Nature-inspired algorithms for a scheduling problem in operational planning

O E Semenkina¹ and E A Popov²

¹Research Department, Reshetnev Siberian State University of Science and Technologies, Krasnoyarsk, Russia
²System Analysis and Control Department, Reshetnev Siberian State University of Science and Technologies, Krasnoyarsk, Russia

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

Abstract. The scheduling problem is a widespread one, and it is still not automatized mostly because of the so-called combinatorial explosion. The paper describes two different approaches to solving a hierarchical scheduling problem based on solution representation. The first one proposes to find an optimal order of projects and then to solve the resource-constrained project scheduling problem for each of them. The second one assumes that we can find a priority of all activities for all projects and use it in the schedule building process if there is a conflict in the choosing of the next activity. The paper considers some nature-inspired algorithms such as the intelligent water drops algorithm, a genetic algorithm and ant colony optimization as well as a self-configuring version of the last two. The algorithm performance and different solution representation approaches are compared using the results of solving the test problems.

1. Introduction

Customer requirements in the modern world claim a high level of precision in product parameters and the fulfilment of deadlines, which can be achieved by the constant improvement of processes at all levels. Only companies and organizations that are able to adapt faster than others to new technologies may remain in the market. In recent years, there have been many changes in everyday life due to the appearance of new information technologies, and this process could not help but affect the manufacturing area. In industry, the trend of converting factories into smart factories is called the Fourth Industrial Revolution, or, Industry 4.0 [1]. The main goal of this transformation is to make a factory work intelligently with its resources; it can include many spheres from supply chains to risk management.

Nine main blocks that form a kind of backbone of Industry 4.0 were highlighted in [2]:

- Additive Manufacturing allows a sample to be created without loss of raw materials
- Horizontal and vertical system integration to create a collaboration between different levels of operational processes and information flows
- Cybersecurity as an important part of the continuous development of internet-based systems
- Augmented reality is an important part of smart industry that is helpful in maintenance procedures or complicated operations that require parameter control
- Clouds bring flexibility in sharing information with contractors, interaction with information services and accessibility from anywhere in the world
• Autonomous robots for reducing errors in simple tasks
• The industrial internet of things that uses devices and sensors to enable communication between equipment, components and services
• Big data and analytics that are responsible for quick and efficient responses to changes
• Simulation for strategic analysis to test the benefits of process change without its realization

All these blocks can be presented as separate interacting decision support systems and services that are combined into an integrated information ecosystem [3]. Operational planning in manufacturing systems is a kind of near real-time control where simulation runtime is an important aspect. The efficiency of short-term decision-making hangs on the speed of online simulation, which in turn depends on many factors such as time horizon, amount of equipment, number of employees, variety of technological processes and so on.

There are several ways to overcome the problem of combinatorial explosion, for example, the use of problem-oriented heuristics, but in this case, attempts to optimize by a criterion may not be successful. Some general rules such as the First Come First Served rule, Shifting Bottleneck Heuristic or Earliest Due Date may also be helpful in job-shop scheduling problem solving [4]. However, a much more interesting approach is to find how to apply different existing global optimization algorithms to the problem. The scheduling problem as well as other well-known combinatorial problems such as the travelling salesman problem (TSP) [5], the bin packing problem or the knapsack problem can be formulated as resource-constrained project scheduling problems (RCPSP) [6].

The authors propose to consider all the details of a production process using a simulation model and also use optimization algorithms for a top-level problem that allows some system parameters to be found that are optimal in some sense. An earlier investigation shows that the hierarchical structure of the problem [7] makes it possible to be sure that all solutions in a search space are feasible in any case and at the same time significantly reduces the problem dimension as well as the number of constraints.

2. Problem statement
The resource-constrained project scheduling [8] problem is formulated in terms of resources such as machine tools or employees with limited availability and activities (or operations). It is assumed that the duration of activities is known. The problem consists in finding a schedule with minimal makespan by assigning a machine tool, an employee and a start time to all activities of a project. It is important to specify that precedence relations as well as resource availability constraints must be respected. Let us define the problem more accurately.

Let \( n \) be an amount of activity and \( m \) be an amount of resource. We assume that the project for the current scheduling consists of activities defined by the set \( A = \{a_0, \ldots, a_{n+1}\} \), where activities \( 0 \) and \( n + 1 \) are fictitious and represent the start and end time points of the project. The clear set of real activities \( A = \{a_1, \ldots, a_n\} \) need to be scheduled under the constraints of the renewable resource set \( R = \{r_1, \ldots, r_m\} \) where each resource \( k \) is available in the quantity of \( B_k \).

The vector of processing times is denoted as \( p = (p_0, \ldots, p_{n+1}) \in \mathbb{N}^{n+2} \), where \( p_i \) is the processing time of activity \( i \). Recalling that activities \( a_0, \ldots, a_{n+1} \) are fictitious, their processing time should be equal to zero \( p_0 = p_{n+1} = 0 \). We suppose that each activity \( i \) requires \( b_{ik} \) amount of resource \( k \) where \( k = 1, \ldots, m \). If \( S_i \) is a start point of activity \( a_i \) for \( i = 0, \ldots, n + 1 \) than \( S_0 = 0 \) and \( S_{n+1} \) is a makespan because the project endpoint is equal to the difference between the start point and finish point. Let the precedence constraints be expressed as a set \( E \) composed of index pairs \( (i, j) \), which means activity \( i \) must be completed before activity \( j \) starts:

\[
S_i + p_i \leq S_j \quad \forall (i, j) \in E.
\]

To comply with the resource constraints, we can limit the number of current activities that use each resource \( k \) at the same time \( t \) from 0 to \( T \), where \( T \) is the scheduling horizon as follows:

\[
A^k_{\text{process}} = \{i \in A \mid S_i \leq t < S_i + p_i \} \forall k \in \overline{1, m}, \forall t \in [0, T].
\]
This horizon $T$ can be considered as an upper bound for the makespan or just taken to be quite large. It is required that at any time $t$ the sum of activities $A^k_{\text{in process}}$ that uses each resource $k$ does not exceed the resource availability:

$$\sum_{i \in A^k_{\text{in process}}} b_{ik} \leq B_k \quad \forall \ k \in \mathbb{1}, m, \forall t \in [0, T]$$

(2)

Let $S = \{S_0, S_1, ..., S_n, S_{n+1}\}$ be a solution of RCPSP, or in other words, it is a project schedule in the form of activity start times. We can say that $S$ is a feasible solution if it satisfies constraints (1) and (2). In this way, RCPSP is an optimization problem with the criterion $S_{n+1} \rightarrow \min$. The problem is computationally difficult (NP-hard) [9] and one of the most intractable combinatorial optimization problems.

Unfortunatley, in the field of manufacturing scheduling problems have many details, and as a consequence, they require a more specific formulation. For example, some operations require several machine tools, employees have different competencies, employees need breaks or time to move from one operation to another, one operation can use more electricity or reagents then others and so on. However, first of all there usually exist a set of projects (or lots) that need to be scheduled. Let there be a set of workflows (activity sequences) corresponding to all possible project (lot) types. We can also define an activity-to-string mapping because we need to represent it as a vector of $n$-dimensional space (figure 1)

![Figure 1. Activity-to-string mapping example.](image)

Let $L = \{l_1, l_2, ..., l_M\}$ be a set of all lots (projects) where each lot $h$ needs a special sequence of activities that can be defined as set $A^h = \{a^h_1, ..., a^h_{n_h}\}$. Now we can define the objective function as follows:

$$\sum_{i=1}^{H} S^h_{n+1} \rightarrow \min$$

(3)

Constraints (1) and (2) for several lots need to be transformed to

$$S^h_i + p_i \leq S^h_j \quad \forall (i, j) \in E_h \quad \forall h \in \mathbb{1}, H$$

(4)

$$\sum_{i \in A^k_{\text{in process}}} b_{ik} \leq B_k \quad \forall k \in \mathbb{1}, m, \forall t \in [0, T]$$

(5)

where $A^k_{\text{in process}} = \{i \in A^k \cap ... \cap A^m | S^h_i \leq t < S^h_j + p_i \} \forall t \in [0, T]$.

In this paper, we consider optimization problem (3)-(5) transformed to a hierarchical optimization problem containing a combinatorial ordering problem and nested RCPSP replaced by a model with some rules. The top-level ordering problem can be regarded as a travelling salesman problem [5]. Modification of this kind makes it possible to simplify the problem in the case of an existing
production model and allows the dimension and number of constraints to be reduced making the application of optimization methods in operational production planning easier.

The real-life manufacturing process is complicated, and it is hard to find a feasible solution using definitions (1)-(3). However, the proposed approach supposes that we can create as complex a model as necessary and still have the opportunity to optimize some model or input parameters. Operational planning has a serious time limit, so it is required to find a solution as quickly as possible, but at the same time, restriction violation is unacceptable.

3. Model description
As mentioned above, the model contains a set of entities such as machine tool, employee, operation (activity), lots and workflows. Each lot corresponds to one of the existing workflows that are kinds of production process and can be presented in the form of the example in figure 1. Each machine tool belongs to one of the equipment types and each employee has one or more competencies from a list. Machine tools and employees are renewable resources that have schedules that can be presented as availability intervals when the resource is not occupied by any operation. An operation or activity has a list of necessary machine tools and employee competencies, and the operation has a predefined duration. Each operation can be used in any workflow in theory. As an input of the model, we consider a list of operations for all lots in a certain order that needs to be processed by one of the existing workflows. The schedule builder uses this information and the production model to create a schedule, and then another model calculates a criterion (makespan), and as a result, the model returns a criterion value. The scheme of this process is presented in figure 2.

![Figure 2. Principal scheme of scheduling model.](image)

The schedule builder takes operations in the given order and puts them at the first accessible point when the machine tool and employee for this operation are available. This approach guarantees that all restrictions on the resources are met. A start point of operation is selected as the nearest free point with an available resource that uses a greedy strategy. The schedule builder realized all the business logic and can construct a valid schedule according to the constraints of the production process. One more important aspect is solution representation because it defines the problem dimension, problem type, and search space. There are four main types of solution representation, which are shown in figure 3.

The first variant supposes that we need to find start times for all activities as it is in the classical RCPSP formulation. However, in this case, a large number of unfeasible solutions can be obtained. In the second variant, we are trying to define real-value priorities for all activities [10]. It can give us flexibility and a wide search space, but at the same time, several solutions may correspond to one schedule [11], for example, sets \{0.2, 2.3, 4.2\} and \{1.3, 1.7, 7.1\}.

Next, there are two forms of permutation-based solution representations, such as activity list permutation and priority-based representation. For the activity permutation form, its location in the permutation signifies the order in which activity will be scheduled. This approach leads to the necessity of using a problem-oriented algorithm to form a feasible solution because of precedence constraints. The priority-based form was used for particle swarm optimization [12], and activity-to-string mapping is required. The location in the permutation denotes the activity and the value means
that its priority is scheduled to start [13]. This variant also means that two permutations \{2,4,6,1,3,5\} and \{2,4,1,6,3,5\} may correspond to the same schedule.

![Figure 3. Solution representation types.](image)

The last approach described in [7] aims to find an optimal order of lots to schedule which means that only after all activities for the first lot are scheduled can the next one be processed by the schedule builder. In this way, we significantly reduce the search space which means a smaller dimension, but at the same time, the probable exclusion of a global optimum. Nevertheless, this price may be reasonable in the face of serious limited computational resources.

In this paper, we compare two approaches, namely permutation of activity priorities and permutation of lots as the most promising. The scheduling process using lot order representation is illustrated in figure 4.

![Figure 4. Solution representation with lot order.](image)

The scheduling process using a permutation of activity priorities means that at the start of the process we defined a list of available activities, the first activities for all lots. On each step, we choose from the list of available activities the one with the best priority value and find for it a place in the schedule. After deleting from the list the used activity, we need to add all the next activities that became available after the current one. The process continues until the activity list becomes empty. This process is illustrated in figure 5. Both formulations of the problem require careful use by some algorithms because it is not clear what a distance between two lots or two activity priorities means.

4. Algorithm description
Nature-inspired algorithms show competitive results on the travelling salesman problem [14] which is why we also used them in this work. In addition to them, Lin-Kernighan Heuristics [15] is used.
4.1. Genetic algorithm
Genetic algorithms (GA) [16] are inspired by the natural evolution process and are a direct illustration of the principle of "survival of the fittest". A typical optimization procedure involving GA is the sequential application of such operators such as selection, recombination, mutation and elitism.

The scope of this algorithm is sufficiently wide and includes scheduling problems in the classical formulation. Operators need to be modified to work with chromosomes in some activities, and resource indexes are encoded as well as start points [17]. Unfortunately, this solution representation did not show very good results, but the real-priorities representation shows good results [11].

![Activity priority order]

Activity order for schedule builder:

1) a31
2) a31 a11
3) a31 a11 a12
4) a31 a11 a12 a21

... ...
8) a31 a11 a12 a21 a22 a23 a32 a24 a32

**Figure 5.** Solution representation with activity priority order.

4.2. Intelligent water drops algorithm
The intelligent water drops algorithm (IWDs) [18] is a population-based meta-heuristic inspired by the movement of water drops in natural rivers, lakes and seas. IWDs construct a better solution through cooperation with other drops and resistance of the environment. This algorithm is also successfully used for scheduling problems in the cloud computing environment [19] or multi-objective job-shop scheduling problems [20].

4.3. Ant colony optimization
The general idea of ant colony optimization (ACO) [21] is based on the natural ability of ants to find the shortest path from an anthill to the food source. Ants can adapt to changing environmental restrictions using pheromone that they left on their path. Thus, other ants can use this cooperative information for choosing a new way to the food or anthill. ACO has also been successful implemented in scheduling problems [22, 23].
4.4. Self-configuring method

All bionic algorithms have many parameters that must be chosen, and this is their great disadvantage. Besides, the best settings on a particular task may differ, and it is impossible to forecast them in advance. Real-world problems do not usually allow resources to be spent on determining the best algorithm settings. The way to fix this disadvantage is through a self-configuring method for the parameter control “on a fly”. This paper considers the self-configuring technique borrowed from [24] which has proven itself in previous works for bionic algorithms solving TSP.

5. Experimental results

Algorithm performance was compared by solving six tasks which were generated using pseudo-random numbers as was described in [7] for lot order problems. It is important to notice that when we use a priority-based solution representation, the dimension increases significantly since each lot has its own set of activities shown in table 1.

Table 1. Task dimensions for different solution representation.

| Task | Lot order problem dimension | Activity priority order problem dimension |
|------|----------------------------|------------------------------------------|
| 1    | 10                         | 75                                       |
| 2    | 20                         | 150                                      |
| 3    | 30                         | 160                                      |
| 4    | 40                         | 270                                      |
| 5    | 50                         | 365                                      |
| 6    | 60                         | 355                                      |

A comparison of the algorithms as well as a comparison of the solution representations was performed on the same objective function calculation amount equal to 1000000. Experiment results were also averaged over 50 runs. The criterion for the scheduling problem is minimum makespan, but the optimum is not known in advance. So, the evaluation of the effectiveness cannot be based on found the solution, rather it is needed to compare the result relative to other algorithms.

Each algorithm has two estimates, namely the result of the algorithm with its best settings for a particular task and the result averaged over all possible settings of this algorithm. This means that the best algorithm (for example GA (best)) is the best result averaged over 50 runs, and the average algorithm is the best result averaged over 50 runs and over all algorithm settings (for GA (average) average by 384 setting variant). Since various tasks have solutions that differ greatly in absolute value, for the convenience of comparison, the result of the algorithm on each task was divided by the worst result on this task among all the algorithms. Given that we are solving a minimization problem, the smaller the result, the better, so the algorithms will have an efficiency from 0 to 1, where 1 has an algorithm with the worst solution.

The results of the experiments described above are shown in figure 6, where the darker colour indicates the solution representation as a permutation of activity priorities. The results of the self-configuring algorithms, namely Self-Configuring GA (ScGA) and Self-Configuring ACO (ScACO), are highlighted separately in blue.

The results of the investigation show that the self-configuring method is an effective standard one modification and has a significant advantage in that the user does not need to select the algorithm settings, but is still able to receive competitive results. Moreover, for solving complex problems, using different settings at different stages of the search process can be a good strategy.

As can be seen from the results, the statement of the problem through the search of the activity priority order shows the better results. Most likely this is due to the fact that after choosing the lot order (in the case of the lot order problem) for each of them the schedule is built in a greedy manner that is, the first available resources are selected. This strategy can lead to a discontinuous schedule for the current resource, where a blank space appears in the schedule of this resource between two already
set activities, which no other activity can fit into, although the schedule of a specific lot is dense. The problem statement with priority ordering significantly increases the dimension of the problem, complicating its solution, but at the same time expands the search space without limiting it to concatenated chains of activities.

![Figure 6. Algorithm comparison on tasks 1-6](image)

One more interesting result is the different behaviour of parameter probability changes of ScACO on different solution representations. This is due to the above-mentioned fact that ACO needs additional information about the distance between two points to select the next one in the permutation. In the case of solving the classical TSP, this distance is just a Euclidean metric. However, for other problems a careful selection of the evaluating heuristic is required. Because of this, it can be seen from figures 7-8 that the algorithm does not significantly change the probabilities of the parameter values, since in fact, without a good heuristic for distance estimation, it is forced to solve the problem operating by only the pheromone trace. On the other hand, the behaviour of ScGA during adaptation of the parameters does not change significantly since it does not require additional information when generating the solution (figure 9).
Figure 7. Probabilities of parameter alpha values through the one run of adaptive ScACO on task 2

Figure 8. Probabilities of parameter beta values through the one run of adaptive ScACO on task 2

Figure 9. Probabilities of selection operator variants through the one run of ScGA on task 2.

6. Conclusion
In this paper, the performance of some heuristic algorithms of combinatorial optimization, such as the Lin-Kernighan heuristic, intelligent water drops algorithm, genetic algorithm, ant colony optimization algorithm and their self-configuring versions, was compared. The self-configuring method shows competitive results that are better than those averaged by the settings. We also consider two different solution representations based on lot order and activity priority order, and the latter shows better results despite a significant increase in the dimension of the problem. This result also shows that it is necessary to move towards some kinds of adaptive methods that allow algorithm parameters to be tuned during their work for all bionic algorithms. This property is especially important when solving scheduling problems in the context of operational production planning when a solution needs to be found very quickly based on the current state of the manufacturing process.

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

[1] Weyer S, et al 2015 Towards Industry 4.0 – Standardization as the crucial challenge for highly modular, multi-vendor production systems IFAC-PapersOnLine 48(3) 579-84
[2] Saucedo-Martinez J A, et al 2017 Industry 4.0 framework for management and operations: a review J. Ambient Intell. Human Comput. 9(3) 789–801
[3] Alcácer V and Cruz-Machado V 2019 Scanning the Industry 4.0: A Literature Review on Technologies for Manufacturing Systems Eng. Sci. Technol. Int. J. 22 899–919
[4] Sousa R A, et al 2017 Job shop schedules analysis in the context of industry 4.0 International Conference on Engineering, Technology and Innovation (ICE/ITMC) pp711-17
[5] Davendra D 2010 Traveling salesman problem, theory and applications (In. Tech. Publishing)
[6] Anichkin A S and Semenov V A 2014 Modern models and methods of scheduling theory Proc. ISP RAS 26(3) 5-50
[7] Semenkina O E, Popov E A and Ryzhikov I S 2019 Hierarchical scheduling problem in the field of manufacturing operational planning IOP Conf. Ser.: Mater. Sci. Eng 537
[8] Artigues C, Demassey S and Neron E 2008 Resource constrained project scheduling: Models, algorithms, extensions and applications. Control systems, robotics and manufacturing series Wiley-ISTE
[9] Blazewicz J, Lenstra J and Rinnooy Kan A H G 1983 Scheduling subject to resource constraints: classification and complexity Discrete Applied Mathematics 5(1) 11–24
[10] Peng W, Wei Y 2008 PSO for solving RCPSP In Proc. Chinese control and decision conference pp 818–22
[11] Chan W T, et al 1996 Construction resource scheduling with genetic algorithms, Journal of Construction Engineering and Management, ASCE 122 (2) 125–32
[12] Kennedy J and Eberhart R C 1995 Particle swarm optimization Proc. IEEE Conf. Neural Netw. 1V 1942–48
[13] Zhang H, Li X, Li H and Huang F 2005 Particle swarm optimization-based schemes for resource-constrained project scheduling Autom. Constr. 14(3) 393–404
[14] Semenkina O E 2013 Research of the effectiveness of bionic algorithms of combinatorial optimization Programmnuye produkty i sistemy 3 126-30
[15] Lin S and Kernigan B W1973 An effective heuristic algorithm for the traveling-salesman problem Operations research 21(2) 498–516
[16] Bäck T, Fogel D B, et al 1997 Handbook of Evolutionary Computation (Oxford University Press)
[17] Gromov S A, Tarasov V B 2011 Integrated intelligent systems for operational production planning News SFU: Technical science 7 60-7
[18] Shah-Hosseini H 2007 Problem-solving by intelligent water drops Proc. of IEEE Congress on Evolutionary Computation pp 3226-31
[19] Elsherbiny S, et al 2018 An extended Intelligent Water Drops algorithm for workflow scheduling in cloud computing environment Egyptian Informatics Journal 19 33–55
[20] Niu S H et al 2013 An improved intelligent water drops algorithm for solving multi-objective job shop scheduling Engineering Applications of Artificial Intelligence 26 2431–42
[21] Dorigo M and Stutzle T 2010 Ant Colony Optimization: Overview and Recent Advances Handbook of Metaheuristics 146(8) 227-63
[22] Merkle D, Middendorf M and Schmeck H 2002 Ant Colony Optimization for Resource-constrained Project Scheduling IEEE Transactions on Evolutionary Computation 6(4) 333–46
[23] Abidi M H, Al-Harkan I M et al. 2014 Ant Colony Optimization for Job Shop Scheduling to Minimize the Total Weighted Tardiness Proc. of the Industrial and Systems Engineering Research Conference
[24] Semenkin E and Semenkina M 2012 Self-Configuring Genetic Algorithm with Modified Uniform Crossover Operator Advances in Swarm Intelligence, ICSI 2012 pp 414-21