FACTA UNIVERSITATIS (NIŠ)
Ser. Math. Inform. Vol. 37, No 5 (2022), 993–1006
https://doi.org/10.22190/FUMI220731068K
Original Scientific Paper

A GENETIC ALGORITHM WITH GREEDY CROSSOVER AND ELITISM FOR CAPACITY PLANNING *

Lev Kazakovtsev[1,2], Elena Kozlovskaya[2], Ivan Rozhnov[1,2] and Olga Patsuk[2]

[1] **Siberian Federal University, 79 Svobodny pr. 660041 Krasnoyarsk, Russia**
[2] **Reshetnev Siberian State University of Science and Technology prosp. Krasnoyarskiy Rabochiy, 31660039 Krasnoyarsk, Russia**

Abstract. We propose a modification to the genetic algorithm with greedy agglomerative crossover operator for the problem of scheduling product types at metal or plastic production factory facilities where the goal is to minimize the number of switchings of the product type of the production lines. Similar algorithms with greedy agglomerative crossover for location problems do not use any elitism in the population. For the considered problem which may also be classified as a location problem, elitism in the population implemened in the form of tournament selection plays a positive role. The article also discusses the dependence of the efficiency of the evolutionary algorithm on the size of the population. As our experiments show, the introduction of elitism into such an algorithm enables us to increase both the rate of convergence of the algorithm and the accuracy of the solution. A special aspect chooses an individual with the best value of the objective function.
Key words: genetic algorithm, greedy crossover, location problem.

1. Introduction

Optimal utilization of production capacity is an indicator of the efficient pro-

Received July 31, 2022, accepted: October 10, 2022
Communicated by Predrag Stanimirović
Corresponding Author: Lev Aleksandrovich Kazakovtsev, Siberian Federal University, 79 Svobodny pr., 660041 Krasnoyarsk, Russia and Reshetnev Siberian State University of Science and Technology, prosp. Krasnoyarskiy Rabochiy, 31660039 Krasnoyarsk, Russia | E-mail: levk@bk.ru
2010 *Mathematics Subject Classification.* Primary 90C57; Secondary 90C27, 90C09

*This work was supported by the Ministry of Science and Higher Education of the Russian Federation (Grant No.075-15-2022-1121)

duction operation. The metal processing factories which include both foundry and a metalworking production lines as well as plastic goods production must fulfill the plan for the release of each type of product, follow the rules concerning the production technology, and minimize the number of switches of the production types among the production lines.

Thanks to their conceptual clarity and intuitive appeal, greedy algorithms have found many applications in the computer sciences. Greedy algorithms are also often used as simple heuristics, even if the optimal solution is not guaranteed.

The application of the genetic algorithm allows us to draw up a schedule on production lines which is close to an optimal schedule. The genetic algorithms with the greedy agglomerative crossover operator are complex hybrid algorithms: they organize the global search with the commonly used evolutionary (genetic) framework, they include a special greedy agglomerative procedure which obtains properties of a local search algorithm, and they include also known commonly used local search procedures for scheduling problems. Local search algorithms allow gradual improvement of certain result in the vicinity of the known solution [1]. Since the greedy crossover procedure is computationally expensive, the population of individuals (solutions) in such algorithms is usually small. The dependence between the population size and the convergence speed has not been investigated. However, the convergence rate depends on the size of the population of individuals for the problem of operational planning of product types at the production facilities of plastic products. Finding a solution for extreme combinatorial problems such as scheduling problems must take into account the requirements for computing resources [2]. Although such problems can be represented in a form of integer linear programming problems, their linearization leads to a huge dimensionality of the resulting problem.

Genetic algorithms are common algorithms for solving such problems. They model evolutionary processes, find the best qualities (for example, genotype and phenotype) in individuals of the population, which lead to the best solution of the problem [3]. To plan the load of production capacities, a genetic algorithm applying greedy heuristics is used [4].

Many methods have been proposed to solve production planning problem such as: dynamic programming, enumeration, but greedy algorithms are no less known and common. Thanks to their conceptual clarity and intuitive appeal, greedy algorithms have found many applications in the computer sciences. Greedy algorithms are also often used as simple heuristics, even if the optimal solution is not guaranteed [5].

Optimization algorithms are usually a sequence of steps, each of which provides a number of options. In a greedy algorithm, a choice is always made that seems to be the best at the moment, that is, a locally optimal choice is made in the hope that it will lead to an optimal solution to the global problem, which is not always true [6]. In the greedy agglomerative procedures, some elements of the solution are succeessively eliminated: each time, the algorithm selects such an element for the elimination, which leads to the least significant deterioration in the value of the

objective function.

The classical problem in the location theory is the p-median problem on a network. It is necessary to find p network nodes in such a way as to minimize the total sum of distances between each network node and the nearest of the selected nodes [7]. Known genetic algorithms with greedy agglomerative heuristics [8, 9] used in many problems including the compiling production schedules [10, 11], reducing such problems to the problems of location of the production switching points on a three-dimensional grid (time-product-production line), traditionally do not use the strategy of elitism. As our experiments show, the introduction of elitism in the population of the genetic algorithm allows us to increase both the rate of convergence of the algorithm and the accuracy of the solution. A special aspect chooses an individual who has the best value of the objective function/functions [12].

The rest of this paper is organized as follows. In Section 2, we give an overview of known algorithm for the problem under consideration. In Section 3, we give a mathematical statement to our problem and describe the genetic algorithm. In Section 4, we provide the computational results with various parameter settings: various population size and elitism in the population. In Section 5, we give a conclusion.

2. Known Algorithms for Capacity Planning

Location problems are often used directly in urban development, in architecture, transportation, and also have indirect application. Starting from the 60s, the location problems were used to determine the optimal composition of various technical systems, or assortment of products and were not logically related to placement in a geometric sense [13, 14]. Local search is effectively applied to such problems. So, the ALA-procedure (Alternating Location-Allocation) used to solve the p-median problem [15] and the k-means procedure (an algorithm for cluster analysis problem which can be also considered as a location problem) have the same structure and are a common algorithm in placement theory.

Classical local search methods can quite simply find local optima of the problem, and existing varieties of the genetic algorithm differ in that with their help it is possible to obtain a solution in the form of a global optimum. However, it must be borne in mind that the task of checking the found optimum for globality is also a difficult task [16]. Local search methods for production planning problems are known from 1970s and include several procedure such as shift of the production type switch in time to the right (future) or left (previous periods), enumeration of possible production types for a production line, insertion of a new production switch moment into the production plan, and its elimination.

To obtain a globally optimal solution, many possible techniques can be used: enumeration, partial enumeration (branch-and-bound algorithms), variable neighborhood search, as well as various evolutionary algorithms.

Alp, Erkut and Drezner considered a genetic algorithm using a special recombination (crosser) procedure, a greedy (agglomerative) heuristic procedure [17]. This

algorithm is used to solve the p-median problem on the network [18]. Its feature is that instead of rearranging the sequences represented by the parent individuals in the population of the genetic algorithm, this heuristic procedure combines the parent set of indices of network nodes that are selected as the medians in the p-median problem, so the child solution begins to contain more medians than the conditions of the problem require. After that, the gradual removal of extra medians, that is, those elements of the solution, the removal of which gives the smallest increase in the p-median objective function (sum of distances), until a feasible solution with the required number of medians is reached [19].

Evolutionary algorithms (including genetic algorithms) use various principles and the conceptual apparatus of natural evolution. Their advantage, compared to classical ones, has been investigated experimentally [20], however, they determine only the general structure of the search. An important feature of greedy agglomerative heuristic methods is the possibility of choosing the next solution of the option that can give a better decrease in the value of the objective function or, when maximizing, the largest increase in the value. It should be borne in mind that these algorithms are local search methods that allow you to gradually improve a certain result in some vicinity of the known solution. The method of greedy heuristics was proposed by relation to the clustering and location problems [21]. The algorithms of this method give results in solving practical problems, which is problematic to improve by other methods in comparable time. Although they are mostly randomized, the results they produce are quite stable, yielding close results at the restarts applied. The same properties of the algorithms with the embedded greedy heuristic procedures remain for the production scheduling problems where the objective function is the number of production type switches. Since the genetic algorithms with greedy agglomerative procedure used as the crossover operator are computationally expensive and include embedded local search procedures, they are usually run with a small population of the solutions. Moreover, such algorithms for location [20] and other problems [21] do not involve any elitism: the individuals are chosen for crossover from the population with equal probability. We investigate modifications of such algorithms with various population sizes and with the elitism impplemented in a form of tournament selection.

3. Mathematical problem statement and genetic algorithm

The problem of production capacity scheduling was considered by Antamoshkin and Masich [22] as a pseudo-Boolean optimization problem. The modified version of the proposed model with integer variables is also used in this article.

Let there be P types of products that are produced on K production lines. Their performance is the same, for the selected pth type of products, the Kth line can produce V_p units of products per shift. The time is discrete and measured in shifts. We assume that there are three shifts per day. The algorithm will result in a plotted graph showing the product type produced by each production line. The limitation is that a part of the assortment produces a certain production line. The matrix Z of Boolean constants $z_{k,p}$, $1 \leqslant k \leqslant K$, $1 \leqslant p \leqslant P$ is set, which contains

1 if the pth line can produce the pth type of product and 0 otherwise. The W_p units for each product is entered in the production plan and it is assumed that it is necessary to fulfill this quantity in T_p days. It is also necessary to establish the minimum total load of the production complex, which reaches the volume of W_{min} units per day. Product types are combined into M classes C_c, $1 \leqslant c \leqslant M$. There is a limitation that the production must be stopped for one shift if a different product class is established for the production by a production line. It is assumed that the non-stop change of production from type p to type r is established by a symmetric matrix of Boolean constants $C_{p,r}$ of dimension $P \times P$: the value $C_{p,r} = 1$ indicates the need to stop the line when changing products from p-type to r-type, if $C_{p,r} = 0$, then there is the possibility of non-stop switching production (naturally, $C_{p,p} = 0$).

In [10, 11], the authors introduced variables $y'_{i,k}$, $0 \leqslant y'_{i,k} \leqslant P$. The value $y'_{i,k} = p$ means the release of the pth type of product on the kth line on the ith day (value $y'_{i,k} = 0$ means no release). It is assumed that $y'_{0,k}$ is a constant that characterizes the initial setting of each Kth production line for a certain type of product at an initial point in time. Additional variables $x'_{i,k}$ were also introduced:

$$x'_{i,k} = \begin{cases} y'_{i,k}, & y'_{i,k} \neq y'_{(i-1),k'} \\ 0, & y'_{i,k} = y'_{i,k} \end{cases} \quad \forall i = \overline{1,I}, k = \overline{1,K}, \tag{3.1}$$

Values $y'_{i,k}$ can be derived from $x'_{i,k}$:

$$y'_{i,k} = \begin{cases} y'_{(i-1),k}, & x'_{i,k} = 0, \\ x'_{i,k} & x'_{i,k} \neq 0 \end{cases} \quad \forall i = \overline{1,I}, k = \overline{1,K}. \tag{3.2}$$

We also use dependent boolean variables:

$$f_3(x) = -\sum_{i=1}^{I} \max\{0, W_{min} - \sum_{k=1}^{K}\sum_{p=1}^{P} V'_{yp}(3 - C'_{y'\,(i-1),k,y'\,i,k})\} \tag{3.3}$$

$$x_{i,k,1} = \begin{cases} 1, & x'_{i,k} = p, \\ 0, & x'_{i,k} \neq p, \end{cases} \quad \forall i = \overline{1,I}, k = \overline{1,K}, p = \overline{1,P}, \tag{3.4}$$

$$y_{i,k,1} = \begin{cases} 1, & y'_{i,k} = p, \\ 0, & y'_{i,k} \neq p, \end{cases} \quad \forall i = \overline{1,I}, k = \overline{1,K}, p = \overline{1,P}. \tag{3.5}$$

In this notation, the problem of production planning is formulated as follows:

$$\min \sum_{i=1}^{I} \sum_{k=1}^{K} \sum_{p=1}^{P} y_{i,k,p}(1 - y_{(i-1),k,p}); \tag{3.6}$$

with constraints (taking into account operation in three shifts):

$$V_p \sum_{i=1}^{T_p} \sum_{k=1}^{K} y_{i,k,p}(3 - C'_{y'_{(i-1),k,y'_{i,k}}}) \geqslant W_1 \quad \forall p = \overline{1,P}, \tag{3.7}$$

$$\sum_{k=1}^{K} \sum_{p=1}^{P} V'_{y'_p}(3 - C'_{y'_{(i-1),k,y'_{i,k}}}) \geqslant W_{min} \quad \forall i = \overline{1,I}. \tag{3.8}$$

$$y_{i,k,p} \leqslant Z_{k,p} \quad \forall i = \overline{1,I}, k = \overline{1,K}, p = \overline{1,P} \tag{3.9}$$

The $C'_{p,r}$ is taken as matrix of Boolean values, and $C_{p,r}$ is an augmented string with a zero index and a column with a zero index ($p, r \in \{\underline{0,P}\}$):

$$C'_{p,r} = \begin{cases} C_{p,r}, & p > 0, r > 0, \\ 0, & r = 0, \\ 1, & p = 0, r > 0 \end{cases} \tag{3.10}$$

Here, V' is the vector of the production rate by type of products per shift, supplemented by V a zero element:

$$V'_p = \begin{cases} V_p, & p > 0, \\ 0, & p = 0. \end{cases} \tag{3.11}$$

In this work, for the algorithm, it is proposed to replace all restrictions with the following penalty functions:

$$f_2(x) = -\sum_{p=1}^{P} \max\{0, W_p - V_p \sum_{i=1}^{T_p} \sum_{k=1}^{K} Y_{i,k,p}(3 - C'_{y'(i-1),k,y'i,k})\}. \tag{3.12}$$

$$f_3(x) = -\sum_{i=1}^{I} \max\{0, W_{min} - \sum_{k=1}^{K} \sum_{p=1}^{P} V'_{y'p}(3 - C'_{y'(i-1),k,y'i,k})\}. \tag{3.13}$$

We also define the objective function:

$$f(x) = |X| = \sum_{i=1}^{I} \sum_{k=1}^{K} \sum_{p=1}^{P} x_{i,k,p}, \tag{3.14}$$

The economic meaning of the function $f_2(X)$ is the total amount of products produced with a lag behind the plan, the function $f_3(X)$ is the total underperformance of the daily minimum of products produced.

The general scheme of the algorithm for problem (3.5) - (3.8) can be described using a genetic algorithm with greedy heuristics for planning continuous production [10, 11] and it will look like this:

Step 1. The initial array of sets of lattice nodes "change-line-type of production" represented by the threes of indices (i, k, p) is formed: $A = \{X_j\} = \{(i_1, k_1, p_1), \ldots, (i_{n_j}, k_{n_j}, p_{n_j})\}, j = \overline{1, N}$.) A population of N individuals of the genetic algorithm is formed. It is assumed that the number of lattice nodes n_j in each of the elements of the array A can be different.

Step 2. Randomly selected are two indices of parent "individuals" $j_1, j_2 \in \{\overline{1, N}\}, j_1 \neq j_2$. Then randomly selected $j_3 \in w$. Here w is some set of indices of "individuals" (array elements) rated as "bad."

Step 3. Assign $X_{j_3} = X_{j_1} \cup X_{j_2}$.

Step 4. For a selected X_{j_3} a mutation procedure is performed (optional, not used in practice for genetic algorithms with greedy crossover).

Step 5. For each node $V = (i_1, k_1, p_1) \in X_{j_3}$, do:

Step 5.1. If $\exists V_2 = (i_2, k_2, p_2) \in X_{j_3} : p_2 \neq p_1, i_2 = i_1, k_2 = k_1$, then choose randomly with equal probability index $p \in \{p_1, p_2\}$, to assign $X_{j_3} = X_{j_3} \backslash \{(i_1, k_1, p_1)\}$.

Step 5.2. Next iteration of loop 5.

Step 6. Calculate the boolean and integer variables $[x_{i,k,p}], [y_{i,k,p}], [x'_{i,k}], [C'_{y'_{i,k}}]$, corresponding to the set of matrices X_{j_3} according to formulae (3.9), (3.1)-(3.3). Due to the fact that the sets X_j in this algorithm are represented by matrices of integer variables, this step and the rest are reduced to calculation $[y'_{i,k}]$ according to the expression (3.1).

Step 7. Assign $FOUND = 0$.

Step 8. The nodes X_{j_3} of the set are arranged in random order, then, for each node $V = (i', k', p') \in X_{j_3}$ the following steps are performed:

Step 8.1. Assign $\xi = X_{j_3} \backslash \{V\}$. Find matrices of Boolean and integer variables $[x^{\xi}_{i,k,1}], [y^{\xi}_{i,k,1}], [x^{\xi}_{i,k}], [C'_{y'_{i,k}}]$ which correspond to set ξ. If $f_4(\xi) < f_4(X_{j_3})$, then assign $X_{j_3} = \xi, FOUND = 1$, the corresponding matrices of Boolean and integer variables $[x_{i,k,p}], [y_{i,k,p}], [x'_{i,k}], [C'_{y'_{i,k}}]$. To the next iteration of cycle 7. It is important that site $V = (i', k', p')$ exclusion from set and recalculation of corresponding

integer variables is reduced to zero value $x'_{i',k'}$ and recalculation of values $y'_{i'',k'}$ for $i'' = \overline{i', T_1}$.

Step 8.2. The application of local search procedures in the vicinity of the node V described below significantly increases the efficiency of the algorithm. Such procedures are known since 1970s and were described in [11] for our problem.

Step 8.3. Next iteration of loop 8.

Step 9. If $FOUND = 1$, then go back to Step 7.

Step 10. Check the shutdown conditions, if they are not met, go to Step 2.

This algorithm sequentially excludes one node from the set X_{j_3}, which corresponds to the work of the genetic algorithm with greedy heuristics for the p-median problem [8].

Steps 8-8.3 implement the greedy procedure which consequently eliminates the lattice nodes. This procedure includes local search procedures embedded in it. Thus, this procedure is computationally expensive. Algorithms with the greedy crossover operator are usually used with a small population. The extreme version of such algorithm is the (1+1)-Evolutionary Algorithm [26] with only one individual in the population. For several applications in clustering and location theory, such algorithms outperform the genetic algorithms.

Thus, the size of the population is a highly important parameter which predefines the efficiency of the whole algorithm. For algorithms with the greedy agglomerative crossover operator, the constant small number of individuals can be used. Other algorithms use a permanently expanded population [27]. In the next Section, we show that an optimal population size is relatively small, however, algorithms with an extremely small populations of 3 or 5 individuals are less efficient. An optimal size of the population for practically important production planning problems can be found by a reconnaissance search which can be used for constructing a self-configuring algorithm.

In addition, we demonstrate that including the elitism implemented in a form of a tournament selection improves the results of the algorithm, and this feature of the genetic algorithm distinguishes this algorithm from greedy crossover genetic algorithms for location problems where elitism is not used.

4. Dependence of algorithm effectiveness on population size

In our experiments, we used various values of population size were established (3, 5, 10, 30, 100). The condition for forced shutdown of the algorithm was the time limitation: 30 minutes. For each population size, 30 algorithm launches were performed (see Table 4.1).

The best result corresponds to 33 production type switches for populations with a size of 10, 30 and 100 individuals. Smaller populations, due to the lack of diversity, are unable to reach this result. Table 4.2 shows the ratio of attempts of all launches of our algorithm that achieve the best result, as well as the running time of the program during which the best solution was found.

Table 4.1: Results of the algorithm without elitism for various population sizes

Population size	Best result	Worst result	Average value
100	33	44	34.1
30	33	34	33.1
10	33	38	34.6
5	34	36	34.9
3	34	36	34.3

Table 4.2: Ratio of attempts achieving the best results

Population size	Ratio of attempts achieving the best result	Program running time for the best achieved result, sec.
100	0.9	838
30	0.9	277
10	0.2	744
5	0	204
3	0	158

Additional data on the results of the algorithm are given in Tables 4.3 and 4.4 as well as Fig. 4.1.

We used the genetic algorithm in two versions: original, with equiprobabilistic selection of the parent solutions at Step 2, and its version with modified Step 2. During the tournament selection at Step 2, all individuals of the population are divided into subgroups with the subsequent choice of individuals with the best fitness in each of them [23].

Despite the fact that with a population size of 30 individuals, the algorithm requires a slightly longer time to stop, it was with this population size in 90 percentages of program launches that a better solution was found. The experience showed that for a population with an initial size of 10 individuals there is no direct dependence. The best time in this case is higher than the population of 30 individuals. However, the algorithm converges to the best solution in 20 percent of attempts only.

The extremes of the graph are in direct dependence - the largest value of the population (100 individuals) corresponds to the greatest time of the best solution

Table 4.3: Actial tiome consumption to achieve the best result

Population size	Running time limit	Minimum	Maximum
100	18000	31	1486
30	18000	240	1376
10	18000	113	1288
5	18000	70	1762
3	18000	158	1794

Table 4.4: Median and average time for reaching the best achieved solution

Population size	Median	Average value
100	1217	1096.5
30	514	596.3
10	365	473.8
5	493	608.4
3	1161	1098.6

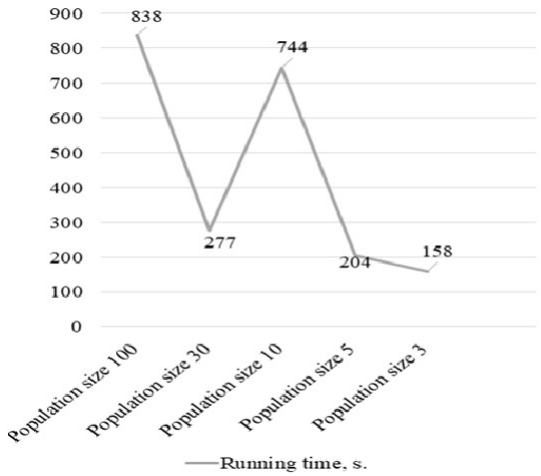

FIG. 4.1: The time needed to find a better solution depends on the original population size

(838 seconds), and the smallest value of the population (3 individuals) corresponds to the smallest time of the best solution (158 seconds).

On average, the solution is most quickly found with an initial population size of 10 individuals, and this time is 2 times less than the largest averages. Figure 4.2 shows this data as a graph.

Conclusion from Figure 4.2: the graph of results is a parabola. On average, the program worked longest in populations where the sample was maximum and minimum. For a population with an initial size of 10 individuals, the minimum average time it took the program to find a better solution is observed.

Subsequently, populations of 30 individuals were used to study the influence of elitism on the result of the algorithm. Tournament selection scheme: from a population containing N lines, t lines are randomly selected, and the best line is written to the intermediate array (a tournament is held between the selected lines). This operation is repeated N times. The rows in the resulting intermediate array are then used to cross (randomly). The size of the group of lines selected for the tournament is often 2. In this case, they talk about a binary or doubles tournament.

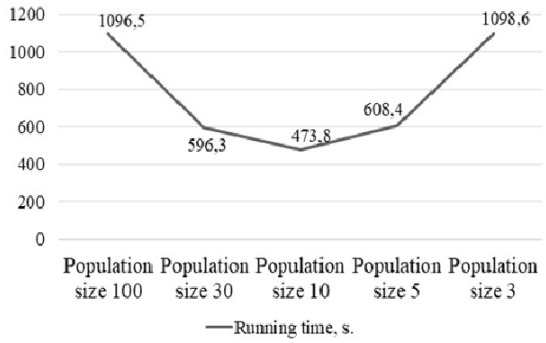

FIG. 4.2: The average program time taken to find a better solution depending on the original population size

Table 4.5: Results for an algorithm with tournament selection, population size 10

Minimum operating time, sec.	Maximum value, s	Ratio of successful attempts	Average program time, sec.
233	1105	0.6	546.7

The number of the tournament is taken as t. The more subgroups participate in the tournament, the less chances for individuals to get into the selection [24].

The advantage of using tournament selection as an elitism strategy is that there is no need for additional calculations, as well as ordering the lines in the population by increasing adaptability. This selection option is closer to reality, because the success of an individual is largely determined by its environment - decision points that are included in the area of permissible/best solutions, as far as they are better or worse than it [25]. We introduced the following modification into the algorithm with greedy heuristics.

Step 2 of our genetic algorithm will take the form:

Step 2. Randomly select three indices of parents j_{11}, j_{12}, j_2, $j_{11} \neq j_{12} \neq j_2$. Select randomly $j_3 \in w$. Here w is a set of indices of individuals (array elements A) rated as "bad."

Step 2.1. If $f_2(\chi_{j11}) + f_3(\chi_{j11}) > f_2(\chi_{j12}) + f_3(\chi_{j12})$ then $j_1 = j_{12}$;

Step 2.2. If $f_2(\chi_{j11}) + f_3(\chi_{j11}) < f_2(\chi_{j12}) + f_3(\chi_{j12})$ then $j_1 = j_{11}$;

Step 2.3. If $f_2(\chi_{j11}) + f_3(\chi_{j11}) = f_2(\chi_{j12}) + f_3(\chi_{j12})$ and $f(\chi_{j11}) > f(\chi_{j12})$ then $j_1 = j_{12}$, other $j_1 = j_{11}$.

The results of the program using tournament selection are presented in Table 4.5. The original population size is 10 individuals. The launch was carried out 30 times. The program limit time is 1800 seconds.

Using tournament selection, the best result is 33 switches, which was found in

233 seconds, which is slower than the result of finding the best solution with a size of the original population of 10 individuals without applying the strategy of elitism. However, with elitism, 60 percent of all launches of the program found the best solution (without it, we had much less successful attempts). Thus, elitism improves the likelihood of finding an optimal solution.

5. Conclusions

The introduction of the elitism strategy in the form of tournament selection, as well as the study of the dependence of the efficiency of the evolutionary algorithm for planning capacity utilization on population size, made it possible to increase the effectiveness of the genetic algorithm with a greedy agglomerate crossing operator for the task of planning production. The effectiveness of the algorithm is determined by the size of the population, which requires additional research in the direction of developing algorithms with a dynamic population, as well as the study of the possibility of applying evolutionary algorithms "without population" to this problem, namely the (1 + 1)-EA algorithm [26] ad similar ones.

REFERENCES

1. E. B. Patsuk and L. A. Kazakovtsev: *Formal model of dynamic scheduling of the educational center*. Economics and management of management systems. **28** (2018), 182.
2. L. A. Kazakovtsev, M. N. Gudyma, D. V. Stashkov, A. A. Stupin and N. N. Dzhioeva: *Evolutionary algorithms with a heterogeneous population for clustering and placement problems*. Monograph. Scientific and Publishing Center "Relevance". Moscow, (2017), 196.
3. L. Kazakovtsev, I. Rozhnov, I. Nasyrov and V. Orlov: *Self-adjusting genetic algorithm with greedy agglomerative crossover for continuous p-median problems*. In: Mathematical Optimization Theory and Operations Research: Recent Trends. MOTOR 2021. Communications in Computer and Information Science, **1476** (2021), 184-200. Springer, Cham. doi.org/10.1007/978-3-030-86433-0-13.
4. O. V. Patsuk: *Investment activity of the region*. Bulletin of SibGAU named after academician M. F. Reshetnev. **3(43)** (2012), 184-189.
5. A. A. Lazarev and E. R. Gafarov: *Schedule theory. Tasks and algorithms*. Economics . MOSCOW STATE UNIVERSITY, Moscow, (2011), 220.
6. E. G. Coffman: *Schedule Theory and Computing Machines*. Publishing Center "Academy", Moscow, (2010), 156.
7. A. Kapoor: *Hands-On Artificial Intelligence for IoT*. Packt Publishing, Mumbai, (2019), 267.
8. L. Kazakovtsev, I. Rozhnov, G. Shkaberina and V. Orlov: *K-Means Genetic Algorithms with Greedy Genetic Operators*. Mathematical Problems in Engineering. **2020** (2020), ArticleID 8839763. doi.org/10.1155/2020/8839763.
9. L. Kazakovtsev, I. Rozhnov, A. Popov and E. Tovbis: *Self-Adjusting Variable Neighborhood Search Algorithm for Near-Optimal k-Means Clustering*. Computation. **8(4)** (2020), ArticleID 90. doi.org/10.3390/computation8040090.

10. L. A. Kazakovtsev and A. N. Antamoshkin: *Algorithm for scheduling.* Bulletin of KrasGAU. **4(103)** (2015), 215-219.

11. L. A. Kazakovtsev., M. N. Guyma and A. N. Antamoshkin: *Genetic Algorithm with Greedy Heuristic for Capacity Planning.* International Congress on Ultra Modern Telecommunications and Control Systems and Workshops. **4** (2014), 607-613.

12. A. V. Eremeev and Yu. B. Kovalenko: *The task of compiling schedules with the grouping of machines by technology.* Discrete analysis and research of operations. **18** (2011), 54-79.

13. E. A. Beltz: *Optimizing the location of enterprises taking into account the minimum permissible distances.* Bulletin of Omsk University. **4** (2012), 13-16.

14. Yu. A. Kochetov: *Comparison of metaheuristics for solving the two-level problem of enterprise placement and factory pricing.* Discrete analysis and research of operations. **3 (123)** (2015), 36-54.

15. A. V. Panteleev: *Application of evolutionary methods of global optimization in problems of optimal control of deterministic systems.* MAI Publishing House, Moscow. (2013), 128.

16. S. Gonzalez-Martin, A. Ferrer, A. Juan and D. Riera: *Solving non-smooth arc routing problems throughout biased- randomized heuristics.* Computer-based Modeling and Optimization in Transportation, Springer International Publishing. (2014), 451.

17. B. A. Bozkaya: *Genetic Algorithm for the p-Median Problem.* Facility Location Applications and Theory. **7** (2002), 179-232.

18. N. S. Grigorvea: *The task of compiling maximum time offset schedules for parallel processors.* MAI Publishing House, Moscow. (2016), 246.

19. I. H. Sigal and A. P. Ivanova: *Introduction to applied discrete programming: models and computational algorithms.* 2nd edition, supplemented and corrected, FIZMATLIT, Moscow, (2007), 140.

20. D. Kirszenblat: *Dubins networks: Thesis.* Melbourne Department of Mathematics and Statistics of the University of Melbourne. **2** (2011), 56-89.

21. L. A. Kazakovtsev and A.N. Antamoshkin: *Genetic algorithm with fast greedy heuristic for clustering and location problems.* Informatica. **38** (2014), 229–240.

22. A. Antamoshkin and I. Masich: *Pseudo-Boolean Optimization in Case of an Unconnected Feasible Set.* In: "Models and Algorithms for Global Optimization", Optimization and Its Applications. **4** (2007), 111–122.

23. G. N. Dubin and A. A. Korbut: *FBehavior on average of greedy algorithms for the minimization problem about the rant - general distributions of coefficients.* Journal of Computational Mathematics and Mathematical Physics. **48** (2008), 1556-1579.

24. I. L. Vasiliev: *New lower grades for the problem of placement with client preferences.* Journal of Computational Mathematics and Mathematical Physics. **6** (2009), 1055-1097.

25. R. A. Neidorf, V. G. Kobak and D. V. Titov: *Comparative analysis of the effectiveness of tournament selection of a genetic algorithm for solving homogeneous distributive problems.* Bulletin of the Don State Technical. un-ta. **3** (2009), 410-432.

26. L. Kazakovtsev, I. Rozhnov and G. Shkaberina: *Self-Configuring (1 + 1)-Evolutionary Algorithm for the Continuous p-Median Problem with Agglomerative Mutation.* Algorithms. **14** (2021), 130. https://doi.org/10.3390/a14050130.

27. L. Kazakovtsev, D. Stashkov, M. Gudyma and V. Kazakovtsev: *Algorithms with Greedy Heuristic Procedures for Mixture Probability Distribution Separation*. Yugoslav Journal of Operations Research. **29**(1) (2019), 51-67. https://doi.org/10.2298/YJOR171107030K.