New Crossover Operator in a Hybrid Genetic Algorithm for the Single Machine Scheduling Problem with Sequence-dependent Setup Times

Aymen Sioud[1], Marc Gravel[1] and Caroline Gagné[2]
[1]*Département d'Informatique et Mathématique, Université du Québec à Chicoutimi, 555 Boulevard Université, Chicoutimi, Canada*
[2]*Département des Sciences Économiques et Gestion, Université du Québec à Chicoutimi, 555 Boulevard Université, Chicoutimi, Canada*

Keywords: Genetic Algorithm, Hybrid Crossover, Constrained based Scheduling, Total Tardiness, Single Machine.

Abstract: This paper presents a new crossover operator based on Constraint Based Scheduling (CBS) approach in a Genetic Algorithm (GA) for solving a scheduling problem. The proposed hybrid crossover, noted as HCX, is applied in Hybrid Genetic Algoritym (HGA) to a single machine scheduling problem with sequence-dependent setup times for the objective of minimizing the total tardiness. Through numerical experiments we compare the performance of the GA and the HGA approaches on different benchmarks from the literature. These results indicate that the HGA is very competitive and generates solutions that approach those of the known reference sets.

1 INTRODUCTION

In a survey of industrial projects, (Conner, 2009) has pointed out, in more 250 projects, that 50% of these projects contain sequence-dependent setup times, and when these setup times are well applied, 92% of the order deadline could be met. Production of good schedules often relies on management of these setup times, and decision makers must therefore organize the job scheduling by trying to minimize downtime while respecting the different deadlines (Pinedo, 2002; Allahverdi et al., 2008; Zhu and Wilhelm, 2006).

The single machine problem has been used in the literature to investigate scheduling issues relating to more complex scheduling problem. But, in many industries, including the pharmaceutical industry, metallurgical production, electronics and automotive manufacturing, the production system can limited by a machine bottleneck, where scheduling might be done by considering only this bottleneck machine (Pinedo, 2002; Leung et al., 2004). This present paper considers the single machine scheduling problem with sequence dependent setup times with the objective to minimize total tardiness of the jobs (SMSDST). This problem, noted as $1|s_{ij}|\Sigma T_j$ in accordance with the notation of (Graham et al., 1979), is an NP-hard problem (Du and Leung, 1990).

The $1|s_{ij}|\Sigma T_j$ may be defined as a set of n jobs available for processing at time zero on a continuously available machine. Each job j has a processing time p_j, a due date d_j, and a setup time s_{ij} which is incurred when job j immediately follows job i. It is assumed that all the processing times, due dates and setup times are non-negative integers. A sequence of the jobs $S = [q_0, q_1,..., q_{n-1}, q_n]$ is considered where q_j is the subscript of the j^{th} job in the sequence. The due date and the processing time of the j^{th} job in sequence are denoted as d_{q_j} and p_{q_j}, respectively. Thus, the completion time of the j^{th} job in sequence will be expressed as $C_{q_j} = \sum_{k=1}^{j}(s_{q_{k-1}q_k} + p_{q_k})$ while the tardiness of the j^{th} job in sequence will be expressed as $T_{q_j} = max(0, C_{q_j} - d_{q_j})$. The objective of the scheduling problem studied is to minimize the total tardiness of all the jobs which will be expressed as $\sum_{j=1}^{n} T_{q_j}$.

In this present paper, we propose a new crossover operator called HCX in a genetic algorithm. The HCX crossover integrates Constraint Based Scheduling (CBS). This approach has become a widely used form for modeling and solving scheduling problems using the constraint programming approach. Computational testing is performed on a set of test problems available from literature. We report on our experimental results and conclude with some remarks and future research directions. As a constraint programming environment, we use the ILOG IBM CP envi-

Sioud A., Gravel M. and Gagné C..
New Crossover Operator in a Hybrid Genetic Algorithm for the Single Machine Scheduling Problem with Sequence-dependent Setup Times.
DOI: 10.5220/0004113101440151
In *Proceedings of the 4th International Joint Conference on Computational Intelligence* (ECTA-2012), pages 144-151
ISBN: 978-989-8565-33-4

ronment using ILOG Solver and ILOG Scheduler via the C++ APIs (ILOG, 2003b; ILOG, 2003a). The use of this kind of platforms has been encouraged by the steady improvement of general purpose solvers over the past decade. Such solvers have become significantly more effective and robust (Yunes et al., 2010).

The body of this paper is organized into five sections. Section 3 presents the used pure GA of (Sioud et al., 2009), the CBS approach and the hybrid crossover *HCX*. The computational testing and discussion are presented in Section 4. Finally, we conclude with some remarks and future research directions.

2 LITERATURE REVIEW

Different approaches have been proposed by a number of researchers to solve the $1|s_{ij}|\Sigma T_j$ problem. (Rubin and Ragatz, 1995) proposed a Branch and Bound approach, which quickly showed its limitations. It could optimally solve only small instances of benchmark files of 15, 25, 35 and 45 jobs proposed by these authors. (Bigras et al., 2008) have optimally solved all instances proposed by (Rubin and Ragatz, 1995) using a Branch and Bound approach with linear programming relaxation bounds. They also demonstrated and used the problem's similarity with the time-dependent traveling salesman problem (TSP). This Branch and Bound approach solved some of these instances in more than 7 days. For their part, (Sioud et al., 2010b) introduce a constraint based programming approach proposing an ILOG API C++ model. This exact approach also shown its limits and fail to resolve small 25 job instances in less than 8 hours. Because this problem is NP-hard, many researchers used a wide variety of metaheuristics to solve this problem, such as a genetic algorithm (Sioud et al., 2009; Franca et al., 2001), a memetic algorithm (Rubin and Ragatz, 1995; Franca et al., 2001; Armentano and Mazzini, 2000), a simulated annealing (Tan and Narasimhan, 1997), a GRASP (Gupta and Smith, 2006) and an ant colonies optimization (ACO) (Gagné et al., 2002; Liao and Juan, 2007). Heuristics such as Random Start Pairwise Interchange (RSPI) (Rubin and Ragatz, 1995) and Apparent Tardiness Cost with Setups (ATCS) (Lee et al., 1997) have also been proposed for solving this problem. More robust methods such the hybrid metaheuristics have also been used. Indeed, (Gagné et al., 2005) introduce a Tabu/VNS which propose a version of the tabu metaheuristic that incorporates variable neighbourhood search. (Sioud et al., 2010a) integrate a CBS approach in a crossover operator using direct precedence constraints to enhance the CBS search. The CBS approach has also been integrated in the intensification search space process using additional constraints. (Sioud et al., 2012) introduce for their part a hybrid genetic algorithm which is based on the integration between a genetic algorithm and concepts from constraint programming, multi-objective evolutionary algorithms and ant colony optimization.

Concerning the pure genetics algorithms (GA), only (Sioud et al., 2009) succeeded in proposing an efficient GA, suggesting that this metaheuristic is not well suited to deal with the specificities of this problem. Indeed, the authors have proposed a GA integrating the RMPX crossover operator which takes greater account of the relative and absolute position of a job. Indeed, (Rubin and Ragatz, 1995; Armentano and Mazzini, 2000; Tan and Narasimhan, 1997) have shown the importance of relative and absolute order positions for solving the $1|s_{ij}|\Sigma T_j$ problem. The proposed GA outdoes the performance of all the GAs found in the literature but is still less efficient than the Tabu/VNS of (Gagné et al., 2005) and the hybrid genetic algorithm of (Sioud et al., 2012) which represent the best approaches found in the literature.

3 THE HYBRID GENETIC ALGORITHM

Several researchers have used metaheuristics variations and hybridizations to improve the effectiveness of these methods (Talbi, 2009; Puchinger and Raidl, 2005). In general, hybridization combines two or more methods in a single algorithm to solve combinatorial optimization problems. (Puchinger and Raidl, 2005) divide hybrid methods into two categories : collaborative and integrative hybridization. The algorithms that exchange information in a sequential, parallel or interlaced way fall into the category of collaborative hybridization. We talk about an integrative hybridization when a technique is an embedded component of another technique. In this paper, we introduce a collaborative hybridization which incorporates CBS approach sequentially in a crossover operator into a genetic algorithm. We introduce first the used pure algorithm genetic, then we describe the *HCX* hybrid crossover.

3.1 Genetic Algorithm

Genetic algorithms are methods based upon biological mechanisms such as the genetic inheritance laws of Mendel and the natural selection concept of

Charles Darwin, where the best adapted species survive. The basic concepts of GA have been described by the investigation carried out by Holland (1992) who explained how to add intelligence into a program computing with the crossover exchange of genetic material and transfer as a source of genetic diversity. In a GA, a population of individuals or chromosomes incurs a sequence of transformations by means of genetic operators to form a new population. Two main operators are used for this purpose : crossover and mutation. Crossover creates new individuals by combining parts of two individuals and mutation creates new individuals by a small change in a single individual.

Based on the GA proposed by (Sioud et al., 2009), we redefine a simple genetic algorithm. A solution is coded as a permutation of the considered jobs. The population size is set to *n* to fit with the considered instance size. The initial population is randomly generated for 60% and also for 20% using a pseudo-random heuristic which favors setup times and promotes a relative order for the jobs. The last 20% is generated using a pseudo-random heuristic which depends on due dates and promotes an absolute order for the jobs. A binary tournament selects the chromosomes for the crossover. The proposed GA uses the OX crossover (Michalewicz, 1996) to generate 30% of offspring and the RMPX crossover (Sioud et al., 2009) to generate the rest of the children population. Both of the OX and RMPX crossover maintain both of the relative and the absolute order positions, but the RMPX crossover seems to give better results. The RMPX crossover can be described in the following steps : *(i)* two parents P1 and P2 are considered and two distinct crossover points *C1* and *C2* are selected randomly, as shown in Figure 1; *(ii)* an insertion point p_ins is then randomly chosen in the offspring E as $p_ins = random\ (n - (C2 - C1))$; *(iii)* the part [*C1*, *C2*] of P1, shaded in Figure 1, is inserted in the offspring E from p_ins. The insertion is to be done from the position 2 showing in Figure 1; and *(iv)* the rest of the offspring E is completed from P2 in order of appearance since its first position.

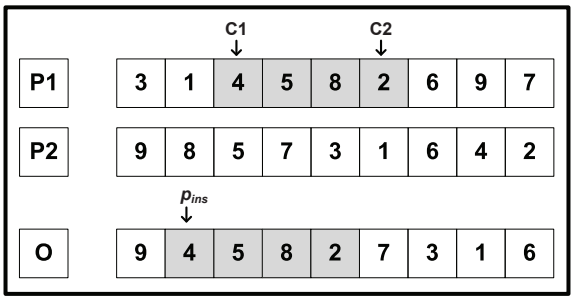

Figure 1: Illustration of RMPX.

The crossover probability *pc* is set to 0.8, i.e. therefore *n*0.8* offspring are generated at each generation in which a mutation is applied with a probability *pm* equal to 0.3. The mutation consists of exchanging the position of two distinct jobs which are randomly chosen. The replacement is elitist and the duplicates individuals in the population were replaced by chromosomes which are generated by one of the pseudo-random heuristics used in the initialization phase. The stop criterion is set to 3000 generations.

3.2 The Constraint based Scheduling

Constraint solving methods such as domain reduction and constraint propagation have proved to be well suited for a wide range of industrial applications (Baptiste et al., 2001). These methods are increasingly combined with classical solving techniques from operations research, such as linear, integer, and mixed integer programming (Talbi, 2002), to yield powerful tools for constraint-based scheduling by adopting them. In the recent years, the CBS has become a widely used form for modeling and solving scheduling problems using the constraint programming approach (Baptiste et al., 2001; Allahverdi et al., 2008). A scheduling problem is the process of allocating tasks to resources over time with the goal of optimizing one or more objectives (Pinedo, 2002). A scheduling problem can be efficiently encoded like a constraint satisfaction problem (CSP).

The activities, the resources and the constraints, which can be temporal or resource related, are the basis for modeling a scheduling problem in a CBS problem. Based on representations and techniques of constraint programming, various types of variables and constraints have been developed specifically for scheduling problems. Indeed, the domain variables may include intervals domains where each value represents an interval (processing or early start time for example) and variable resources for many classes of resources. Similarly, various research techniques and constraints propagation have been adapted for this kind of problem.

In Constraint Based Scheduling, the single machine problem with setup dependent times can be efficiently encoded in terms of variables and constraints in the following way. Let *M* be the single resource. We associate an activity A_j for each job *j*. For each activity A_j four variables are introduced, $start(A_j)$, $end(A_j)$, $proc(A_j)$ and $dep(A_j)$. They represent the start time, the end time, the processing time and the departure time of the activity A_j, respectively. The departure time represents the needed setup time of an activity when the latter starts the schedule.

A setup time s_{ij} is introduced and it is incurred when job j immediately follows job i. In our case, the setup times are activity related and not resource-related. For this purpose, we assign a type to each activity and a lattice to the unary machine. Then, when we calculate the objective function, it is possible to associate the transition times between two distinct types of activities. The tardiness criterion is represented by an additional variable *Tard*. Its value is determined by $Tard = \sum_{A_j=1}^{n} max(end(A_j) - d_{A_j}, 0)$.

(ILOG, 2003a) provides several predefined search algorithms named as *goals* and activity selectors. We used the *IloSetTimesForward* algorithm with the *IloSelFirstActMinEndMin* activity selector. The *IloSetTimesForward* algorithm schedules activities on a single machine forward initializing the start time of the unscheduled activities. The activity selector defines the heuristic scheduling variables representing start times, which chooses the next activity to schedule. The *IloSelFirstActMinEndMin* tries first the activity with the smallest start time and in case of equality the activity with the smallest end time. For his part, (ILOG, 2003b) provides four strategies to explore the search tree : the default *Depth-First Search* (DFS), the *Slice-Based Search* (SBS) (Beck and Perron, 2000), Interleaved Depth-First Search (IDFS) (Meseguer, 1997) and the *Depth-Bounded Discrepancy Search* (DDS) (Walsh, 1997). In this paper, we use the CBS model introduced by (Sioud et al., 2010b) where the *IloSetTimesForward* algorithm with the *IloSelFirstActMinEndMin* activity selector are used driven by the *Depth-Bounded Discrepancy Search* (DDS) (Walsh, 1997) algorithm search engine.

3.3 The Hybrid Crossover

When we handle a basic single machine model, there is no precedence constraint between activities as is the case in a flow-shop or job-shop where adding constraints improves the CBS approach. The main idea of integrating the CBS in a crossover is to provide to this latter precedence constraints between activities when generating offspring. In this work, we consider all the precedence constraints during the introduced new crossover HCX while (Sioud et al., 2010a) considered only the direct constraints between two jobs. Also, (Sioud et al., 2010a) extracted these direct constraints from two selected parents while the HCX crossover extract this information from the whole population. Thereby, using more precedence constraints the HCX crossover extract more information and use it to generate better offspring taking greater account of the relative position of a jobs.

The introduced new crossover HCX can be described in the following steps : *(i)* from a population at time t we build a precedence job matrix noted as *MPREC*; *(ii)* for each job j in *MPREC* we calculate the number of time that j precede the other jobs; *(iii)* using a roulette wheel based on the sum calculated at the second step we choose nbr_{job} jobs; *(iv)* using a threshold t_{job} we extract some precedence constraints from the x considered jobs in the third step; and *(v)* from a random chosen individual in the actual population, the CBS approach tries to solve the problem while adding the precedence constraints built in the previous step and an upper bound consisting of the objective function value of the considered individual. The upper bound is added to discard faster bad solutions when branching during the solver process. As a reminder, the ILOG Solver uses a Branch and Bound approach to solve a problem (ILOG, 2003b). The HCX crossover will be done under probability p_{HCX}.

To better understand how the HCX crossover works, consider the 5-jobs example in Figure 2. The Figure (a) represents a population at time t with five individuals. At the HCX crossover first step the *MPREC* matrix is build from this population as shown in Figure (b). So, we can remark, for example, that the job 2 precedes five times the job 5 and the job 4 precedes only once the job 3. Then, from the *MPREC* matrix and for each job, we calculate the sum of the precedence constraints. As shown in Figure (c), the job 2 have 14 precedence constraints while job 5 have only 5. After, at the third step, using a roulette wheel we choose nbr_{job} jobs. We consider in this example that we choose the two jobs 2 and 4. We also consider a threshold t_{job} equals to 2. Next, in the fourth step, we keep only two precedence constraints without considering mutuals constraints. So, the constraints "*2 before 4*" and "*4 before 2*" will be remove. These two precedence constraints are those with the greatest values. In the example in Figure (d), we keep the two precedence constraints "*2 before 5*" and "*2 before 1*" for the job 2 shaded in gray. Concerning the job 4, we keep the two precedence constraint "*4 before 1*" and "*4 before 5*" also shaded in gray. In the case that we have more than two equal values, we keep randomly only two constraints. Finally, using the CBS approach from a randomly chosen individual from the population we try to solve the problem while adding objective function value of the considered individual and the four precedence constraints : "*2 before 5*", "*2 before 1*", "*4 before 1*" and "*4 before 5*".

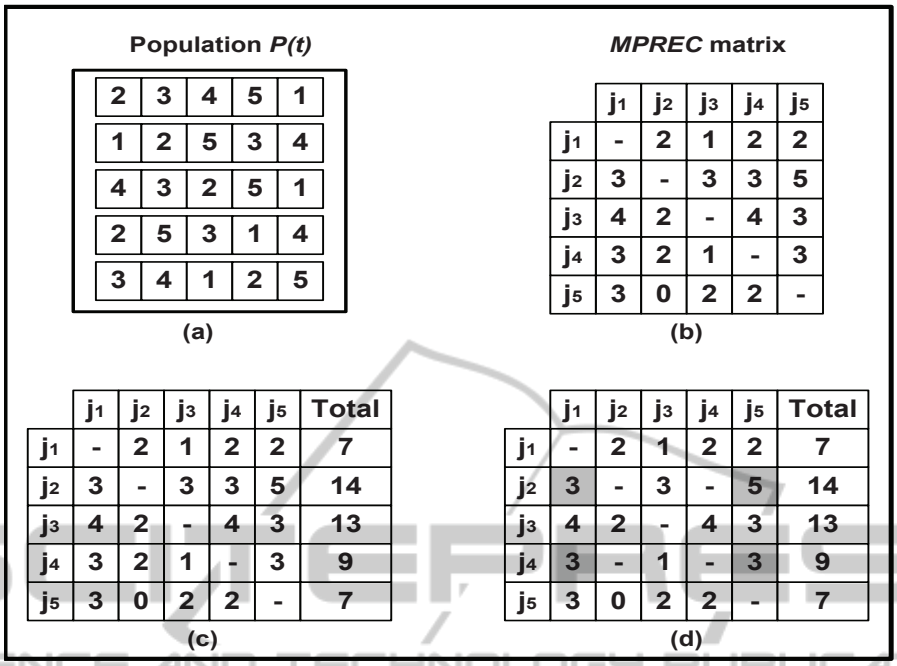

Figure 2: Illustration of HCX.

4 COMPUTATIONAL RESULTS AND DISCUSSION

The benchmark problem set consists of eight instances, each with a number of jobs of 15, 25, 35 and 45 jobs, and it is taken from the work of (Ragatz, 1993). These instances are available on the Internet at https://www.msu.edu/˜rubin/files/c&ordata.zip. The job processing times are normally distributed with a mean of 100 time units and the setup times are also uniformly distributed with a mean of 9.5 time units. Each instance has three factors which have both high and low levels. These factors are due date range, processing time variance and tardiness factor. The tardiness factor determines the expected proportion of jobs that will be tardy in a random sequence. All the experiments were run on an Itanium with a 1.4 GHz processor and 4 GB RAM. Each instance was executed 5 times and the results presented represent the average deviation with the optimal results of (Bigras et al., 2008). All the algorithms are coded in C++ language under the ILOG IBM CP constraint environment using ILOG Solver and Scheduler via the C++ API (ILOG, 2003b; ILOG, 2003a).

Table 1 summarize the comparison of different algorithms approaches. In this table, PRB denotes the instance names and OPT the optimal solution found by the B&B of (Bigras et al., 2008). These authors have not given information about the execution time of their approach. They only said that some instances have been resolved after more than seven days. The GA column shows the results average deviation to the optimal solution of the genetic algorithm described in the section 3.1 which gives the best results among all pure genetic algorithms in the literature without an intensification process (Sioud et al., 2009). The GA average CPU time is equal to 13.4 seconds for the 32 instances. The GA generally obtained fairly good results only for the instances 601, 605, 701 and 705.

The CBS column shows the deviations of the CBS approach minimizing the total tardiness proposed by (Sioud et al., 2010b) and used in the HCX crossover. For this approach, the execution time is limited to 60 minutes. It can be noticed that the CBS approach results deteriorate with increasing the instances size and especially for the **4, **5 and **8 instances. The GA_{PCX} column shows the average deviation of the genetic algorithm in which the hybrid crossover operator PCX of (Sioud et al., 2010a) is integrated. This crossover used only direct constraints between two jobs from two selected parents. The CBS approach execution time in this hybrid crossover is limited to 15 seconds. The GA_{PCX} average time execution is equal to 15.2 minutes for the 32 instances.

The GA_{HCX} column shows the average deviation of the genetic algorithm in which the hybrid crossover operator HCX presented in this paper is integrated. The probability p_{HCX} is equal to 0.05 and the CBS ap-

Table 1: Comparison of different algorithms.

PRB	OPT	GA	CBS	GA_{PCX}	GA_{HCX}
401	90	0.0	0.0	0.0	0.0
402	0	0.0	0.0	0.0	0.0
403	3418	0.5	0.0	0.0	0.0
404	1067	0.0	0.0	0.0	0.0
405	0	0.0	0.0	0.0	0.0
406	0	0.0	0.0	0.0	0.0
407	1861	0.0	0.0	0.0	0.0
408	5660	0.2	0.9	0.0	0.0
501	261	0.5	0.4	0.0	0.0
502	0	0.0	0.0	0.0	0.0
503	3497	0.2	2.5	0.0	0.0
504	0	0.0	0.0	0.0	0.0
505	0	0.0	0.0	0.0	0.0
506	0	0.0	0.0	0.0	0.0
507	7225	0.7	1.8	0.0	0.0
508	1915	0.0	35.8	0.0	0.0
601	12	169.4	41.7	6.7	3.9
602	0	0.0	0.0	0.0	0.0
603	17587	1.8	6.5	0.8	0.2
604	19092	1.8	21.1	1.1	0.6
605	228	13.0	122.4	2.6	1.2
606	0	0.0	0.0	0.0	0.0
607	12969	1.6	17.7	0.7	0.3
608	4732	1.7	156.6	0.7	0.0
701	97	30.7	20.6	6.8	2.9
702	0	0.0	0.0	0.0	0.0
703	26506	1.9	2.8	1.2	0.7
704	15206	3.4	94.8	1.6	0.8
705	200	33.7	72.5	6.1	3.1
706	0	0.0	0.0	0.0	0.0
707	23789	2.2	20.4	1.0	0.4
708	22807	2.8	50.0	1.5	1.0
Execution Time (min)	-	0.225	60	15.2	14.5

proach execution time is limited to 15 seconds. The two parameters nbr_{job} and t_{job} are fixed to 0.15 * n and 0.2 * n where n is the jobs considered number. These two parameters were adjusted following empirical tests on different instances. The first observation is that the GA_{HCX} algorithm is always optimal for 15 and 25 jobs instances. It should be noted that the integration of the HCX crossover improves all of the GA results and especially for the instances **1 and **5 where the deviation became less than 4%. For example, the deviation was reduced from 169.4% to 3.9% for the 601 instance. Using the precedence constraints allows the HCX crossover to enhance both the GA exploration and the CBS search; and consequently reaching better schedules. So, this new hybrid crossover can reach better space solution using more precedence constraints.

The GA_{HCX} average time execution is equal to 14.5 minutes for the 32 instances. This hybrid algorithm improves all the results found by the GA_{PCX}. These improvements are more pronounced with the integration of all the precedence constraints. This integration improves essentially the **1 and the **5 instances. Also, the optimal schedule is always reached by GA_{HCX} for the 608 instance. The GA_{HCX} found the optimal solution for all the instances at least one

time and this was not the case either for GA_{PCX}.

The convergence of both GA and the GA_{PCX} algorithms are similar. Indeed, the average convergence generation is equal to 1837 and 1845 generations for GA and GA_{PCX}, respectively. The GA_{HCX} average convergence generation is equal to 1358 and compared to the GA_{PCX}, the integration of the precedence constraints speeds up the convergence of the solution with reaching better results.

Exact methods are well known to be time expensive. The same applies to the hybridization of them with metaheuristics. Indeed, execution times increases significantly with such hybridization policies due to some technicality during the exchange of information between the two methods (Talbi, 2009; Talbi, 2002; Puchinger and Raidl, 2005; Jourdan et al., 2009) and this is what has been observed here. However, in this paper, the solution quality is our main concern. So, we concentrated our efforts on it.

5 CONCLUSIONS

In this paper, we introduce a hybrid crossover into a Genetic Algorithm to solve the sequence-dependent setup times single machine problem with the objective of minimizing the total tardiness. The proposed hybrid crossover extracts precedence constraints from the population. These constraints improve the CBS search and consequently the schedules quality.

Compared to a simple GA, the use of the HCX crossover improves all the results but for some instances the difference is still noticeable. Also, the results of this crossover outdoes those of a hybrid crossover taken from literature. Indeed, using the direct and indirect precedence constraints from the population improves the results and speeds up the convergence of the solution

Our results encourage us to use such hybridization for other scheduling problems in particular and other optimization problems in general. It is in this direction that our work is directed in the future. Also, to making a self-adaptive method, we will work on refining the individual selection process for the hybdrid HCX crossover and its two parameters : nbr_{job} and t_{job}.

REFERENCES

Allahverdi, A., Ng, C., Cheng, T., and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985 – 1032.

Armentano, V. and Mazzini, R. (2000). A genetic algorithm for scheduling on a single machine with setup times and due dates. *Production Planning and Controly*, 11(7):713 – 720.

Baptiste, P., LePape, C., and Nuijten, W. (2001). *Constraint-Based Scheduling : Applying Constraint Programming to Scheduling Problems*. Kluwer Academic Publishers.

Beck, J. C. and Perron, L. (2000). Discrepancy bounded depth first search. In *CP-AI-OR'2000: Fourth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 7–17.

Bigras, L., Gamache, M., and Savard, G. (2008). The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. *Discrete Optimization*, 5(4):663–762.

Conner, G. (2009). 10 questions. *Manufacturing Engineering Magazine*, pages 93–99.

Du, J. and Leung, J. Y. T. (1990). Minimizing total tardiness on one machine is np-hard. *Mathematics and Operations Researchs*, 15:438–495.

Franca, P. M., Mendes, A., and Moscato, P. (2001). A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operational Research*, 132:224–242.

Gagné, C., Gravel, M., and Price, W. L. (2005). Using metaheuristic compromise programming for the solution of multiple objective scheduling problems. *The Journal of the Operational Research Society*, 56:687–698.

Gagné, C., Price, W., and Gravel, M. (2002). Comparing an aco algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times. *Journal of the Operational Research Society*, 53:895–906.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. G. H. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326.

Gupta, S. R. and Smith, J. S. (2006). Algorithms for single machine total tardiness scheduling with sequence dependent setups. *European Journal of Operational Research*, 175(2):722–739.

Holland, J. H. (1992). *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA.

ILOG (2003a). *ILOG Scheduler 6.0. User Manual*. ILOG.

ILOG (2003b). *ILOG Solver 6.0. User Manual*. ILOG.

Jourdan, L., Basseur, M., and Talbi, E.-G. (2009). Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, 199(3):620–629.

Lee, Y., Bhaskaram, K., and Pinedo, M. (1997). A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions*, 29:45–52.

Leung, J., Kelly, L., and Anderson, J. H. (2004). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, Inc., Boca Raton, FL, USA.

Liao, C. and Juan, H. (2007). An ant colony optimization for single-machine tardiness scheduling with

sequence-dependent setups. *Computers and Operations Research*, 34:1899–1909.

Meseguer, P. (1997). Interleaved depth-first search. In *IJCAI'97: Proceedings of the Fifteenth international joint conference on Artifical intelligence*, pages 1382–1387, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag, London, UK.

Pinedo, M. (2002). *Scheduling Theory, Algorithm and Systems*. Prentice-Hall.

Puchinger, J. and Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation, Las Palmas, Spain, LNCS*.

Ragatz, G. L. (1993). A branch-and-bound method for minimumtardiness sequencing on a single processor with sequence dependent setup times. In *Proceedings twenty-fourth annual meeting of the Decision Sciences Institute*, pages 1375–1377.

Rubin, P. and Ragatz, G. (1995). Scheduling in a sequence-dependent setup environment with genetic search. *Computers and Operations Research*, 22:85–99.

Sioud, A., Gravel, M., and Gagné, C. (2009). New crossover operator for the single machine scheduling problem with sequence-dependent setup times. In *GEM'09: The 2009 International Conference on Genetic and Evolutionary Methods*.

Sioud, A., Gravel, M., and Gagné, C. (2010a). Constraint based scheduling in a genetic algorithm for the single machine scheduling problem with sequence dependent setup times. In *ICEC'2010: Proceedings of the International Conference on Evolutionary Computation*, pages 137–145.

Sioud, A., Gravel, M., and Gagné, C. (2010b). A modeling for the total tardiness smsdst problem using constraint programming. In Arabnia, H. R., de la Fuente, D., Kozerenko, E. B., Olivas, J. A., Chang, R., LaMonica, P. M., Liuzzi, R. A., and Solo, A. M. G., editors, *ICAI*, pages 588–594. CSREA Press.

Sioud, A., Gravel, M., and Gagné, C. (2012). A hybrid genetic algorithm for the single machine scheduling problem with sequence-dependent setup times. *Computers & OR*, 39(10):2415–2424.

Talbi, E. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8:541–564.

Talbi, E.-G. (2009). *Metaheuristics : from design to implementation*. John Wiley & Sons.

Tan, K. and Narasimhan, R. (1997). Minimizing tardiness on a single processor with setup-dependent setup times: a simulated annealing approach. *Omega*, 25:619 – 634.

Walsh, T. (1997). Depth-bounded discrepancy search. In *IJCAI'97: Proceedings of the Fifteenth international joint conference on Artifical intelligence*, pages 1388–1393, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Yunes, T., Aron, I. D., and Hooker, J. N. (2010). An Integrated Solver for Optimization Problems. *Operations Resaearch*, 58(2):342–356.

Zhu, X. and Wilhelm, W. E. (2006). Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE Transactions*, 38(11):987–1007.