Cooperative ant colony optimization for a hierarchical scheduling problem

D Dresvyanskiy\(^1\), O Semenkina\(^2\) and E A Popov\(^3\)

\(^1\) System Analysis and Control Department, Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia
\(^2\) Research Department, Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia
\(^3\) System Analysis and Control Department, Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia

E-mail: ecodenis@yandex.ru, semenkinaolga@gmail.com, epopov@bmail.ru

Abstract. An important part of modern integrated decision support systems in any area is scheduling. At the same time, domain knowledge can be extremely complex and includes numerous restrictions and process rules, which makes the corresponding optimization problem highly complicated. This paper is based on a hierarchical problem structure where the top-level problem is the travelling salesman problem, and the nested resource-constrained project scheduling problem is replaced by the simulation model. Cooperative ant colony optimization as well as other biology-inspired algorithms were evaluated on a set of generated problems, and the obtained experimental results are presented here.

1. Introduction

In the modern world, competition and requirements for the high quality, high speed and low cost of production have become an integral part in the effective management of any successful enterprise. It is also impossible to deal with complex production standards without regular innovations and increased production efficiency. After taking some steps to optimize the production process, such as the purchase or rental of high-quality equipment, the reasonable arrangement of equipment and the introduction of lean production, the next step is to solve the problem of operational production planning.

The operational production planning problem involves organizing the coordinated work of all departments for production with the optimal use of all resources, quality assurance and compliance with contractual deadlines. This problem is often solved by experts, but this way can lead to inconsistency in the decisions made by individual experts and errors caused by the human factor. The use of modern information technologies allows us to avoid these errors, and also to take into account all the parameters and how various decisions affect each other.

As we know, any type of planning needs to be optimized. However, it is often not possible for experts to solve the scheduling problem. For example, complicated production usually contains many types of equipment of various kinds, a large number of types of product nomenclature, and tens or hundreds of various workshops with thousands of workers. The compiling of an effective production plan for such manufacture by experts alone is just impossible. In such cases, we can use technologies
for supporting operational production planning, providing experts with a wide range of various tools for solving this problem.

Today, the main directions in the development of such systems are their active integration into various industries and their use as intelligence software. Now they are systems with complex heuristics and algorithms to solve the problem, covering more data and revealing hidden patterns in these data.

2. Problem statement
Job shop scheduling as a separate field of science originated with the work of Henry Gantt [1] published in 1903. In this work, he developed diagrams that made it possible to plan the work of machines and workers. Currently, these diagrams are called Gantt diagrams and they are still actively used in project management.

The job shop scheduling problem in a general case can be formulated as follows. Let us have a set of activities with some characteristics, for example, the processing time and processing cost. We also have a set of machines (or equipment) in which these activities should be performed in a specific order. Thus, we need to create a schedule of activities on a given number of machines that will minimize one or more of the specified criteria (for example, the total time to complete the activities, known as makespan). Thereby, we have a problem of combinatorial optimization.

As is well known, the problem of job shop scheduling is an NP-complete problem. Other combinatorial NP-complete problems, such as the bin packing problem, knapsack problem, travelling salesman problem and also the scheduling problem can be formulated as resource-constrained project scheduling problems (RCPSP) [2]. An RCPSP is defined formally in [3, 4].

Unfortunately, in practice RCPSPs are much more complicated than their classical formulation; for example, we may have several types of machines as well as human resources (employee competencies). There is the processing of several lots in production at the same time, and these can influence each other. If we define corresponding designations for lots and specially reformulate the objective function and constraints, then we get a hierarchical optimization problem which includes the combinatorial optimization problem for lot ordering and RCPSP. It means that we need to solve an RCPSP for the activities of each lot for the considered lot order. In this case, the top level of the hierarchy is the travelling salesman problem with corresponding restrictions [5].

In general, the RCPSP is NP-hard, but, on the other hand, there are a large number of heuristics and algorithms that can solve it, for example, a genetic algorithm (which, unfortunately, does not produce a good result [6, 7]) or ant colony optimization algorithm, which was originally designed for solving the travelling salesman problem.

In summary, in this work we propose to consider the job shop problem as a hierarchical combinatorial optimization problem, where at the top level of hierarchy we consider the problem of lot order as a travelling salesman problem, and at the lower level of hierarchy RCPSP for each lot is replaced by a simulation model. We chose this approach because a real-world RCPSP is so complex (it has a lot of restrictions and it is too hard to find an optimal solution) and time-consuming that if any feasible solution is found, it can be considered as a successful result. Using a simulation model allows us to relatively quickly find some solution which will optimize some external parameters, for example, an order of lots.

3. Algorithm description
There are a large number of algorithms and heuristics which have been developed to solve the travelling salesman problem. These algorithms can be easily generalized for cases with various restrictions.

3.1. Lin-Kernighan heuristic
This heuristic is one of the representatives of standard zero-order optimization methods for solving combinatorial optimization problems. The main idea is an iterative improvement of the current
solution (initially randomly generated) by replacing $k$ edges by other edges. When it finds a new more valuable solution, it takes this as the current one and repeats the procedure. Usually, it uses a multi-start for achieving greater efficiency and continues repeating iterations until the current solution changes.

3.2. Genetic algorithm

The idea of a genetic algorithm (GA) is based on the genetic processes of biological organisms. The population of individuals evolve over several generations, obeying the laws of natural selection and the principle of “survival of the fittest”. Holland formulates the basic principles of GA in [8]. GA uses a flexible coding of solutions in its chromosome, which allows us to use it in almost any type of optimization problem, thus making it one of the most universal optimization algorithms.

It is important to note that combinatorial GA has a significant difference compared to the standard GA in that the chromosome is now described not by a binary string, but by a permutation of numbers from 1 to $n$, where $n$ is a number of cities. For this reason, the crossover and mutation operators are modified. For example, instead of single-point, two-point or uniform crossovers, here we have a modified type of crossover, where the offspring is the result of a combination of the first parent random region of the chromosome and the remaining sequence of numbers of the second parent. The mutation also has various modified variants [9].

3.3. Intelligent water drops algorithm

The intelligent water drops algorithm (IWDs) [10] is based on natural laws by which rivers lay their route along the path of least resistance to the lowest point on the Earth's surface. This fact explains why the river has so many twists and turns. We can imagine the river as a lot of drops confronting each other and with the environmental resistance of the soil in a riverbed. In this regard, several features characterize each water drop:

1. Each water drop has a speed that depends on environmental resistance.
2. Each water drop can transport a few amounts of soil from one place of the river to another.

Usually, the soil is transported from areas with a higher speed of the river to areas with a lower speed. Therefore, areas with a higher speed become deeper than areas with lower speeds. Since the areas with greater speed are deeper, they can accommodate more water and therefore "attract" more drops.

During this soil transfer from one area to another with the help of a water drop, the amount of soil transported by the drop increases. It is important to note that a drop of water with high speed takes up more soil than a drop with low speed. This means that the amount of soil which can be transferred depends on the speed of the drop. However, the speed of the water drop also depends on the soil – it grows faster on a stretch with a lower amount of soil. Since rivers and therefore water drops flow along the path of least resistance, it transpires that it prefers a path with a small amount of soil. If we look at this situation from a mathematical point of view, we can note that the considered problem is the environment, and the river, which consists of a certain amount of water drops, tries to find an optimal path. The convergence of this algorithm to the optimal solution is proved in [11].

3.4. Ant colony optimization

The idea of an ant colony optimization (ACO) algorithm [12, 13] comes from the natural ability of ants to find quickly the shortest path from an anthill to the food. In nature, ants are able to adapt to changing environmental restrictions. They find a new path when obstacles appear. When ants move, they leave particles of pheromone on the way. Other ants use this information later for choosing their way to the food or the anthill. Thus, when an ant is deciding where it needs to go, it relies on two factors: the visibility of the next point defined as a Euclidean distance to the proposed point and the amount of pheromone that "contains" this path.

The ACO algorithm is a swarm-type algorithm that models a multi-agent system. It is a greedy heuristic, where the probabilities of moving from one point to another correspond to the quality of the
obtained solution on previous iterations. We can use the ACO algorithm to solve most combinatorial optimization problems. For some types of problems, we need to modify it, but the main idea of visibility and pheromone on paths is always used.

The ACO algorithm guarantees convergence to the optimal solution, but the speed of this convergence is not known. In a general case, any ACO algorithm can be algorithmically represented in the following form:

- Initialization
- Until we reach the break condition:
  1. Find the solution
  2. Refresh the pheromone level
  3. Additional actions (modifications)

3.5. Cooperative ant colony optimization

The considered bionic algorithms include many parameters (types of crossover, mutation, selection of GA, visibility and pheromone level for ACO algorithm) and we need to choose them correctly because it significantly affects the quality of the solution. There are various heuristics which can solve this problem, for example the probabilistic self-configuring one presented in [14] or the cooperative approach of self-configuration [15], which we develop for the ACO algorithm in this paper.

The main idea of the cooperative approach is the parallel running of several algorithms during the adaptation period specified by the researcher. At the time, when the algorithms are running, they are fighting with each other for a common general resource and trying to get as much as possible. After the end of the adaptation period with the help of formula (1) we calculate the quality of each algorithm in the adaptation period.

\[ q_i = \frac{T - k}{k + 1} \cdot b_i(k) , \]  

where \( T \) is the adaptation interval, \( b_i(k) = 1 \), if the \( i \)-th population at the moment \( k \) contains the most valuable individual, \( k = 0 \) denotes the current situation, \( k = 1 \) denotes the previous situation and so on. We use the calculated quality estimates of the algorithms to determine the best algorithm and reallocate resources in its favour. During the allocation of resources, each algorithm gives to the winning algorithm some percentage of the initial value of the total resource specified by the researcher. After the reallocation of resources, we repeat the adaptation period.

It is also important to note that each algorithm has a certain “social card", a minimum or immutable percentage of the initial amount of resources. In addition to the reallocation of such a resource as the number of individuals in a population, we also use in this work the amount of pheromone on all edges between the nodes (cities). We recalculate the pheromone level after each adaptation interval by the following formula:

\[ \tau = \sum_{i=1}^{n} \tau_i q_i , \]  

where \( \tau \) is the matrix of the pheromone level, \( \tau_i \) is the matrix of the pheromone level belonging to the \( i \)-th algorithm, \( n \) is the number of algorithms in the cooperative ACO algorithm. The quality of algorithms \( q \), must be normalized using the following formula:

\[ q_i^{new} = \frac{q_i}{\sum_{j=1}^{n} q_j} . \]  

We tested the considered algorithms on various benchmarks in [16]. In this work, we showed that the efficiency of the cooperative ACO algorithm exceeds the efficiency of the self-configured (by
probabilistic approach) genetic algorithm and ACO algorithm in all test problems for TSP. In addition, this testing showed that the efficiency of the considered cooperative ACO algorithm is slightly smaller than the efficiency of the ACO algorithm with the best-found parameters, and in some cases even exceeds it.

4. Experimental results

We develop the considered simulation model in the programming language Java with the help of the framework Spring Boot. The database was developed using PostgreSQL, and this database interacts with the application using the MyBatis framework. The model simulates a production process in terms of concepts such as machines (equipment), employees, operations, lots and processes. It is important to note that employees and machines are a renewable resource, and the model compiles their schedule of work. At the same time, both machines and employees are divided into several types. Technological processes are defined by a certain sequence of operations. Each lot at the input of the simulation model receives one of the technological processes – the lot path in production.

As a result, the input of the simulation model receives the order of lots, and for each lot, a schedule is sequentially prepared taking into account the necessary resources. The model automatically respects all specified restrictions because they are taken into account in the simulation model.

The comparison of the effectiveness of the algorithms was performed according to the previously proposed technique [4, 17] on six randomly generated tasks of various dimensions. When we generate the tasks, the first thing that we do is to generate from one to five various types of machines and employees. Then we generate 50 different operations that select machines and employees and have a random duration. Next, we generate six different technological processes, which contain from three to ten randomly selected operations. The number of lots for each of 1-6 technological processes was 10-60, respectively.

Each algorithm runs 50 times with the same resources, which comprise 1000000 calculations of the objective function. The calculated results were averaged by different algorithm settings and by runs. Unfortunately, due to the number of possible solutions, it is not possible to find out the true global minimum. Therefore, we cannot use such a quality estimation as reliability. On the other hand, it makes sense to compare all the algorithms with each other to choose the most efficient one. A comparison of the efficiency of the algorithms was performed according to the criterion of the total time of the technological process with specified lots (makespan).

For algorithms, we applied the parameters presented in table 1:

| Algorithm                        | Parameters       | Value                                      | Total number of combinations of the parameters |
|----------------------------------|------------------|--------------------------------------------|-----------------------------------------------|
| Genetic algorithm (GA)           | selection        | Proportional, tournament (size of tourament 2, 4, 8), linear ranking, exponential ranking (exponential weight equals 0.5, 0.8, 0.9) | 384                                           |
| Ant colony optimization algorithm (ACO) | mutation | Inversion, 2-exchange, shifting, insertion | 16                                            |
| Inteigent water drops algorithm (IWDs) | $\alpha$ | 1, 2, 5, 10 | 16                                           |
| Inteigent water drops algorithm (IWDs) | $\beta$ | 1, 2, 5, 10 | 16                                           |
| Inteigent water drops algorithm (IWDs) | $\rho_s$ | 0.9, 0.7, 0.5 | 81                                           |
| Cooperative ant colony optimization algorithm | mulet | 10 % | -                                            |
| Cooperative ant colony optimization algorithm | social card | 20 % | -                                            |
The results of the testing are presented in figure 1. For each algorithm, the best setting on each task was chosen, and thus, algorithms with the best settings and the ones averaged over all possible settings were considered. For example, "GA (best)" means GA with the best settings for a task averaged by 50 runs, "GA (average)" means the result averaged by 384 variants of settings and by 50 runs.

As is well known, the effectiveness of GA, ACO or IWDs algorithms for each specific task highly depends on the selected settings. This means that we must tune each algorithm for each specific task. The process of selecting parameters takes significant time, because, as we have seen in table 1, a bionic algorithm usually has many combinations of parameters. The self-configuring ACO algorithm as well as the cooperative ACO algorithm, eliminate this drawback because they configure their parameters during the run and save a large amount of time that can be used, for example, to increase the resources provided to the algorithm.

![Figure 1. Results of testing on tasks 1-6.](image)

Testing showed that the self-configuring ACO algorithm and the cooperative ACO algorithm have greater efficiency than standard algorithms with "average" settings, but slightly worse than standard
algorithms with the "best" settings. These are very good results because finding the "best" settings for a real-word problem can take a large amount of time (which increases with the growing dimension of the task). For example, if you need to find the "best" parameters for a standard ACO on a specific case of the task, you need to run the algorithm 81·50 = 4050 times instead of 50 times for a cooperative algorithm (50 times are needed to average the results). Thus, we can use the current, but slightly worse than the best solution, or spend significant time awaiting the results of the full test of all combinations of parameters. In fast industry processes, we do not have so much time and cannot select the second option.

Another advantage of the cooperative ACO algorithm and the self-configuring ACO algorithm is that they change their search strategy throughout their work. At the initial stage, the algorithm-members of cooperation contributing to the global search have a greater "gain" because they allow us to escape from the gravitation of local minimum. At the end of work, on the contrary, cooperation may give preference to a local search strategy for an approximation of the current solution.

5. Conclusion
In this paper, we considered combinatorial optimization algorithms such as the Lin-Kernighan heuristic, a genetic algorithm, an ant colony optimization algorithm, as well as the self-configuring ACO algorithm and the cooperative ACO algorithm. In addition, we considered an approach for solving the RCPSP by using a hierarchical structure of this problem, which greatly simplifies the application of combinatorial optimization algorithms in practice in the face of a serious lack of time resources. The top-level TSP problem of this hierarchical structure has been solved based on the considered algorithms.

Each considered algorithm has advantages and disadvantages, but they allow us to effectively solve the problem introduced in this paper. The main disadvantage of standard methods is the need to select parameters for each specific task. We can eliminate this drawback by using a self-configuring or cooperative version of these algorithms because they self-configure the desired parameters in each task and show efficiency that is better than the efficiency of the algorithm with "average" settings.

Acknowledgements
The reported study was funded by RFBR according to the research project № 18-37-00433

References
[1] Gantt H L 1903 A graphical daily balance in manufacture Transactions of the American Society of Mechanical Engineers 24 322-36
[2] Anichkin A S and Semenov V A 2014 Modern models and methods of scheduling theory Proc. ISP RAS 26(3) 5-50
[3] Sousa R A, Varela M L R, Alves C, Machado J 2017 Job shop schedules analysis in the context of industry 4.0 Proc. of International Conference on Engineering, Technology, and Innovation (ICE/ITMC) pp 711-17
[4] Semenkina O E, Popov E A 2019 Nature-inspired algorithms for solving a hierarchical scheduling problem in short-term production planning Herald of the Bauman Moscow State Technical University, Series Instrument Engineering 3 46–63
[5] Davendra D 2010 Traveling salesman problem, theory, and applications (Tech. Publishing)
[6] Eiben A E, Smith J E 2003 Introduction to evolutionary computing (Berlin Springer-Verlag)
[7] Gromov S A, Tarasov V B 2011 Integrated intelligent systems for operational production planning News SFU: Technical News 26(3) 60-7
[8] Holland J H 1975 Adaptation in Natural and Artificial Systems (Ann Arbor: University of Michigan Press)
[9] Semenkina O Ev, Popov E A, Semenkina O Er 2017 Self-configuring nature-inspired algorithms for combinatorial optimization problems Journal of SFU Mathematics & Physics 10(4) 463–73
[10] Shah-Hosseini H 2007 Problem-solving by intelligent water drops Proc. of IEEE Congress on Evolutionary Computation pp 3226-31
[11] Shah-Hosseini H 2009 Optimization with the Nature-Inspired Intelligent Water Drops Algorithm International Journal of Bio-Inspired Computation 1(1/2) 71-9
[12] Dorigo M 1992 Optimization, Learning, and Natural Algorithms (Politecnico di Milano)
[13] Dorigo M, Stutzle T 2010 Ant Colony Optimization: Overview and Recent Advances M. Gendreau and J.-Y. Potvin, eds. Handbook of Metaheuristics vol 146 of International Series in Operations Research & Management Science New York: Springer chapter 8 pp 227-63
[14] Semenkina O E, Semenkina O E 2013 Research of the effectiveness of bionic algorithms of combinatorial optimization. Programmye produkty i sistemy 3 126-30
[15] Semenkin E S, Gumennikova A V, Emelianova M N, Sopov E A 2003 About evolutionary algorithms for solving complex optimization problems Vestnik SibGAU 5 14-23
[16] Dresvyanskiy D 2018 About effectiveness of co-operation of ant colony optimization algorithms in solving the problems of combinatorial optimization In Proc. of International Conference Science and Research Conference "Reshetnev Readings" (Krasnoyarsk) 2 116-17
[17] Dresvyanskiy D 2018 On biology-inspired stochastic algorithms for solving combinatorial optimization problems. In Proc. XVII-th International Scientific Conference of bachelor students, master students, post-graduate students and young scientists "YOUTH, SOCIETY, MODERN SCIENCE, TECHNOLOGIES & INNOVATIONS" (Krasnoyarsk) pp 254-56