International Journal of Industrial Engineering Computations 7 (2016) 385–398

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

A hybrid metaheuristic method to optimize the order of the sequences in continuous-casting

Achraf Touil[a*], Abdelwahed Echchtabi[a], Adil Bellabdaoui[b] and Abdelkabir Charkaoui[a]

[a]*Laboratory of Mechanics, Industrial Management and Innovation (LMMII), FST Settat, Morocco*
[b]*Information Technology Team and Corporate Management (TIME), ENSIAS Rabat, Morocco*

C H R O N I C L E

Article history:
Received June 10 2015
Received in Revised Format December 21 2015
Accepted February 10 2016
Available online February 10 2016

Keywords:
Steel-making
Continous-casting
Hybrid metaheuristics
Simulated annealing
Genetic algorithm
Tabu list

A B S T R A C T

In this paper, we propose a hybrid metaheuristic algorithm to maximize the production and to minimize the processing time in the steel-making and continuous casting (SCC) by optimizing the order of the sequences where a sequence is a group of jobs with the same chemical characteristics. Based on the work Bellabdaoui and Teghem (2006) [Bellabdaoui, A., & Teghem, J. (2006). A mixed-integer linear programming model for the continuous casting planning. *International Journal of Production Economics*, *104*(2), 260-270.], a mixed integer linear programming for scheduling steelmaking continuous casting production is presented to minimize the makespan. The order of the sequences in continuous casting is assumed to be fixed. The main contribution is to analyze an additional way to determine the optimal order of sequences. A hybrid method based on simulated annealing and genetic algorithm restricted by a tabu list (SA-GA-TL) is addressed to obtain the optimal order. After parameter tuning of the proposed algorithm, it is tested on different instances using a.NET application and the commercial software solver Cplex v12.5. These results are compared with those obtained by SA-TL (simulated annealing restricted by tabu list).

1. Introduction

The steel industry is an important activity that contributes to the economy development of many countries. Companies have consistently sought innovative solutions in the production process to meet multiple challenges such as competition and fluctuations in energy costs. The production scheduling is an essential tool to improve productivity, minimizing treatment times, production costs and saving energy. The problems that deal with planning and scheduling of steelmaking processes have been widely discussed in the literature. These problems can be classified according to: The structure of the workshop (the number of converters (EAFs), the number of refining stands (RSs) and the number of continuous casting (CCs)). The constraints are taken into account as well as the optimization tools used in this field namely operations research methods and artificial intelligence methods. Thus, several works are proposed based on heuristics, exact models, and expert systems, constraint satisfaction, metaheuristics,

* Corresponding author.
E-mail: achrafchen@gmail.com (A. Touil)

doi: 10.5267/j.ijiec.2016.2.001

human-machine coordination methods and multi-agent methods. Tang et al. (2000) presented a mathematical programming modeling approach to schedule the steelmaking continuous-casting production planning. Chokshi et al. (2000) demonstrated that the problem of achieving minimal stopping at the continuous caster can be considered as a distributed and coordinated scheduling problem. Pacciarelli et al. (2004), Bellabdaoui et al. (2005) proposed a heuristics to solve SCC problem. Kumar et al. (2006) proposed a combinatorial auction-based heuristic and used it as a case study in an SCC factory with four production units at each of the three process stages (CV–RS–CC) and with a planning horizon of 10 hours. Two independent criteria were considered: minimizing the waiting time and maximizing the flow time. In a paper by Bellabdaoui and Teghem (2006), a mathematical programming modeling approach was used to schedule the SCC production planning for Arcelor Group in Liege, Belgium. Xuan and Tang (2007) addressed a hybrid flow shop scheduling with serial batching at the last stage, the model seeks to minimize the completion time. In Atighehchian et al. (2009), both the concept of ant colony optimization and nonlinear optimization methods were introduced to solve the problem for Mobarakeh Steel Company MSC. A bee colony-based algorithm was proposed by Pan et al. (2013) to solve the CV–RS–CC steelmaking planning and scheduling problem. To deal with the SCC scheduling issue, the work discussed in Xujun et al. (2009) developed a constraint programming model. Recently, Slotnick (2011) developed a lead time policy model for the continuous caster where a stochastic dynamic programming was used. The model considers stochastically the arrivals of several products and organizes them in a queue for processing.

In this paper, a hybrid metaheuristic based on genetic algorithm combined with simulated annealing and restricted by a tabu list is proposed to optimize the order of sequences which will maximize the production and also minimize the processing time in continuous casting.

The rest of this paper is organized as follows. In section.2, the literature reviews is presented. Section.3 is about the mathematical model of production scheduling is introduced. Section 4 presents the proposed algorithm to optimize the order of sequences in continuous casting. Section 5 deals with the numerical results. And in Section 6, a summary and conclusion are given.

2. Literature reviews

The steel production process is considered as a hybrid flow shop (HFS) problem because all jobs (charges) visit the three steps in the same direction and at every stage there are several parallel and identical machines. However, it is different from classical HFS regarding the processing jobs in the last step. Processing at this stage requires a continuous treatment of a set of charges (sequences) on the same refining stand and in the same continuous casting. In literature, a wide range of metaheuristics is proposed to solve the hybrid flow shop problem (HFS). In other words, these metaheuristics applied to HFS can solve our problem (optimize the order of sequences in continuous casting).

Haouari and M'Hallah (1997) proposed two-phase methods based on simulated annealing algorithm (SA) and tabu search algorithm (TS) to minimize the maximum completion time of two-stage hybrid flow shop problem. In a paper by Allaoui and Artiba (2004), a method that integrates the simulation and simulated annealing algorithm was proposed for the k-stage HFS with maintenance constraints to optimize several objectives such as flow time and due date. It was shown, that the performance of this approach could be affected by the percentage of the breakdown times. Further, this method performs better than the NEH heuristic under certain conditions. Nearchou (2004) presented a hybrid simulated annealing algorithm, which integrates the basic structure of the simulated annealing with some features of genetic algorithm and local search techniques for minimizing the makespan criterion. The proposed algorithm, used a population of candidate schedules and generates new populations of neighbor solutions by using a proper perturbation scheme. The advantage of this hybridization is to escape from possible local minima. It was also shown, that the proposed algorithm performs better than the standard algorithm SA and GA. Bertel and Billaut (2004) proposed a greedy algorithm and a genetic algorithm for solving

the hybrid flow shop problem in which machines that have different speeds and jobs may be re-visited. The model seeks to minimize the weighted number of tardy jobs. Kurz and Askin (2004) proposed a mixed integer programming model and the random keys genetic algorithm for minimization of makespan criterion for hybrid flow shop problem with sequence-dependent setup time constraint. Low (2005) proposed a heuristic based simulated annealing to deal with flexible flow shop scheduling problem with unrelated parallel machines where the objective was to minimize the total flow time. Several restrictions have been taken into accounts such as independent setup and dependent removal times. Sadegheih (2006) dealt with the flow shop scheduling problem with minimizing the makespan criterion. He presented a general purpose schedule optimizer for manufacturing shop scheduling using simulated annealing and genetic algorithm. Also, it was shown that the cooling rate, crossover rate and mutation effect on objective value. Ruiz and Stützle (2007) proposed a new iterated greedy algorithm with two phases named destruction and construction. The first phase, some jobs are eliminated from the incumbent solution. During, the second phase, the eliminated jobs are reinserted into the sequence using the well-known NEH construction heuristic. After this step, a local search can be applied optionally. The proposed algorithm performs better than the state-of-the-art methods. Janiak et al. (2007) considered hybrid flow shop by minimizing three criterions: the summation of the total weighted earliness, the total weighted tardiness and the total weighted waiting time. To solve this problem, they proposed three constructive algorithms and three metaheuristics based one tabu search and simulated annealing. Shiau et al. (2008) proposed a hybrid constructive genetic algorithm (HCGA) to address the HFS. The proposed procedure incorporates two fitness functions to evaluate a chromosome. The first fitness function reflects the total cost of a given schedule while the second function drives the evolutionary process to a population trained by a local improvement search based on tabu search. Bandyopadhyay et al. (2008) proposed a simulated annealing based multi-objective optimization algorithm that incorporates the concept of the archive to provide a set of compromise solutions for the considered problem. They proposed a procedure that takes into account the domination status of the new solution with the current solution to count the probability of acceptation of the solution. The proposed algorithm provided an efficient in terms of the complexity analysis compared with two other existing and well-known multi-objective evolutionary algorithms (MOEAs). Laha et al. (2008) used simulated annealing in conjunction with a constructive heuristic Nawaz et al. (1983) for minimization of makespan in permutation flow shop scheduling. Figielska (2009) considered the problem of scheduling in a two-stage flow shop with unrelated parallel machines and additional renewable resources at the first stage and a single machine at the second phase. To solve this problem, a heuristic that combines column generation technique with a genetic algorithm and a simulated annealing algorithm was proposed where the objective was the minimization of makespan. The proposed approach yielded good quality solutions using reasonable computation time compared with simulated annealing and genetic algorithm. Naderi et al. (2009) presented a compromise between intensification and diversification to increase the competitiveness of the simulated annealing method for HFS with sequence dependent setup times and transportation times to minimize total completion time and total tardiness. Czapiński (2010) presented parallelization between simulated annealing and genetic algorithm is presented for hybrid flow. Allahverdi and Al-Anzi (2006) studied scheduling multi-stage parallel-processor services to minimize average response time

3. SCC production process description and the mathematical model

In this section, the overall(sweeping) assumptions of this problem are defined following a description of the SCC production process.

3.1 Process description

Iron and steel production include several phases including iron making, steelmaking, continuous casting and steel rolling. The steel process under consideration is the application of SCC in a Belgian site of Arcelor Group. The basic processes to be scheduled are shown in block form in Fig. 1. It consists of three major steps (stages): (1) two identical converters CV (the basic oxygen furnace) with the same processing

time, (2) two refining stands RS with the same processing time at each machine, and (3) two continuous casting CC with a different processing time at each charge. Different material-handling and traveling crane are considered, but we just hold back the transfer time between successive stages. The converter refines pig iron to produce the crude steel on burnt by blowing in pure oxygen. The refining stand eliminates impurities and adjusts the chemical composition of the steel. Finally, the liquid steel is continuously poured into a bottomless mold. The molded metal descends, guided by a set of rollers and continues to cool and solidify (Arcelor, 2004; Kahraman et al., 2008).

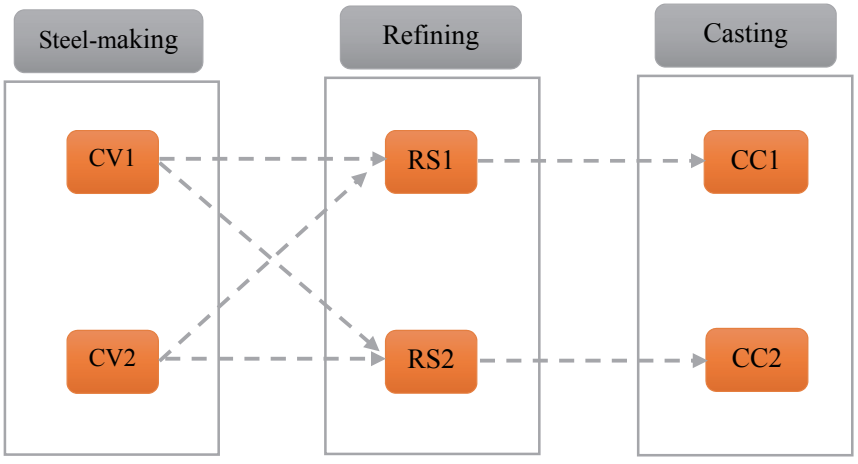

Fig. 1. Scheme of the layout

3.2 Mathematical model

The mathematical model is based on mixed integer linear programming, which concerns the scheduling of several sequences (a sequence consists of multiple jobs with the same characteristics, treated in a given order). A sequence is dedicated to one of the two production lines (RS - CC). It is part of an already ordered set on each of the continuous castings. The objective of this scheduling is to maximize the production and to minimize the processing time and should ensure the two following points:

- The jobs of the two sequences should be assigned to converters.
- Scheduling the execution of the jobs in the three steps of the steel by respecting the constraints, which involves the determination of processing sequences start in continuous casting (possible delays about availability date of those) and determine the load processing time on continuous casting (possible slowdown about the minimum duration of treatment).

4. The proposed hybrid algorithm

The MILP described in Bellabdaoui and Teghem, (2006), concerns several sequences already ordered. In this section, we analyze the possibility to optimize the order of sequences additionally. It is very clear that it is unrealistic to introduce directly this approach in the MILP model. Indeed, binary variables already numerous in this model, are based on the known order of jobs i and j. If this order is unknown, the extension of the MILP model would require, the introduction of additional binary variables to represent this order. But this, unfortunately, generates nonlinearities (binary variables product). Other approaches may be considered: The enumerative algorithms such as the complete enumeration, in practice this approach is more accurate in term of the quality of the solution found. However, the major drawback of this approach is the significant execution time for large instances. Hence, the need to consider other methods to find a compromise between the optimality of the solution and the execution time such as metaheuristics. In this paper, a hybrid method based on simulated annealing, genetic algorithm and restricted by a tabu list (SA-GA-TL) is used to solve this problem. The proposed method contains three main components: Simulated annealing, genetic algorithm, and tabu list. The SA is often presented as the oldest metaheuristics method; it inspired by the process of annealing applied to

metallurgy to solve an optimization problem. In SA not only better solutions are accepted, but also, worse neighboring solutions have a probability of being accepted. By this strategy and the cooling scheme, the algorithm is capable of escaping local optimum and exploring the search space as much as possible. A genetic algorithm is a population-based metaheuristic, inspired by the biological processes of genetics and evolution, which have successfully been applied to a wide variety of problems. By using of so-called genetic operators such as crossover and mutation, GA can explore the search space very well. The crossover operator speeds up the process to reach better solutions while the mutation operator explores a wider search space to avoid trapping in local optima. In another hand, the tabu list is one of the key factors that determine the quality of a TS algorithm to prevent the search process turning back to the individuals generated in the last n iterations.

The main advantage of combining these components is to use the advantages of the genetic algorithm such as population-based solutions, the genetics operators and the diversification strategy of simulated annealing algorithm. In other words, the transition probability of SA that helps to accept and move the offspring which has worse fitness function value than their parents to the next generation. The tabu list is implemented to (1) store the individuals generated in the previous generations; (2) maintain the elitism and the diversity of the population; (3) escape the local optima. The length of tabu list is a fixed size. When the tabu list is full, the oldest element of the list is replaced by the best new individuals. The mechanism of the proposed method could be described as follows. The algorithm begins with an initial solution generated by the NEH algorithm. After the initialization of the parameters (the population size, initial temperature), the first generation of solutions is created by applying the method described in Algorithm 1. For each chromosome, we calculate the fitness function. A list of chromosomes is selected from the current population to act as parents using the elitist strategy. The genetic operators are applied to produce the offsprings. For each offspring, we calculate the fitness function. After that, a replace strategy based on the Metropolis criteria is used to form the next generation. And to avoid premature convergence in the population we use two methods, named restart and generational schemes Ruiz & Maroto (2006). Finally, we update the temperature using the geometric scheme. And the process continues until the stop criteria are reached. The primary step of the proposed algorithm is described in Algorithm

4.1 Encoding the problem

The application of genetic algorithm requires a chromosome representation of the solution. In this paper, each chromosome is associated with the sequences prepared by the "logistics" service within a given order depending on the demand and the quality of the steel. These are divided into blocks; each block contains three sequences (the last block may hold the rest). This choice is inspired by the industrial reality since the steel does not receive all of these sequences once for all. It receives packets, each of these packages allows it to have the material to make it's scheduled for 2-3 days. Afterward, the rest of the sequence comes progressively. Fig.3. Shows the structure of chromosome (CC1 contains the sequences dedicated to continuous casting 1, CC2 contains the sequences dedicated to continuous casting 2).

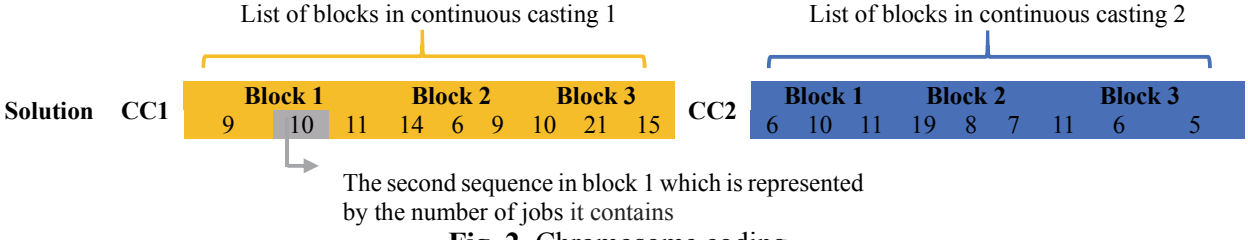

Fig. 2. Chromosome coding

4.2. Generation of the initial population

The performance and the efficiency of a GA are strongly related to the quality and the genetic diversity of the first generation of individuals. Extensive tests show that genetic algorithms based on randomly generated individuals for hybrid flow shop problems have a slow convergence. In the initialization

proposed by Reeves (1995) all individuals of the initial population is generated randomly, but one of this individuals is generated by the NEH heuristic (Nawaz et al., 1983). In this paper, to increase the convergence speed of the proposed algorithm, the NEH algorithm is implemented to generate the initial chromosome. The NEH algorithm generates a first sequence by allotting the jobs in a sequence based on their completion times. Jobs with higher processing times are given higher priority compared to jobs with lower processing times. The presence of the NEH algorithm ensures a better initial chromosome and guarantees the same first chromosome each time for a particular problem. The other chromosomes are generated using the Algorithm 1.

Algorithm 1. Initial-population:
Inputs:
Initial Chromosome C, Psize, Tabu-List Ω
Outputs:
Initial population P.

1: **boolean** Find=false;
2: **For** Psize
3: **Repeat**
4: Swap two blocks randomly in CC1;
5: Choose a block in CC1 randomly and swap randomly two sequences;
6: Swap two blocks randomly in CC2;
7: Select a block in CC2 randomly and swap randomly two sequences;
8: Find = Tabu_Search (C', Ω);
9: **Until** Find = false;
10: P.add (C')
11: **end** for
12: **Return** P

Evaluation function

The evaluation of fitness function is carried out using CPLEX. In this paper, we consider the minimization of makespan.

Genetic operators

To best explore the search space and to generate good quality solutions, three operators are applied: selection, crossover, and mutation. Thus, the evolution of the population from kth generation to $kth + 1$ depends on the power of these operators.

Selection

The selection operator determines the number of individuals which are selected for reproduction. The present work utilizes the elitist strategy. This strategy consists a sorting the population according to the ascending order and chosen $N_s$best individuals, where N_s the number of individuals selected for reproduction.

Crossover

The fundamental role of crossover is to permit the exploration of the search space and enriches the diversity of the population by manipulating the structure of chromosomes. A detailed description is given in **Algorithm 2.**

Algorithm 2. Crossover operator:

Inputs:

Two parent's P1 and P2 from the current population, Tabu-List Ω.

Outputs:

Two new offsprings C1 and C2.

1: For the parent's P1 and P2 choose randomly two points j_1 and j_2 in cast CC1 with $j_1 < j_2 <$ N-1 , N is the length of CC1 (resp. two points k_1 and k_2 in cast CC2 with $k_1 < k_2 < M - 1$, M is the length of CC2).
2: Cut P1 and P2 chromosomes into left, center and right segments;
3: Generate a new offspring namely C1
4: Inherit the middle part of P1 chromosome to the center segment of the C1 chromosome;
5: Copy the blocks in P2 chromosome that does not appear in the center part of P1 chromosome to the left and right segment of C1 chromosome **Fig. 6 (a)**.
6: Generate a new offspring namely C2 with the same manner that is generated the offspring C1
7: For each offspring applied a local swap into each block of the two casts CC1 and CC2. **Fig. 6 (b)**.Verify the occurrence of the two offsprings in the tabu list. If the offsprings have been already explored, go to step 1.Otherwise, return the two new offsprings. Figure 2: Illustration of crossover operator

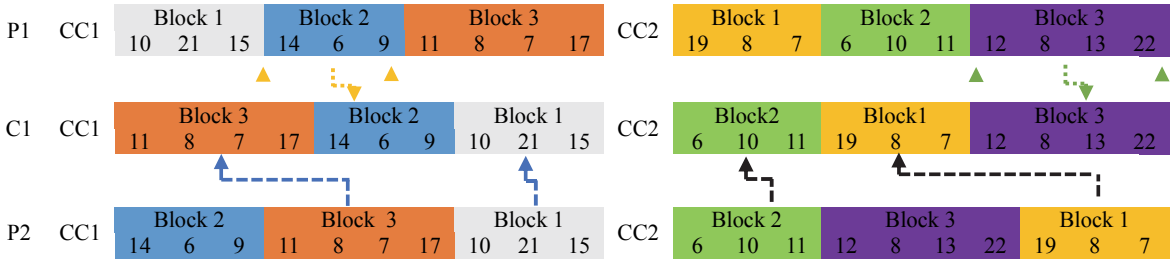

Fig. 3. Illustration of crossover operator

Mutation

The mutation operator changes a gene with a certain probability (mutation probability). The mutation operator focuses on maintaining the diversity of the population so as to enlarge the search space, and helps the search algorithm to escape from local optimal solutions or cooperates with crossover operator to achieve a ‘‘better’’ solution. A detailed description is given in **Algorithm 3.**

Algorithm 3. Mutation operator:

Inputs:
A randomly chosen chromosome.
Outputs:
A new chromosome.
1: For each cast CC1 and CC2 in the chromosome;
2: Choose randomly a B_i ;
3: Select randomly a location j ;
4: Insert the block B_i at the location j;
5: Verify the occurrence of the new chromosome in the tabu list. If the chromosome has been already explored, go to step 1.Otherwise, return the new chromosome.

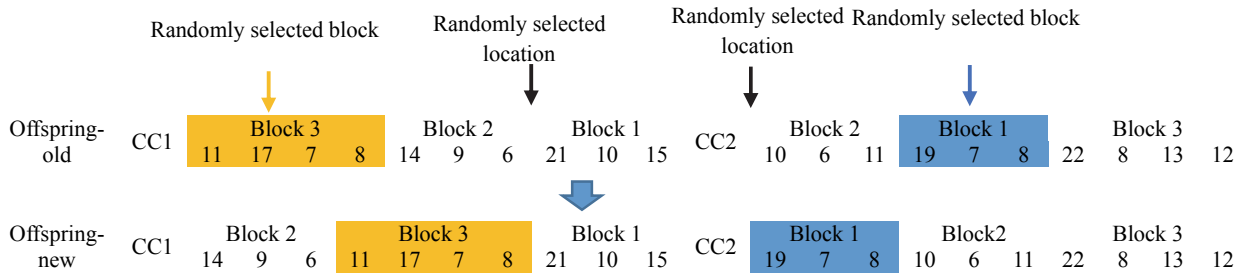

Fig. 4. Illustration of crossover operator

Replacement

The offspring population created by selection, recombination, and mutation replaces the original parental population. Many replacement techniques such as elitist replacement, generation-wise replacement, and steady-state replacement methods are used in GAs. In our case, the replacement is applied via the metropolis criteria Metropolis et al. (1953). For each generated child, the fitness function value is calculated, and if it is better than their parent's fitness function values, the child is moved into next generation. Otherwise, the child is transferred to the next generation with the probability equal to $e^{(\frac{\Delta F}{T_i})}$ .Otherwise, the child is rejected and, their parents move to the next generation. A detailed description is given in **Algorithm 4.**

Algorithm 4. Replacement strategy:

Inputs:

Parents P, Children Q, Current temperature T

Outputs:

Next generation P'

for Children Q
ΔF1=Fitness (child1)-Fitness (parent1);
ΔF2=Fitness (child2)-Fitness (parent2);
if ΔF1 < 0 **and** ΔF2 < 0 **then**
    The children move to the next generation P';
else
if $e^{\frac{\Delta F2}{T}} > Rnd()$ **and** $e^{\frac{\Delta F2}{T}} > Rnd()$**then**
        The children are accepted and moved to the next generation P';
        A=A+1;
else
        The children are rejected, and the parents move to the next generation P';
        R=R+1;
end if
end if
end for

Return P'.

Restart and generational schemes

To further avoid premature convergence in the population a restart mechanism based on the scheme proposed in Ruiz and Maroto (2006) is used. At each generation, we store the minimum value of makespan. If the fitness value is not changed for more than a pre-defined number ($countmak$) the restart phase commences to regenerate the population when $countmak$ become higher than a specific number (G_r). In the present work, the restart algorithm described as follow:

Algorithm 5. Restart and generational scheme:

- At each temperature, store the minimum makespan mak_T.
- If $mak_T = mak_{T-1}$then $countmak = countmak + 1$. Else $countmak = 0$
- If $countmak > G_r$ (number of iterations without improve) then
- Sort the population in ascending order.
- Skip the 20% individuals from the sorted list (the best individuals)
- From the remaining 80% individuals, 50% of them are replaced by simple SHIFT mutations of the best individual and 50% are replaced by newly randomly generated schedules.
- $regeneration = regeneration + 1$
- $countmak = 0$

Algorithm 6. The primary step of the proposed algorithm:

Inputs:
Chromosome C, Temp-in, Temp-fin, L, Amax, Rmax, Psize, Pc, Pm, Reg,
Outputs:
Best Solution found C*

```
Call Algorithm 2 to create an initial population;
EvaluationFitness (P);                    * Evaluate the initial population *
Best individual C* = Best-Individual (C ∈ P); f(C*) =f (Best-Individual (C ∈ P));
T=Temp-in
while T≥Temp-fin and Amax≥A and Rmax≥R and Reg≥r  do* the cooling loop *
l=0
while L≥land Amax≥A and Rmax≥R  and Reg≥r  do * the equilibrium loop *
Parents P =Selection (P);
Children Q =Crossover (Parents P, Pc);
Children Q =Mutation (Children Q, Pm);
EvaluationFitness (Q);
Call Algorithm 3 to create the next generation;
if f(C*) ≥ f (Best-Individual (C ∈ Q)) then* update best individual *
C*= f (Best-Individual (C ∈ Q));
f(C*) = f (Best-Individual (C ∈ Q));
end if
l=l+1;
Population Prestart=Restart();* Restart and generational schemes *
end while
Update (T);  * Update T with geometric scheme *
end while
Return C*
```

Parameter settings

The implementation of the proposed algorithm requires the adjustment of a set of parameters that influence the computation time and the objective function value. The proposed algorithm parameters are as follows.

- **Population size (P_{size})**: it is the number of individuals who are represented by their chromosomes in each generation. The increase of this parameter increases the objective function convergence. However, the process time increases very significantly.

- **Crossover probability (p_c)**: it is the probability of the crossover occurrence. The increase of crossover probability can help the process to reach the best optimum result. Figure 5.
- **Mutation probability (p_m)**: it is the probability of mutation occurrence. Generally smaller than the crossover. Figure 6.
- **Gr & regeneration:** the number of iteration without improvement, help to avoid the premature convergence of genetic algorithm and escape the local optimum.
- **The initial temperature (T_0)**: the initial temperature is an essential parameter in simulated annealing, it will change the execution time of the algorithm based on the effort required to the research. Based on the maximum estimated difference in the objective function (Sanvicente-Sánchez et al. 2004), as follows: $T_0 = \frac{-\Delta Z_{max}}{Ln(P^A(\Delta Z_{max}))}$ With $\Delta Z_{max} = Max\{Z(S_j) - Z(S_i)\}$and $P^A(\Delta Z_{max}) \in [0,1]$ the acceptance probability of a solution to the initial temperature. In the context of this problem, is chosen to have a solution that degrades the objective function accepted with a probability of 0.99. So the initial temperature is $T_0 \approx 100 * \Delta Z_{max}$.
- **The final temperature**: The final temperature is an essential parameter simulated annealing, it will stop the algorithm. The final temperature (T_f) selected such that: the probability of accepting a move is too low $P^A(\Delta Z_{ij}) \approx 0$; $T_f = \frac{-\Delta Z_{min}}{Ln(P^A(\Delta Z_{min}))}$with $\Delta Z_{min} = Min\{Z(S_j) - Z(S_i)\}$ and $P^A(\Delta Z_{min})$the acceptance probability of a solution to the initial temperature. In the context of this problem, is chosen to have a solution that degrades the objective function accepted with a probability of 0.01. So the final temperature is $T_f \approx \frac{\Delta Z_{min}}{5}$.
- **Total number of individual to accept**: this number must be sufficient to allow the proper exploration of the search space, but not too large so that it becomes disadvantageous regarding the process time
- **A total number of individual to reject**: this number must be sufficient to control the process search.
- **The length of the epoch**: At each temperature, a sufficient number of transitions is required to reach the equilibrium state.
- **The cooling scheme:** There is always a compromise between the quality of solutions and cooling scheme. If the temperature is slowly decreasing, the best solutions are obtained. However, the process time increases very significantly. In our case, the most used in the literature was chosen geometric scheme: $T_i = \alpha * T_{i-1}$ with $\alpha \in [0,1]$ is the cooling ratio.

Stopping Criterion

The stopping criteria are used to determine when the proposed method should stop. In the context of this work the method terminates when:

- The population ceases to evolve or not evolve enough, exceeds the specific number initially.
- The theory suggests a final temperature equal 0. In practice, the search stops when the probability of accepting a move is negligible. In our case, the algorithm terminates when one of the following conditions is fulfilled. $R_{max} \leq R$; $A_{max} \leq A$; $T_0 \leq T_f$.

5. Experimental results

In this section, we describe the numerical results obtained on a set of instances conducted to evaluate the performance of the proposed algorithm (SA-GA-TL). Our algorithm (SA-GA-TL) was implemented

in.NET the computational experiments were carried out on a computer with an Intel 2.66 GHz i5 core CPU with the windows seven operating system. The application allows the connection to the CPLEX solver v12.5 to evaluate the solution generated by (SA-GA-TL). To overcome this problem, we benchmark the quality of (SA-GA-TL) against SATL (A simulated annealing approach without the genetic algorithm component). The SA-TL have the same representation and the same parameters (initial temperature, final temperature, and the epoch length, and the Tabu list, total number of solutions accepted and the total number of solutions rejected. The geometric cooling scheme is used to decrease the temperature). Also, the performance of the proposed method depends on a set of the parameters such as:

- **Population size**: 50,100.
- **Crossover probability**: is set to 0.90 for all instances.
- **Mutation probability**: 0.1, 0.2.
- **Gr & regeneration**: 30, 20.
- **The initial and final temperatures**: For each instance, the temperatures are determinates using the equation described in the parameters section for a sample size of 20 solutions generated randomly.
- **The cooling ratio**: is set to 0.98 for all instances.
- **The epoch length**: 30, 40, 50.
- **Total number of individual to accept**: 50, 100
- **A total number of individual to reject**: 25, 30, 50.
- **The length of tabu list**: 15, 20, 30.

The total number of runs is 2×2×3×2×3×3=216. For instance, we choose 50 for initial population, 0.8 for the mutation probability, 30 for the epoch length, 50 and 25 for the total number of individual to accept and reject, and 20 for the length of tabu list. The numerical results found through.NET application are given in Table 1 column 2 represents the size of the problem where n_1 is the number of sequences in continuous casting 1 and n_2 represents the number of sequences in continuous casting 2. Column 3 represents the value of the objective function of the initial order proposed by the logistics service. The column 4 and 5: show the best results for each algorithm. For the column 6: shows the gap calculated using Eq. (1) between the SA-TL and SA-GA-TL. For all instances, the two algorithms obtain better solutions than the model mathematical. In Addition, the SA-GA-TL algorithm could obtain better solutions than the SA-TL.

$$Gap = \left(\frac{Best(SATL) - Best(SAGATL)}{Best(SATL)}\right) \times 100 \quad (1)$$

Table 1
The results of the proposed algorithm and SATL

Instance	Problem Size (n×m)	Initial solution (min)	SATL	SAGATL	Gap
			BS (min)	BS(min)	
Inst1	9×9	64,158,444	62,986,127	62,659,479	0,52
Inst2	9×12	75,280,415	74,107,724	74,107,724	0,00
Inst3	9×15	83,148,992	82,816,107	82,709,459	0,13
Inst4	11×9	71,292,797	70,997,913	70,608,818	0,55
Inst5	11×12	93,748,901	93,213,180	93,102,916	0,12
Inst6	11×15	119,249,618	118,021,181	11,770,889	0,26
Inst7	12×9	84,063,031	83,169,478	82,743,605	0,51
Inst8	12×12	130,218,320	129,158,888	128,664,731	0,38
Inst9	12×15	145,107,200	144,908,972	144,776,087	0,09
Inst10	15×9	163,273,747	162,800,359	162,761,406	0,02
Inst11	15×12	124,050,377	123,999,149	123,606,091	0,32
Inst12	15×15	141,348,693	141,270,787	141,270,787	0,00

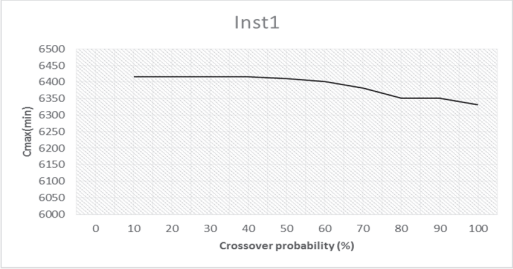

Fig. 5. Crossover influence on the objective function

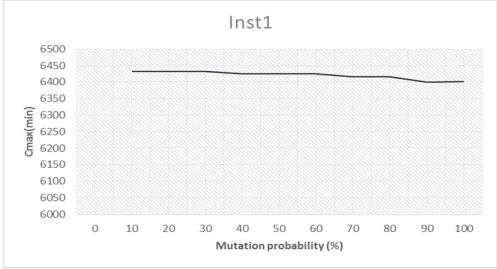

Fig. 6. Mutation influence on the objective function

6. Conclusions

This paper describes an optimization problem of the order of the sequences in continuous casting. The objective is to find the optimal order that minimizing the processing time and maximizing the productivity. A hybrid metaheuristic algorithm based on simulated annealing and the genetic algorithm was proposed to deal with this problem. Results showed that the proposed algorithm SA-GA-TL obtains better solutions than SA-TL for all instances. Use this approach, it was possible to reduce the processing time compared to the time obtained by the initial order proposed by the "logistic" service.

Acknowledgement

The authors would like to thank the anonymous referees for constructive comments on earlier version of this paper.

References

Arcelor (2004). Arcelor: a portrait. www.arcelor.com.

Allahverdi, A., & Al-Anzi, F. S. (2006). Scheduling multi-stage parallel-processor services to minimize average response time. *Journal of the Operational Research Society*, *57*(1), 101-110.

Allaoui, H., & Artiba, A. (2004). Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. *Computers & Industrial Engineering*, *47*(4), 431-450.

Atighehchian, A., Bijari, M., & Tarkesh, H. (2009). A novel hybrid algorithm for scheduling steel-making continuous casting production. *Computers & Operations Research*, *36*(8), 2450-2461.

Bandyopadhyay, S., Saha, S., Maulik, U., & Deb, K. (2008). A simulated annealing-based multiobjective optimization algorithm: AMOSA. *Evolutionary Computation, IEEE Transactions on*, *12*(3), 269-283.

Bellabdaoui, A., & Teghem, J. (2006). A mixed-integer linear programming model for the continuous casting planning. *International Journal of Production Economics*, *104*(2), 260-270.

Bellabdaoui, A., Fiordaliso, A., & Teghem, J. (2005). A heuristic algorithm for scheduling the steelmaking continuous casting process. *Pacific Journal of Optimization*, *1*(3), 447-464.

Bertel, S., & Billaut, J. C. (2004). A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. *European Journal of Operational Research*, *159*(3), 651-662.

Chokshi, N. N., Matson, J. B., & McFarlane, D. C. (2000). Distributed co-ordination of steel-making operations for reduced production stoppages.*Proceedings of MCPL*.

Czapiński, M. (2010). Parallel simulated annealing with genetic enhancement for flowshop problem with C sum. *Computers & Industrial Engineering*, *59*(4), 778-785.

Figielska, E. (2009). A genetic algorithm and a simulated annealing algorithm combined with column generation technique for solving the problem of scheduling in the hybrid flowshop with additional resources. *Computers & Industrial Engineering*, *56*(1), 142-151.

Haouari, M., & M'Hallah, R. (1997). Heuristic algorithms for the two-stage hybrid flowshop problem. *Operations research letters*, *21*(1), 43-53.

Janiak, A., Kozan, E., Lichtenstein, M., & Oğuz, C. (2007). Metaheuristic approaches to the hybrid flow

shop scheduling problem with a cost-related criterion. *International journal of production economics*, *105*(2), 407-424.

Kahraman, C., Engin, O., Kaya, I., & Kerim Yilmaz, M. (2008). An application of effective genetic algorithms for solving hybrid flow shop scheduling problems. *International Journal of Computational Intelligence Systems*, *1*(2), 134-147.

Kumar, V., Kumar, S., Tiwari, M. K., & Chan, F. T. S. (2006). Auction-based approach to resolve the scheduling problem in the steel making process.*International journal of production research*, *44*(8), 1503-1522.

Kurz, M. E., & Askin, R. G. (2004). Scheduling flexible flow lines with sequence-dependent setup times. *European Journal of Operational Research*, *159*(1), 66-82.

Laha, D., & Chakraborty, U. K. (2009). An efficient hybrid heuristic for makespan minimization in permutation flow shop scheduling. *The International Journal of Advanced Manufacturing Technology*, *44*(5-6), 559-569.

Low, C. (2005). Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. *Computers & Operations Research*, *32*(8), 2013-2025.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, *21*(6), 1087-1092.

Naderi, B., Zandieh, M., Balagh, A. K. G., & Roshanaei, V. (2009). An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. *Expert systems with Applications*, *36*(6), 9625-9633.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91-95.

Nearchou, A. C. (2004). Flow-shop sequencing using hybrid simulated annealing. *Journal of Intelligent manufacturing*, *15*(3), 317-328.

Pacciarelli, D., & Pranzo, M. (2004). Production scheduling in a steelmaking-continuous casting plant. *Computers & Chemical Engineering*, *28*(12), 2823-2835.

Pan, Q. K., Wang, L., Mao, K., Zhao, J. H., & Zhang, M. (2013). An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process. *Automation Science and Engineering, IEEE Transactions on*, *10*(2), 307-322.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing.*Computers & operations research*, *22*(1), 5-13.

Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, *169*(3), 781-800.

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, *177*(3), 2033-2049.

Sadegheih, A. (2006). Scheduling problem using genetic algorithm, simulated annealing and the effects of parameter values on GA performance. *Applied Mathematical Modelling*, *30*(2), 147-154.

Sanvicente-Sánchez, H., & Frausto-Solís, J. (2004). A method to establish the cooling scheme in simulated annealing like algorithms. In *Computational Science and Its Applications–ICCSA 2004* (pp. 755-763). Springer Berlin Heidelberg.

Shiau, D. F., Cheng, S. C., & Huang, Y. M. (2008). Proportionate flexible flow shop scheduling via a hybrid constructive genetic algorithm. *Expert Systems with Applications*, *34*(2), 1133-1143.

Slotnick, S. A. (2011). Optimal and heuristic lead-time quotation for an integrated steel mill with a minimum batch size. *European Journal of Operational Research*, *210*(3), 527-536.

Tang, L., Liu, J., Rong, A., & Yang, Z. (2000). A mathematical programming model for scheduling steelmaking-continuous casting production. *European Journal of Operational Research*, *120*(2), 423-435.

Xuan, H., & Tang, L. (2007). Scheduling a hybrid flowshop with batch production at the last stage. *Computers & Operations Research*, *34*(9), 2718-2733.

Xujun, Z., & Zhimin, L. (2009, July). Model and solution for steelmaking-continuous casting scheduling problem based on constraint programming method. In *Information Technology and Computer Science, 2009. ITCS 2009. International Conference on* (Vol. 1, pp. 19-22). IEEE.

Zhang, Y., Li, X., & Wang, Q. (2009). Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization.*European Journal of Operational Research*, *196*(3), 869-876.