruitment problem with enterprise performance*, J Intelligent Manufacturing, Vol. 28(3), Springer, 2014, 295-704. doi:10.1007/s10845-014-0997-1