Cooperative self-configuring nature-inspired algorithm for a scheduling problem

O E Semenkina¹, E A Popov² and E S Semenkin²

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, eugenesemenkin@yandex.ru

Abstract. One of the crucial challenges related to operational manufacturing planning is an optimal plan search in the current situation using a workflow model. The problem solving is greatly hindered by the rapid growth of the search space with an increase in dimension or the so-called combinatorial explosion. This paper uses two different approaches to solving a hierarchical scheduling problem based on different solution representations. The first approach assumes a search of an optimal project order and then solving of resource-constrained project scheduling problem (RCPSP) for each of the projects using a model based on the greedy principle. In the second approach we are searching for priorities of all actions of all projects and then use them in the process of building a schedule if there are any conflicts when choosing the next action. To solve the problem with both approaches, the paper considers some nature-inspired algorithms such as the intelligent water drops algorithm (IWDs), a genetic algorithm (GA) and ant colony optimization (ACO) as well as a self-configuring version of the last two. The paper shows the efficiency of the application of the coevolution algorithm using IWDs, self-configuring GA and ACO

1. Introduction

The development of information technology opens up great opportunities for the active transition of many organizations to Industry 4.0 [1]. Such innovations directly affect all areas of both human activity and business, because, in just a few years, such technologies go from a competitive advantage in the market to a standard of use necessary to carry out activities. And if the production monitoring systems like manufacturing execution systems (MES) are already widespread and are going through the stages of revision and automation, then the widespread implementation of effective systems for operational manufacturing planning is complicated by many problems.

First of all, of course, this is due to the deficiency of proper process formalization, which produces great difficulties already at the stage of creating a system model. But even if such a model exists, the dimension of the scheduling problem in its classical formulation allows it to be classified as a large-scale optimization problem. This problem is also strengthened by the fact that the complexity of the domain creates the complexity of the workflow model, which increases calculation time for one value of the objective function dramatically. Thus, the key problem in creating operational manufacturing planning systems is that the issue of computational speed is extremely aggravated because, on the one
hand, it is necessary to search in a high-dimensional space, on the other hand, there is a strict time limit for finding a solution since the plan constructed belatedly loses its relevance.

To solve the problem discussed above, the article has two approaches from different sides. Firstly, there are two ways to reduce problem dimension with the use of a hierarchical scheduling problem formalization suggested in [2]. Another way to speed up the search for solutions is to use a self-configuring algorithm, which, due to the flexibility of the algorithm, helps to automate the adjustment of an algorithm to the problem in the process of solving it. Of course, it is possible to use one of problem-oriented heuristics or general rules for job-shop scheduling [3] like the First Come First Served rule, Shifting Bottleneck Heuristic, or Earliest Due Date but sometimes this provides only a quick solution instead of searching of global optima, especially in cases with dynamic and complicated business-processes. It can lead to a dead-end due to the problem of scalability and adaptability to changing business processes.

2. Problem statement

One of the classic formulations of the scheduling problem is resource-constrained project scheduling problem (RCPSP) [4, 5]. The problem is to find a schedule of minimal duration by assigning a start time to each activity of a project such that the precedence relations and the resource availabilities are respected. In terms of manufacturing a project is a lot that must be processed by some kind of workflow and operations in this workflow correspond to project activities. Thus, in the operational manufacturing planning, it is necessary to deal with several RCPSP (because of several lots), which must be solved simultaneously. At the same time, RCPSP is computationally difficult as an NP-hard problem [6].

In the field of manufacturing planning, scheduling problems have many details associated with the complexity of the domain area, and therefore often require more specific formulation for a current situation. For example, some operations require multiple machines, employees have different competencies, employees need breaks or time to move from one operation to another, one operation may consume more electricity or reagents than others, and so on. Therefore, the classical formulation of the problem or its complication for one specific situation is not a universal method, and there are no formulations that combine all possible difficulties. Therefore, it makes sense to use simulation modeling without the need to shape it up to an analytical form and then use the parameters of the model as controlled variables. More details about the developed model were described earlier [7], so here we will give only the basic principles.

First, we need to represent a schedule for one lot (project) as a vector of n-dimensional space, so let us define workflows (activity sequences) for each type of lot as some non-linear sequence of activities can be transformed with activity-to-string mapping (figure 1).

Let \( L = \{l_1, l_2, ..., l_n\} \) be a set of all lots (projects) where each lot \( h \) needs a special sequence of activities that can be defined as a set \( A^h = \{a^h_1, ..., a^h_n\} \). In this case, the objective function can be defined as follows:
\[ \sum_{h=1}^{H} S_{n_{h+1}} \rightarrow min \]

where \( S_{n_{h+1}} \) is a start point of last activity \( a_{n_{h+1}} \) for lot \( h = 0, \ldots, H \), where \( n_h \) is a number of operations for lot \( h \) in corresponding workflow. \( S_{h+1} \) is a makespan because the project endpoint is equal to the difference between the start point and finish point.

In this paper, we consider the optimization problem transformed into a hierarchical optimization problem containing a combinatorial ordering problem and nested RCPSP replaced by a model with some greedy rules. In this case, the ordering problem can be regarded as a traveling salesman problem (TSP) [8] since in both problem statements it is necessary to find the optimal permutation of a certain entities (numbered lots or numbered all activities of all lots), taking into account that each entity can be used only once. Such a transformation helps to reduce the problem dimension and make it easier to implement in operational production planning. As an input of the model, we use a list of operations for all lots in a certain order that needs to be processed. The module with a schedule builder takes defined operation in the order given by an algorithm and puts them at the feasible point when the machine tool and employee for this operation are available. The schedule builder module implements a simulation model and therefore guarantees accordance with all restrictions.

There are four main types of solution representation with real values and in terms of permutations (figure 2). The first one corresponds to classical RCPSP and requires finding start points for all activities of each project which leads to problems a large number of unfeasible solutions that can be obtained. The second one with real-value priorities is flexible, but in this case, search space has a lot of different solutions with the same criterion value. The third type means activity permutation to define which one has to be scheduled first. This approach leads to the necessity of using a problem-oriented algorithm to form a feasible solution because of precedence constraints. And the last one looks the most promising because every solution can be unique, and we can delegate all precedence constraints problems to the module of schedule builder. The location in the permutation denotes the activity and the value means that its priority is scheduled to start [9].

![Figure 2. Solution representation types.](image)

So, in this paper, we consider two approaches – one is a permutation of priorities of all activities of all projects and the second one is to find an optimal order of lots themselves to be scheduled described in [10]. It means that only after all activities for the first lot were scheduled, the module of schedule builder will obtain activities for the next lot. This approach helps to significantly reduce the search space.
from about $n \times H$ dimensions to just $H$. But the big disadvantage of this approach is that it is possible to exclude global optimum from search space.

Thereby, we compare these two approaches, called a permutation of activity priorities and permutation of lots, and a more simple scheduling process for lot order representation is illustrated in figure 3 and a more complex process for permutation of activity priorities in figure 4. The location in the permutation denotes the activity and the value means that its priority when scheduling. At the start of the process we defined a list of available activities which consist of the first activities for each lot. On each step, we choose from the list of available activities the one with the best priority value and find a place in the schedule for it. After deleting used activity from the list, we need to add all the next activities that became available after the current one. The process continues until the activity list becomes empty.

**Figure 3.** Solution representation with lot order.

**Figure 4.** Solution representation with activity priority order.

### 3. Algorithm description

Efficiency research was conducted for several standard algorithms, such as Lin-Kernighan heuristics [11], genetic algorithms (GA) [12], intelligent water drops algorithm (IWDs) [13], and ant colony optimization (ACO) [14]. Also, for GA and ACO was added self-configuring technique borrowed from
[15] and thus were used self-configuring GA (SCGA) and self-configuring ACO (SCACO). The self-configuring method helps to select the parameters of the algorithm while solving the problem, since nature-inspired algorithms often have many adjustable parameters. In this work, algorithms were considered for sets of their parameters, evaluating them from the point of view of the average efficiency for all parameters, as well as the efficiency with best parameters. The self-configuring versions of the algorithms used the same settings to choose between them.

For GA there are 8 types of selection operators that were considered: tournament selection (with the tournament size equals 2, 4, 8), fitness proportional selection and rank selection with linear or exponential ranking (exponential weight equals 0.5, 0.8, 0.9). Also 3 mutation probabilities were considered (low, medium, and high) as well as 4 types of mutation: by inversion, by 2-exchange, by shifting and by insertion. An algorithm with and without the use of elitism was also considered. So, totally $8 \cdot 3 \cdot 4 \cdot 2 \cdot 2 = 384$ variants of genetic algorithm settings were considered. For ACO this work considers following variants $\alpha = 1, 2, 5, 10$, $\beta = 1, 2, 5, 10$, in total $4 \cdot 4 = 16$ variants. The paper does not consider such parameters as evaporation rate and other, since in earlier studies the authors found that statistically they have little effect on the result. In case of IWDs this paper considers the following setting variants: significance of the best solution in updating of soil matrix $\alpha = 0.1, 0.3$ or 0.5, local soil updating parameter $\rho_n = 0.9, 0.7$ or 0.5, initial soil on each edge of the graph $\text{InitSoil} = 1000, 10000$ or 100000, initial velocity of each drop $\text{InitVelocity} = 20, 100$ or 200. So, totally $3 \cdot 3 \cdot 3 = 81$ variants of IWDs settings were considered.

Another adaptive technique used in this article is coevolution [16]. The coevolution approach is based on the parallel execution of several population algorithms during a certain adaptation period. All populations rival each other for a common computing resource, and at the end of each adaptation period, the quality of each algorithm in this period is calculated using the following formula:

$$q_i = \sum_{k=0}^{T-1} \frac{T-k}{k+1} \cdot b_i(k)$$

where $T$ is the adaptation interval, $b_i(k) = 1$, if the $i$-th population at the moment $k$ contains the fittest individual, $k = 0$ denotes the current situation, $k = 1$ denotes the previous situation, and so on. Further, the calculated estimates of the quality of the algorithms are used to determine the best algorithm, which means that resources will be redistributed in its favor. During resource allocation, each population reduces its number of individuals and transfers them to the winning algorithm. After the reallocation of resources, a new adaptation period begins. A flowchart of the coevolution algorithm is shown in figure 5.

Multipopulational algorithms consist of several subpopulations that are periodically exchanged by individuals, this exchange of individuals is called migration and is controlled by several parameters. In this paper, we selected the topology of the exchange of individuals where the best algorithm in an adaptation period gives its best solution to other algorithms. This best solution replaces the worst individual of the population (for example, in the genetic algorithm), or, if this is not possible, contributes to the history of the population (for example, in the intelligent water drops algorithm, contributes to a soil matrix).

In this paper, we consider two coevolutionary algorithms with self-configuring nature-inspired algorithms (self-configuring GA, self-configuring ACO, as well as the standard IWDs) using and without the migration mechanism, respectively designated as CoScNI-M and CoScNI.

4. Experimental results

The effectiveness investigation of all the above algorithms was carried out on six problems that were generated using pseudo-random numbers, as described in [2]. A solution representation for the problem significantly affects its dimension, for example, for the lot order problem the dimension is the number of lots that should be scheduled. For the representation based on priorities, the dimension depends not only on the number of lots but also on the number of operations for each lot, provided that each lot should be processed according to a certain workflow and the number of operations in different
workflows may not coincide. The dimension of six problems for both representations is shown in table 1 below.

**Figure 5.** Flowchart of coevolutionary algorithm.

**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                                      |
To ensure equal conditions, in this paper, all algorithms were compared using the same number of calculations of the objective function - 1,000,000. The conclusions were made based on statistical data obtained by averaging the results of 50 independent runs for each setting of each algorithm. Since the criterion for solution estimation is makespan, and the problem dimension does not allow us to find out the true global optimum, the effectiveness was assessed by comparing the algorithms with each other.

In the materials and conclusions below, each algorithm has two results. The first one was obtained for the algorithm with its best of all the considered settings for a particular problem, and the second result is averaging over all the considered settings of this algorithm. For example, GA (best) is the best GA result out of all results for each settings averaged over 50 runs, and GA (average) is the result averaged not only over 50 runs but also over all the algorithm settings under consideration, in this case, 384 setting variants were considered for GA. Besides, various problems have solutions that differ greatly in absolute value. Therefore, for the convenience of comparing the worst result out of all algorithms was chosen for each problem, and all other results were divided by this worst result. Such a transformation allows us to evaluate the efficiency of the algorithm in comparison with the others on a scale from 0 to 1, where 1 is obviously the worst result, and the closer the value is to 0, the more the algorithm's solution differs from the worst on the problem, which means that it is better.

Described experiment results are shown in figures 6-12, where the darker color indicates the solution representation as a permutation of activity priorities and the lighter one refers to the result for lot order problem. Figure 6 shows the result for all algorithm additionally averaged by 6 tasks. The results of the self-configuring algorithms, namely Self-Configuring GA (ScGA) and Self-Configuring ACO (ScACO), are highlighted separately in blue. The result of coevolutionary algorithms is highlighted in green.

The results show that the self-configuring method is an effective modification of algorithms because, although it is inferior to the algorithm with the best settings, it exceeds the variant averaged over all settings. Since bionic algorithms are quite unstable, their application in practice is complicated by the need to select the correct settings for each particular problem, which significantly increases the necessary computing resources, which are so limited in operational manufacturing planning. That is, the use of self-configuring algorithms guarantees a result no worse than the average with no need to spend resources on the choice of parameters, which means that these resources can be used to provide more calculations of the objective function to the algorithm, which will ultimately lead to a much better result. Moreover, for solving complex problems, using different settings at different stages of the search process can be a good strategy.

At the same time, the coevolutionary algorithm also shows a competitive result and allows the user not to make a decision about which algorithm should be used for a specific task. Also, it can be observed that an algorithm with isolated coevolution without migration shows a slightly better result, which may indicate
that it is more preferable to have independent populations and thereby explore a large area of the search space, avoiding the stagnation of all algorithms near the same local optimum.

It is also worth noting a comparison of the two variants of solution representation. Representation by order of activity priority shows the best results despite the much larger dimension of the problem. It is possible that for representation by lot order, for each lot the schedule is built greedy-like, that is, the first available resources are selected for a given order of operations. This strategy sometimes leads to an intermittent schedule for some machine tools or employees, because there can be a space between two already scheduled operations so, that no other operations can fit in it, although the schedule for a particular lot has no breaks. Representation by order of activity priority significantly increases the problem dimension, complicating its solution, but at the same time expands the search space without limiting it to chained lists of actions.

![Figure 7. Algorithm comparison on tasks 1.]

![Figure 8. Algorithm comparison on tasks 2.](image-url)
Figure 9. Algorithm comparison on tasks 3.

Figure 10. Algorithm comparison on tasks 4.

Figure 11. Algorithm comparison on tasks 5.
5. Conclusion
In this paper, some bionic algorithms for combinatorial optimization have been considered and investigated, such as the Lin-Kernighan heuristic, intelligent water drops algorithm, genetic algorithm, ant colony optimization algorithm. Self-configuring versions of GA and ACO were also presented, as well as a co-evolutionary algorithm using self-configuring bionic algorithms with and without the possibility of solution migration. The self-configuring method shows competitive results better than average one over all settings of each algorithm while allowing not to waste resources on choosing the algorithm settings for the current problem. The coevolutionary algorithm also shows high efficiency, comparable to the best algorithm for the problem, and at the same time eliminates the need to spend resources on choosing the best algorithm for the current problem.

This result also shows that it is necessary to move forward in the development of adaptive methods that allow us to customize the way of solving a problem while solving this problem, simplifying the application of intelligent technologies in practice. This feature is especially important when solving scheduling problems in the context of operational manufacturing planning when it is needed to find a solution very quickly based on the current state of the manufacturing process.

Acknowledgements
This work was supported by the Ministry of Science and Higher Education of the Russian Federation within limits of state contract № FEFE-2020-0013.

References
[1] A Weyer S, Schmitt M., Ohmer M and Gorecky D 2015 Towards Industry 4.0 – Standardization as the crucial challenge for highly modular, multi-vendor production systems IFAC-PapersOnLine 48(3) 579-84
[2] 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 032001
[3] Sousa R A, Varela M L R, Alves C and Machado J 2017 Job shop schedules analysis in the context of industry 4.0 International Conference on Engineering Technology and Innovation (ICE/ITMC) 711-7
[4] Anichkin A S and Semenov V A 2014 Modern models and methods of scheduling theory Proc. ISP RAS 26(3) 5-50
[5] Artigues C, Demasse S and Neron E (Eds.) 2008 Resource constrained project scheduling: Models, algorithms, extensions and applications Control systems, robotics and manufacturing series Wiley-ISTE
[6] 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
[7] Semenkina O E, Popov E A 2020 Nature-inspired algorithms for a scheduling problem in operational planning *IOP Conf. Ser.: Mater. Sci. Eng* **734** 012107
[8] Davendra D 2010 Traveling salesman problem theory and applications (In. Tech. Publishing)
[9] 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
[10] 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** 032001
[11] Lin S and Kernigan B W 1973 An effective heuristic algorithm for the traveling-salesman problem *Operations research* **21**(2) 498-516
[12] Bäck T, Fogel D B, Michalewicz Z eds. 1997 *Handbook of Evolutionary Computation.* – *Oxford University Press*
[13] Shah-Hosseini H 2007 Problem-solving by intelligent water drops Proc. of *IEEE Congress on Evolutionary Computation* 3226-31
[14] Dorigo M and Stutzle T 2010 Ant Colony Optimization: Overview and Recent Advances *Handbook of Metaheuristics International Series in Operations Research & Management Science* **146**(8) 227-63
[15] Semenkin E and Semenkina M 2012 Self-Configuring Genetic Algorithm with Modified Uniform Crossover Operator Advances in Swarm Intelligence ICSI 2012 *Springer* **1(733)** 414-21
[16] Ficici S G 2004 Solution Concepts in Coevolutionary Algorithms: A Doctor of Philosophy *Diss. Brandeis University*