Lot-Sizing in a Foundry Using Genetic Algorithm and Repair Functions

Jerzy Duda

AGH University of Science and Technology,
Faculty of Management, Dept. of Applied Computer Science,
ul. Gramatyka 10, 30-067 Kraków, Poland
jduda@zarz.AGH.edu.pl

Abstract. The paper presents a study of genetic algorithms applied to a lot-sizing problem, which has been formulated for an operational production planning in a foundry. Three variants of genetic algorithm are considered, each of them using special crossover and mutation operators as well as repair functions. The real size test problems, based on the data taken from the production control system, are presented for assessment of the proposed algorithms. The obtained results show that the genetic algorithm with two repair functions can generate good suboptimal solutions in the time, which can be acceptable from the decision maker point of view.

1 Introduction

The lot-sizing models allow to determine the production quantities at all production planning levels. The reviews of them can be found in [1], [3] or in [5]. The problem presented in this paper comes from a real production environment in a foundry and focuses on a short-term production planning.

The considered foundry is a typical foundry, which produces iron castings and uses hand-operated moulding machines. Among the shops existing in such a foundry two are the most important regarding operational production planning: a melting shop in which hot iron is prepared and a moulding shop where the moulds are made. Pouring and moulding operations must be coordinated, as melted iron cannot wait too long to be poured into the moulds and the space for the moulds waiting for pouring is limited.

Thus the main weekly task for the planners is to prepare a moulding plan together with a pouring schedule for the furnaces. While building those plans many technological and organizational constraints must be taken into consideration. The most significant are:

- capacities of furnaces and moulding machines,
- the number, desired delivery date and cast iron grade of ordered castings,
- the number of different castings, which can be produced during one shift (setup times are included in moulding times),
- the number of flasks of various size available during a working shift.

G.R. Raidl and J. Gottlieb (Eds.): EvoCOP 2005, LNCS 3448, pp. 101 – 111, 2005.

2 Optimisation Model

A mathematical model is built around the classical discrete capacitated lot-sizing problem with single level and multi item production. Presented model can be classified as a small bucket model, because only limited number of different items can be produced during one period of time.

Only few models dedicated strictly to planning in iron foundries can be found in literature. Van Voorhis et al. [7] provide a description of they work for Steel Foundries Society of America to develop software for generating pouring schedules. The objective function proposed by them is the sum of the non-utilization costs of heats and moulding lines, the costs of putting production for a given order into a particular lot, inventory costs and the penalty value for lateness. The constraints reflect all the capacity limitations as well as metallurgical ones. They used two stage heuristic, which solves an LP problem in the first stage and an IP problem in its second stage.

A model which is closer to the classical lot-sizing model can be found in dos Santos-Meza et al. paper [6]. The authors present a lot-sizing problem in a foundry with automated moulding machines. They use a minimization of item production costs as the only objective function and apply a relaxation method for the problems, which are then solved using CPLEX 4.0 library.

The objective function used in the model presented herein is similar to the objective function proposed by Van Voorhis et al. Instead of the non-utilization costs of furnaces and moulding lines, which may not always be estimated precisely, the combined utilization value is used directly. Also the inventory costs are omitted, as they are more or less fixed for the considered foundry (to some, but high enough limit).

The following symbols are used:

Decision variables:

x_{ijtz} – number of castings planned for order i to be manufactured on machine j during day t and shift z,

v_{htz} – number of heats of grade h during day t and shift z,

Data:

τ – week for which the plan is created,

k – number of working days in a week,

l – number of machine type,

m_j – number of working shifts for machines type j,

n_j – number of active orders for machines type j,

C_P – daily furnaces melting capacity [kg],

W – weight of single heat [kg],

C_{Fj} – capacity of moulding machines type j during a working shift [minutes],

w_{ij} – total iron weight needed to produce single i casting [kg],

a_{ij} –time of making a mould for casting i on machine j [minutes],

d_{ij} – ordered number of castings of type i to be produced on machine j,

γ – number of iron grades,

g_{ij} – iron grade for casting i, $g_{ij} \in \{1,...,\gamma\}$,

ω – number of flask types,

S_o – flask number of type o available during a working shift,

q_{ij} – flask type in which a mould for casting i is prepared, $q_{ij} \in \{1,...,\omega\}$,
κ_j – number of different castings which can be produced on machine type j during one working shift,
δ_{ij} – due week for castings of type i to be produced on machine j,

Maximize:

$$\sum_{j=1}^{l}\sum_{i=1}^{n_j}\sum_{t=1}^{k}\sum_{z=1}^{m_j}\left(\frac{x_{ijtz}w_{ij}}{kC_P}+\frac{x_{ijtz}a_{ij}}{km_jC_{Fj}}\right)-\sum_{j=1}^{l}\sum_{i=1}^{n_j}\left((d_{ij}-\sum_{t=1}^{k}\sum_{z=1}^{m_j}x_{ijtz})(\tau-\delta_{ij})(\tau<\delta_{ij})\right)/1000 \quad (1)$$

Subject to:

$$\sum_{h=1}^{\gamma} v_{hzt}W \le C_P, \quad t=1,...,k, \quad z=1,...,m_j \quad (2)$$

$$\sum_{i=1}^{n_j} x_{ijtz}a_{ij} \le C_{Fj}, \quad j=1,...,l, \quad t=1,...,k, \quad z=1,...,m_j \quad (3)$$

$$\sum_{t=1}^{k}\sum_{z=1}^{m_j} x_{ijtz} \le d_{ij}, \quad j=1,...,l, \quad i=1,...,n_j \quad (4)$$

$$\sum_{j=1}^{l}\sum_{i=1}^{n_j}(x_{ijtz}w_{ij}(g_{ij}=h)) \le v_{htz}W, \quad h=1,...,\gamma, t=1,...,k, z=1,...,m_j \quad (5)$$

$$\sum_{i=1}^{n_j}(x_{ijtz}>0) \le \kappa_j, \quad j=1,...,l, \quad t=1,...,k, \quad z=1,...,m_j \quad (6)$$

$$\sum_{j=1}^{l}\sum_{i=1}^{n_j}(x_{ijtz}(q_{ij}=o)) \le S_o, \quad o=1,...,\omega, \quad t=1,...,k, \quad z=1,...,m_j \quad (7)$$

The objective function (1) maximizes two elements. The first is the utilization level of furnaces and moulding machines, which are the main bottlenecks in the production system. Both utilization values are treated equally, however in reality the decision maker may use a weighted sum of them. The second element of the sum maximizes the penalty for the backlogging. The penalty function is proportional to the backlogged quantity and the number of overdue weeks. Those two criteria have been indicated directly by the planners in the considered foundry.

Constraints (2) and (3) are the capacity constraints for the furnaces and the moulding machines, respectively. Constraint (4) limits the production of a given casting to the quantity ordered by the customer. Constraint (5) limits the weight of the planned castings of a particular cast iron grade to the weight of the metal which is to be melted. Constraint (6) limits the number of different items which may be produced during one working shift. The last constraint (7) limits the flask availability.

The model is formulated as a discrete nonlinear problem. It was changed into an integer programming formulation by entering additional binary variables for the sake of the comparison between the results obtained by genetic algorithm and CPLEX solver.

3 Test Problems

Two test problems have been chosen from the data existing in the production control system, which is used in the described foundry.

The first test problem (*fixed1*) consists of 84 orders while the second one (*fixed2*) has 100 orders. There are four moulding lines in the considered factory, each consisting of two moulding machines, one for making a cope and one for making a drag (top and bottom part of a flask). However, there are only three types of moulding machines (denoted here as *A*, *B* and *C*). The type of a machine, which has to be used for making a mould for a particular casting is stated in a casting operation sheet.

Detailed orders specification for problems *fixed1* and *fixed2* are shown in Table 1 and Table 2, respectively.

Table 1. Detailed specification of *fixed1* problem

machine	order no.	flasks left to make	weight [kg]	moulding time [min]	iron grade	due week	machine	order no.	flasks left to make	weight [kg]	moulding time [min]	iron grade	due week	machine	order no.	flasks left to make	weight [kg]	moulding time [min]	iron grade	due week
A	1	282	61.2	30.5	4	-3	B	13	91	25.0	13.9	4	0	B	41	184	23.8	12.1	4	5
A	2	37	82.0	32.1	4	0	B	14	212	10.4	12.1	2	0	B	42	52	8.3	13.0	5	5
A	3	26	61.6	29.0	4	0	B	15	159	12.2	12.1	2	0	B	43	59	4.2	11.5	5	5
A	4	3	54.0	31.8	4	0	B	16	4	9.0	12.6	5	0	B	44	545	28.9	13.2	4	5
A	5	125	43.0	27.3	4	0	B	17	47	13.8	12.0	5	0	C	1	3	15.6	17.2	4	-3
A	6	226	65.0	32.6	4	1	B	18	16	12.4	13.9	5	0	C	2	257	18.2	15.1	4	-3
A	7	102	48.0	25.6	4	2	B	19	16	11.6	13.9	5	0	C	3	26	10.7	16.4	4	-2
A	8	16	30.4	25.4	4	3	B	20	16	11.0	13.9	5	0	C	4	25	10.8	18.1	4	-2
A	9	16	37.3	25.4	4	3	B	21	16	12.0	13.9	5	0	C	5	58	52.2	19.0	4	-2
A	10	22	34.7	30.2	5	3	B	22	133	12.0	12.3	5	1	C	6	196	29.6	17.7	4	-1
A	11	14	51.0	27.3	4	3	B	23	16	12.9	13.2	5	1	C	7	4	70.0	19.2	5	0
A	12	249	62.8	29.3	4	3	B	24	37	12.8	13.2	5	1	C	8	26	18.6	17.3	4	0
A	13	30	43.0	26.1	4	4	B	25	26	15.9	13.2	5	1	C	9	37	62.0	16.5	5	0
A	14	6	54.6	31.8	4	5	B	26	24	21.4	15.1	5	1	C	10	43	29.5	19.0	5	0
A	15	44	80.0	35.0	4	5	B	27	229	13.5	11.5	4	2	C	11	265	23.0	18.0	4	0
A	16	548	79.0	37.4	4	5	B	28	8	6.8	14.3	5	3	C	12	67	18.3	15.9	4	1
B	1	32	24.0	14.2	5	-3	B	29	16	6.0	14.3	5	3	C	13	36	6.9	2.9	5	2
B	2	35	24.0	14.2	5	-3	B	30	31	1.8	3.6	5	3	C	14	36	3.4	1.4	5	2
B	3	231	18.0	11.8	2	-3	B	31	6	10.4	13.9	5	3	C	15	83	5.8	2.4	5	2
B	4	424	9.3	11.6	4	-3	B	32	11	9.1	13.9	5	3	C	16	122	9.0	6.8	5	3
B	5	8	3.4	5.5	4	-2	B	33	16	10.8	13.9	5	3	C	17	96	23.6	16.7	5	3
B	6	31	15.6	13.9	2	-2	B	34	5	10.9	14.0	5	3	C	18	249	13.6	10.2	5	3
B	7	404	15.1	14.2	4	-2	B	35	5	13.1	14.0	5	3	C	19	22	21.7	18.6	5	3
B	8	538	15.1	14.2	4	-1	B	36	19	15.2	13.1	2	3	C	20	62	26.8	18.4	4	5
B	9	432	16.2	15.3	5	0	B	37	10	13.9	12.3	2	3	C	21	108	30.2	14.1	4	5
B	10	44	14.3	12.7	4	0	B	38	112	9.6	13.3	5	3	C	22	27	36.6	14.7	4	5
B	11	28	18.1	14.3	4	0	B	39	458	12.2	12.7	5	3	C	23	401	30.4	17.4	4	5
B	12	83	25.0	13.9	4	0	B	40	32	12.6	13.0	5	3	C	24	53	39.2	18.6	4	5

Table 2. Detailed specification of *fixed2* problem

machine	order no.	flasks left to make	weight [kg]	moulding time [min]	iron grade	due week	machine	order no.	flasks left to make	weight [kg]	moulding time [min]	iron grade	due week	machine	order no.	flasks left to make	weight [kg]	moulding time [min]	iron grade	due week
A	1	19	143	38.6	1	-2	B	3	45	9.9	13.0	4	-3	B	37	78	23.1	12.5	4	3
A	2	19	48.0	25.6	3	-2	B	4	220	15.1	14.2	3	-2	B	38	275	14.4	13.6	4	4
A	3	155	31.1	28.7	3	-2	B	5	66	19.3	12.0	4	-2	B	39	108	16.0	12.7	3	4
A	4	38	26.5	28.7	4	-2	B	6	212	31.0	14.7	3	-2	B	40	57	4.2	11.5	4	4
A	5	59	61.2	30.5	3	-2	B	7	52	16.0	12.7	3	-2	B	41	52	8.3	13.0	4	4
A	6	31	44.8	27.5	1	-1	B	8	135	18.0	13.2	3	-2	B	42	45	9.9	13.0	4	4
A	7	131	51.0	27.3	3	-1	B	9	39	23.0	19.6	2	-2	B	43	138	11.6	12.7	4	4
A	8	212	31.1	28.7	3	-1	B	10	33	28.9	13.2	3	-2	C	1	42	20.0	16.8	3	-3
A	9	52	32.6	28.8	1	-1	B	11	520	13.5	12.6	3	-2	C	2	156	10.6	14.7	4	-2
A	10	110	35.0	29.6	3	0	B	12	324	16.3	12.7	3	-2	C	3	35	41.6	20.2	1	-2
A	11	168	44.8	27.5	1	0	B	13	23	23.8	12.1	3	-1	C	4	28	13.2	15.8	3	-2
A	12	32	37.3	25.4	3	0	B	14	106	12.2	12.1	1	0	C	5	293	23.0	18.0	1	-2
A	13	44	73.0	29.5	3	0	B	15	106	10.4	12.1	1	0	C	6	305	27.5	18.0	1	-2
A	14	52	32.6	28.8	1	0	B	16	299	35.0	16.8	3	0	C	7	16	22.4	16.8	3	-1
A	15	109	51.0	27.3	3	1	B	17	17	13.1	14.0	4	0	C	8	20	58.8	17.1	2	-1
A	16	27	51.4	31.7	2	1	B	18	17	10.9	24.5	4	0	C	9	43	29.8	18.4	3	0
A	17	232	31.1	28.7	3	1	B	19	33	18.4	12.2	4	0	C	10	364	37.5	18.0	2	0
A	18	197	26.1	30.0	3	1	B	20	110	9.0	12.6	4	0	C	11	69	20.4	18.1	4	0
A	19	26	32.6	28.8	1	1	B	21	132	15.1	14.2	3	0	C	12	42	22.4	16.8	3	0
A	20	75	26.5	28.7	4	1	B	22	324	16.2	15.3	4	0	C	13	108	23.0	18.0	3	1
A	21	31	30.4	25.4	3	2	B	23	74	24.0	14.4	2	0	C	14	47	41.0	19.8	3	1
A	22	232	31.1	28.7	3	2	B	24	43	31.0	14.7	3	0	C	15	47	60.0	17.1	2	2
A	23	197	26.1	30.0	3	2	B	25	65	20.8	14.8	1	0	C	16	55	14.2	15.4	1	2
A	24	26	32.6	28.8	1	2	B	26	42	15.0	13.2	4	1	C	17	27	58.8	17.1	2	2
A	25	75	26.5	28.7	4	2	B	27	165	19.3	12.0	4	1	C	18	162	58.8	17.1	2	2
A	26	206	44.8	27.5	1	3	B	28	168	13.8	13.2	4	1	C	19	394	41.5	17.1	2	2
A	27	108	51.4	31.7	2	3	B	29	258	19.8	14.7	3	1	C	20	55	18.5	15.9	1	2
A	28	232	31.1	28.7	3	3	B	30	244	31.0	14.7	3	1	C	21	63	15.7	14.5	3	3
A	29	118	26.1	30.0	3	3	B	31	110	18.0	11.8	1	2	C	22	63	11.8	14.5	3	3
A	30	103	32.6	28.8	1	3	B	32	121	28.7	18.0	4	2	C	23	106	28.5	18.5	3	3
A	31	34	26.5	28.7	4	3	B	33	73	23.1	12.5	4	2	C	24	17	14.0	20.2	3	4
A	32	44	52.3	27.5	4	4	B	34	44	8.6	11.7	4	3	C	25	364	37.5	18.0	2	4
B	1	19	8.4	12.7	3	-3	B	35	147	24.0	14.4	2	3							
B	2	11	18.2	13.2	2	-3	B	36	38	8.8	11.7	4	3							

The number of flasks, which are to be made is calculated as the number of castings ordered by the customers divided by the number of castings which fit in a single flask. Thus the weight and forming time refer to the whole flask, not to a single casting. Due week is a week which has been agreed with the customer as a term of delivery. A negative number indicates that the remaining castings are already overdue.

There are 3 working shifts for the lines of machine type *A* and *C* while there are only 2 working shifts for the lines of machine type *B*. A common practice in the considered foundry is that only two different castings can be produced during one working shift, so κ_1 and κ_3 are set to 2 and κ_2 is set to 4. The total daily capacity of the furnaces is 21000 kg and a single heat weighs 1400 kg, i.e. at most 15 heats a day are possible. The number of flasks available for all moulding machines during one working shift is limited to 50 big flasks (machine type *A*), 100 medium (machine type *C*) and 120 small ones (machine type *B*).

The goal for optimisation is to create a plan for a week, which consists of 5 working days or for two weeks, consisting of 10 working days.

4 Genetic Algorithm

A weekly plan for moulding machines and a pouring schedule are coded in a single chromosome using integer gene values. First $n*k*(m_1+m_2+..+m_l)$ genes represent the quantity of castings planned for production or equals zero if the production for a particular order during a given shift is not planned. Last $\gamma*k*\max\{m_j\}$ genes represent the number of heats of a particular iron grade. This can be presented as the matrix shown in Figure 1.

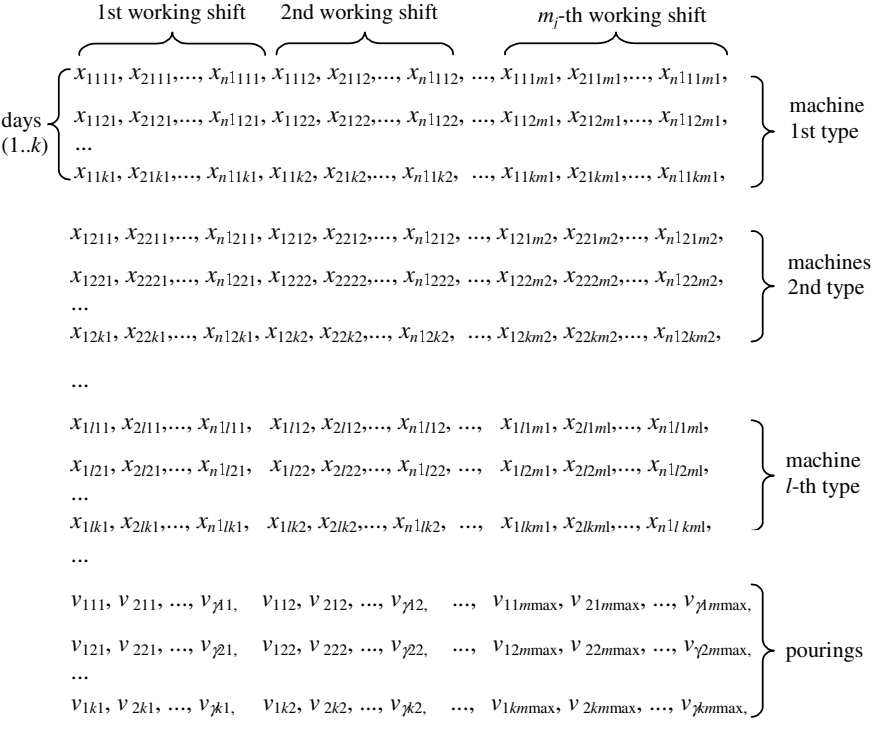

Fig. 1. Moulding plan and pouring schedule coded in a chromosome

4.1 First Variant (GA1)

The proposed chromosome structure is simple and natural. However, there is a lot of zeroes in a chromosome representing a valid plan, regarding the constraint (6). To avoid keeping incorrect individuals in a population, a simple repair algorithm has been introduced. Whenever constraint (6) is violated for one of the machines and working shifts, the smallest lots planned so far are eliminated successively from the plan until the number of different lots which are allowed for production during one working shift is reached. Only the non overdue castings are taken into consideration at the first stage. If the reduction in this stage is not enough the same procedure is applied also for the overdue castings. The scheme of the algorithm can be presented as follows:

Step 1. For each day t, working shift z and a machine j:
$K \leftarrow \mathrm{SUM}(i=1..n_j)\{x_{ijtz}\}$
Step 2. While $K > \kappa_j$
Step 3. Find a lot, for which the smallest weight of castings has been planned:
$x_s \leftarrow \mathrm{MIN}(i=0..n_j)\{x_{ijtz} w_i\}$
Step 4. Remove the lot x_s from the plan
Step 5. $K \leftarrow K-1$. Go to step 2

The above algorithm is used also in the remaining two variants of genetic algorithm, as it improves solutions by 20–30%, on average.

Also a new crossover operator has been introduced. It creates one child from two parents in the following way. A string of genes representing a single shift is chosen randomly in two parents. If the fitness value for the first parent is better than for the second parent, lots from the chosen shift in the first parent are placed in the second parent. If the second parent has better fitness, then the lots from it replace the lots in the first parent. The crossover operator simultaneously alters the pouring schedule for the affected shift.

The irregular mutation in the version proposed by Michalewicz and Janikow [4] has been chosen as a mutation operator with a one modification, which has been applied to it. The probability of increasing a gene value is 0.75 for the overdue castings, while it equals to standard 0.5 for the rest of the castings.

Most of genes in initial population are set to zero and only about 3% of them are set to random values. The genes representing pouring schedule are set randomly in such a way that their sum equals to the limit of the number of heats allowed. Experiments have shown that the solution quality obtained after first 1000 generations had a great impact on the quality of the final solution. That is why the proposed algorithm uses initially 10 populations, starting from 10 different points. The evolution is continued only for the best population after first 500 generations.

The remaining parameters, common for all algorithm variants look as follows:

- population size (fixed): 100 individuals
- number of generations: 10000 for problems with 5 days, 20000 for 10 days problems (+10000 for 10 initial populations)

- selection type: binary tournament with elitism
- crossover type: crossover changing shifts with the probability of 0.8
- mutation type: irregular mutation with the probability of 0.001
- penalty function: sum of squares of constraint violation values multiplied by 10000

4.2 Second Variant (GA2)

In the second variant of genetic algorithm two sets of variables representing moulding plans (x_{ijtz}) and pouring schedule (v_{htz}) are treated as two separate chromosomes.

Irregular mutation operator alters genes only in the first chromosome. For the second chromosome another mutation has been defined. It works as follows. The number of heats of randomly chosen iron grade is decreased by 1 and simultaneously the number of heats of another randomly chosen iron grade within the same working shift is increased by 1. This mutation is used with the probability of 0.05.

All other parameters of the genetic algorithm remain the same as in the first variant.

4.3 Third Variant (GA3)

In the third variant of genetic algorithm the genes in which pouring schedule is coded has been removed from the chromosome structure. Instead of this a second repair algorithm has been used. Its role is to keep moulding plans always acceptable from a pouring schedule point of view. This means there is always enough hot iron for filling all the moulds, which has been prepared. The idea of this algorithm is similar to the first repair algorithm. If the maximum number of heats of a particular iron grade is exceeded than the lot with the minimum weight of castings is removed from the plan. The details of the algorithm are shown below:

Step 1. $t \leftarrow 1$

Step 2. For each iron grade h:
Calculate the summary weight of iron grade h (SW_h) necessary for pouring the moulds prepared on day t:
$SW_h \leftarrow \mathrm{SUM}(j=1..l, i=1..n_j, z=1..m_j)\{(x_{ijtz}w_i)(g_i=h)\}$

Step 3. Calculate the number of heats (lw):
$lw \leftarrow \mathrm{INT}(\mathrm{SUM}(h=1..l)\{SW_h/W\})+1$

Step 4. If $lw \le lw_{max}$ go to step 9

Step 5. For the day t find a machine j, a shift z and a lot i, for which the smallest weight of castings has been planned:
$x_s \leftarrow \min(j=1..l, i=1..n_j, z=1..m_j)\{x_{ijtz}w_i\}$

Step 6. Remove the lot x_s from the moulding plan

Step 7. Correct lw by the removed lot weight

Step 8. If $lw > lw_{max}$ go to step 4

Step 9. $t \leftarrow t + 1$

Step 10. If $t \le k$ go to step 2

Experiments have shown that the role of the crossover operator defined earlier is virtually meaningless for the third variant of genetic algorithm. Thus a modified version of it has been introduced for this variant. The lots in randomly chosen shift from the first parent are swapped with the lots in another randomly chosen shift from the second parent. In that way two parents create two children instead of one, as it was in the previous case. However, this crossover plays a role of another mutation operator, rather than crossover itself. It is used with the probability of 0.1.

5 Results and Comparison to Integer Programming

Each variant of genetic algorithm was run for 10 times for fixed1 and *fixed2* test problems, assuming 5 days and 10 days planning horizon. A single run took about 4 minutes for 5-day problems and 8 minutes for 10 days (computer with Pentium 560 processor, 1 GB RAM). The results were then compared with the solutions given by branch-and-bound algorithm, implemented in CPLEX 9.0 mixed integer programming solver. Solving time for CPLEX was limited to the time of 10 runs of a single genetic algorithm. The best results and average results obtained from ten runs of the three genetic algorithm variants and the results generated by CPLEX 9.0 (denoted as bb) are collected in Table 3. The GA1 variant gave only 5 valid solutions for *fixed2* problem with 10 days.

Table 3. Results obtained by the genetic algorithm variants

Problem		GA1	GA2	GA3	bb	GA3-bb
fixed1 with 5 days	best	0.61	1.38	1.92	1.98	3.1%
	avg.	0.21	1.15	1.89		4.6%
fixed1 with 10 days	best	0.52	0.79	1.81	1.89	4.5%
	avg.	0.12	0.50	1.77		6.7%
fixed2 with 5 days	best	-0.35	0.74	1.84	1.96	6.4%
	avg.	-0.48	0.39	1.77		10.0%
fixed2 with 10 days	best	-0.58	-0.15	1.77	1.90	6.9%
	avg.	-1.24	-0.41	1.69		10.7%

The last column in the table shows a relative difference in the objective function value between the third variant of genetic algorithm and the branch-and-bound solution provided by CPLEX 9.0. The best of ten runs solution obtained by the third GA variant is not more than 5% behind the branch-and-bound algorithm for the first problem and less than 7% for the second problem.

Next, the experiments for the problems with the objective function, which consisted only of the first summand in equation (1), i.e. the utilization level of bottleneck aggregates, have been carried out. The same genetic algorithms were tested and only the overdue castings were not treated in a special way by the mutation operator and the repair functions, as it was in the previous case. Table 4 shows the obtained results in the same form as earlier.

Table 4. Results obtained by the genetic algorithm variants for a simplified objective

Problem		GA1	GA2	GA3	bb	GA3-bb
fixed1 with 5 days	Best	1.82	1.89	1.97	1.98	0.8%
	Avg.	1.71	1.76	1.93		2.5%
fixed1 with 10 days	Best	1.04	1.83	1.89	1.90	0.6%
	Avg.	0.72	1.77	1.80		5.1%
fixed2 with 5 days	Best	1.69	1.87	1.96	1.97	0.4%
	Avg.	1.58	1.79	1.92		2.8%
fixed2 with 10 days	Best	1.19	1.85	1.92	1.96	2.3%
	Avg.	0.86	1.65	1.85		5.4%

This time the best of the three genetic algorithm variants remains only less then 1% behind the CPLEX 9.0 algorithm in 3 of 4 test tasks, if the best result from ten runs is taken into consideration. The average solution generated by GA3 is within 3% of the branch-and-bound limit for the 5 days planning problems and within 5.5% for the 10 days instances.

This shows that the combined, but competitive criteria have a significant negative impact on the quality of solutions generated by the genetic algorithms. One of the methods to overcome this problem is to treat all the objective functions independently and use multiobjective evolutionary algorithms. The results of such an approach for a similar production planning problem are described in the recently published author's paper [2].

6 Final Remarks

The results presented in the paper show how much a repair function is important to a genetic algorithm, at least for certain real world problems. Both entering new operators and suiting a chromosome structure to a specific problem can significantly improve the quality of the obtained solutions. Nerveless, it is the introduction of even simple repair functions that lets a genetic algorithm to generate solutions of the high quality.

The third variant of genetic algorithm presented in this paper (GA3) can provide good solutions, which differ by 0.5–7% from the results obtained by the advanced branch-and-bound methods implemented in CPLEX 9.0. However, an integer programming approach cannot always be applied easily, because it requires all the objective functions and constraints to be non-linear. This is not a problem for metaheuristics such as evolutionary algorithms, for which the models can be written in the natural way.

The optimisation model for operational production planning in a foundry proposed in this paper will be successively complemented with new technological and organizational constraints, which have an impact on the overall production costs. The most interesting seems to be the assessment of the costs of a particular heat sequence, resulting from the costs of changing from one iron grade to another. The possibility of

making a given casting from different iron grades (usually higher), if such an operation is acceptable by the customer, will be also introduced into the planning model.

The data for problems *fixed1* and *fixed2* can be downloaded from the author's website at http://www.zarz.agh.edu.pl/jduda/foundry.

Acknowledgment

This study was supported by the State Committee for Scientific Research (KBN) under the Grant No. 0224 H02 2004 27.

References

1. Drexl, A., Kimms, A.: Lot sizing and scheduling – Survey and extensions, European Journal of Operational Research vol. 99, 2 (1997) 221–235
2. Duda, J., Osyczka, A.: Multiple criteria lot-sizing in a foundry using evolutionary algorithms (in:) Coello Coello C.A. et al. (eds.): EMO 2005, Lecture Notes in Computer Science vol. 3410 (2005) 651–663
3. Karimi, B., Fatemi Ghomi, S.M., Wilson, J.M.: The capacitated lot sizing problem: a review of models and algorithms, Omega, vol. 31, 5 (2003) 409–412
4. Michalewicz, Z., Janikow, C.Z.: Genetic algorithms for numerical optimization, Statistics and Computing, vol. 1, 2 (1991) 75–91
5. Pochet, Y.: Mathematical programming models and formulations for deterministic production planning problem (in:) Jünger M., Naddef D. (eds.), Computational Combinatorial Optimization, vol. 2241, Berlin, Springer-Verlag, Berlin (2001)
6. dos Santos-Meza, E., dos Santos, M.O., Arenales, M.N.: A Lot-Sizing Problem in An Automated Foundry. European Journal of Operational Research, vol. 139, 3 (2002) 490–500
7. Voorhis, T.V., Peters, F., Johnson, D.: Developing Software for Generating Pouring Schedules for Steel Foundries. Computers and Industrial Engineering, vol. 39, 3 (2001) 219–234