APPLICATION OF EVOLUTIONARY ALGORITHMS FOR OPTIMIZATION OF PRODUCTION SCHEDULE IN FOUNDRY

Kubiński W.[1], Krawczyk K.[2]
[1] Faculty of Management
[2] Faculty of Metallurgy and Industrial Informatics, University of Science and Technology, Al. Mickiewicza 30, 30-059 Cracow
E-mail: kubinski@uci.agh.edu.pl, E-mail: christobar@poczta.onet.pl

APLIKÁCIA EVULUČNÝCH ALGORITMOV PRE OPTIMALIZÁCIU VÝROBNÝCH PLÁNOV V ZLIVÁRENSTVE

Kubiński W.[1], Krawczyk K.[2]
[1] Fakulta manažmentu
[2] Fakulta metalurgie a priemyselnej informatiky, Univerzita vedy a technológie, Krakow – Poľsko
E-mail: kubinski@uci.agh.edu.pl, E-mail: christobar@poczta.onet.pl

Abstrakt

Zámerom príspevku je poukázať na možnosti, ktoré poskytuje evolučný algoritmus pre riešenie problémov spojených s operatívnym plánovaním výroby. Článok ukazuje princípy operačného evolučného algoritmu, napr. pre zlieváreň. Zlieváreň s automatickým zlievárenským formovacím strojom bola analyzovaná. Pre predmetnú zlieváreň bol napísaný program, ktorý využíva evolučný algoritmus pre operatívne plánovanie výroby. Základným problémom je hľadanie sekvencie, v ktorej spoločnosť môže akceptovať objednávku s tým, že všetci zákazníci sa vybavia načas. Základnou funkciou je teda minimalizácia zdržania. Uprednostnili sme objednávky tak, že zákazníci ktorí sú veľmi dôležití pre spoločnosť, (napr. stáli zákazníci, veľkí zákazníci) sú vybavení načas. Hlavné obmedzenia sú: kapacita automatického formovacieho stroja a pecná kapacita. Pred vlastnou optimalizáciou sa musel vykonať zber prevádzkových údajov zameraný na použitie zliatiny, ktorý bol zaradený medzi priority. Pec musí byť zaplnená iba s jedným druhom zliatiny a preto niektoré objednávky zákazníkov sa musia deliť. To znamená, že liatie niektorej objednávky bude vykonané v dvoch dávkach. Tiež musí dôjsť k naplneniu kapacity pece, lebo inak by jej prevádzka bola neekonomická. Na koniec ukazujeme niektoré výsledky, analýzy a závery vyplývajúce z týchto údajov. Evolučný algoritmus v porovnaní s inými metódami optimalizácie, ktorými sa proces simuluje, dáva o 15% lepšie výsledky. Preto bol použitý evolučný algoritmus riešiaci iné problémy vyskytujúce sa v každej prevádzke, zvlášť nasledovné problémy: optimalizácia v rozhodovaní, plánovanie investičnej stratégie, minimalizácia výdavkov, pracovné postupy atď.

Abstract

The goal of this article is to show one of many possibilities which give evolutionary algorithms in solving problems which are connected with operational planning for production. Article shows principle of operation evolutionary algorithms for instance of foundry. Therefore foundry with automatic foundry moulding machine was analyzed and for this foundry we wrote a program that use evolutionary algorithms to operational planning for production .In short problem depends on finding sequence in which company should produce accepted orders so that

all orders were finished on time Thus the fitness function is minimizing the delays. We can favor some orders so that orders which are for company very important (for example orders from regular customers, very large orders, etc) will be made certainly on time. The main constraints are: capacity of automatic moulding machine and furnace capacity However before we start optimization we have to arrange received orders according to due dates, kind of alloy from which it has to be made and priorities. Furnace has to be filled only with one kind of alloy therefore some customer's orders have to be divided. This means that castings from some order will be made in two batches .Furnace should also be entirely filled because otherwise production will be unprofitable. At the end we show some results, analyses and we draw a conclusion from this data. Evolutionary algorithm in comparison with another method of optimization that is simulated annealing gives much better results - around 15% better results. Therefore we should use evolutionary algorithms to solve other problems occurring in each enterprise especially such problems as: optimization in making decision, planning investment's strategy, cost's minimization, work scheduling etc.

Keywords: evolutionary algorithms, optimization of production schedule, operational planning in foundry

1. Introduction

In recent years more often we are looking for solution of different problems by watching and analyzing rules which take place in nature – for instance it may be evolutionary algorithms. They derive from Charles Darwin's theory of evolution. This theory is connected with natural selection which says that only strong and better fitted organisms can survive in given environment. In natural selection we can distinguish: inheritance features which are passed from generation to generation; genetic variability like crossing genetic information and gene mutation. It leads to situation when next generations are better fitted to changes which take places in their environment. The idea of evolutionary algorithms was presented by John Holland at the turn of the sixties and seventies last century. Computer program in difference to natural evolution which follows very slowly creates and assesses thousands generations just in a few seconds. Evolutionary algorithms in spite of random elements effectively generate solutions which are better fitted to the environment – objective function.

2. How classic evolutionary algorithm works?

Classic evolutionary algorithm (Figure 1) called also elementary or basic genetic algorithm consists of:

a) Genetic representation of potential solutions.
b) Selection initial population.
c) Fitness function which assess adjustment specimens in population.
d) Selection.
e) Basic genetic operators.
f) Some constant parameters - for instance: size of population, probability of using genetic operators etc.
g) End conditions.

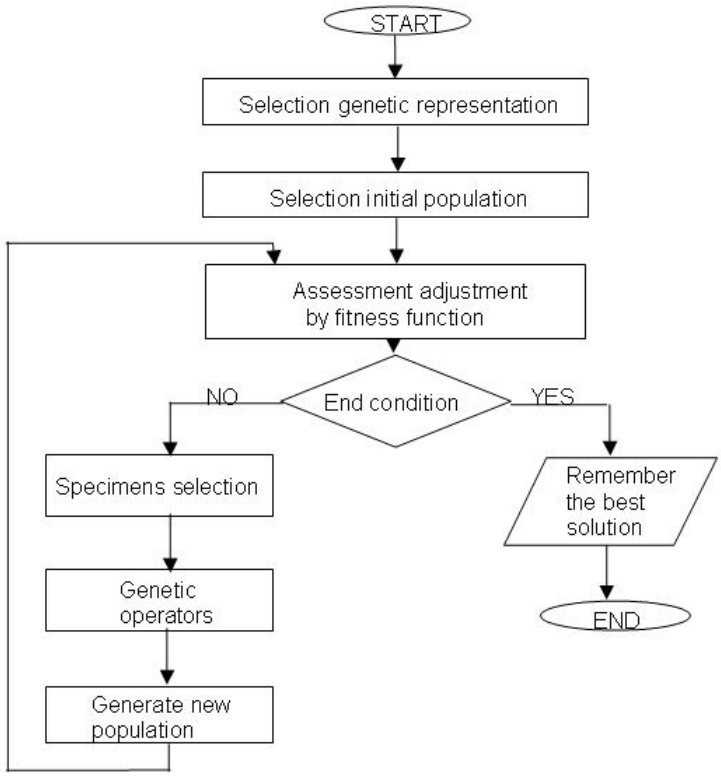

Fig.1 Classic evolutionary algorithm – own study on the basis of literature [1,2]

a) When we create evolutionary algorithm we have to represent problem which we want to solve in the form of chromosome code. There is a lot of way of coding and we can choose any of them but we must remember that chromosome structure and genetic operators should meet constraints which are present in our problem. In our evolutionary algorithm, which optimizes production schedule in foundry, we applied vectorial representation consist of sequence of integers, which represents successive "casts". By "cast" we understand a single furnace filled with specified kind of alloy. The single solution consists of ordered integers which range depends on quantity of orders received from customers. From one furnace ("cast") we can make one or more customer's orders.

b) In literature we can find that initial population is created on a random basis. In practice if we know searched space we can use some deterministic algorithm in order to evolutionary algorithm could start calculations with better solutions. In our program initial population is created at random from set of all possible solutions - $n!$ where n is a quantity of "casts" in given period. Sampling prevents form selection the same "cast" twice so that each product is made only once and can't be omitted.

c) Fitness function enables to choose better fitted specimens to their reproduction. On the one hand fitness function should select the best specimens, but on the other hand it can't lead to domination one or more specimens in population. Such domination can eliminate other specimens which have a good genetic material. Therefore can be find only good solution (local optimum) but not the best solution.

Details about fitness function will be presented later. Generally all orders should be performed on time, and if it is impossible we should make the least delays. Fitness function contains also priorities what means that some orders may be more important than others.

d) Selection consists in choosing on the basis of fitness function specimens which take part in reproduction. In reproduction we create new generation in this way that specimens which are better fitted are more likely to be chosen to create new generation, but worse specimens have also some chance to create new generation. The most known methods of selection are: roulette wheel selection and tournament selection which we applied in our program. We draw form population two subgroups consist of five specimens. From each subgroup is chosen the best fitted specimen which moves on to the next operation: crossing and mutation.

e) The basic genetic operators are:

- **Crossover**. It consists in exchanging features between two specimens. Couples of specimens which will be crossing are chosen at random according to crossover probability (p_c from 0,5 to 1). There are many ways how to perform crossover but in our program we applied order crossover (OX) [3]. In this method the most important is task order not their position.
- **Mutation.** This operator is not as important as crossover but sometimes mutation helps regain good genetic material. Mutation probability is low (p_m from 0 to 0,1), that is consistent with natural world where mutation also occur rarely. We apply mutation which select two tasks and exchange their position.

By means of genetic operators we create new generation. We calculate fitness function for each chromosome and check end condition. If end condition is fulfilled we take the best solution found so far. If the end condition is not fulfilled we make selection but we replace old generation by new one.

f) The most important constant parameters in evolutionary algorithm are: size of population and probability of using genetic operators i.e. crossover and mutation. Size of population has influence on variety of population and selection. If size of population is too small then may be found only local optimum. On the other hand, when size of population is too big, time to find solution can be lengthen. The main sources of knowledge about genetic operators are experiences, therefore to find optimal value we have to make a lot of tests. In our program population consist of 25 specimens and the first specimen is the same in all populations. That specimen is ordered from the first to the last “cast” and is used in calculation for other specimens. Such size of population is not too big and not too small, so it seems to be correct. Crossover and mutation probability can be chosen by program user. According to literature [3] and on the basis of our results we can assume that the best crossover probability is $p_c = 0,65$, and the best mutation probability is $p_m = 0,01$.

g) End conditions depend mostly on the problem that we try to solve. The simplest way to end evolutionary algorithm is by determination number of iteration or determination some time after which algorithm ends calculations. Sometimes it is difficult to say how many iteration should be carried out or how long the program should work to find satisfactory solution. Therefore there are methods which allow end evolutionary algorithm when chance to improve solution is low. In our program we determine number of iteration after which program present the best solution find so far. On the basis of experiences 1000 iteration is sufficient to find good solution.

3. Advantages and disadvantages of evolutionary algorithms

Table 1 Advantages and disadvantages of evolutionary algorithms.

Advantages	Disadvantages
• Effective technique with wide possibility in application. • Obtained solutions are reliable. • Possibility to search multidimensional and complex solution spaces. • Relatively easy to create and implement. • There is no constraint as regards to form of fitness function. • Detailed knowledge about the problem which we want to solve is not need. • You can optimize multi-criterion function. • You can modify the form of algorithm. • Many information is processed – evolutionary algorithms are parallel. • You can obtain partial solutions and stop calculation whenever you want. • Evolutionary algorithms easy interact with other techniques.	• Heuristics character does not give certainty to find optimum. • Calculation takes a lot of time but technical progress mitigates this effect. • Frequently ineffective at the end of calculations. • Do not have theoretical basis. • It is difficult to determine correct values of parameters.

The main advantages and disadvantages presents Table 1.

4. Production process in analyzing foundry

Evolutionary algorithm optimized production plan in foundry, which produces grey cast iron and spheroid cast iron. That casts irons are used to produce:

- industrial fittings such as: bodies, valves, flanges, bolts and screws;
- parties of machines and appliances;
- subassemblies for motorization;
- elements for railway tractions, etc.

Foundry possess certificate ISO 9002, what means that all produced assortments have a high quality. Foundry consists of two cupola furnaces, automatic molding machine called Disamatic, and devices used to clean casts. When orders from specific period of time (few days or week) are collected then created is production schedule. Made is also casting project which defines its shapes and kind of alloy used in production, as well time of each activities and all technical documentation. Some castings need cores in order to achieve a specific shape. Cores are made from special resin which is melted in high temperature. In our program we do not take into consideration time which is need to made cores because cores are made in other part of foundry – but cores must be delivered before production will start. In analyzing foundry cupolas have 6000 kg capacity, but one by one is filled only 4800 kg. The rest is left to accelerate melting of consecutive alloys. During one day (24 hours) furnace is filled four times. For each time may be used different alloys which have different chemical composition (contents of carbon, iron, silicon etc.) and different properties (strength, hardness etc.). When alloy is ready it is casting to ladle which has capacity of 600 kg and then individual moulds are filled. Moulding takes place in automatic moulding machine. Average weight of cast is 18 kg, but the range of weight is from 8 to 27 kg. In one day we can formed around 650 – 670 casts. Capacity of automatic moulding machine is around 12 000 kg per day. Then casts solidify and are fettling from moulds. The last stage of production is mechanical cleaning and quality inspection. Casts

which come up to all technological requirements are stored in warehouse. There is used only one furnace and the other is used only when the first is repaired. It means that capacity of automatic mouldnig machine is considerably lower then capacity of furnaces. Fettling shop capacity is also sufficient besides it can be easily increased by adding another worker therefore bottleneck in production process is automatic mouldnig machine.

5. Mathematical notation – operational planning production in foundry by means of evolutionary algorithms.

In analyzing foundry problem which is solved by means of evolutionary algorithm depends on finding sequence in which company should produce accepted orders so that all orders were finished on time. Fitness function is minimization delays. The main constraints are: furnace capacity and automatic moulding machine capacity. Program which we made enables to favour some orders which are very important for us.

Operating production planning make up of two parts and consist in:

- Joining or dividing customers' orders in order to fill furnace only with one kind of alloy – furnace should be always entirely filled because otherwise production will be unprofitable (Figure 2).
- Organizing production schedule so that all orders were made on time especially that orders which have priorities. Priorities define which orders should be made before other orders and they can take into consideration such criterions like: permanent customers, customers which order a lot of products, etc. To achieve optimal sequences in production schedule evolutionary algorithm was applied.

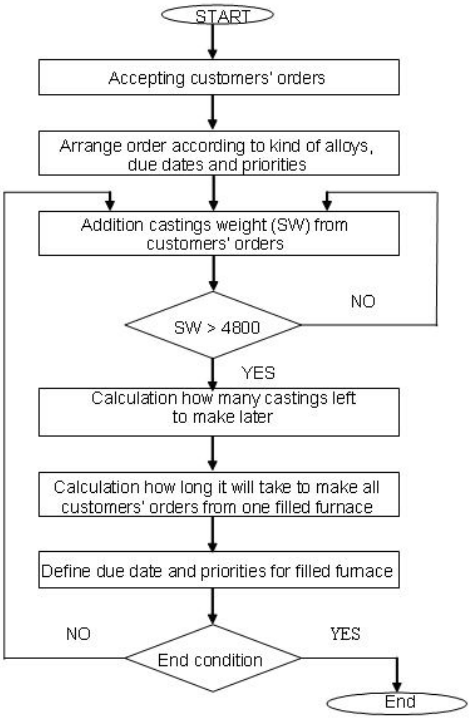

Fig.2 Dividing customers' orders in order to fill furnace only with one kind of alloy

Before we start optimization production schedule we have to choose period from which we start optimization and establish some fixed data such as: date when we start production, capacity of automatic moulding machine, furnace capacity, number of employers in fettling shop. We also have to define parameters for evolutionary algorithms i.e. crossover and mutation probability and number of iteration.

All orders which we received we arrange according to due dates – from the earliest to the latest, kind of alloy from which it has to be made and priorities – from the most to the less important orders. In the first place we take that orders which have the earliest due date and the highest priorities. When the furnace will be filled that is when weight of castings coming form one or more customers' orders exceed 4800 kg, then the last order will be made in parts.

Some castings from that order will be made from alloy which is in furnace presently and the rest castings from that order have to wait until furnace will be filled with the same kind of alloy. In such a situation we have a problem to define priorities and time which we need to make orders coming from alloy which is in furnace. So we treat one filled furnace with define kind of alloy as one "order" ("cast"). In this way we have to choose the earliest due date from customers' orders which will be made from one furnace, and priorities are counted as average. When customers' orders do not fill furnace with one kind of alloy then we check customers' orders form another kind of alloy. This is the end condition. Finally we receive number of "orders" – filled furnaces with due dates and priorities which sequence we will be optimized by means of evolutionary algorithms. Way how we divide customers' orders is presented on Figure 2.

5.1 Denotations in mathematical model.

In mathematical model we assume following symbols:

- Period of time when we receive customers' orders – t (it can be week, month etc.).
- Kinds of alloys - $j = \{150, 200, 250, 400, 450, 500\}$.
- Number of orders accepted in period t – o. Each order o is made with define kind of alloy j and it is single casting $o = 1,...,O$.
- Spoilage for each casting – b_0 *[kg]*.
- Alloy used to make single casting g_0 *[kg]*.
- Time when customers want to have ready products (due date) – d_0.
- Quantity castings which have to be made from order o - k_o *[piece]*.
- Time which foundry has to have in order to made individual order – p_0 *[day]:*

$$p_o = (f_o + l_o)* k_o / 60 / 24$$

where: f_o – time needed to mould individual casting [minutes],
l_o – time needed to finish individual casting [minutes].

- Daily capacity of automatic moulding machine DISAMATIC - $F_t = 12000$ *[kg per day]*.
- Priorities ascribe to each order – w_0. It was not previously in foundry and we insert it according to literature [8]. Priorities ascribe each order number from 1 to 5 and point at that orders which are really important for us. If order has number 5 it means that this order will be made first. Priorities can take into consideration one or more criterions – for example keep customers' due date with maximization production capacity. In analyzing foundry the most important criterion is keeping due date for strategic

customers – regular customers, customers that order great deal of casts, etc. Priorities are ascribed by worker that accepts orders. They help to establish in which sequence orders should be made. In our program priorities were chosen at random.

- Time when we start production according to received orders in period t - S_t. It is equal to time when we finish production in period $(t-1)$. Exception can be overhauls and other random accidents. We start production process when alloy is ready in furnace.
- Furnace filled with define kind of alloy j is $i = 1, ..., N$. From one furnace we can make a lot of orders o. If order is very big (over 9600 kg of alloy) then furnace can be filled many times with the same kind of alloy but it will be treated as one furnace i. When orders' weigh less then 9600 kg then we connect orders with the same kind of alloy to filled furnace. Evolutionary algorithm looking for the best sequence in which furnace has to be filled by different kinds of alloy in order to meet customers due date.
- The earliest customers' due date from that orders o which will be made from furnace i is $d_i = min\ \{d_o,...,d_m\}$ where m – quantity of orders make from furnace i.
- Total time needed to make orders o from furnace i is p_i:

$$p_i = \sum p_o$$

When orders are divided then we take only part of time which is needed to make them.

- Average priorities from orders o make from furnace i is w_i and it is round off to integer:

$$w_i = \frac{\sum w_o}{m}$$

- Time (week) when finish products o from furnace i will be pass on to warehouse c_i.

5.2 Fitness function

Time to make all furnaces i = 1, …,N from orders accept in define period t is:

$$c_i = S_t + p_i$$

Delays for furnaces $i = 1,...N$ is:

$$T_i = c_i - d_i, \text{ when } T_i < 0 \text{ then } T_i = 0$$

We must find sequence in which furnaces will be filled with alloys of different kinds so that delays will be minimized. We also have to take into account priorities so fitness function has the form:

$$F = \sum_{i=1}^{N} w_i * T_i \rightarrow MIN$$

5.3 Constraints

We take following constraints:

1. Constraint connected with capacity of automatic moulding machine:

$$\sum_{o=1}^{O}(k_o * g_o + k_o * g_o * b_o) \le F_t * \sum_{o=1}^{O} f_o * k_o$$

Left side of this constraint shows us how much alloy [kg] we need to make all orders from analyzing period of time. Right side shows capacity of automatic moulding machine. When the automatic moulding machine could not mould all customers' orders then we turn down the latest orders until this constraint will be fulfill. Rejected orders will be made in next period, when more customers' orders we will receive.

2. Make orders with the highest priorities on time:

$S_t + p_i \le d_i$, where $w_i = \{4,5\}$

3. We start production after end production in previous period:

$$S_{(t+1)} \ge S_t + \sum_{i=1}^{N} p_i$$

4. Quantity of castings must be larger than zero: $k_o \ge 0$.

5. We can not make order doubly. It is similar to salesman's problem where we can not visit the same city more then one time. This constraint is fulfilled by taking to evolutionary algorithm only acceptable solutions.

6. Example of receiving solutions

Table 2 Received solutions by means of evolutionary algorithms.

Sequence *i* which gives the best solution	Time *pi*	Priorities *wi*	Customers' due date *di*	Time when we finish production *ci*	Number of furnaces	Kind of alloy *j*
11	0,436	3	40	40	1	500
3	2,074	3	40	40	1	200
10	0,742	4	39	40	2	500
5	0,296	2	40	40	1	400
6	1,029	3	40	41	1	400
4	1,109	3	41	41	1	250
2	3,844	3	40	41	2	200
9	3,553	3	45	42	2	400
1	0,838	3	42	42	1	150
12	1,385	3	43	42	2	500
7	0,292	3	44	42	1	400
8	0,44	2	44	42	1	400

We can present received solutions in table (look at Table 2). In first column there is sequence of furnaces *i* filled with define kind of alloy *j* which gives us minimal delays in production. Second column shows time which is need to make all orders from alloy in each furnace. Third column shows priorities ascribe to each furnace. Fourth column presents time

when customers want to collect finished products and fifth column shows time when customers' products are really finished. Number of furnaces means how many times furnace has to be filled the same kind of alloy in case of order was very big (more than 9600 kg). Thanks to received data we can determine time when production of individual orders will be finished. Way which evolutionary algorithm has to make in order to find the best solution is presented in Figure 3. The best solution is found very quickly – in 266-th generation. If we increase number of iterations (to 10000) we receive the same solution, so we can assume that this is the best possible solution.

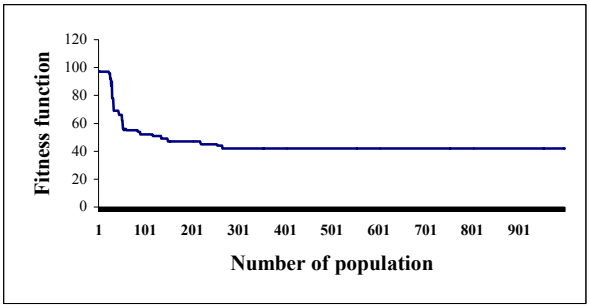

Fig.3 Way which evolutionary algorithm has to make in order to find the best solution.

Table 3 The best solutions for 1000 iterations which were made in 25 tests by means of evolutionary algorithms and simulated annealing.

Test	Fitness function		
	Evolutionary algorithm	Simulated annealing	Due date
1	44	71	
2	42	58	
3	42	61	
4	42	57	
5	42	60	
6	42	65	
7	42	72	
8	42	63	
9	42	68	
10	42	55	
11	42	57	
12	42	64	
13	42	60	83
14	42	59	
15	42	61	
16	42	66	
17	42	68	
18	42	70	
19	42	56	
20	42	62	
21	42	56	
22	42	57	
23	42	63	
24	42	67	
25	42	58	
Minimum	42	55	83
Average	42,08	62,28	
Standard Deviation	0,39	4,86	
Average deviation from average value	0,1536	4,15	

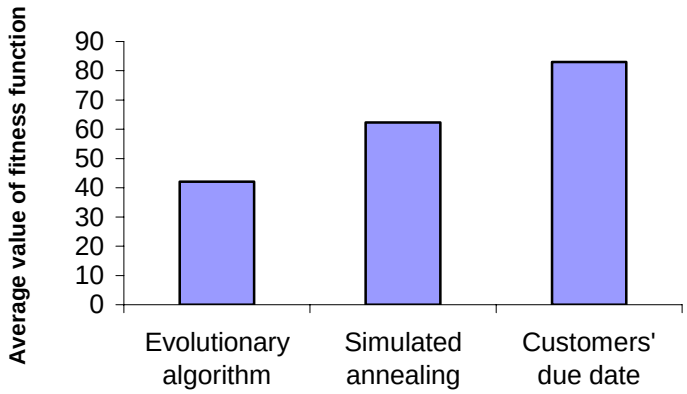

Fig.4 Graphical comparison received solutions.

In order to show effectiveness of evolutionary algorithm we made optimization of production schedule for orders accepted during 2 weeks – 118 different orders, which give us 34 filled furnaces. Received solutions we compare with other methods of optimization that is simulated annealing and with arrangement furnaces of different alloys according to the earliest customers' due date. The best solutions for 1000 iterations which were made in tests are presented in Table 3. Graphical comparison of received solutions is presented in Figure 4. On the basis of received solutions we can compare average values of fitness function between two methods: evolutionary algorithm and simulated annealing. We consider two hypotheses:

H_0: Average value of fitness function received by evolutionary algorithm is equal to average value of fitness function received by simulated annealing.

H_1: Average value of fitness function received by evolutionary algorithm is different from average value of fitness function received by simulated annealing.

We established significance level $\alpha = 0,05$, for which critical value is $z_{\alpha/2} = z_{0,025} = 1,96$. We can count value of statistic on the basis of sample from formula [9]:

$$z_{obl} = \frac{\overline{x}_1 - \overline{x}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} = 20,7$$

Since $z_{\alpha/2} < z_{obl}$ so we reject H_0 hypothesis. Average value of fitness function received by means of evolutionary algorithm is different from average value of fitness function received by means of simulated annealing. We can see that in Figure 4. Results which we received by means of evolutionary algorithms are much better – around 15-20 % than results received by other methods.

Despite the fact that in reality we can not eliminate all delays, we can limit their amount. We can also concentrate on that orders which are really important for us. To achieve better results we should take to optimization single orders but this is impossible because furnace has to be full. Otherwise production will be unprofitable.

7. Conclusions

On the basis of results which we received from our program we can draw conclusions:

1. We can very easily and cheap improve efficiency in planning production schedule in foundry. Program which we made gives answers on questions:

- Which orders we should accept to production, which castings and how many?
- When we should produce and in which sequence, according to priorities?
- When we finish production?

2. We can find many others problems which we can solve by means of evolutionary algorithms or other methods of artificial intelligence. In enterprises we can use evolutionary algorithms in such problems like: optimization in making decisions, planning investment's strategy, cost's minimization, work scheduling, etc.

3. In foundries and steel mills evolutionary algorithms can be used to solve difficult and complex tasks like: classic optimization problem called mix problem. This problem consists in defining quantity of each material which is needed in production so that costs of production were minimal. Another problem which can be solved by means of evolutionary algorithms is defining optimal quantity of products which can be produced on different moulding machines or defining whole production schedule that is locating in the time individual operations.

4. Although evolutionary algorithms do not solve all problems which appear in enterprise they should be used because they make these problems easier.

Literature

[1] Krawczyk K.: Optymalizacja planu produkcyjnego dla odlewni za pomocą algorytmów genetycznych. Praca magisterska. Wydział Zarządzania AGH. Kraków 2003 (nie publikowana)
[2] Rutkowska D. i inni: Sieci neuronowe, algorytmy genetyczne i systemy rozmyte. Wydawnictwo Naukowe PWN, Warszawa 1999.
[3] Michalewicz Z.: Algorytmy genetyczne + Struktura danych = Programy ewolucyjne. Wydawnictwo Naukowo-Techniczne, Warszawa 1999.
[4] Biethahn J., Nissen V.: Evolutionary Algorithms in Management Applications. Sprinter-Verlag, Berlin, Heidelberg 1995.
[5] Goldberg D.E.: Algorytmy genetyczne i ich zastosowanie. Wydawnictwo Naukowo-Techniczne, Warszawa 1998.
[6] Cytowski J.: Algorytmy genetyczne: podstawy i zastosowania. PLJ, Warszawa 1996.
[7] http://www.odlewniepolskie.pl/.
[8] European Journal of Operational Research 139 (2002) 490 - 500.
[9] Żyżyński J.: Podstawy metod statystycznych dla zarządzania. Wydawnictwo Naukowe Wydziału Zarządzania Uniwersytetu Warszawskiego, Warszawa1999.
[10] http://cs.felk.cvut.cz/~xobitko/ga/