RAIRO-Oper. Res. 52 (2018) 1351–1376
https://doi.org/10.1051/ro/2018023

RAIRO Operations Research
www.rairo-ro.org

A GENETIC ALGORITHM FOR THE STEEL CONTINUOUS CASTING WITH INTER-SEQUENCE DEPENDENT SETUPS AND DEDICATED MACHINES

Abdelkader Sbihi[1,*] and Makram Chemangui[2]

Abstract. The steel continuous casting planning and scheduling problem namely SCC is a particular hybrid (flexible) flowshop that includes stages: (i) the converters (CV), (ii) the refining stands (RS) and (iii) the continuous casting (CC) stages. In this paper we study the SCC with inter-sequence dependent setups and dedicated machines at the last stage. The batch sequences are assumed to be pre-determined for one of the CC devices with a non preemptive scheduling process. The aim is to schedule the batches for each CC machine including the times setup between two successive sequences. We model the problem as a MILP where the objective is to minimize the makespan $C_{\max}$ that we formulate as the largest completion time taking account of the setup times for each CC. Then, we propose an adapted genetic algorithm that we call Regeneration GA (RGA) to solve the problem. We use a randomly generated instances of several sizes to test the model and for which we do not know an optimal solution. The method is able to solve the problems in an acceptable time for medium and large instances while a commercial solver was able to solve only small size instances.

Mathematics Subject Classification. 90B35, 90B50, 90C11, 90C59

Received May 4, 2017. Accepted March 11, 2018.

1. Introduction

The steel continuous casting problem (SCC) is a particular scheduling problem that arises from the steel making industry. The problem comprises three main stages that are the converters (CV), the refining stands (RS) and the continuous castings (CC) where each stage includes one or more parallel devices. Each charge is to be process on one device at each stage while each device has to process only one charge separately. With this configuration (Figs. 1 and 2), the SCC can be seen as a hybrid flowshop (HFS) (or flexible flowshop (FFS)) problem (see [14, 28]). However, the difference with HFS resides in the continuous processing (non preemptive process) of the charges while taking into account the inter-sequence dependent setup times at the last stage (CC).

To show the difference between the HFS and the SCC, we make in the following a comparative study.

For a manufacturing system, the production is organized with regards to a given number of operations or stages to be accomplished with respect to the same order. Hence, the devices are supposed to carry setup

Keywords and phrases: SCC, GA, scheduling, $C_{\max}$, setup.

[1] Université Le Havre Normandie, LMAH, FR CNRS 3335, ISCN, 76600 Le Havre, France.
[2] Audencia Business School, 8 route de la Jonelière, 44312 Nantes, France.

* Corresponding author: abdelkader.sbihi@univ-paris1.fr

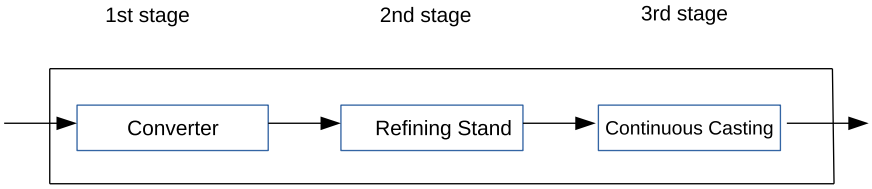

FIGURE 1. A generic SCC configuration.

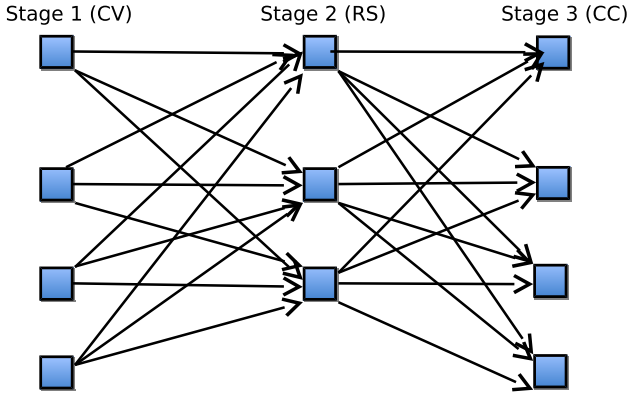

FIGURE 2. A $((\mathrm{P}_i)_{i=1}^{m_1},(\mathrm{Q}_j)_{j=1}^{m_2},(\mathrm{R}_k)_{k=1}^{m_3} \mid\mid \sigma \mid C_{\max})$ configuration.

times and two different cases are considered: (i) asynchronous transfer of the product flow from a current stand to the next one, (ii) and synchronized transfer of the product flow from one device to another. In general, the asynchronous transfer case is called a flowshop. The classical flowshop problem (FSP) considers only one machine at each stage, while its generalization considers a set of devices in series with multicapacity in parallel, called a hybrid flowshop problem (HFS). On one hand, the hybrid flowshop problem (HFS) can be seen as a set of parallel machines in series. On the other hand, the SCC planning and scheduling problem is a complex hybrid flowshop (HFS) scheduling problem with the added properties: (i) job grouping in sequences, (ii) precedence constraints, (iii) no preemptive scheduling process for the same sequence of charges, (iv) and setup time constraints on the last stage (CC stage).

More generally, the objective of the SCC is to determine the sequences order, the schedules and the system of devices assignments for the entire production processes. Moreover, we may notice that the SCC problem is known in the literature as a hard scheduling problems (see [11, 34, 36]).

In the following, we represent the SCC problem as a 3 stages scheduling problem with m_r $(r = 1, \ldots, 3)$ parallel machines at each stage r:

(1) m_1 identical parallel machines CV at the 1st stage (noted $(\mathrm{P}_i)_{i=1}^{m_1}$),
(2) m_2 uniform parallel machines RS at the 2nd stage (noted $(\mathrm{Q}_j)_{j=1}^{m_2}$),
(3) m_3 unrelated parallel machines CC at the 3rd stage (noted $(\mathrm{R}_k)_{k=1}^{m_3}$).

Similarly to the notation by [7], we write the SCC problem as a 3-stages hybrid flowshop (HFS): $(\mathrm{P}_i)_{i=1}^{m_1},(\mathrm{Q}_j)_{j=1}^{m_2},(\mathrm{R}_k)_{k=1}^{m_3} \mid\mid \sigma \mid C_{\max}$, where $\mid\mid$ stands for a parallel system and σ is the inter-sequence setup time (Fig. 2).

At the last stage the system includes the setup times that represent the inter-sequence times necessary to a CC machine between the processing termination of the current sequence and the processing start of the next sequence (between the last charge of the current sequence and the first charge of the next sequence) such that:

- the sojourn (transit) time is minimized;
- the due date is met;
- the continuity constraints are satisfied;
- no machine breakdown is considered.

The scheduling problems are mostly NP-hard optimization problems (see [5, 13]) and in the papers by [22], the authors showed that HFS problems with the objective to minimize the makespan are NP-complete. Also, the m-stages HFS with unrelated parallel machines and dependent setup times has been proven to be NP-hard (see [14, 16]). Also, there exist many variants of the SCC problem (see [25, 29, 37]) which they differ by the objective to optimize and/or by the set of constraints to handle. In many literature works, the setup time is considered as a constant or a parameter given in an interval (see [3, 31]).

In this paper we propose an MILP to model the SCC production problem. The objective function includes the setup time as a decision variable which contributes to minimize the makespan by reducing the gap between sequences on the same CC machine at the last stage. Also, we consider the makespan to minimize as the largest completion time for all the CC machines. Moreover, we recall that the proposed problem has a very complex structure, where small size problems could be difficult to solve to their optimum in a reasonable run time.

To overcome this difficulty, we intend to develop an approximate approach to solve the SCC. The novelty of the proposed planning and scheduling approach resides in optimizing the casting sequencing and the steelmaking scheduling simultaneously. The Regeneration GA is designed with specific components based on the schedule optimization search. We, first, compare the numerical experiments obtained by the standard GA with those obtained by the RGA approach. This enables us to claim the high efficiency of RGA regarding the standard GA. We used the quality of the solution and the consuming time by both GA and RGA. Then, we set the algorithm parameters to their best settings to test a set of instances of different sizes.

In the remainder, the paper is organized as follows. In Section 2, we give a literature survey on the hybrid (flexible) flowshop and the steel-making scheduling problems. We recall several exact and approximate approaches that have been developed for such problems. Section 3 describes the problem and details its structure. Section 4 presents the SCC problem, formulates the makespan $C_{\max}$ and details the mathematical formulation as a MILP subject to a set of constraints where the objective is to minimize $C_{\max}$. In Section 5, we develop a genetic based approach to approximately solve the SCC problem. We explain the start time computation for each charge at the first stage (CV) and detail the procedures used to compute an efficient solution. We also explain and discuss the main steps of the RGA approach and the auxiliary procedures. In Section 6, we detail and discuss the numerical experiments by showing the efficiency of our approach. Finally in Section 7, we give a conclusion that summarizes the study and shows some potential research perspectives to improve the current results.

2. Literature survey and problem positioning

Numerous research works have been developed to solve several HFS and SCC variants. In this section, we review the most known approaches (exact and approximate) that have been developed in previous research works to address the HFS in general and the SCC in particular. In the paper by [36], the authors reviewed and proposed a classification of several SCC models. In [2, 11], the authors detailed both mathematical and non-mathematical techniques used to solve these variants.

Moreover, optimization methods such as local search, genetic algorithms (see [42]), swarm intelligence optimization (see [12]) or mathematical programming (see [3, 31]) have been developed to solve the SCC problem. In the papers by [8, 24, 43] several types of metaheuristics have been applied to the SCC problem. Hybrid approaches have also been tailored for a continuous steel plant (see [1, 4, 10]).

We also may recall that simulation techniques (see [44]) have been applied to simulate the production environment and efficient solutions have been computed for such problems. In [9], the authors have designed a multi-agent based system to evaluate a dynamic casting schedule. Other techniques such as graphs theory and mathematical programming approaches have also been used. For instance, in [20], the authors have developed

a continuous slab caster schedule using interval graphs class. In the paper by [17], the authors have applied a multiple knapsack problem based ILP to avoid distortion between the surplus inventory and orders.

In [35], the authors have used a general heuristic based Lagrangean relaxation and dynamic programming for a non-linear programming model to address the machines conflicts. In [15], the authors developed a decomposition method of the MILP to solve large size scheduling problems. One can cite the paper by [37] in which the authors have relaxed the capacity constraints, then applied a dynamic programming for solving the relaxed problem. In the paper by [40], an approach based on the production control has been developed to identify the best steelmaking operations level. The paper by [38] proposed another approach to minimize the total weighted termination time by relaxing the precedence constraints.

In [26], the SCC problem has been solved by a beam search method. Tabu search (see [39]) has been applied for the SCC to schedule the production in order to maximize the production quality while minimizing the costs and other penalties. In [41], the authors presented an approximate approach to minimize the completion time based on batch decoupling. Also, in [25], the authors developed a Lagrangean relaxation coupled with cuts generation to compute better bounds for the relaxed capacity constraints problem. Particular SCC problems with different configurations have also been solved. For instance, in the article by [23], the authors have developed a 3-stages heuristic procedure to schedule jobs for a complex steel plant with 3 CV, 2 RS and 4 CC to improve the schedule by using an LP model.

Also several evolutionary based algorithms have been applied to the SCC. One can cite the paper by [19] in which an evolutionary algorithms has been developed for a two-strands CC scheduling system. In the paper by [6], the authors proposed a GA to identify the efficient solutions in the Pareto front after running some extensive search. In the paper by [33], the authors have developed a GA they called hybrid constructive genetic algorithm (HCGA) to solve the HFS. Their approach used two fitness functions (total cost and local search optimization) to evaluate an individual. A multi-objective GA to minimize the makespan and tardiness penalties has been developed by [18]. Recently, in [21], the authors have presented a self-adaptive GA to optimize the casts in order to minimize the makespan and the total idle times.

During the last two decades, other evolutionary approaches such as ant colony (ACO), bee colony (BCO) or swarm optimization (SO) have also been applied for the SCC. In the article by [1], one can find an approach based on the combination of ACO and nonlinear optimization methods. In [28], the authors have developed a BCO heuristic. In the paper by [27], the author has proposed a cooperative co-evolutionary artificial BCO algorithm with two sub-swarms to solve the SCC by using some self-adaptive neighborhood operators.

Due to the NP-hardness of SCC, we develop an adapted genetic algorithm that we call Regeneration GA (RGA) based heuristic to solve the SCC. The choice of the genetic based optimization is efficient since this algorithm fits well with the structure of the SCC problem. In particular, coding the chromosome fits well with the solution structure of the problem to assign the charges to each CV, RS or CC machine while ensuring the validity of the individual. The proposed approach solves the problem by optimizing the order and the schedules with inter-sequence dependent setup times on each dedicated CC device. The approach combines: (i) the planning of the casting of the batches and (ii) the steelmaking scheduling of the total charges. These two components are integrated in the model. In the next sections, we describe the GA features by detailing the used genetic operators. We also present the approach performance obtained by the experiments on a random generated set of instances inspired by a real production situation.

3. Problem statement

In the SCC problem $\equiv \mathrm{P}_{m_1},\mathrm{Q}_{m_2},\mathrm{R}_{Mm_3} \mid\mid \sigma \mid C_{\max}$, the production orders are represented by a set of the cast sequences (batch) where each sequence is dedicated to a CC machine at the last stage. At the first stage, the charges can be processed individually on any one of the available converters (CV). Then they are transferred to the first available refining stand (RS). However, at the stage 3, the processing of the charges on the CC is predetermined. That means that the actual allocation of charges and sequence of charges on each CC line is predetermined and only the timing of the charges needs to be established.

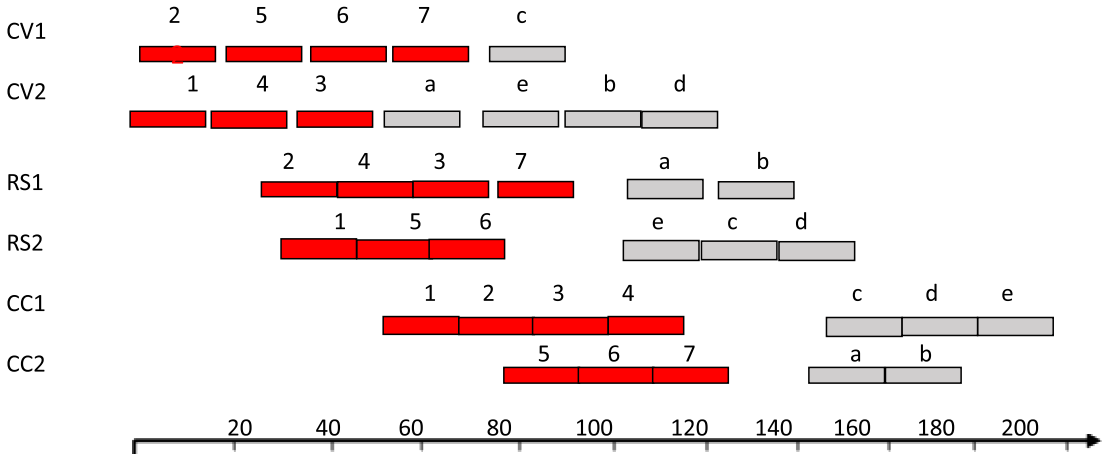

FIGURE 3. A 2 CV-2 RS-2 CC Gantt chart.

We assume that the sojourn (transit) time for the charges between their termination at the first stage and their start at the last stage (CC) is limited. The processing time for each charge at the last stage is a bounded value. Also, we add an availability date for each converter so that the start time at the first stage is different for all the converters. We set the objective as to maximize the productivity by minimizing the $C_{\max}$.

3.1. Process, sequences and continuity constraints

The planning system is detailed in the following:

- a sequence is a set of charges dedicated to a CC with precedence constraints for the charges;
- any converter CV can be used in the first stage with a constant processing time;
- the charge has a limited sojourn time between the termination in the CV and the start at CC;
- there is no idle time for the charges on the CC due to the continuous constraints;
- the processing times at the CC stage are decision variables belonging to a given interval;
- the sequence start times at CC are bounded and the waiting time between two successive sequences represents the setup time.

The example in Figure 3 shows the sequencing of a batch composed of 4 sequences on a (2CV-2RS-2CC) system. The first batch is composed of 2 sequences noted $(1,2,3,4)\times(c,d,e)$ with respectively 4 and 3 charges which are dedicated to CC_1, while the second batch is composed of 2 sequences noted $(5,6,7)\times(a,b)$ with respectively 3 and 2 charges and are dedicated to CC_2. Between CV and RS stages the charges are processed individually, while the sequences are re-assembled to be processed on their assigned CC. For example, the charge noted "a" is belonging to the second sequence and is to be processed on CC_2. It starts at time 50 on CV_2, is transferred to RS_1 at time 101 then it is transferred to CC_2 and starts at time 142.

4. Continuous casting of the pre-ordered sequences

In this section we give a mathematical formulation for the SCC problem taking into account the set of constraints and the objective function to optimize. The obtained MILP aims at minimizing the makespan. The production environment is a deterministic one and the number of charges together with their intrinsic data are assumed to be known in advance. Also, we assume that the number of parallel machines at each stage is known in advance.

4.1. Notations

To better understand the problem, we detail the used notations and all the data in the next sections.

4.1.1. Sets, constants and indices

(1) $(r = 1, 2, 3)$ denote the stages (CV,RS,CC);
(2) m_r: number of machines at the stage r;
(3) $i = 1, \ldots, m_1$ (resp. $j = 1, \ldots, m_2$) (resp. $k = 1, \ldots, m_3$) denotes the index of a machine at the 1st stage (resp. the 2nd stage) (resp. the 3rd stage) such that (CV_i,RS_j,CC_k) is a production line;
(4) S_k: number of the sequences s_k to process on the machine CC_k ($s_k = 1, \ldots, S_k$);
(5) s_k: a sequence to process on CC_k ($k = 1, \ldots, m_3$);
(6) $n_{s_k} = |s_k|$: number of the pre-ordered charges (jobs) c for a sequence s_k on CC_k;
(7) $n_k = \sum_{s_k=1}^{S_k} n_{s_k}$: total number of charges to process on CC_k ($k = 1, \ldots, m_3$);
(8) c: index of a charge of the sequence s_k dedicated to CC_k ($c = 1, \ldots, n_k$);
(9) $N = \sum_{k=1}^{m_3} n_k$: total number of charges in the system.

We also have the following assumptions:

(1) π_i: position of a charge on the first $(m_1 - 1)$ CV_i ($\pi_i = 1, \ldots, \Pi_i$; $i = 1, \ldots, m_1 - 1$) where $\Pi_i = \lfloor \frac{\sum_{k=1}^{m_3} n_k + 1}{m_1} \rfloor$;
(2) π_{m_1}: position of a charge on the last machine CV_{m_1} ($\pi_{m_1} = 1, \ldots, \Pi_{m_1}$) where $\Pi_{m_1} = \lfloor \frac{\sum_{k=1}^{m_3} n_k}{m_1} \rfloor$.

4.1.2. Data, parameters and settings

(1) p^1: processing time of a charge c ($c = 1, \ldots, n_k$) on the converters CV_i ($i = 1, \ldots, m_1$) at the 1st stage;
(2) p_j^2: processing time of a charge c on the refining stand RS_j ($j = 1, \ldots, m_2$) at the 2nd stage;
(3) $[P_{k,c}^{\min}, P_{k,c}^{\max}]$: interval of the processing time $p_{k,c}^3$ of a charge c dedicated to CC_k;
(4) $[\sigma_{s_k}^{\min}, \sigma_{s_k}^{\max}]$: interval of σ_{s_k}, the inter-sequence dependent setup time between sequences s_k and $(s_k + 1)$ $(s_k = 1, \ldots, S_k - 1)$;
(5) d_i: availability date for CV_i;
(6) $T_{k,c}$: maximum allowed sojourn time for a charge c between the processing termination in any CV_i and its start in CC_k;
(7) τ_{12} (resp. τ_{23}): transfer time of a charge between CV_i (stage 1) and RS_j (stage 2) (resp. RS_j (stage 2) and CC_k (stage 3)).
(8) $d_{k,c}$: due date for a charge c dedicated to CC_k

4.1.3. Decision variables

We have identified both continuous and binary decision variables for the proposed model. For any CC_k $k = 1, \ldots, m_3$:

(1) $x_{k,c}^r$: start time of the charge c ($c = 1, \ldots, n_k$) dedicated to CC_k at stage r ($r = 1, 2, 3$);
(2) $p_{k,c}^3$: processing time of charge c ($c = 1, \ldots, n_k$) dedicated to CC_k at the 3^{rd} stage;
(3) σ_{s_k}: setup time between two successive sequences s_k and $(s_k + 1)$ to process at CC_k (between the last charge n_{s_k} of s_k and the first charge $1_{(s_k+1)}$ of s_{k+1}) for $s_k = 1, \ldots, S_k - 1$.

(4)

$$y_{k,c}^{\pi_i} = \begin{cases} 1 & \text{if the dedicated charge } c \text{ to } \mathrm{CC}_k \text{ is assigned to a position } \pi_i \text{ in } \mathrm{CV}_i, \\ 0 & \text{otherwise.} \end{cases}$$

$c = 1, \ldots, n_k$, $\pi_i = 1, \ldots, \Pi_i$, $i = 1, \ldots, m_1$.

4.2. A SCC mathematical model

The model computes the processing start time of the charge defined by its position at the first stage (CV) regarding the precedence constraint. It also respects the charge assignment to the CC machine.

4.2.1. Constraints

For each CC_k ($k = 1, \ldots, m_3$), the constraints are formulated as follows:

$$\sum_{i=1}^{m_1} \sum_{\pi_i=1}^{\Pi_i} y_{k,c}^{\pi_i} = 1; \quad c = 1, \ldots, n_k; \tag{4.1}$$

$$\sum_{k=1}^{m_3} \sum_{c=1}^{n_k} y_{k,c}^{\pi_i} = 1; \quad \pi_i = 1, \ldots, \Pi_i; \tag{4.2}$$

and $$\sum_{k=1}^{m_3} y_{1,1}^{1} = 1;$$

$$y_{k,c+1}^{t_i+1} \leq \sum_{i=1}^{m_1-1} \sum_{\pi_i=1}^{t_i} y_{k,c}^{\pi_i}; \quad c = 1, \ldots, n_k - 1;\ t_i = 1, \ldots, \Pi_i - 1; \tag{4.3}$$

$$y_{k,c+1}^{t_{m_1}} \leq \sum_{i=1}^{m_1} \sum_{\pi_i=1}^{t_i} y_{k,c}^{\pi_i} - y_{k,c}^{t_{m_1}}; \quad c = 1, \ldots, n_k - 1;\ t_i = 1, \ldots, \Pi_i; \tag{4.4}$$

$$x_{k,c}^{1} = \sum_{i=1}^{m_1} \sum_{\pi_i=1}^{\Pi_i} [d_i + p^1(\pi_i - 1)] y_{k,c}^{\pi_i}; \quad c = 1, \ldots, n_k \tag{4.5}$$

Refining stage

$$x_{k,c+1}^{2} \geq x_{k,c}^{2} + p_j^2; \quad c = 1, \ldots, n_k - 1,\ j = 1, \ldots, m_2; \tag{4.6}$$

$$x_{k,c}^{2} \geq x_{k,c}^{1} + p^1 + \tau_{12}; \quad c = 1, \ldots, n_k; \tag{4.7}$$

Continuous casting stage

$$x_{k,c}^{3} \geq x_{k,c}^{2} + p_j^2 + \tau_{23};\ c = 1, \ldots, n_k,\ j = 1, \ldots, m_2; \tag{4.8}$$

$$x_{k,c+1}^{3} = x_{k,c}^{3} + p_{k,c}^{3};\ \forall c \notin \{n_1, n_1 + n_2, \ldots, \sum_{s_k=1}^{S_k} n_{s_k}\}, \forall k = 1, \ldots, m_3; \tag{4.9}$$

$$x_{k,c+1}^{3} \geq x_{k,c}^{3} + p_{k,c}^{3} + \sigma_\ell;\ \forall c = \sum_{s_k=1}^{\ell} n_{s_k},\ \ell = 1, \ldots, S_k - 1; \tag{4.10}$$

$$p^3_{k,c} \in [P^{\min}_{k,c}, P^{\max}_{k,c}];\ c = 1, \ldots, n_k; \tag{4.11}$$
$$x^3_{k,c} - (x^1_{k,c} + p^1) \leq T_{k,c},;\ c = 1, \ldots, n_k; \tag{4.12}$$
$$\sigma_{s_k} \in [\sigma^{\min}_{s_k}, \sigma^{\max}_{s_k}];\ s_k = 1, \ldots, S_k - 1; \tag{4.13}$$
$$x^3_{k,c} + p^3_{k,c} \leq d_{k,c};\ c = 1, \ldots, n_k. \tag{4.14}$$

Constraint (4.1) means that a charge c ($c = 1, \ldots, n_k$) is assigned to one and only one machine CV_i and at one and only one position π_i.

Constraint (4.2) means that a converter CV_i has to process one and only one charge c at a specific position π_i ($i = 1, \ldots, m_1$). Also, only the charge $c = 1$ dedicated to CC_1 has to be assigned to the first position at the converter CV_1.

Constraint (4.3) requires that the charge $(c + 1)$ has to be processed on the converter CV_1 ($i = 1$) at the position $(t_1 + 1)$ only if the charge c is to process on one of the $(m_1 - 1)$ first converters CV_i and at any positions between 1 and t_i.

Similarly to the constraint above, constraint (4.4) means the same in the case of the last converter CV_{m_1} for two successive charges c and $(c + 1)$.

Constraint (4.5) stands for the start time to process the charges in the first stage CV_i ($i = 1, \ldots, m_1$);

Constraint (4.6) means that for two successive charges to process on the same refining stand (RS_j), the second charge can be processed only if the first one has reached its end time.

Constraints (4.7) and (4.8) require that for two successive operations for the same charge, the second operation can only start once the first one has reached its end time and the charge has been transferred to the next stage.

Moreover, constraints (4.7) (resp. constraint (4.8)) stands for the precedence rules between CV_i and RS_j (resp. between RS_j and CC_k).

Constraint (4.9) stands for the continuity constraints between charges c and $(c + 1)$ for a given sequence s_k (no preemptive process).

Constraint (4.10) stands for inter-sequence dependent setup times between two successive sequences (*i.e.* between the last charge of the current sequence and the first charge of the next sequence) for the same CC_k machine at stage 3.

Constraints (4.11) defines the interval for the processing time on the machine CC_k at the stage 3 for a charge c.

Constraint (4.12) means that the sojourn time (transit) of a charge c is limited.

Constraint (4.13) defines the interval of the inter-sequence setup times at stage 3.

Constraint (4.14) defines the due dates for the charges that are processed on a CC device.

4.3. The objective function

First we formulate the total completion time on each CC_k ($k = 1, \ldots, m_3$). Then we formulate the makespan ($C_{\max}$) as the maximal total value of the completion time of the last sequence S_k and the corresponding setup time for all the CC_k devices. This formulation allows us to minimize both the completion and setup times of the entire production process and by the way to reduce the waiting time between two successive sequences.

Hence the $C_{\max}$ has the following mathematical formulation:

$$\mathrm{C}_{\max} := \max_{1 \leq k \leq m_3} \left\{ \sigma_{S_k} + x^3_{k,n_{S_k}} + p^3_{k,n_{S_k}} \right\}$$

On one hand, the identification of charges is independent of the sequences. However, their order numbers in the corresponding sequence are preserved. On the other hand, for each device, the sequencing is achieved by mean of the charges assignment variables. Also, constraints (4.9), (4.12) and (4.14) ensure that the continuity at the last stage, the allowed sojourn (transit) times and the due dates are met.

To solve the problem, we set the objective as to minimize the makespan $C_{\max}$ subject to all the identified constraints. The problem can formally be written as follows:

$$(\min C_{\max}) \begin{cases} \mathrm{C}^{\star}_{\max} = & \text{Minimize } \mathrm{C}_{\max} \\ \text{Subject to} & (1)-(14); \\ & C_{\max} \geq \sigma_{S_k} + x^3_{k,n_{S_k}} + p^3_{k,n_{S_k}}, \quad \forall k = 1,\ldots,m_3; \qquad (4.15) \\ & x^3_{k,n_{S_k}},\ p^3_{k,n_{S_k}},\ \mathrm{C}_{\max} \geq 0,\ \sigma_{S_k} \in [\sigma^{\min}_{s_k}, \sigma^{\max}_{s_k}]. \qquad (4.16) \end{cases}$$

The MILP above brings a novelty by considering the makespan as the sum of the inter-sequence setup time at the last stage and the total completion time. This formulation permits: (i) to minimize the setup time values, (ii) to respect the continuity constraints and (iii) to tighten the charges flows in order to optimize the productivity.

The MILP can be extended to a multi-stage batch production with continuity constraints and setup times at the last stage. This latter is generalizing in turn the multi-stage hybrid flowshop with continuity constraints. The production is processed on multi-sequential production stages where each stage is guaranteeing a multicapacity of parallel machines. However, all the devices located on the first $(r-1)$ stages are allowed to process jobs of any sequence where at the last stage, the sequence jobs are processed on their dedicated machine. Continuity constraints are imposed at the last stage with new introduced decision variables that are the times setup between two successive sequences of a batch production. This constitutes a novelty in our MILP.

5. A genetic algorithm for the SCC with inter-sequence dependent setup time

For HFS problems, [30] were the first to apply a hybrid GA to the HFS with setup times. The method has been applied to minimize the makespan for a m-stages system with unrelated parallel machines, setup times and machines eligibility. For HFS problems, most of the GA approaches usually separate assignment and sequencing (see [29, 32]). Generally, these methods, are first sequencing by permutation flowshop procedures and then are assigning charges to devices at each stage by some priority rules or by machine availability. However, these strategies yield poor solutions quality in general. In the next sections, we detail the RGA based heuristic to solve the proposed problem.

5.1. Encoding chromosomes and an initial solution

To obtain high quality solutions, we used an encoding strategy which integrates both sequencing and assignment of the charges at stages 1 and 2 and continuity constraints at stage 3. The chromosome is defined as a sequence of the total N charges that are assigned to their CC machines. Each gene is represented by a couple (k, c) meaning that a charge c of a certain sequence is assigned to CC_k. The gene has a machine assignment index k that refers to CC_k and a sequencing index c that refers to the charge of a certain sequence s_k.

So that $(1, c)$, $c = 1,\ldots,n_1$, represents all the genes associated with sequences to process on machine CC_1, $(2, c)$ the sequences to be processed on machine CC_2,... The total number of genes is the total number of charges $N = \sum_{k=1}^{m_3} n_k$ where $n_k = \sum_{s_k=1}^{S_k} n_{s_k}$.

At stage 1, the order in which the charges start their processing time is obtained by the position of the genes in the chromosomes, that means we assign charges to converters with regards to their availability and possible slowdown dates.

The chromosome shown in Figure 4 means that a total of 10 charges have to be processed on 3 CC machines (CC_1,CC_2,CC_3) such that CC_1 must process a sequence of 4 charges, CC_2 a sequence of 3 charges and CC_3 has to process a sequence of 3 charges. For example, the couple $(3, 1)$ represents the gene (k, c) such that the machine $\mathrm{CC}_{(k=3)}$ has to process the first charge $c = 1$.

(1,1)	(3,1)	(2,1)	(2,2)	(1,2)	(1,3)	(3,2)	(1,4)	(3,3)	(2,3)

FIGURE 4. A chromosome encoding configuration.

Input: *chromosome*(1), *size*
Output: Initial population $\mathcal{P}_1$

$\mathcal{P}_1 \leftarrow \{chromosome(1)\}$
accept() = `false`;
`While not` (accept())
  `For` $i = 2$ `to` *size* `do`
    `For` $k = 1$ `to` m_3 `do`
      Swap randomly 2 batches in CC_k;
      Choose randomly one batch in CC_k and swap randomly 2 sequences s_k and s'_k;
    `EndFor`;
    `If` *chromosome*(i) is not valid then repair(); //Repair the chromosome
    chromosome(i) $\leftarrow$ accept(); //check whether the chromosome is not already in the population
    $\mathcal{P}_1 \leftarrow \mathcal{P}_1 \cup \{chromosome(i)\}$;
  `EndFor`;
`EndWhile`;
`Return` $\mathcal{P}_1$.

FIGURE 5. Gener(): an initial population $\mathcal{P}_1$.

In the following, we show how to generate a first feasible solution. We also consider the fitness function $f = \dfrac{1}{1 + C_{\max}}$ to build fittest solutions. A chromosome is said to be healthier if it has the lowest makespan value $C_{\max}$. We add 1 to the denominator of the ratio to avoid a possible dividing by 0. The schedule with the minimal makespan value is considered as the best solution.

We recall that for HFS it is not always easy to determine the start time at the first stage. Based on the assignment constraints (4.3) and (4.4) on one hand, and on the precedence constraints (4.6), (4.7) and (4.8) on the other hand, we compute (constraint (4.5)) the start time of the charges at the first stage according to the minimal sojourn time.

One can also use a heuristic procedure to fix the machine assignments and precedence relationships as parameters so that the obtained model carries only continuous variables that is easy to solve in this case. We call the first feasible solution obtained in Section 5.1 *chromosome*(1). Then from *chromosome*(1), a first population $\mathcal{P}_1$ is generated using the following steps:

Once a first feasible solution is obtained, we apply the Gener() procedure to generate an initial population $\mathcal{P}_1$ of size *size* (Fig. 5).

For each generated individual, we consider the objective function such that:

- to minimize the makespan $C_{\max}$,
- to meet with the continuity constraints at the CC stage,
- to reduce the charges sojourn times,
- to meet with due dates.

5.2. GA operators

5.2.1. Crossover

Several crossover techniques were developed by the past for GA. To ensure constructing feasible sequences while performing crossover operations, we use a tailored order crossover that is adapted from the classical OX operator. Our crossover operator consists: (i) to generate two children by swapping genes information of two

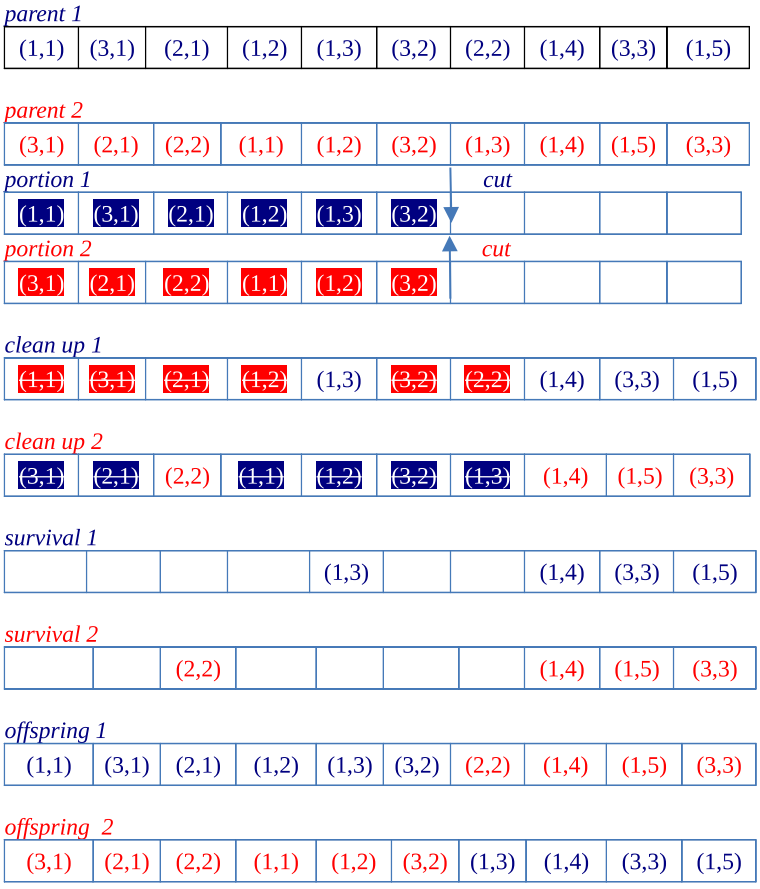

FIGURE 6. A crossover mechanism.

parents that are selected with a selection probability, (ii) to identify by mean of a random cut the left and right cut sides of each parent and (iii) to generate the children by selecting the first part of the first parent while keeping the order of the genes of the second parent.

The example (Fig. 6) shows how we adapt the crossover operation. We consider 2 chromosomes representing a configuration of 10 charges dedicated to 3 CC where CC_1 has to process 5 charges, CC_2 has to process 2 charges and CC_3 has to process 3 charges. The cut is selected at the 6th gene.

The crossover operator works as follows:

- we randomly select a cut position;
- for each portion and starting from the left side, we remove one by one all the genes that are common with the other parent;
- we complete each survival with the remaining genes of the other survival in their recorded order to obtain an offspring.

We can see that the crossover is automatically ensuring the feasibility of the obtained chromosome since it is preserving the CC machines assignment at the last stage.

Input: A Population $\mathcal{P}_1$ of size *size*
Output: An efficient solution for SCC with the best $C^{\star}_{max}$

Intialization
1. $g = 1$, $i = 0$, *counter* $= 0$; // generation index, individual index, no improvement counter
2. f = Eval($\mathcal{P}_1$); // the fittest solution
3. $C^{\star}_{max}$= Min_Makespan($\mathcal{P}_1$); // record the best makespan of the initial population

Main steps
4. `While` ($g \leq G$)
5. $g = g + 1$;
6. $\mathcal{P}_g = \emptyset$;
7. `For` $i = 1$ `to` *size* `do`;
8. $parent_k$ = Random_Select($\mathcal{P}_{g-1}$), $k = 1, 2$;
9. *child* = Crossover() `with probability` π_χ;
10. *child* = Mutation() `with probability` π_μ;
11. `If` *child* is not valid `then` Repair();
12. $\mathcal{P}_g \leftarrow \mathcal{P}_g \cup \{child\}$;
13. `End_For`;
14. f = Eval($\mathcal{P}_g$);
15. `call` Find_Fittest($\mathcal{P}_g$); //find the fittest chromosome in the population;
16. `If` Min_Makespan($\mathcal{P}_g$) $\geq C^{\star}_{max}$ `then` *counter* $\leftarrow$ *counter* $+ 1$; //no improvement
17. `Else` *counter* $\leftarrow 0$;
18. $C^{\star}_{max}$= Min_Makespan($\mathcal{P}_g$);
21. `If` (*counter* $=$ *maxCount*) `then` reorder the population $\mathcal{P}_g$ in $\uparrow$ order and renumber them;
23. $\mathcal{P}_g \leftarrow$ `Regeneration()`; //regenerate population
24. f=Eval($\mathcal{P}_g$);
25. $C^{\star}_{max}$= Min_Makespan($\mathcal{P}_g$);
26. *counter* $= 0$;
27. `EndWhile`;
28. `Exit with the best SCC schedule with the best makespan` $C^{\star}_{max}$.

FIGURE 7. A RGA for SCC.

5.2.2. Mutation

We use some classical mutation operators (the exchange, swap or shift operators). We compute the mutation probability for each gene with regards to its position in the chromosome. Then, the mutation operator shifts a random selected position in the sequence and relocates it at a random position (at the left or the right) where the charges between the two shifted positions change also their positions regarding the shifted charges. If the obtained child is not valid, then a repairing step is performed.

5.3. The GA algorithm

We use the following parameters: the population size (*size*), the number of generations (G), the selection probability, the crossover probability (π_χ) and the mutation probability (π_μ). The population evolves for a fixed number of generations. However we may encounter the difficulty such that for a given generation, the solution is not improved enough due to poor population diversity. In this case, we integrate a regeneration mechanism based on the idea proposed by [30].

A new feature for our RGA is that it is generalizing the regeneration mechanism since we adapt it to a set of jobs sequences. Also, at each generation, we record the value that realizes the minimum makespan $C_{\max}$ value. If for a certain number of iterations (or treshold) no improvement is observed for the fitness value, then we apply the regeneration mechanism to regenerate the population in order to escape local optimum (Fig. 7).

1. $\mathcal{P}_g \leftarrow \mathcal{P}_g \setminus \{chromosome(1), ..., chromosome(\lfloor (1/4)size \rfloor\}$; //skip the first 25% fittest //chromosomes
2. For $i = \lfloor (1/4)size \rfloor + 1$ to $\lfloor (3/8)size \rfloor$
3. $chromosome(i)$ = Mutation(); //mutate the first half remaining chromosomes
4. For $i = \lfloor (3/8)size \rfloor + 1$ to $size$ call Gener(); //generate randomly the second half //of the remaining chromosomes
5. $\mathcal{P}_g \leftarrow \mathcal{P}_g \cup (\bigcup_{i=\lfloor (3/8)size \rfloor + 1}^{size} \{chromosome(i)\})$;
6. EndIf;

FIGURE 8. The Regeneration() procedure.

The Eval() function computes the fitness value of the individuals. Then GA generates at each generation g a new population $\mathcal{P}_g$ by mating two random selected parents $parent_1$ and $parent_2$ from $\mathcal{P}_{g-1}$. The selection is performed thanks to the selection probability $\frac{f_i}{\sum_{i=1}^{size} f_i}$.

The derived GA namely RGA comprises 2 phases and is detailed as follows:

In the GA phase, we apply the Crossover() operator to generate a child noted *child*. Then, if the mutation probability π_μ is acceptable, we apply the Mutation() operator on *child* and we add *child* to $\mathcal{P}_g$.

The repairing is applied to eliminate infeasibility such as discontinuities and large sojourn times by modifying the sequences start time. We reiterate the process until we generate *size* chromosomes to add to the new population $\mathcal{P}_g$. We evaluate the individuals and select the fittest one as the solution (best $C_{\max}$) to SCC.

Another new feature of our approach resides in the procedure Find_Fittest() which uses the MILP model of Section 4.2 as a neighborhood search optimization to improve locally the solution. If the continuity and sojourn time constraints are not violated, then we reach a local optimum for the makespan and the SCC obtains an efficient solution. Otherwise, there exists no feasible solution in the considered neighborhood space.

At a given generation g which is not improved enough, a counter of no improvement records the current state and is incremented by one unit until it reaches a maximum value fixed in advance, then the Regeneration() phase (Fig. 8) of the algorithm is activated (Steps 23–26).

6. Experimental tests

In this section, we detail the numerical experiments carried out in order to test the proposed approach. We have coded the GA with C++ language on a Ubuntu Linux 14.04 Hp ProBook Core i3, 2.4 Ghz and 6 Go of Ram. The CPU time, the completion time and the makespan are computed in seconds while the model run time is related to the instance configuration (sequences and charges numbers). Thus attempting to solve the SCC to its optimality may leads the computation time to grow fast with the size of the problem. Even though only small problems can be solved by mathematical programming techniques, there are still benefits to study the model in order to develop efficient approximate approaches.

The mathematical model that we present helps to better understand the problem structure and its complexity. However to the best of our knowledge, the model herein presented has not been studied so far. Consequently, no existing results are available in the literature to compare with our numerical results. We have based the design of our benchmark on the paper by [31] and other instances of large size were randomly generated with the same industrial context.

The population size *size* defines the number of chromosomes which conformed the population. If we have too small population sizes, the algorithm converges faster to a local optimum. In the case of a large population size, the computation time becomes disadvantageous. The number of generated populations G stands for the number of evolutions steps to apply and a small number of generations usually leads to an early optimization break. Also a large number of generations is too time consuming and no possible improvement can be performed after a certain number of generations.

TABLE 1. Cplex results for a 2 CC problem with different sequences size.

# Inst.	$(n_{1_1}, \ldots, n_{S_1})x(n_{1_2}, \ldots, n_{S_2})$	CPU	CC_1 compl. time	CC_2 compl. time
1	$(5,5) \times (5,5)$	27.67	**887.58**	864.43
2	$(14,21) \times (5,5,5)$	917.25	**1087.01**	974.26
3	$(15,10,12) \times (5,5,5)$	out of mem. err.	N/A	N/A
4	$(10,13,17,8) \times (5,5,5,5,5)$	out of mem. err.	N/A	N/A
5	$(18,11,23,15,27) \times (5,5,5,5,5,5,5,5,5,5)$	out of mem. err.	N/A	N/A

TABLE 2. Cplex results for a 3 CC problem with different sequences size.

# Inst.	$(n_{1_1}, \ldots, n_{S_1}) \times (n_{1_2}, \ldots, n_{S_2})$ $\times(n_{1_3}, \ldots, n_{S_3})$	CPU	C_1 compl. time	C_2 compl. time	C_3 compl. time
1	$(5) \times (5) \times (5)$	2.56	363.45	**396.78**	382.88
2	$(10,10) \times (10,10)$ $\times(10,10)$	1261.27	676.31	692.26	**713.10**
3	$(5,15,19) \times (7,10,8)$ $\times(5,7,12)$	2321.38	**1496.34**	1389.38	1214.19
4	$(5,5,8) \times (6,4)$ $\times(5,4,5) \times (6,6,7)$	out of mem. err.	N/A	N/A	N/A
5	$(5,15,5,10)$ $\times(10,10,10,5) \times (5,5,5,5)$	out of mem. err.	N/A	N/A	N/A
6	$(5,10,5,10,5)$ $\times(10,10,10,5,5) \times (5,5,5,5,5)$	out of mem. err.	N/A	N/A	N/A

TABLE 3. Optimization performance study for RGA and standard GA.

Configuration	Pop. size	Approach	Av. $C_{\max}$	Av. Std. Dev	Av. 100 $\left(\frac{C_{\max_{\mathrm{RGA}}} - C_{\max_{\mathrm{GA}}}}{C_{\max_{\mathrm{GA}}}}\right)$
2CV-2RS-2CC (14,21) × (5,5,5)	100	RGA	1245.15	0.58	−17.6%
		GA	1512.23	0.37	
	200	RGA	1207.23	0.59	−15.6%
		GA	1431.40	0.44	
3CV-3RS-3CC (15,15,19) × (7,10,8)	100	RGA	1625.54	0.57	−23.5%
		GA	2125.17	0.46	
	200	RGA	1572.71	0.54	−18.6%
		GA	1932.35	0.38	

TABLE 4. Comparison of the averages consuming time for RGA and standard GA.

Configuration	Pop. size	Av. Comp. time for RGA	Av. Comp. time for std. GA
2CV-2RS-2CC (14,21) × (5,5,5)	100	1643.24	6812.27
	200	1843.12	7207.13
3CV-3RS-3CC (15,15,19) × (7,10,8) × (5,7,12)	100	1987.22	8145.30
	200	2317.38	9045.87

TABLE 5. A $C_{\max}$ sensitivity analysis for SCC($[14, 21] \times [5, 5, 5]$).

Size	$C^{\star}_{\max}$	$C^{\star\star}_{\max}$	$\overline{C}_{\max}$	sd	Err.
25	1230.202		1231.642	0.737	2.670
30	1223.023		1224.186	1.002	2.071
35	1219.332	1198.202	1219.922	0.442	1.763
40	1213.063		1214.439	1.217	1.240
45	1200.001		1201.783	0.770	0.150
50	1198.202		1199.277	0.933	0.0

TABLE 6. A $C_{\max}$ sensitivity analysis for SCC($[15, 15, 19] \times [7, 10, 8] \times [5, 7, 2]$).

Size	$C^{\star}_{\max}$	$C^{\star\star}_{\max}$	$\overline{C}_{\max}$	sd	Err.
25	1590.304		1591.817	0.967	0.762
30	1588.031		1589.492	0.689	0.618
35	1584.086		1585.042	0.632	0.368
40	1586.147	1578.274	1586,993	0.555	0.498
45	1578.274		1579.113	0.577	0.0
50	1578.788		1579.305	0.416	0.032

The crossover probability π_{χ} stands for the intensity to pair chromosomes and the mutation probability π_{μ} determines the level to modify the configuration of the chromosomes. All these parameters needs to be well defined for high quality solutions.

As a first step of our study, we show the validity of the model and its hardness to obtain an optimal solution for medium and large size instances. We use a commercial solver (CPLEX V12.71) and we report the CPU time. All the tested instances by the solver are 2 CC and 3 CC configurations at the last stage (Tabs. 1 and 2).

The numerical results presented herein and based on a real-life sequences show the validity of our model for the planning of several sequences for a 2 CC and a 3 CC systems. CPLEX failed to solve some of the instances with 2 CC and 3 CC and for all the systems with 4CC and higher due to a lack of memory.

6.1. Performance of the RGA

To study the performance of the RGA, we have first applied the standard GA without Regeneration() procedure and have compared the obtained results to those of the RGA. We have run two tests with population sizes of 100 and 200 each (for a crossover probability $\pi_{\chi} = 0.8$ and a mutation probability $\pi_{\mu} = 0.03$) on a 2CV-2RS-2CV and a 3CV-3RS-3CC configurations. For each case, we used 10 sets of instances that have been tested 10 times repeatedly. In Table 3, we report the performance of RGA based an a comparison study of the average results from each test case (population size). We can observe that our approach is able to provide on average a smaller makespan $C_{\max}$ than the standard GA for the 2 population sizes and a lower standard deviation. The gaps between GA and RGA represent -17.6% and -15.6% for the 2CV-2RS-2CC and -23.5% and -18.6% for the 3CV-3RS-3CC configuration for 100 and 200 individuals respectively. From these outputs, we can easily state that the solutions obtained by RGA are likely to be of better quality that those obtained by the standard GA.

Moreover, we have considered the computational time to measure the performance of the RGA. We reported in Table 4 the average time obtained for all the tests cases. From these results, we can remark that the RGA is nearly 4 times less consuming time than the standard GA. We can explain it by the fact that our algorithm manages well the SCC scheduling by saving the time to build a schedule for a batch.

TABLE 7. A $C_{\max}$ sensitivity analysis for SCC($[5,5,8]\times[6,4]\times[5,4,5]\times[6,6,7]$).

Size	$C^{\star}_{\max}$	$C^{\star\star}_{\max}$	$\overline{C}_{\max}$	sd	Err.
25	2611.066		2611.406	0.246	0.717
30	2604.069		2605.442	0.888	0.447
35	2601.176		2602.449	0.938	0.336
40	2594.216	2592.458	2595.521	0.927	0.067
45	2593.058		2594.143	0.732	0.023
50	2592.458		2593.048	0.400	0.0

6.2. Sensitivity analysis and parameters setting

To test the efficiency of our algorithm, we make a sensitivity analysis study. The aim is:

- to test the influence of the population size;
- to investigate the relationship between the total number of populations for a population with a fixed size.

For each chosen instance, we have launched 20 runs. Tables 5–8 detail the best makespan noted $C^{\star}_{\max}$ for each fixed size, the best found makespan noted $C^{\star\star}_{\max}$ which corresponds to the best settings for all the fixed sizes, the average makespan of the 20 runs noted $\overline{C}_{\max}$, the standard deviation noted sd and the relative percentage error noted *err*.

Since we do not know any optimal solution for the tested instances or any alternative approach that has been tested by the past, we compare the results with the best found solution so far. We compute the relative percentage error as $\text{err} = 100 \times \frac{C^{\star}_{\max} - C^{\star\star}_{\max}}{C^{\star}_{\max}}$ and compute the standard deviation noted sd.

6.2.1. Sensitivity analysis of the population size

We have set the number of populations to $G = 150$, varied the size of each population in the set $\{25, 30, 35, 40, 45, 50\}$ and we have run the RGA algorithm 20 times. Tables 5–7 give the results for the three instances SCC($[14,21]\times[5,5,5]$), SCC($[15,15,19]\times[7,10,8]\times[5,7,2]$) and SCC($[5,5,8]\times[6,4]\times[5,4,5]\times[6,6,7]$).

For the 3 problem instances, RGA has been running 20 times for each chosen population size (*size*). $C^{\star}_{\max}$ corresponds to the best found solution so far among the 20 computed ones and $C^{\star\star}_{\max}$ corresponds to the best settings.

Note that as we increase the size of the population, the variance value (sd^2) decreases quickly to zero value from $size = 30$ to $size = 40$ and with a small decrease from $size = 40$ to $size = 50$. This means that from a certain size of the population RGA gives the solutions for 20 runs with comparable quality. For example, for the problem instance SCC($[15,15,19]\times[7,10,8]\times[5,7,2]$), the algorithm always gives the best $C^{\star}_{\max}$ for sizes value starting from 150. The *size* value can then be limited to 150 with no need to increase it more. This allows us to save computational time.

TABLE 8. A $C_{\max}$ sensitivity analysis for SCC($[5,5,8]\times[6,4]\times[5,4,5]\times[6,6,7]$).

G	$C^{\star}_{\max}$	$C^{\star\star}_{\max}$	$\overline{C}_{\max}$	sd	Err.	CPU(s)
50	2609.10		2699.23	1.27	4.25	863.8
100	2663.02		2665.61	2.28	2.93	1014.1
150	2599.96	2587.04	2602.44	1.02	0.49	1523.5
250	2590.25		2592.28	3.32	0.12	1987.4
300	2587.04		2590.79	2.54	0.0	2021.5

6.2.2. Sensitivity analysis of the number of populations

For this study, we set the population size to 50. Table 8 shows the best $C^{\star}_{\max}$ resulting from 20 runs of the RGA algorithm for a SCC instance and for different number of generations G taken from the set $\{50, 150, 200, 250, 300\}$. The Computation time was also reported.

Table 8 shows that the RGA is able to obtain good solutions with an "acceptable" computational time. The best generations calibration is around 150. For big generations number, RGA consumes huge amount of time without a substantial improvement of the solution. Also for a small generations number, RGA leads "rapidily" to a solution whose quality is less than the best known so far.

6.3. Numerical tests

For the model presented in Section 4.2, we have developed and tested a 2-phases RGA based heuristic on a set of SCC real instances and other randomly generated ones that simulate a production plan. For memory amount considerations, we have limited our examples to 2 SCC configurations. The first configuration is a 2 CC machines and the second one is a 3 CC machines. All the related parameters to the 2 SCC systems are known and the solution is highlighted in bold characters.

Table 9. The RGA paramaters.

Parameters	RGA
# of generations G	150
Population size $size$	50
Crossover prob. π_{χ}	0.8
Mutation prob. π_{μ}	0.03
Regeneration() $maxCount$	25

Table 10. Performance comparaison between RGA and HCGA.

$N \times 3 \times m$	HCGA	RGA	%Improvement	CPU_{HCGA} (s)	CPU_{RGA} (s)	%Improvement
$60 \times 3 \times 4$	227.0140	227.0140	0.0000	27.5000	11.3792	−0.5862
$60 \times 3 \times 8$	126.9580	126.9580	0.0000	40.1000	21.4347	−0.4656
$60 \times 3 \times 12$	94.3647	94.0580	−0.0033	65.5000	39.9107	−0.3907
$60 \times 3 \times 16$	78.1323	77.7080	−0.0054	95.4000	63.2675	−0.3368
$80 \times 3 \times 4$	319.0330	319.0330	0.0000	36.4000	34.7705	−0.0448
$80 \times 3 \times 8$	174.6420	174.6420	0.0000	65.3000	55.7230	−0.1467
$80 \times 3 \times 12$	127.4397	127.0000	−0.0035	92.8000	78.2667	−0.1566
$80 \times 3 \times 16$	103.8413	103.3000	−0.0052	122.6000	89.2809	−0.2718
$80 \times 3 \times 20$	90.1456	89.4000	−0.0083	238.8000	234.5156	−0.0179
$100 \times 3 \times 8$	338.7683	338.5000	−0.0008	79.3000	78.8036	−0.0063
$100 \times 3 \times 12$	241.1030	240.0000	−0.0046	155.8000	87.5064	−0.4383
$100 \times 3 \times 16$	192.7576	191.8447	−0.0047	247.6000	170.8258	−0.3101
$100 \times 3 \times 20$	164.4252	163.7000	−0.0044	498.8000	455.9293	−0.0859
$120 \times 3 \times 6$	334.6000	329.9237	−0.0140	698.0000	533.5807	−0.2356
$120 \times 3 \times 10$	287.5000	283.9800	−0.0122	765.0000	613.0737	−0.1986
$120 \times 3 \times 14$	423.7000	420.6253	−0.0073	987.0000	586.7541	−0.4055
$150 \times 3 \times 4$	298.6000	294.7603	−0.0129	1002.0000	591.6787	−0.4095
$150 \times 3 \times 8$	409.7000	406.3901	−0.0081	1579.0000	918.2093	−0.4185
$150 \times 3 \times 10$	475.6700	469.9805	−0.0120	1987.0000	1894.6503	−0.0465
Average			−0.0056			−0.2501

TABLE 11. A RGA results for CC_1 for Problem SCC($[14, 21] \times [5, 5, 5]$).

Charge (k, c)	$x^1_{k,c}$	CV_i	$x^2_{k,c}$	$x^3_{k,c}$	$p^3_{k,c}$	Comp. time	Sojourn	cont.
1	30	1	89	126	24	150	52	**
2	50	1	112.3698	149.3698	42.9453	192.3151	55.3698	<0.1
3	94	2	154.6849	191.6849	42.9453	234.6302	53.6849	<0.1
4	138	1	197.0000	234.0000	30.3619	264.3619	52	<0.1
5	162	2	226.7317	263.7317	40.8049	304.5366	57.7317	<0.1
6	206	2	266.9064	303.9064	42.9453	346.8517	53.9064	<0.1
7	226	2	309.2215	346.2215	30.3619	376.5834	76.2215	<0.1
8	270	2	338.9532	375.9532	30.3619	406.3151	61.9532	<0.1
9	294	1	368.6849	405.6849	42.9453	448.6302	67.6849	<0.1
10	314	1	411	448	42.9453	490.9453	90	<0.1
11	382	2	453.3151	490.3151	28.5115	518.8266	64.3151	<0.1
12	402	1	475.3151	518.1964	28.5115	546.7079	72.1964	<0.1
13	426	1	509.0777	546.0777	35.8389	581.9166	76.0777	<0.1
14	470	2	531.0777	581.2864	37.3438	648.6302	67.2864	<0.1
1	514	2	573	648	20.6302	668.6302	90	**
2	534	2	631	668	24.6302	692.6302	90	<0.1
3	558	1	655	692	36.6641	728.6641	90	<0.1
4	602	2	677	728.0339	23.8634	751.8973	82.0339	<0.1
5	646	1	705	751.2671	37.8265	789.0936	61.2671	<0.1
6	666	2	751.4634	788.4634	24.9705	813.4339	78.4634	<0.1
7	710	1	773.4634	812.8037	37.8265	850.6302	58.8037	<0.1
8	754	2	813	850	39.0664	889.0664	52	<0.1
9	778	1	837	888.4362	28.1953	916.6315	66.4362	<0.1
10	798	1	879.0013	916.0013	23.2397	939.241	74.0013	<0.1
11	842	2	901.6108	938.6108	24.0194	962.6302	52.6108	<0.1
12	866	2	925	962	24.0194	986.0194	52	<0.1
13	886	1	947	985.3892	24.0194	1009.4086	55.3892	<0.1
14	910	1	969	1008.7784	24.0194	1032.7978	54.7784	<0.1
15	930	1	991	1032.1676	24.0194	1056.187	58.1676	<0.1
16	954	2	1018.5568	1055.5568	24.0194	1079.5762	57.5568	<0.1
17	974	2	1041.946	1078.946	24.0194	1102.9654	60.946	<0.1
18	998	1	1063.946	1102.3352	24.0194	1126.3546	60.3352	<0.1
19	1018	1	1088.7244	1125.7244	27.1379	1152.8623	63.7244	<0.1
20	1042	2	1115.2321	1152.2321	36.7771	1189.0092	66.2321	<0.1
21	1062	1	1151.379	1188.379	36.7771	**1225.1561**	82.379	<0.1

The analysis study (Sect. 6.2) allows to better tune the parameters. Table 9 summarizes the parameters settings for RGA.

At a first stage and based on the above tuned parameters, we have tested the performance of our RGA on some HFS benchmark taken from [33] and another generated one.

We have adapted the RGA algorithm to the HFS model in order to solve a set of instances and their results are presented in Table 10.

As it was noted above, a SCC problem is a particular HFS problem with 3 stages. A HFS problem is such that jobs are single and are not belonging to a sequence. The batch is simply a set of jobs while in a SCC configuration, a batch is represented by a set of sequences of charges to process.

Table 10 presents the performance of RGA with regards to HCGA presented in [33]. To do so, we solve each combination of the total number of jobs in the system ($N = 60, 80, 100, 120$ and 150 and total number of

TABLE 12. A RGA results for CC_2 for SCC($[14, 21] \times [5, 5, 5]$).

Charge (k, c)	$x^1_{k,c}$	CV_i	$x^2_{k,c}$	$x^3_{k,c}$	$p^3_{k,c}3$	Comp. time	Sojourn	cont.
1	15	2	74	121	41.5	162.5	62	**
2	35	1	115.5976	162.5976	58.1391	220.7367	83.5976	<0.1
3	103	1	173.8343	220.8343	58.1391	278.9734	73.8343	<0.1
4	147	2	232.071	279.071	59.959	339.03	88.071	<0.1
5	211	1	292.1276	339.1276	59.959	399.0866	84.1276	<0.1
1	279	2	382.1842	429.1842	44.1837	503.3679	106.1842	**
2	367	2	426	473.4655	61.8963	535.3618	62.4655	<0.1
3	387	1	458	535.4594	61.8963	597.3557	104.4594	<0.1
4	475	1	534	597.4533	61.8963	659.3496	78.4533	<0.1
5	519	1	578	659.4472	63.5079	722.9551	96.4472	<0.1
1	607	2	706.0527	753.0527	39.123	822.1757	102.0527	**
2	651	2	738.0527	792.2733	54.8113	847.0846	97.2733	<0.1
3	719	2	800.1822	847.1822	54.8113	901.9935	84.1822	<0.1
4	763	1	855.0911	902.0911	54.8113	956.9024	95.0911	<0.1
5	851	2	910	957	54.8113	1011.8113	62	<0.1

machines $m = m_1 + m_2 + m_3 = 4, 8, 12, 16$ and 20) with no specification on the number of machines m_r per stage $r = 1, \ldots, 3$.

As we have noted above, a SCC problem is a HFS (FFS) problem where jobs are single and are not belonging to a sequence. The batch is simply a set of jobs where in a SCC configuration, a batch is a set of sequences of charges to process.

Table 10 presents the performance of RGA with regards to HCGA presented in [33]. To do so, we solve each combination of number of jobs n=60, 80, 100,120 and 150 and number of machines $m = 4, 8, 12, 16$ and 20. We report the solution obtained by both CGHA and RGA in columns 2 and 3. Column 4 represents the improvement rate recorded by RGA with regards to HCGA. The average CPU times consumed by both HCGA and RGA are shown in columns 6 and 7.

We report the solutions obtained by both CGHA and RGA in columns 2 and 3. Column 4 represents the improvement rate recorded by RGA with regards to HCGA. The average CPU times consumed by both HCGA and RGA are shown in columns 5 and 6.

For medium instances, RGA obtain on average the same solution found by CGHA but in a very small time while for large size instances RGA performs better and in a small time as well. The improvement average for the solution is about 5.6% (column 4) and the improvement average for the CPU consuming time is about 25% (column 7) which shows that RGA is much more faster than HCGA.

For each Tables (11 and 15), column 1 represents the charge as a couple representing the dedicated CC and the charge number of a sequence, columns 2-4-5 represent the computed start times $x^r_{k,c}$ ($r = 1, \ldots, 3$) at each stage (CV_i-RS_j-CC_k) of a charge. Column 3 shows the CV_i number for which a charge is assigned at the first stage. Column 6 represents the computed processing time $p^3_{k,c}$ of a charge at the last stage CC_k. Columns 7-8-9 represent respectively the completion time, the sojourn (transit) time and the continuity constraints for each charge c dedicated to CC_k.

We have run our numerical tests in order to study the behavior of the GA algorithm when its is running on problem instances of several types and configurations.

From Tables 11 and 12, the maximum completion times for both CC devices are respectively equal to 1225.15 and 1011.81. Thus the makespan $C_{\max}$ is equal to 1225.15 min of the production order. This efficient solution is reached for the charge number 21 of the sequence number 2 on CC_1 device. Our approach was able to compute the obtained solution with a CPU time of 2321.60 s.

TABLE 13. A RGA results for CC_1 for SCC($[15, 15, 19] \times [7, 10, 8] \times [5, 7, 12]$).

Charge (k, c)	$x^1_{k,c}$	CV_i	$x^2_{k,c}$	$x^3_{k,c}$	$p^3_{k,c}3$	Comp. time	Sojourn	Cont.
1	10	1	69	106	21.3	127.3	52	**
2	24	3	89	127.24	41.094	168.334	59.24	<0.1
3	68	1	127	168.27	29.795	198.065	56.27	<0.1
4	104	2	163	198	37.385	235.385	50	<0.1
5	112	3	200.32	235.32	29.795	265.115	79.32	<0.1
6	148	2	225	265.05	29.795	294.845	73.05	<0.1
7	186	2	245	294.79	29.795	324.585	64.79	<0.1
8	200	1	289.52	324.52	29.795	354.315	80.52	<0.1
9	244	3	309.52	354.25	27.944	382.194	66.25	<0.1
10	274	3	333	382.13	27.944	410.074	64.13	<0.1
11	288	1	353	410.01	27.944	437.954	78.01	<0.1
12	324	3	402.89	437.89	27.944	465.834	69.89	<0.1
13	332	2	430.77	465.77	35.272	501.042	89.77	<0.1
14	368	1	465.98	500.98	35.272	536.252	88.98	<0.1
15	412	2	485.98	536.19	16.647	602.837	80.19	<0.1
1	450	3	567.78	602.78	22.556	625.336	108.78	**
2	464	1	590.27	625.27	22.556	647.826	117.27	<0.1
3	500	2	612.76	647.76	23.296	671.056	103.76	<0.1
4	508	1	636	671	22.565	693.565	119	<0.1
5	544	3	658.5	693.5	22.565	716.065	105.5	<0.1
6	552	1	681	716	23.527	739.527	120	<0.1
7	582	2	701	739.46	22.672	762.132	113.46	<0.1
8	632	3	721	762.07	22.672	784.742	86.07	<0.1
9	670	1	749.68	784.68	22.672	807.352	70.68	<0.1
10	684	2	772.29	807.29	23.452	830.742	79.29	<0.1
11	714	2	795.68	830.68	23.452	854.132	72.68	<0.1
12	758	1	819.07	854.07	23.452	877.522	52.07	<0.1
13	772	1	842.46	877.46	39.825	917.285	61.46	<0.1
14	808	3	882.22	917.22	23.452	940.672	65.22	<0.1
15	816	1	905.61	940.61	23.452	1014.062	80.61	<0.1
1	904	3	979	1014	25.474	1039.474	66	**
2	940	2	999	1039.4	23.452	1062.852	55.4	<0.1
3	948	3	1027.8	1062.8	39.825	1102.625	70.8	<0.1
4	992	1	1067.6	1102.6	39.825	1142.425	66.6	<0.1
5	1028	3	1107.3	1142.3	39.825	1182.125	70.3	<0.1
6	1080	2	1147.1	1182.1	23.452	1205.552	58.1	<0.1
7	1110	3	1169	1205.5	23.452	1228.952	51.5	<0.1
8	1124	1	1193.9	1228.9	23.452	1252.352	60.9	<0.1
9	1154	2	1217.3	1252.3	23.452	1275.752	54.3	<0.1
10	1168	1	1240.6	1275.6	26.571	1302.171	63.6	<0.1
11	1198	2	1267.1	1302.1	36.21	1338.31	60.1	<0.1
12	1212	1	1303.3	1338.3	36.21	1374.51	82.3	<0.1
13	1242	3	1339.4	1374.4	23.452	1397.852	88.4	<0.1
14	1292	2	1362.8	1397.8	23.452	1421.252	61.8	<0.1
15	1300	3	1386.2	1421.2	23.452	1444.652	77.2	<0.1
16	1344	2	1409.6	1444.6	23.452	1468.052	56.6	<0.1
17	1374	3	1433	1468	26.571	1494.571	50	<0.1
18	1388	2	1453	1494.5	36.21	1530.71	62.5	<0.1
19	1424	3	1495.7	1530.7	36.21	1566.91	62.7	<0.1

TABLE 14. A RGA results for CC_2 for SCC($[15, 15, 19] \times [7, 10, 8] \times [5, 7, 12]$).

Charge (k, c)	$x^1_{k,c}$	CV_i	$x^2_{k,c}$	$x^3_{k,c}$	$p^3_{k,c}3$	Comp. time	Sojourn	Cont.
1	25	1	84	124	43.85	167.85	54	**
2	69	2	128	168	69.734	237.734	54	<0.1
3	127	3	186	237.88	61.966	299.846	65.88	<0.1
4	201	2	260	300	59.907	359.907	54	<0.1
5	251	3	320.06	360.06	71.918	431.978	64.06	<0.1
6	333	1	392.12	432.12	46.94	479.06	54.12	<0.1
7	377	3	439.21	479.21	74.243	583.453	57.21	<0.1
1	427	1	543.61	583.61	74.243	657.853	111.61	**
2	559	1	618	658	61.844	719.844	54	<0.1
3	611	2	679.99	719.99	61.844	781.834	63.99	<0.1
4	647	2	741.99	781.99	39.071	821.061	89.99	<0.1
5	699	1	781.21	821.21	54.759	875.969	77.21	<0.1
6	735	3	836.12	876.12	54.759	930.879	96.12	<0.1
7	817	2	891.03	931.03	54.759	985.789	69.03	<0.1
8	867	1	945.94	985.9	54.759	1040.659	73.9	<0.1
9	905	3	1000.8	1040.8	39.071	1079.871	90.8	<0.1
10	949	2	1040.1	1080.1	54.759	1164.859	86.1	<0.1
1	993	3	1125	1165	54.759	1219.759	127	**
2	1037	1	1179.9	1219.9	54.759	1274.659	137.9	<0.1
3	1175	2	1234.8	1274.8	54.759	1329.559	54.8	<0.1
4	1219	1	1289.7	1329.7	39.071	1368.771	65.7	<0.1
5	1227	3	1328.9	1368.9	54.759	1423.659	96.9	<0.1
6	1307	2	1383.8	1423.8	54.759	1478.559	71.8	<0.1
7	1351	3	1438.7	1478.7	54.759	1533.459	82.7	<0.1
8	1389	1	1493.6	1533.6	55.942	**1589.542**	99.6	<0.1

From Tables (13–15), the maximum completion times for the three CC devices are respectively equal to 1566.91, 1589.54 and 1470.58. The makespan $C_{\max}$ is then obtained for the value 1589.54 min of the production order. The sojourn time of this charge is equal to 99.6 min while the maximum allowed sojourn time is equal to 150 min. The makespan corresponds to the charge number 8 of the sequence number 3 of the production order on CC_2. The solution is computed within a CPU time of 1267 s.

One can also remark that the continuity constraints are met for all the tests (col. 9) where the gap is less that 0.1. This ensures computing feasible schedules for all the system.

In Table 16, column 1 represents the sequence number dedicated to a CC, column 2 represents the charge number c of a sequence noted as a couple (k, c), columns 3-5-6 represent the start times at each stage (CV_i-RS_j-CC_k) of a charge $c = 1, \ldots, n_k$. Column 4 represents the CV number for which a charge c is assigned to at the first stage. Column 7 represents the processing time computed for a charge at the last stage CC. Columns 8-9-10 represent respectively the completion time ($C_{\max}$), the sojourn (transit) time and the continuity constrains for each charge. Column 11 represents the setup time between two successive sequences for the same CC device. Column 12 represents the dedicated CC machine to which a charge of a sequence is assigned. Moreover, we can remark that all the obtained continuity constraints are mostly met with an acceptable charges continuity.

The solutions are presented in Table 16 and the efficient makespan for the problem is equal to $C_{\max}$= 2605.966 and was obtained for a total CPU of 1750 s. The sojourn time of this charge is of 128 min where the maximum allowed sojourn time for the CC_2 is equal to 150 min. This efficient makespan is delivered for the processing termination time of the charge number 4 of the sequence number 2 dedicated to CC_2 machine at the last stage.

In the Table 16, the RGA approach was running on a set of production orders with a 4 CC devices system at the last stage.

TABLE 15. A RGA results for CC_3 for SCC($[15, 15, 19] \times [7, 10, 8] \times [5, 7, 12]$).

Charge (k, c)	$x^1_{k,c}$	CV_i	$x^2_{k,c}$	$x^3_{k,c}$	$p^3_{k,c}3$	Comp. time	Sojourn	cont.
1	7	3	77	118	53.2	171.2	59	**
2	54	3	121	171.14	71.924	243.064	65.14	<0.1
3	136	1	203	243	73.08	316.08	55	<0.1
4	186	1	253	316.02	62.056	378.076	78.02	<0.1
5	230	1	338.01	378.01	62.056	440.066	96.01	<0.1
1	312	2	398	470.01	62.056	532.066	106.01	**
2	370	2	461	532	44.062	576.062	110	<0.1
3	414	2	486	576	42.311	618.311	110	<0.1
4	488	1	574	618.25	40.283	658.533	78.25	<0.1
5	532	2	599	658.47	54.971	713.441	74.47	<0.1
6	590	1	673.38	713.38	54.971	768.351	71.38	<0.1
7	620	3	698.38	768.29	54.971	823.261	96.29	<0.1
1	714	3	813.2	853.2	54.971	908.171	87.2	**
2	796	3	868.11	908.11	54.971	963.081	60.11	<0.1
3	846	1	923.01	963.01	39.283	1002.293	65.01	<0.1
4	854	1	962.23	1002.23	39.283	1041.513	96.23	<0.1
5	884	2	1001.46	1041.5	54.971	1096.471	105.5	<0.1
6	978	2	1045	1096.4	54.971	1151.371	66.4	<0.1
7	1030	1	1111.3	1151.3	54.971	1206.271	69.3	<0.1
8	1066	3	1166.2	1206.2	54.971	1261.171	88.2	<0.1
9	1110	2	1221.1	1261.1	54.971	1316.071	99.1	<0.1
10	1154	1	1276	1316	71.062	1387.062	110	<0.1
11	1280	1	1347	1387	44.364	1431.364	55	<0.1
12	1324	3	1391.3	1431.3	39.283	1470.583	55.3	<0.1

The numerical results detailed in Table 16 show that RGA obtains the solutions in less than 1 s for the first set of instances (small ones) (see Tab. 9). On the other hand, RGA computes the solution in a CPU time of 2105.31 s for instances that comprise up to 144 charges partitioned in 2 sequences and for a 2 CC system. However, for larger problems instances one should allow extra time to run the RGA and for big sets. For all these , there were no generated discontinuities (the gap between the end time and start time of a charge on a CC for two successive charges). We remark that there is no charge that has been exceeding the maximum allowed transit time $T_{k,c}$.

Table 17 shows that RGA succeed to obtain an efficient solution within a CPU time equal to 85.31 s for the case of a 2 CC with 2 sequences of a total of 144 charges. While the algorithm was able to compute the solution within a CPU time of 1678.23 s for the case of a 3 sequences of a total of 100 charges for a 3 CC system. Also, one can remark that the problem becomes big time consuming for the problems with more than 2 CC system.

Based on the tables above, we have shown the performance of the proposed RGA for several SCC instances and with different parameters.

We also have demonstrated that in the most cases the obtained solution gives the best trade-off between an acceptable running time, a minimal inter-sequence dependent setup times and minimal discontinuities. Also RGA ensures that in all cases, the maximum allowing sojourn time is respected.

7. CONCLUSION

This article considers a novel model to study a particular configuration of the SCC that is the inter-sequence dependent setup times SCC planning and scheduling problem. It is able to assign and sequence a big number of batches into several continuous casting devices. The problem can be seen as a particular case of the hybrid

TABLE 16. A RGA results for a 4 CC with large problem SCC($[5,5,8]\times[6,4]\times[5,4,5]\times[6,6,7]$).

Seq. s_k	Charge (k,c)	$x^1_{k,c}$	CV$_i$	$x^2_{k,c}$	$x^3_{k,c}$	$p^3_{k,c}$	Comp. time	Sojourn	Cont.	σ_{s_k}	CC$_k$
1	1	2	1	72	135	64.967	199.967	88			1
	2	27	2	97	200	64.579	264.579	128	0.033		1
	3	132	1	202	265	64.589	329.589	88	0.421		1
	4	177	2	247	330	64.079	394.079	108	0.411		1
	5	222	3	292	485	64.546	639.546	128	90	90	1
2	1	327	2	397	550	64.888	614.888	88			1
	2	392	2	462	615	64.489	679.489	88	0.112		1
	3	437	1	507	680	64.986	744.986	108	0.511		1
	4	487	3	557	745	64.996	809.996	123	0.014		1
	5	587	1	657	855	64.259	964.259	88	45	45	1
3	1	627	2	697	920	64.67	984.67	113			1
	2	717	3	787	985	64.454	1049.454	88	0.33		1
	3	767	2	837	1050	64.375	1114.375	103	0.546		1
	4	807	3	877	1115	64.329	1179.329	128	0.625		1
	5	1002	2	1072	1180	95.982	1275.982	88	0.671		1
	6	1067	1	1137	1275	95.305	1370.305	118	−0.982		1
	7	1157	3	1227	1370	95.448	1465.448	123	−0.305		1
	8	1257	1	1327	1510	95.581	1650.581	118	45	45	1
1	1	1342	2	1412	1605	95.677	1700.677	128			2
	2	1437	2	1507	1700	95.156	1795.156	128	−0.677		2
	3	1632	1	1702	1795	85.985	1880.985	88	−0.156		2
	4	1677	3	1747	1880	85.863	1965.863	128	−0.985		2
	5	1782	3	1852	1965	85.994	2050.994	108	−0.863		2
	6	1882	2	1952	2110	85.334	2255.334	93	59.006	60	2
2	1	1937	2	2007	2195	85.932	2280.932	123			2
	2	2017	1	2087	2280	85.721	2365.721	128	−0.932		2
	3	2142	1	2212	2365	85.942	2450.942	88	−0.721		2
	4	2187	2	2257	2485	85.966	**2605.966**	128	34.058	35	2
1	1	132	1	202	305	50.188	355.188	128			3
	2	182	3	252	355	50.392	405.392	128	−0.188		3
	3	237	3	307	405	50.492	455.492	123	−0.392		3
	4	282	1	352	455	50.652	505.652	128	−0.492		3
	5	332	1	402	550	50.422	600.422	128	44.348	45	3
2	1	482	3	552	600	50.019	650.019	88			3
	2	532	2	602	650	50.223	700.223	88	−0.019		3
	3	582	2	652	700	50.282	750.282	88	−0.223		3
	4	632	1	702	785	50.93	940.93	88	0.718	35	3
3	1	672	1	742	802	50.417	955.417	98			3
	2	752	1	822	852	60.915	1015.915	128	−0.417		3
	3	812	2	882	912	60.594	1075.594	128	−0.915		3
	4	912	1	982	972	60.251	1135.251	88	−0.594		3
	5	932	2	1002	1062	60.511	1255.511	128	29.749	30	3
1	1	992	3	1062	1122	60.875	1285.875	128			4
	2	1092	2	1162	1182	60.639	1345.639	88	−0.875		4
	3	1112	2	1182	1242	60.389	1405.389	128	−0.639		4
	4	1212	2	1282	1302	60.414	1465.414	88	−0.389		4
	5	1292	2	1362	1362	50.684	1515.684	128	−0.414		4
	6	1342	3	1412	1472	50.426	1685.426	128	59.316	60	4

TABLE 16. (Continued)

Seq. s_k	Charge (k,c)	$x^1_{k,c}$	CV_i	$x^2_{k,c}$	$x^3_{k,c}$	$p^3_{k,c}$	Comp. time	Sojourn	Cont.	σ_{s_k}	CC_k
2	1	1392	1	1462	1522	50.074	1675.074	128			4
	2	1482	3	1552	1572	50.343	1725.343	88	−0.074		4
	3	1492	1	1562	1622	50.834	1775.834	128	−0.343		4
	4	1542	3	1612	1672	50.354	1825.354	128	−0.834		4
	5	1677	1	1762	1722	55.11	1880.11	103	−0.354		4
	6	1747	1	1817	1807	55.522	1995.522	88	29.89	30	4
3	1	1802	2	1872	1862	55.24	2020.24	88			4
	2	1827	3	1927	1917	55.068	2075.068	118	−0.24		4
	3	1872	1	1942	1972	55.866	2130.866	128	−0.068		4
	4	1927	3	1997	2027	55.6	2185.6	128	−0.866		4
	5	1982	3	2052	2082	55.026	2240.026	128	−0.6		4
	6	2037	3	2117	2137	55.306	2295.306	128	−0.026		4
	7	2092	1	2162	2192	55.754	2350.754	128	−0.306		4

TABLE 17. RGA experiment results for different CC systems.

# Inst.	Batch	Best $C_{\max}$	CPU (sec)	Tot. charges
1	$(5) \times (5) \times (5)$	427.58	1.09	15
2	$(14,21) \times (5,5,5)$	1225.15	2.12	50
3	$(15,10,12) \times (5,5,5)$	1559.00	5.75	52
4	$(15,14,30) \times (5,5,5)$	1807.01	6.55	74
5	$(10,13,17,8) \times (5,5,5,5,5)$	1756.72	40.99	73
6	$(18,11,23,15,27) \times (5,5,5,5,5,5,5,5,5,5)$	2506,04	85.31	144
7	$(10,10) \times (10,10) \times (10,10)$	706.07	7.45	60
8	$(15,15,19) \times (7,10,8) \times (5,7,12)$	1589.54	8.31	98
9	$(5,15,5,10) \times (10,10,10,5) \times (5,5,5,5)$	1239.34	67.23	90
10	$(5,10,5,10,5) \times (10,10,10,5,5) \times (5,5,5,5,5)$	1678.23	217.12	100
11	$(5,5,5) \times (6,4) \times (5,4,5) \times (6,6,7)$	2605.96	90.86	58
12	$(10,10,10,10) \times (10,10,10,) \times (5,5,5) \times (5,5)$	1543.27	387.12	95
13	$(5,5,5,5,5) \times (10,5,10,5) \times (5,5,5) \times (5,5) \times (10,5)$	1278.65	567.35	95

flowshop problem (HFS) with 3 stages which is also known as the flexible flowshop problem. The objective stated as the Makespan is to minimize the maximum completion time of the last charge of the last sequence while considering the inter-sequence dependent setup times between sequences at the last stage. We developed a generalized model that could consider any number of the underlying devices at each stage and for any number of the dedicated sequences. One of the remarks that we can make is that the studied problem is very difficult and handles a huge number of decision variables that make it intractable for big size instances with a production bottleneck at the last stage.

The proposed RGA approach provides suitable operations to avoid infeasible solutions by including two informations in each gene (k,c) that are the controlling and coding operations. The first information controls coding of the genes. The second information contains the genetic information. These strategies show the flexibility and ability to produce efficient solutions for the studied problem.

We have recorded some promising results by considering setup times that exist between sequences. In fact, setup times are only to take into account for the last charge for a sequence and the first charge of the next sequence to process on the same CC device. We have ensured that: (i) the sojourn time must be satisfied, (ii) the setup times must be minimized and (iii) the continuity constraints must be satisfied. The RGA approach

that we developed gives satisfactory results in terms of the number of the processed sequences with different sizes and densities. Our approach were specifically tailored for the studied problem with specific chromosomes encoding and specific evolutionary operators (crossover, mutation,...). The algorithm was set to run on both real life similar problems and other randomly generated ones to establish the limits of the run time.

To overcome the complexity and the hardness of the SCC, some future research may consider to embed an high performance computing based GPU to minimize the total run time to be able to solve very large instances for the SCC industrial system.

References

[1] A. Atighehchian, M. Bijari and H. Tarkesh, A novel hybrid algorithm for scheduling steelmaking continuous casting production. *Comput. Oper. Res.* **36** (2009) 2450–246.

[2] S. Basu and G. Dutta, A Survey of the Non-Optimization techniques used in an integrated steel plant. *Manag. Dyn.* **6** (2006) 33–68.

[3] A. Bellabdaoui and J. Teghem, A mixed-integer linear programming model for the continuous casting planning. *Int. J. Prod. Econom.* **104** (2006) 260–270.

[4] A. Bellabdaoui, A. Fiordaliso and J. Teghem, A heuristic algorithm for scheduling the steelmaking continuous casting process. *Pac. J. Optim.* **1** (2005) 447–464.

[5] J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt and J. Weglarz, Scheduling Computer Manufacturing Processes. Springer (1996).

[6] N. Chakraborti, R. Kumar and D. Jain, A study of the continuous casting mold using a pareto-converging genetic algorithm. *Appl. Math. Model.* **25** (2001) 287–297.

[7] P.C. Chang and S.H. Chen, Integrating Dominance Properties with Genetic Algorithms for Parallel Machine Scheduling Problems with Setup Times. *Appl. Soft Comput.* **11** (2011) 1263–1274.

[8] L. Chen, N. Bostel, P. Dejax, J.C. Cai and L.F. Xi, A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. *Eur. J. Oper. Res.* **181** (2007) 40–58.

[9] P.I. Cowling, D. Ouelhadj and S. Petrovic, Dynamic scheduling of steel casting and mill using multi-agents. *Prod. Plan. Control* **15** (2004) 495–501.

[10] B. de Schutter, Designing optimal timing and sequencing strategies for a continuous steel foundry, in *Proceedings of the European Control Conference 1999 (ECC'99), Karlsruhe, Germany, Paper 160/BP–2.6, Aug.–Sept.* (1999).

[11] G. Dutta and R. Fourer, A survey of mathematical programming application in integrated steel plants. *Manuf. Service Oper. Manag.* **3** (2001) 387–400.

[12] I. Ferretti, S. Zanoni and L. Zavanella, Production-inventory scheduling using ant system metaheuristic. *Int. J. Prod. Econom.* **104** (2008) 317–326.

[13] M. Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of Np-Completness. W.H. Freeman and Company, San Francisco (1979).

[14] J.N.D. Gupta, Two-stage, hybrid flowshop scheduling problem. *J. Oper. Res. Soc.* **39** (1988) 359–364.

[15] I. Harjunkoski and I.E. Grossmann, A decomposition approach for the scheduling of a steel plant production. *Comput. Chem. Eng.* **25** (2001) 1647–1660.

[16] M. Helal, G. Rabadi and A. Al-Salem, A tabu search algorithm to minimize the makespan for the unrelated parallel machines scheduling problem with setup times. *Int. J. Oper. Res.* **3** (2006) 182–192.

[17] J.R. Kalagnanam, M.W. Dawande, M. Trumbo and H.S. Lee, The surplus inventory matching problem in the process industry. *Oper. Res.* **48** (2000) 505–516.

[18] C.-H. Ko and S.-F. Wang, Precast production scheduling using multi-objective genetic algorithms. *Expert Syst. Appl.* **38** (2011) 8293–8302.

[19] H.S. Lee, S.S. Murthy, S.W. Haider and D.V. Morse, Primary production scheduling at steelmaking industries. *IBM J. Res. Develop* **40** (1996) 231–252.

[20] K. Lee, S.Y. Chang and Y. Hong, Continuous slab caster scheduling and interval graphs. *Prod. Plan. Control* **15** (2004) 495–501.

[21] L. Li, Q. Tang, P. Peng Zheng, L. Zhang and C.A. Floudas, An improved self-adaptive genetic algorithm for scheduling steelmaking continuous casting production, in *Proceedings of the 6th International Asia Conference on Industrial Engineering and Management Innovation (IEMI2015), Core Theory and Applications of Industrial Engineering, 1: 399–410, Tianjin, July 25–26th* (2015).

[22] R. Linn and W. Zhang, Hybrid flow shop scheduling: a survey. *Comput. Ind. Eng.* **31** (1999) 57–61.

[23] H. Missbauer, W. Hauber and W. Stadler, A scheduling system for the steelmaking-continuous casting process. A case study from the steel-making industry. *Int. J. Prod. Res.* **47** (2009) 4147–4172.

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

[25] T. Nishi, Y. Hiranaka and M. Inuiguchi, Lagrangian relaxation with cut generation for hybrid flow shop scheduling problems to minimize the total weighted tardiness. *Comput. Oper. Res.* **37** (2010) 189–198.

[26] D. Pacciarelli and M. Pranzo, Production scheduling in a steelmaking-continuous casting plant. *Comput. Chem. Eng.* **28** (2004) 2823–2835.
[27] Q.K. Pan, An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling. *Eur. J. Oper. Res.* **250** (2016) 702–714.
[28] Q.K. Pan, L. Wang, K. Mao, J.H. Zhao and M. Zhang, An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process. *IEEE Trans. Autom. Sci. Eng.* **10** (2013) 307–322.
[29] C. Rajendran and D. Chaudhuri, A multi-stage parallel processor flowshop problem with minimum flowtime. *Eur. J. Oper. Res.* **57** (1992) 11–122.
[30] R. Ruiz and C. Maroto, A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *Eur. J. Oper. Res.* **169** (2006) 781–800.
[31] A. Sbihi, A. Bellabdaoui and J. Teghem, Solving a mixed integer linear program with times setup for the steel-continuous casting planning and scheduling problem. *Int. J. Prod. Res.* **52** (2014) 7276–7296.
[32] H. Sherali, S. Sarin and M. Kodialam, Models and algorithms for a two-stage production process. *Prod. Plan. Control* **1** (1990) 27–39.
[33] D.F. Shiau, S.C. Cheng and Y.M. Huang, Proportionate flexible flow shop scheduling via a hybrid constructive genetic algorithm. *Expert Syst. Appl.* **34** (2008) 1133–1143.
[34] L. Tang, and G. Wang, Decision Support system for the batching problems of steelmaking and continuous-casting production. *Omega Int. J. Manag. Sci.* **36** (2008) 976–991.
[35] L. Tang, J. Liu, A. Rong and Z. Yang, A mathematical programming model for scheduling steelmaking-continuous casting production. *Eur. J. Oper. Res.* **120** (2000) 423–435.
[36] L. Tang, J. Liu, A. Rong and Z. Yang, A review of planning and scheduling systems and methods for integrated steel production. *Eur. J. Oper. Res.* **133** (2001) 1–20.
[37] L. Tang, P.B. Luh, J. Liu and L. Fang, Steel-making process scheduling using Lagrangian relaxation. *Int. J. Prod. Res.* **40** (2002) 55–70.
[38] L. Tang, H. Xuan and J. Liu, A new lagrangian relaxation algorithm for hybrid flow shop scheduling to minimize total weighted completion time. *Comput. Oper. Res.* **33** (2006) 3344–3359.
[39] L. Tang, X. Wang and J. Liu, Color-coating production scheduling for coils in inventory in steel industry. *Autom. Sci. Eng. IEEE Trans.* **5** (2008) 544–549.
[40] W.S. Um, Computer simulation of the steelmaking process with ARENA. *J. Korean Soc. Maint. Eng.* **7** (2002) 77–90.
[41] H. Xuan and L. Tang, Scheduling a hybrid flow shop with batch production at the last stage. *Comput. Oper. Res.* **34** (2007) 2178–2733.
[42] J. Yang, H. Che, F.P. Dou and T. Zhou, Genetic algorithm-based optimization used in rolling schedule. *J. Iron Steel Res. Int.* **5** (2008) 18–22.
[43] V. Yaurima, L. Burtseva and A. Tchernykh, Hybrid flowshop with unrelated machines, sequence dependent setup time, availability constraints and limited buffers. *Comput. Ind. Eng.* **56** (2009) 1452–1463.
[44] D.F. Zhu, Z. Zheng and X.Q. Gao, Intelligent optimization-based production planning and simulation analysis for steelmaking and continuous casting process. *J. Iron Steel Res. Int.* **17** (2010) 19–24.