ELSEVIER

Journal of Materials Processing Technology 109 (2001) 90–95

Journal of Materials Processing Technology

www.elsevier.com/locate/jmatprotec

A production scheduling problem using genetic algorithm

R. Knosala*, T. Wal

Faculty of Mechanical Engineering, Silesian Technical University, Konarskiego 18a, 44-10 Gliwice, Poland

Abstract

The way of flexible manufacturing cell work scheduling with the aid of genetic algorithm and draft of code strings, which are used by this algorithm, have been presented in this paper. Moreover, some results obtained by computer program based on this method also have been presented. In first case it has been assumed that the cell works in optional mode — every operation can be done on every machine, and in second case cell works in sequential mode — the first operation is executed on the first machine, the second operation on the second machine and so on. The only criterion of schedule evaluation is the time of cell work. This time must be the shortest for definite number of jobs and machines.

Keywords: Genetic algorithms; Scheduling

1. Introduction

To project schedules as good as possible, many analytical and heuristic methods, from mathematical programming to the game theory are applied. One of many tools, used for this purpose, are genetic algorithms, one of artificial intelligence techniques [1,2]. Genetic algorithms are search algorithms, based on natural selection mechanisms and heredity. They join the survival principle of the best fitted strings with systematic, but carried out in a stochastic way, information exchange. In every generation the new group of artificial organisms (for example, bit strings), made from the fusion of the best fitted representatives fragments of previous generation, come into existence. Besides, the new part is tested sporadically. In spite of fate element, genetic algorithms do not reduce themselves to ordinary casual stray. They efficiently use the past experience to determine new searching field of expected higher efficiency (see [3]).

2. Activity of genetic algorithms

Genetic algorithms work on adequately transformed (coded) task parameters. If the task is to find a maximum of some function, then task parameters — values of function domain, must be transformed to the code strings. Such code can be a chain of zeros and ones (binary alphabet), chain of zeros, ones and twos (alphabet folded of three signs) or chain which is using alphabet of any, constant number of signs. It is recommended to limit number of signs and the most often one use binary alphabet.

Genetic algorithms are characterized by the following proprieties (see [3]):

1. they do not directly transform task parameters, but their coded form;
2. they lead searching, coming out not from one point, but from some population of points;
3. they use only fitness function, but do not use her derivative or other auxiliary information.

3. Program construction

Computer program, made for the purpose of scheduling with the aid of genetic algorithm, is composed of modules presented in Fig. 1. At first the input data — machining time of particular operations is fed in. After data read, program leads operations of genetic algorithm for 600 generations (it is constant, assumed number). There are 30 individuals (code strings) in every generation.

3.1. Code strings

The structure of the string for optional working cell has been presented in Fig. 2 and string for sequential working cell has been presented in Fig. 3. For optional working cell first part of code string is responsible for succession of distribution of particular jobs in the second part of this string. In ‘‘succession of jobs block’’ in Fig. 2 the jobs are put in order: 3, 1 and 2. That means in ‘‘operations block’’

* Corresponding author.

0924-0136/01/$ – see front matter
PII: S0924-0136(00)00780-9

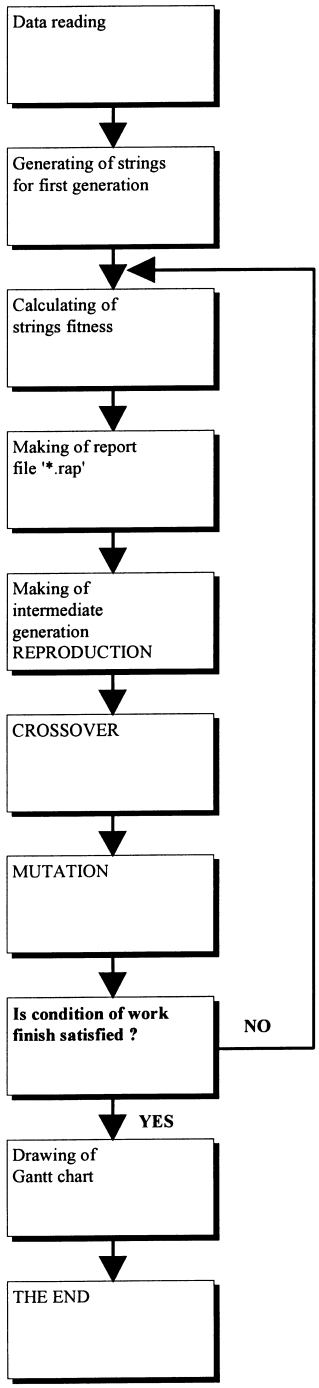

Fig. 1. Program structure.

operations belonged to third job are taken into consideration at first (two operations), then operations belonged to first job (three operations) and at last operations belonged to second job (three operations). It has been assumed that operations in range of job have placed according to succession of their machining. In ''operations block'' numbers of particular machines are assigned to individual positions, which are represent operations. For example, in Fig. 2 there are only digits 1 and 2 in this block. That mean distribution of operations has place for two machines only. Code string

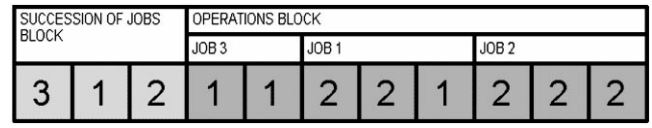

Fig. 2. Code string for optional working cell.

Fig. 3. Code string for sequential working cell.

for sequential working cell has been built of digits, which are a numbers of jobs, put in that order, in which operations, belonging to jobs, will be done on particular machines. In Fig. 3 you can see, that job number 2 will be done first, then job number 6, 3, 8, 1 and so on.

This construction of code strings should assure a good heredity of parents strings properties into the children strings.

Together with the change of jobs number, the length of strings changes too, but during the work of genetic algorithm, for definite number of jobs, the string length is constant. The data, which are necessary for program operation, are machining times for particular operations. On this base program is counting number of jobs and number of operations for every job.

3.2. Fitness function

One of the most difficult tasks in creating genetic algorithm is the proper designing of fitness function. This function is responsible for evaluation of quality of given code string. In the considered cases of cell's work scheduling it has been assumed, that time of cell's work is a value, which has to be minimized. This time is calculated for precisely specified number of jobs, consisting of a certain definite number of operations. The algorithm of fitness function calculating, that means time of cell's work, for optional working cell is presented in Fig. 4. Algorithm for sequential working cell is showed in Fig. 5.

For the proper use of fitness function values during the reproduction process, some operations on these values have to be performed. During reproduction the simulation of proper calibrated dial (roulette) in program was applied, therefore it was necessary to modify fitness function so that instead of searching the minimum, its maximum had to be found. In other words, up till now the code strings with the smallest value of fitness function have been regarded as ''the best fitted'' but after modification of fitness function ''the best fitted'' would be strings with maximum value of fitness function.

3.3. Crossover

In the first case choice of crossover's sort depends on crossover's site. If crossover point is placed in ''succession

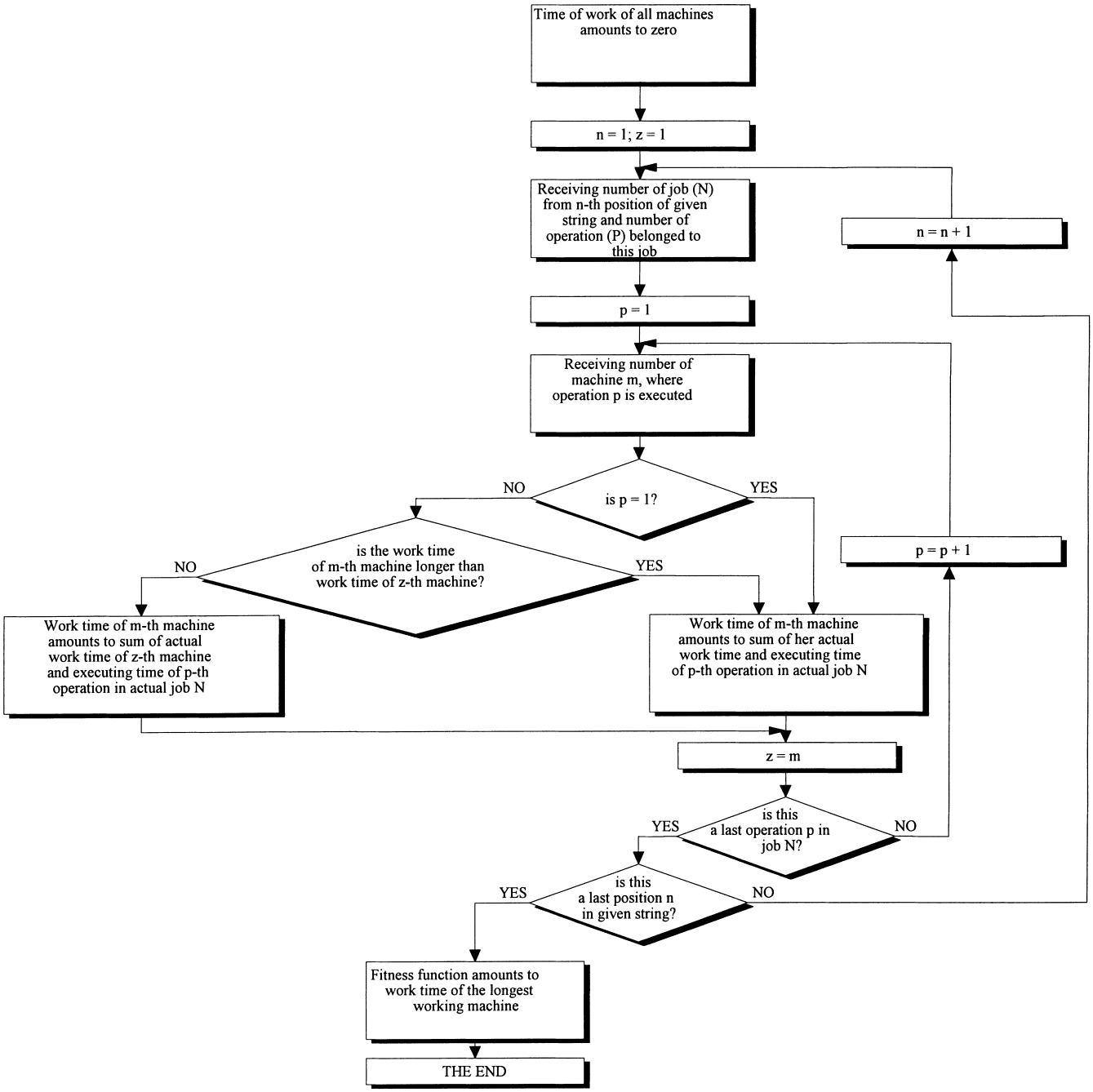

Fig. 4. Algorithm of fitness function calculating for optional working cell.

of jobs block'' then order crossover has place (see [4]), but if crossover point is placed in ''operations block'' or between blocks then simple crossover has place. In the second case, crossover is carried out in the same way like in ''succession of jobs block'' of previous string; because in string all jobs must be taken into consideration (none of them can be omitted) and jobs must not repeat, during strings crossover operation the so-called order crossover has been introduced. After probabilistic choice of crossover point, substituting to the first offspring part of the first parent, placed on the left side from the crossover point, takes place. Remaining positions in the first offspring's string are completed by lacking values originating from the second parent. Succession of putting in of these values to positions of the offspring's string placed on the right from crossover point is imposed by the second parent.

The second offspring comes into existence in the same way, fragment of string on the left from the crossover point originating from the second parent and the fragment on the right from the crossover point is completed according to the

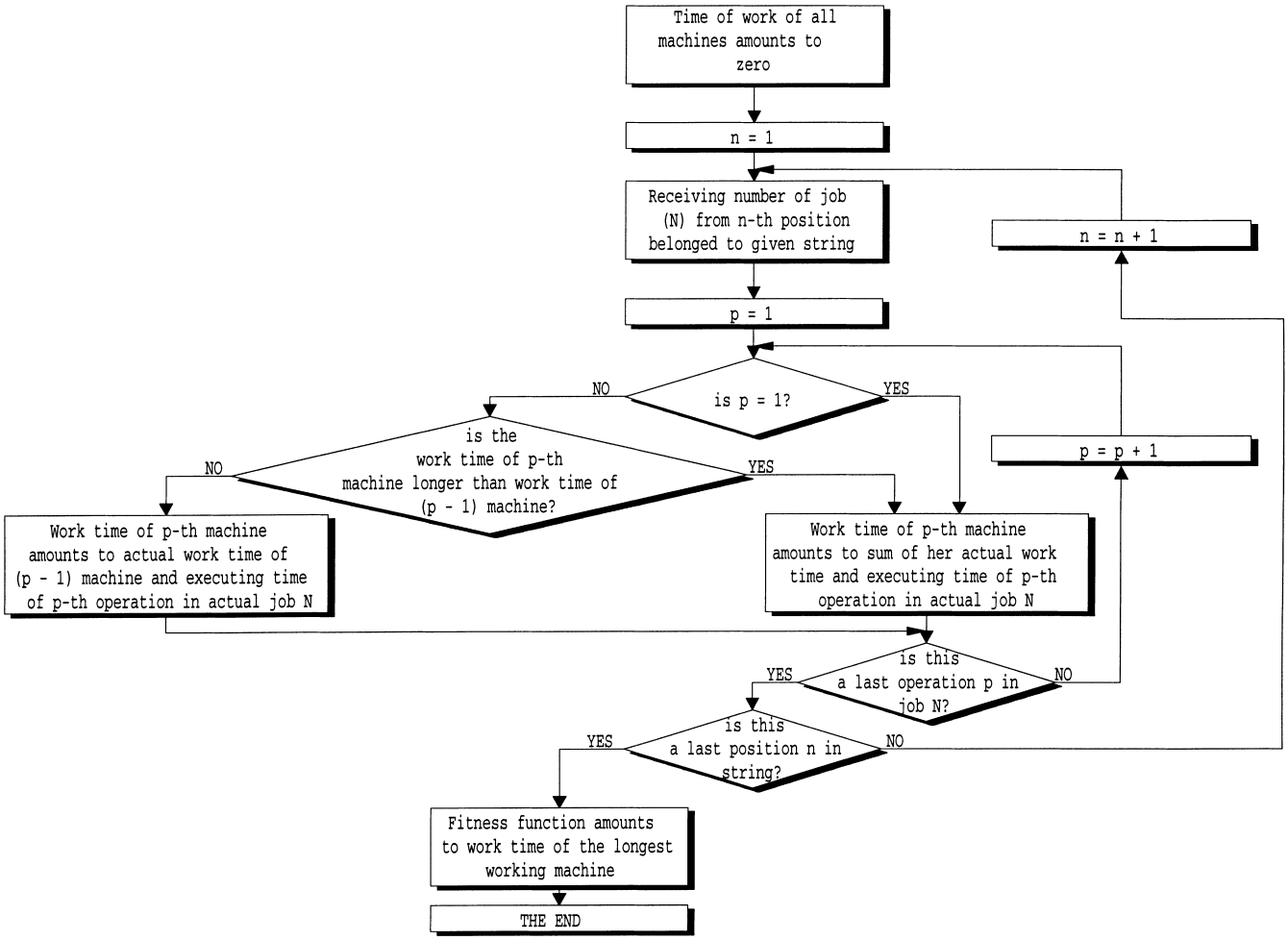

Fig. 5. Algorithm of fitness function calculating for sequential working cell.

first parent. This type of crossover, unlike the simple crossover (based on simple fragments exchange) prevents formation of irregular strings (see Fig. 6), and its correct work was proved in [5].

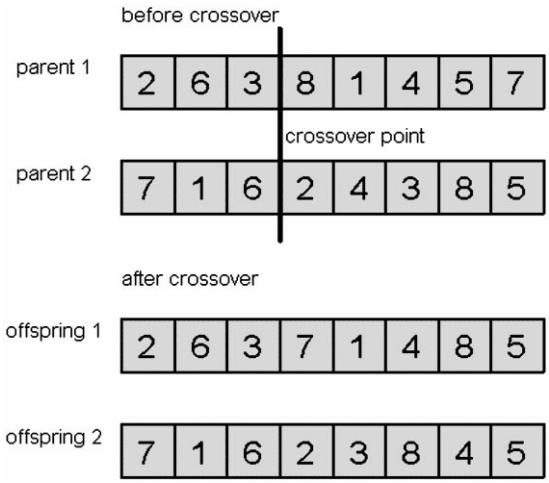

Fig. 6. Scheme of order crossover.

3.4. Mutation

The similar situation appears during mutation. For first case, if it is carried out in the first block then change of job's succession take place and if it is carried out in the second block then number of machine has been changing. For second case, it is not just a simple exchange of value on single position. If value on a given position has to undergo a mutation, then one more position will be automatically drawn out by lots and mutation will be based on the exchange of places of these two values.

4. Results of program's work

After work of genetic algorithm the Gantt chart is drawn. For optional working cell there are presented results of simple scheduling three jobs to two machines in this paper. The Gantt chart for this case is presented in Fig. 7. For sequential working cell the created program has been tested for nine jobs and every job contained five operations. That mean cell has been contained five machines. Times of

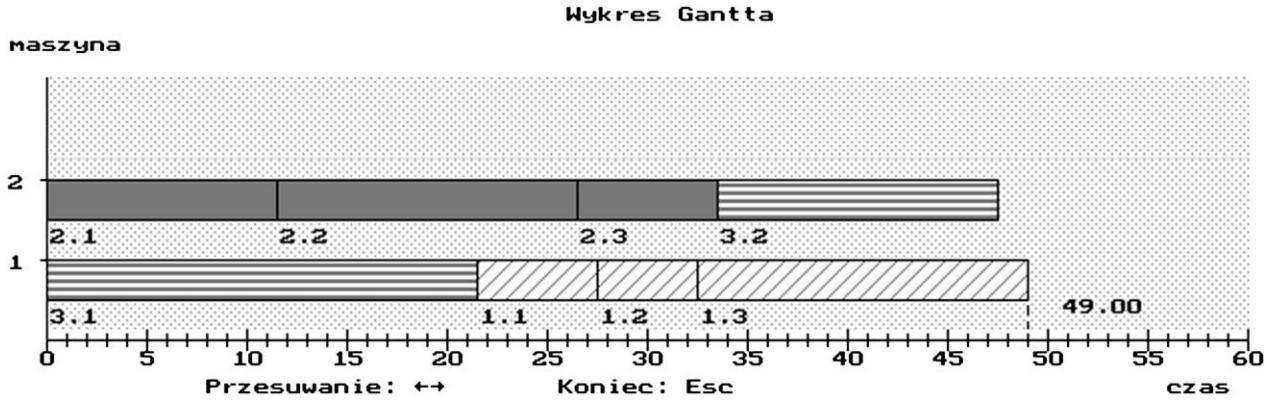

Fig. 7. Gantt chart for optional working cell.

Table 1
Operation's machining time for sequential working cell (in time units)

	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job 7	Job 8	Job 9
Operation 1	1	4	4	9	11	6	4	1	7
Operation 2	2	5	2	7	4	4	7	4	13
Operation 3	3	7	6	9	7	10	12	2	6
Operation 4	5	4	7	10	1	4	4	10	5
Operation 5	8	3	8	4	5	10	6	7	8

particular operations are compared in Table 1 and Gantt chart for this mode is presented in Fig. 8. Besides report files are generated for every generation. Example one of these files is presented in Fig. 9. Data gained from report files are shown in Fig. 10. These are: the shortest time of schedule realization obtained from fitness function of the best string, which has appeared, the shortest time of schedule realization obtained from fitness function of the best string in given generation and average time of schedule realization for given generation. As you can see, for sequential working cell genetic algorithm reached correct, near to optimal, solution very fast (in the course of 200 generations). The

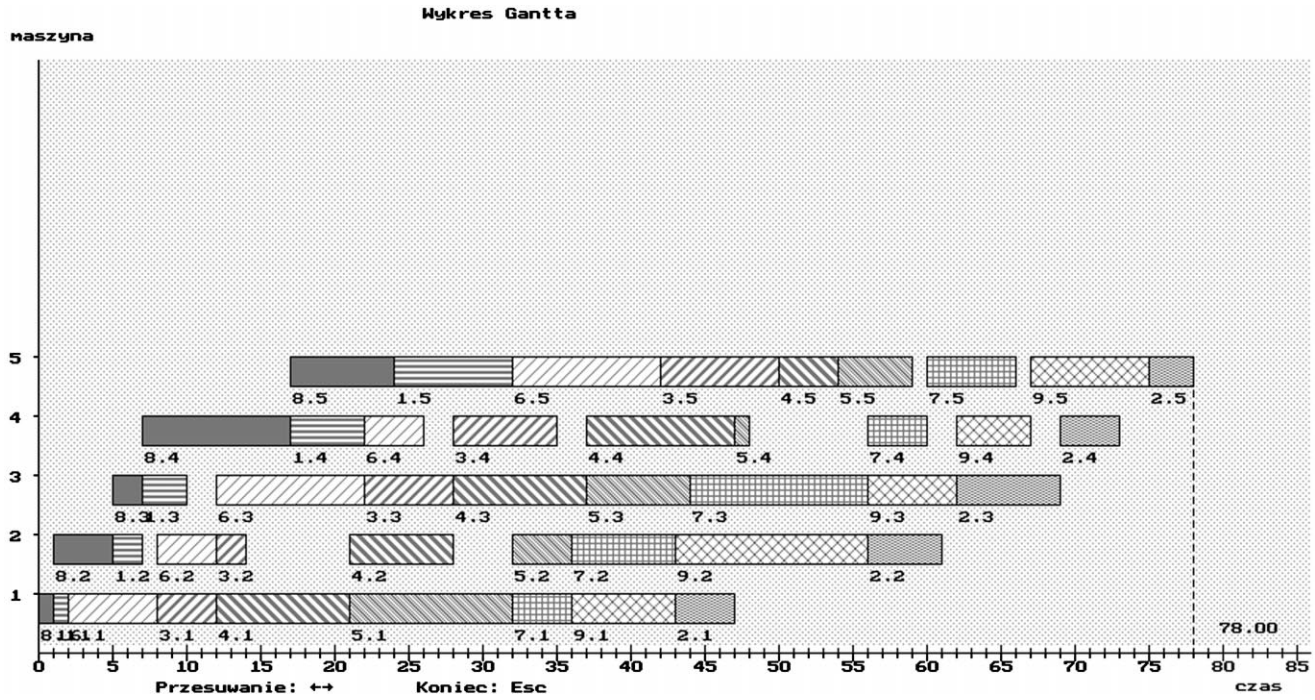

Fig. 8. Gantt chart for sequential working cell.

```
----------------------------------------
  1)   4  7  9  1  8  3  5  2  6    97.00
  2)   7  1  3  4  2  9  8  6  5    92.00
  3)   3  8  9  5  7  4  2  1  6    96.00
  4)   8  6  9  2  7  3  4  1  5    95.00
  5)   7  2  8  3  5  9  6  4  1    94.00
  6)   4  8  7  3  1  2  6  9  5   100.00
  7)   6  1  9  3  8  5  4  2  7    86.00
  8)   1  4  5  7  6  3  2  8  9    97.00
  9)   4  8  2  7  5  9  1  3  6   100.00
 10)   2  1  3  8  4  7  5  9  6    85.00
                    ...
 25)   1  6  4  3  9  2  8  5  7    86.00
 26)   4  3  8  7  5  2  1  6  9    99.00
 27)   1  8  3  2  6  9  5  7  4    84.00
 28)   4  2  5  8  9  7  6  3  1   104.00
 29)   3  4  7  8  1  5  6  2  9    96.00
 30)   3  1  7  6  2  8  4  9  5    87.00
----------------------------------------
Statystyka:
Najkrótszy zanotowany czas pracy:        84.00 jednostek
dla ciągu:   1  8  3  2  6  9  5  7  4
Najkrótszy czas pracy w tym pokoleniu:   84.00 jednostek
średni czas pracy w tym pokoleniu:       94.93 jednostek
----------------------------------------
```

Fig. 9. Fragment of report file for first generation.

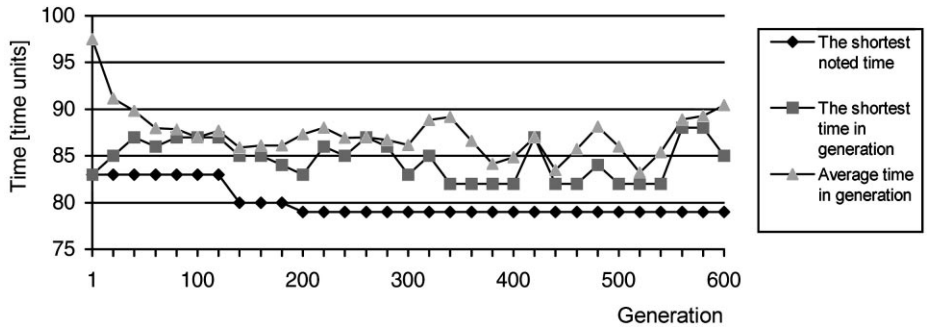

Fig. 10. Changes of fitness function value.

jobs have been put in order: 8, 1, 6, 3, 4, 5, 7, 9, 2 and this causes, that cell realizes orders in 78 time units.

5. Summary

Genetic algorithm used in the created program has generated correct, with regard to time, schedules, but we are not sure that the solution is optimal. Number of jobs and their operations have not had influence on quality of obtained results for sequential working cell. Gained schedules have been correct for all cases, that means strings assure right succession of operations. Applied structure of code string has assured good, but not the best, efficacy of creation and propagation of schemes, which assured high adjustment of strings (see [3,4]).

References

[1] H. Tamaki, M. Ochi, M. Araki, Application of genetic-based machine learning to production scheduling, in: Proceedings of the Japan/USA Symposium on Flexible Automation, Vol. 2, ASME 1996, pp. 1221–1224.

[2] E. Goldberg, Algorytmy genetyczne i ich zastosowania, Warszawa, WNT, 1995.

[3] K. Terano, Y. Yao, K. Okamoto, Y. Hashimoto, I. Nishikawa, T. Watanabe, H. Tokumaru, Application of simulated annealing method and genetic algorithm to scheduling problems in plastic injection molding, in: Proceedings of the Japan/USA Symposium on Flexible Automation, ASME 1996, Vol. 2, pp. 1225–1228.

[4] Z. Michalewicz, Algorytmy genetyczne + struktury danych = programy ewolucyjne, Warszawa, WNT, 1996.

[5] E. Biegel, J.J. Darven, Genetic algorithm and job shop scheduling, Comp. Ind. Eng. 19 (1/4) (1990) 81–91.