Sequencing in a Non-permutation Flowshop with Constrained Buffers: Applicability of Genetic Algorithm versus Constraint Logic Programming[*]

Gerrit Färber[1], Said Salhi[2], Anna M. Coves Moreno[1]

[1] Institut d'Organització i Control de Sistemes Industrials (IOC), Universitat Politècnica de Catalunya (UPC), Av. Diagonal 647, Planta 11, 08028 Barcelona, España. gerrit_faerber@gmx.de, anna.maria.coves@upc.es
[2] The Centre for Heuristic Optimisation (CHO), Kent Business School (KBS), University of Kent, Canterbury, Kent CT2 7PE, UK, s.salhi@kent.ac.uk

Abstract

Mixed model production lines consider more than one model being processed on the same production line in an arbitrary sequence. Nevertheless, the majority of publications in this area are limited to solutions which determine the job sequence before the jobs enter the line and maintains it without interchanging jobs until the end of the production line, which is known as permutation flowshop. This paper considers a non-permutation flowshop. Resequencing is permitted where stations have access to intermediate or centralized resequencing buffers. The access to the buffers is restricted by the number of available buffer places and the physical size of the products. Two conceptually different approaches are presented in order to solve the problem. The first approach is a hybrid approach, using Constraint Logic Programming (CLP), whereas the second one is a Genetic Algorithm (GA). Improvements that come with the introduction of constrained resequencing buffers are highlighted. Characteristics such as the difference between the intermediate and the centralized case are analyzed, and the special case of semi dynamic demand is studied. Finally, recommendations are presented for the applicability of the hybrid approach, using CLP, versus the Genetic Algorithm.

Keywords: Non-permutation flowshop; Constrained buffers; Mixed model assembly line Genetic Algorithm; Constraint Logic Programming.

1. Introduction

This paper is located in the area of mixed model non-permutation flowshop production lines where jobs of more than one model are being processed on the same production line in an arbitrary sequence. Unlike the majority of publications in this area, Potts et al. (1991), and Liao et al. (2006) study the improvements when the possibility of resequencing jobs between selected stations is regarded; the considerable improvements are even more evident when setup cost/time exists. The case of infinite buffers is basically a theoretical case in which no limitation exists with respect to the number of jobs that may be buffered between two stations. Roy (1962), presents a graph-theoretical interpretation which allows the calculation of the makespan of a sequence in the flowshop that is not a permutation sequence. Surveys on heuristics treating the case of infinite buffers are presented by Liesegang and Schirmer (1975), and Park et al. (1984).

As explained by Pinedo (1995), mathematically a buffer can be realized as a station with zero processing time. Approaches which consider a limited number of buffer places for the flowshop problem are studied by Dutta and Cunningham (1975), Reddi (1976), Papadimitriou and Kanellakis (1980), Nowicki (1999), Smutnicki (1998), and Leisten (1990). Mascic and Pacciarelli (2002) consider limited resequencing possibilities for jobshop problems, also studied by Brucker et al. (2006). Previous to these studies, the same research group, Brucker et al.

[*] This work is partially supported by the Ministry of Science and Technology, Spain, and the funding for regional research DPI2004-03472.

(2003), presented a Tabu search for a flowshop with a buffer positioned between all consecutive stations. The number of buffer places is the same for all resequencing buffers and the variable parameter is the number of buffer places which is 0, 1, 2, or infinite. The authors use the test-bed from Taillard (1993), which was originally designed for permutation flowshop problems, and present next to their solutions the optimal solutions from other authors, and if not available, the so far best lower bounds and the best known solutions.

The introduction of resequencing possibilities generally leads to additional costs, caused by additional equipment to be mounted, like buffers, but also extra efforts in terms of logistics complexity may arise. The problem in NP-hard and as highlighted by Lahmar et al. (2003), only few resequencing possibilities are necessary in order to achieve the greatest benefits. In the case in which there exist jobs with large and small physical size, the investment for additional resequencing equipment can be reduced by, e.g., only giving small jobs the possibility to resequence. Consequently, only small resequencing buffer places are installed. Following this concept, in a chemical production line with client orders of different volumes, only resequencing tanks that permit client orders of relatively small volume to be resequenced. The work of Witt (2005), considers that different jobs have different physical sizes and occupy only a portion of the continuous intermediate resequencing buffer which is used for decoupling and resequencing.

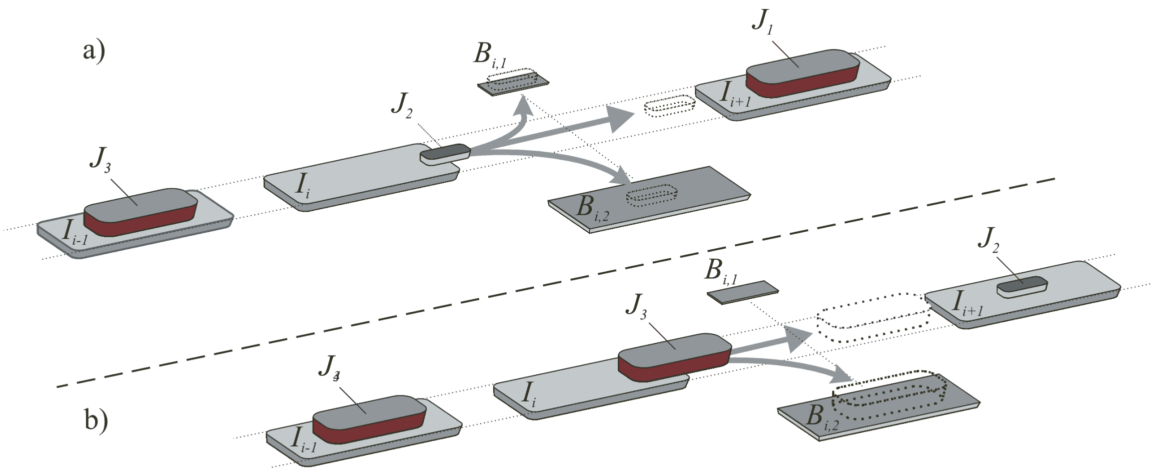

Figure 1. Scheme of the considered flowshop. The jobs J_j pass consecutively through the stations I_i. The buffer B_i permits to temporally store a job with the objective of reinserting it at a later position in the sequence. a) Job J_2 can pass through any of the two buffer places $B_{i,1}$ or $B_{i,2}$ of buffer B_i. b) Job J_3 can pass only through buffer place $B_{i,2}$, due to its physical size.

This paper considers a mixed model non-permutation flowshop with the possibility to resequence jobs between consecutive stations. The buffers are located off-line, either accessible from a single station (intermediate case) or from various stations (centralized case). In both cases, it is considered that a job may not be able to be stored in a buffer place, due to its extended physical size, see figure 1. The primary objective is the minimization of the makespan, but also setup-cost and setup-time is contemplated. In what follows the problem is formulated with more detail and two heuristic approaches are described. Thereafter, the accomplished performance comparison is presented, followed by the conclusions.

2. Problem under Study

The realized work is based on the classical flowshop in which the jobs (J_1, J_2, …, J_j, …, J_n) pass consecutively through the stations (I_1, I_2, …, I_i, …, I_m). Furthermore, after determined stations, off-line buffers B_i permit to resequence jobs. The buffer provides various buffer places ($B_{i,1}$, $B_{i,2}$, …) and each buffer place is restricted by the physical size of the jobs to be stored. As can be seen in figure 1a, job J_2 can be stored in buffer place $B_{i,1}$ as well as in $B_{i,2}$. Whereas, the next job J_3 can be stored only in buffer place $B_{i,2}$, because of the physical size of the job exceeding the physical size of the buffer place $B_{i,1}$, see figure 1b. The objective is to minimize the makespan, the setup-cost and the setup-time. The objective function is the weighted sum of the makespan and the sequence dependent setup cost, where the sequence dependent setup time is not concerned with a weight but is indirectly included in the calculation of the makespan. In a first step, the resequencing buffers are located intermediate, between two consecutive stations. In this case the buffer is assigned to the precedent station and may be accessed only by this station. Then, for an additional benefit, a single resequencing buffer is used, with access from various stations, while the limitations on the physical size of the buffer places are maintained. The considered problem is relevant to various flowshop applications such as chemical productions dealing with client orders of different volumes and different sized resequencing tanks, or in productions where split-lots are used for engineering purpose, such as the semiconductor industry. Even in the production of prefabricated houses with, e.g., large and small walls passing through consecutive stations where electrical circuits, sewerage, doors, windows and isolation are applied.

3. Hybrid-Constraint Logic Programming (CLP)

The name CLP was first introduced by Jaffar and Lassez (1987) and can be described as a powerful extension of conventional logic programming. It involves the incorporation of constraint languages and constraint solving methods into logic programming languages, see also Cohen (1990), Marriot and Stuckey (1998), Apt (2003) and Rossi et al. (2006). The concept of CLP was used for sequencing and scheduling problems by Caseau and Laburthe (1994) and Filipe (1997). The formulation, used here, is described in detail in Färber (2006). The formulation was implemented in OPL Studio version 3.7, see e.g. Hentenryck (2002) for reference on OPL. Apart from the job and station precedence, the CLP formulation determines the jobs which are to be taken off the line for the purpose of resequencing, given that a free buffer place is available. The CLP formulation is utilized in different arrangements. First the basic arrangement is presented and due to its limitations to small problems, several alternative arrangements are proposed to approach the problem under study. These alternative arrangements are hybrids of the exact approach with heuristic concepts and therefore do not ensure optimality.

3.1. Basic arrangement (A_{NP})

In this case, the non-permutation model is directly applied to the problem under study. Unless the time for the execution of this arrangement is restricted, it solves the problem to optimality. However, due to the complexity of the problem, the execution time is limited to t_{max}, and if the optimal solution is not found within this time, the best solution found up to that point is used for comparison. In order to study the limitation of the basic arrangement, an example of a 10-station flowshop was used with two intermediate resequencing buffers, accessible after station 2 and 5, each with one resequencing buffer place. The total time for solving a four job instance to optimality was 4314 seconds on a Pentium 4, 3.0 GHz, 512MB RAM. On account of the elevated time spent on the calculation, more efficient alternatives to the basic arrangement were needed.

3.2. Hybrid arrangement I: Cascading (A_C)

In this arrangement first the permutation model is applied and in a first cascade the non-permutation model. The intention of the first cascade is to determine the optimum permutation sequence. Then, in the second step and with the use of the non-permutation model, the additional constraint is introduced. This additional constraint uses a semi dynamic demand with a fixed job sequence for the first station to the one previously determined by the first cascade. The sum of the maximal execution time for both cascades is set to $t_{\max}$.

3.3. Hybrid arrangement II: Multistart cascading (A_M, A_{MSS})

As indicated by its name, the multistart arrangement consists of solving the problem several times, where the execution time is limited to a fraction of $t_{\max}$. Every time a different sequence is fed to the first cascade (evaluation with the completely constrained permutation model) only its corresponding value of the objective function is determined. This operation requires a very small amount of time, compared to the second cascade which then determines the improvement, taking into account resequencing after the first station. The sequence of the first station is fixed to the one determined by the first cascade. As a consequence, and in contrast to pure cascading (section 3.2), the multistart arrangement is able to consider permutation sequences that are not optimal within the set of permutation sequences, but which may result in a relatively better final solution after applying the second cascade. The multistart arrangement is applied in two conceptional different ways.

3.3.1 Without Feedback (A_M)

In this case, a certain number of randomly generated permutation sequences is evaluated with the completely constrained permutation model, summarized, and the most promising *R* sequences are passed to the non-permutation model pretending a semi dynamic demand with fixed job sequence for the first station ($\Pi_1=\Pi_{perm}$). The arrangement is shown in figure 3 with a total of 10 *R* randomly generated permutation sequences.

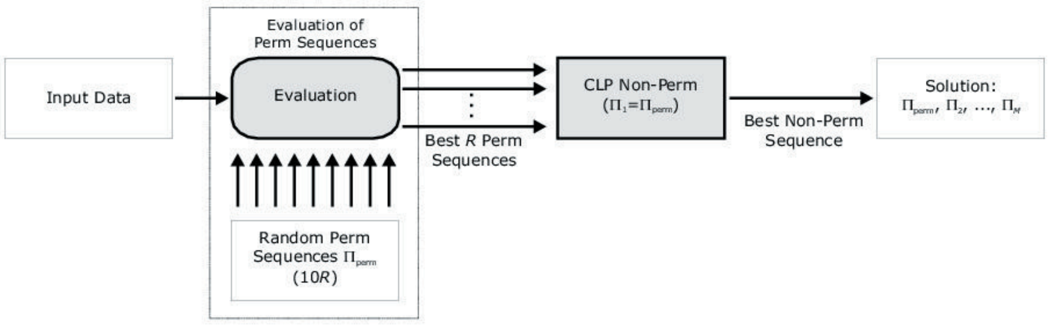
Evaluation of Perm Sequences
Input Data
Evaluation
CLP Non-Perm ($\Pi_1=\Pi_{perm}$)
Solution: $\Pi_{perm}, \Pi_2, \ldots, \Pi_M$
Best *R* Perm Sequences
Best Non-Perm Sequence
Random Perm Sequences Π_{perm} (10*R*)

Figure 3. The CLP works in a multistart cascading mode without feedback, a total of 10 *R* randomly generated permutation sequences are used.

3.3.2 With Feedback and shift and swap operations of jobs (A_{MSS})

The schematic for this arrangement is shown in figure 4. The execution is performed in an iterative way, while the number of randomly generated permutation sequences is constantly reduced (from *R* to 1), whereas the information obtained by the non-permutation model is fed back in an increasing way (from 0 to *R*-1) in order to preserve valid information.

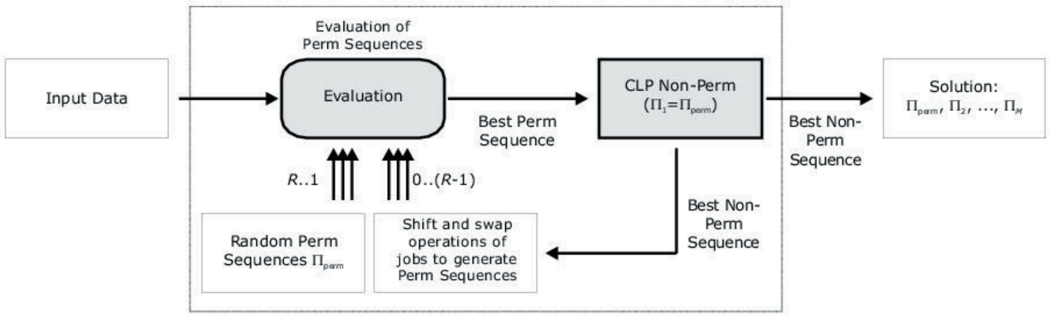
Evaluation of Perm Sequences
Input Data
Evaluation
Best Perm Sequence
CLP Non-Perm ($\Pi_1=\Pi_{perm}$)
Solution: $\Pi_{perm}, \Pi_2, \ldots, \Pi_M$
Best Non-Perm Sequence
R..1
0..(*R*-1)
Best Non-Perm Sequence
Random Perm Sequences Π_{perm}
Shift and swap operations of jobs to generate Perm Sequences

Figure 4. The CLP works in a multistart cascading mode with feedback of the sequence which obtained the best solution in the second cascade.

3.4. Limited flexibility

During the course of this work it was observed that the CLP can be positively influenced by the introduction of additional constraints such as the maximum number of positions a job can move upwards or downwards, say W_{up} and W_{down}, respectively.

4. Genetic Algorithm (GA)

The concept of the GA was first formulated by Holland (1973) and various GAs were designed for mixed model assembly lines such as Potts (2003), Levitin et al. (2006) and Wang et al. (2006). The formulation used here is a variation of the GA explained in Michaelewicz (1996) and further details can be found in Färber and Coves (2006). The genes represent the jobs which are to be sequenced. The chromosome h, determined by a series of genes, represent a sequence of jobs. A generation is formed by *R* chromosomes and the total number of generations is *G*. In the permutation case, the size of a chromosome is determined by the number of jobs, the sequence Π. In the non-permutation case, the chromosomes are *L*+1 times larger, resulting in the sequences Π'_1, ..., Π'_{L+1}, *L* being the number of resequencing possibilities. We also call one sequence Π', which includes every job exactly one time, a main fraction of a chromosome. The relevant information for each chromosome is its fitness value (objective function), the number of job changes and the indicator specifying if the chromosome is feasible. A chromosome is marked infeasible and is imposed with a penalty, if a job has to be taken off the line and no free buffer place is available or the physical size of the job exceeds the size limitation of the available buffer places. When two solutions result in the same fitness value, the one with fewer job changes is preferred.

4.1. Genetic operators

The genetic operators specify in which way the subsequent population is generated by reproduction of the present population, taking into account that "fitter" solutions are more promising and therefore are more likely to reproduce. Even an unfeasible solution is able to reproduce, because of the fact that it may generate valuable and feasible solutions in one of the preceding generations. The used genetic operators are inheritance, crossover and mutation. The value p_X is the percentage with which a genetic operator *X* is applied to a chromosome.

4.1.1 Inheritance

This operator is determined by two parameters. The parameter p_{BS} determines the percentage of the best solutions which will be copied directly to the next generation, called the cluster of promising chromosomes, and ensures that promising chromosomes are not extinct. Then, in order to avoid being trapped in a local minimum, the parameter p_{b} is used to determine the percentage of chromosomes which are removed from this cluster.

4.1.2 Crossover

Crossover is a genetic operator that combines one or more chromosomes to produce a new chromosome. We apply the classical one-point crossover (crossover-I), see figures 5a, 5b, and the two-point crossover (crossover-II), see figures 5c, 5d. The probabilities with which these operations are applied to a chromosome are $p_{\mathrm{c\text{-}I}}$ and $p_{\mathrm{c\text{-}II}}$, and the crossover points are defined by the random number pos for crossover-I, and the pair pos_1 and pos_2 for crossover-II, respectively. If the crossover point pos, pos_1 and pos_2 is a multiple of a main fraction, the crossover operation is simple and takes place between two main fractions of the chromosome, see figure 5a (crossover-I) and figure 5c (crossover-II), i.e. after the crossover point the chromosomes are completely crossed over. In the complex case, however, the crossover points are located within a main fraction of the chromosome and it has to be assured explicitly that each job is sequenced exactly one time for each main fraction, see figure 5b (crossover-I) and figure 5d (crossover-II).

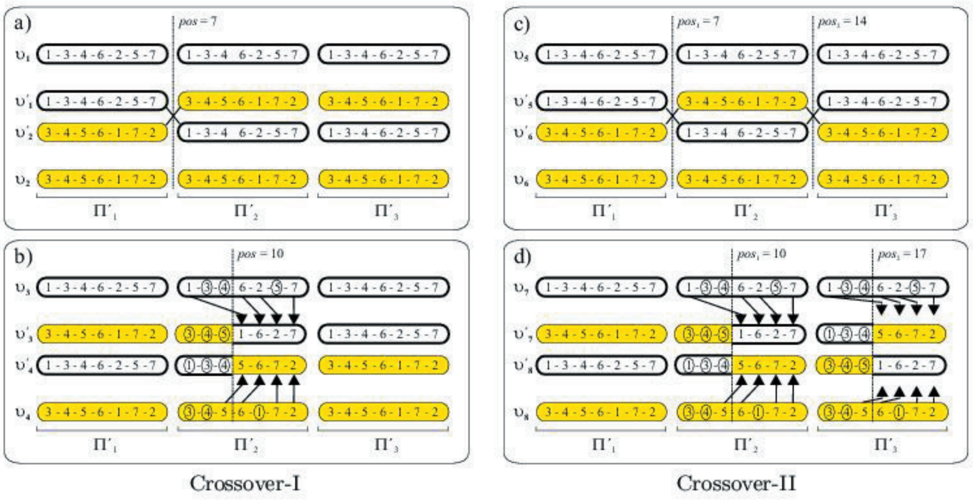

Figure 5. Operators crossover-I and crossover-II. a) and c) In the simple case the crossing takes place between two main fractions of the chromosome. After the crossover point the chromosomes are completely crossed over. b) and d) In the more complex case it has to be assured that each job is sequenced exactly one time for each main fraction of the chromosome.

4.1.3 Mutation

Two classical mutation operators are applied. The first one is the one-point mutation (mutation-I), which relocates the job at position pos_1 to position pos_2 within the same main fraction of a chromosome. The second one is the two-point mutation (mutation-II), which interchanges the jobs at position pos_1 and position pos_2, within a main fraction. Furthermore, there exist two cases for mutation-I: forward mutation ($pos_1 < pos_2$); and backward mutation ($pos_1 > pos_2$). In the first case, a single job has to be taken off the line, and in the second case, in order to let a single job pass; a group of succeeding jobs has to be taken off the line, resulting in a larger effort to realize. The probabilities of this operator are $p_{\text{m-I(f)}}$, $p_{\text{m-I(b)}}$ and $p_{\text{m-II}}$.

4.2. Cascading

In order to further improve the GA, it is partitioned into two steps. In the first step, the possibility of resequencing jobs within the production line is ignored. Furthermore, only permutation sequences are considered as possible solutions and the chromosome size is reduced to the number of jobs. The last generation, together with the best found solution, form the initial generation for the next cascade where the resequencing possibilities, provided by stations with access to resequencing buffers, are taken into account.

5. Performance Comparison

In order to evaluate the performance of the two presented approaches, GA and CLP, a flowshop which consists of 10 stations is used. Station 3, 5 and 8 have access to a single intermediate (centralized) buffer place (L=3). The range of the production time is [1...100], for the setup cost [2...8] and for the setup time [1...5]. The number of jobs is varied in the range of 4 to 10 and the objective function is the weighted sum of the makespan (factor 1.0) and the setup cost (factor 0.3), where the setup time is not concerned with a weight but is indirectly included in the calculation of the makespan.

5.1. Difference in physical size of jobs

After station 3, 5 and 8 access to resequencing buffer places is made available. Also three differently sized buffer places are used and the ratio for the physical size of jobs is 4/10, 3/10 and 3/10 for small (1), medium (2) and large (3), respectively. The allocation of the buffer places to the buffers considers five scenarios for the intermediate case ("I111", "I231", "I132", "I222", "I333") and three scenarios for the centralized case ("C1", "C2", "C3"). For instance, "I132" represents one small, one large and one medium buffer place, located as intermediate resequencing buffer places after stations 3, 5 and 8, respectively. "C2" represents one medium (2) buffer place, located as a centralized buffer place, accessible from station 3, 5 and 8. "I333" and "C3" are the two cases which provide the largest flexibility in terms of physical size restrictions.

Table 1. Difference in physical size of jobs for the GA.

Jobs	GA_P	I111	I231	I132	I222	I333	C1	C2	C3
4	481,2	476,1	**474,9**	**474,9**	**474,9**	**474,9**	476,1	476,1	476,1
5	470,5	470,5	464,2	467,6	464,2	**449,4**	470,5	464,2	**449,4**
6	564,6	541,1	542,1	541,1	540,9	**537,6**	543,9	538,8	537,8
7	568,7	566,8	564,9	566,8	564,9	**563,7**	566,8	566,8	563,8
8	589,6	589,6	589,3	588,8	589,2	**586,0**	589,2	589,6	589,5
9	679,3	676,3	668,8	**666,7**	675,9	671,7	677,2	675,7	670,8
10	736,1	735,5	729,7	725,8	735,4	**724,0**	735,5	735,5	725,3

Table 2. Comparison of the GA and the CLP for the intermediate case.

			Minimum				Average			
Jobs	A_P	GA_P	A_{NP}	A_C	A_{MSS}	GA	A_{NP}	A_C	A_{MSS}	GA
4	481,2	481,2	481,2	480,0	477,7	**474,9**	481,2	480,5	478,2	**475,1**
5	470,5	470,5	470,5	470,5	451,8	**449,4**	470,5	470,5	463,7	**463,2**
6	564,6	564,6	564,6	563,4	**537,1**	537,6	564,6	564,2	**538,1**	540,6
7	568,7	568,7	568,7	566,6	565,8	**563,7**	570,3	567,9	568,2	**565,4**
8	589,6	589,6	589,6	589,6	589,1	**586,0**	605,6	589,8	593,4	**588,6**
9	684,3	679,3	684,3	681,0	685,5	**666,7**	694,8	683,1	689,6	**671,9**
10	736,1	736,1	742,4	735,5	740,9	**724,0**	750,1	738,2	744,2	**730,1**

Table 3. Comparison of the GA and the CLP for the centralized case.

			Minimum				Average			
Jobs	A_P	GA_P	A_{NP}	A_C	A_{MSS}	GA	A_{NP}	A_C	A_{MSS}	GA
4	481,2	481,2	481,2	480,0	477,7	**476,1**	481,2	480,0	478,1	**476,1**
5	470,5	470,5	470,5	470,5	451,8	**449,4**	470,5	470,5	462,2	**461,4**
6	564,6	564,6	564,6	564,0	**537,7**	537,8	564,6	564,0	**538,2**	540,1
7	568,7	568,7	568,7	567,5	567,0	**563,8**	570,7	567,5	567,9	**565,8**
8	589,6	589,6	589,6	589,6	**588,0**	589,2	610,9	589,6	591,0	**589,4**
9	684,3	679,3	684,3	680,7	684,8	**670,8**	696,2	682,5	689,0	**674,5**
10	736,1	736,1	742,4	735,2	736,9	**725,3**	749,1	735,4	741,6	**732,1**

The results of the GA are presented in table 1. The algorithm performs best for the case "I333" and nearly as good in the case "C3" in all instances. It outperforms the solutions which are limited to permutation sequences. The comparison of the GA and the Constraint Logic Programming are shown in table 2 and table 3 for the intermediate and the centralized case, respectively. The permutation solutions found by the GA (GA_p) are the same as for the permutation solutions of the CLP (A_p), except for one case (684,3 versus 679,3). Furthermore, the GA in general achieves similar or slightly better results for the intermediate and the centralized case for less than 9 jobs. In the case of 9 and 10 jobs, better solutions are obtained by the GA. In the studied flowshop, the CLP achieved an average of 0.74% and 1.09% for the intermediate and the centralized case, respectively. Whereas the GA achieved an average of 1.49% and 1.41% for the intermediate and the centralized case, respectively. The overall best results were achieved with up 4.9% for the CLP and 4.8% for the GA, compared to the best result achieved for permutation sequences.

5.2. Semi-Dynamic Demand

In the case of the semi dynamic demand, the same input values are used as in the previous case. The only difference is that the sequence for the first station is set to a fixed sequence. For comparison the same sequence for the first station is used for the CLP and the GA.

Table 4. Solution for the semi dynamic demand using the CLP.

Jobs	Perm	I111	I231	I132	I222	I333	C1	C2	C3
4	483,1	480,9	**480,7**	**480,7**	480,9	**480,7**	480,9	**480,7**	**480,7**
5	552,0	**490,7**	**490,7**	**490,7**	**490,7**	**490,7**	**490,7**	**490,7**	**490,7**
6	647,5	627,0	**620,1**	**620,1**	**620,1**	**620,1**	628,5	622,2	622,2
7	636,5	627,7	627,7	625,0	625,0	**609,5**	628,0	627,7	616,1
8	673,6	646,6	646,6	644,8	644,8	644,8	646,6	644,8	**632,3**
9	744,3	719,4	719,4	716,1	716,1	716,1	719,4	716,1	**712,1**
10	813,7	786,5	763,2	**762,5**	785,9	791,2	786,5	786,5	764,0

Table 5. Solution for the semi dynamic demand using the GA.

Jobs	Perm	I111	I231	I132	I222	I333	C1	C2	C3
4	483,1	480,9	**480,7**	**480,7**	**480,7**	**480,7**	480,9	**480,7**	**480,7**
5	552,0	**490,7**	**490,7**	**490,7**	**490,7**	**490,7**	**490,7**	**490,7**	**490,7**
6	647,5	627,0	**620,1**	**620,1**	**620,1**	**620,1**	628,5	622,7	622,2
7	636,5	627,7	628,8	626,1	625,0	**609,5**	629,1	628,5	616,3
8	673,6	669,2	651,3	646,0	647,2	638,2	672,4	653,4	**637,0**
9	744,3	736,5	724,2	728,6	721,8	**709,4**	736,5	729,5	714,7
10	813,7	808,3	788,0	781,7	805,1	**757,5**	809,2	805,9	772,2

Table 6. Comparison of the GA and the CLP for semi dynamic demand. The values show the relative improvement of the CLP with respect to the GA.

Jobs	Perm	I111	I231	I132	I222	I333	C1	C2	C3
4	483,1	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**
5	552,0	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**
6	647,5	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	**0,0%**	0,1%	**0,0%**
7	636,5	**0,0%**	0,2%	0,2%	**0,0%**	**0,0%**	0,2%	0,1%	**0,0%**
8	673,6	3,4%	0,7%	0,2%	0,4%	-1,0%	3,8%	1,3%	0,7%
9	744,3	2,3%	0,7%	1,7%	0,8%	-0,9%	2,3%	1,8%	0,4%
10	813,7	2,7%	3,1%	2,5%	2,4%	-4,5%	2,8%	2,4%	1,1%

Table 4 shows the result for the CLP. In any case, when offline resequencing buffers are considered, the results are improved compared to the permutation sequence. In the studied flowshop, an average of 4.3% is achieved for the use of the CLP. Whereas in the case of the GA, see table 5, the average is 3.7%.In the case of the Hybrid-CLP, as well as in the GA, the semi dynamic demand with a fixed job sequence for the first station, leads to considerable improvements, even for larger problem sizes. Table 6 shows the improvement of the CLP with respect to the GA. For only a few jobs, both methods achieve the same solutions. When 6 or more jobs are to be sequenced, the CLP in general outperforms the GA, especially when the number of buffer places at each resequencing possibility is limited to a small number.

6. Applicability

The basic arrangement A_{NP} is very limited with respect to the problem size; whereas, the alternative (hybrid) approaches A_C and A_{MSS}, lead to further improvements. The consideration of limited flexibility, as highlighted in section 3.4, positively influences the performance of the CLP. In the case of a semi dynamic demand, the CLP and the GA apparently show different behaviours: for the case of up to 10 jobs, the CLP outperforms the GA for few resequencing possibilities, whereas, the GA performs better when several buffer places but only few resequencing possibilities are considered with several buffer places. It was furthermore discovered that the GA is applicable for small as well as for problems with at least up to 100 Jobs. Figure 6 shows the summary of the applicability of the different approaches, for the case of a maximum of 3 resequencing possibilities and with respect to the number of jobs to be processed. The number of stations (M) is not found to be a very limiting factor, compared to the number of jobs (N) and the number of resequencing possibilities (L).

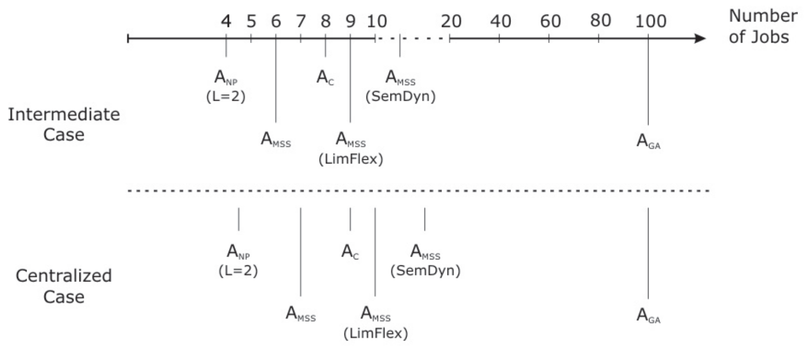

Figure 6. Applicability of the pure CLP and the hybrid CLP versus Genetic Algorithm.

7. Conclusions

This paper has presented a performance comparison of a Hybrid-CLP and a GA which were applied to a mixed model non-permutation flowshop, designed to consider intermediate or centralized resequencing buffers. The primary objective is the minimization of the makespan, but also setup-cost and setup-time is contemplated. Furthermore, the buffer access is restricted by the number of buffer places and the physical size. Comparing the results of the proposed approaches, it can be concluded that the Hybrid-CLP performs well on problems with 10 stations, approximately 10 jobs and only few distributed resequencing buffer places. The Hybrid-CLP furthermore is positively influenced when additional restrictions are present, such as the introduction of the centralized instead of the intermediate buffer location. The GA on the other hand performs better when only few stations are considered with access to offline resequencing buffers but with several buffer places each. The direct comparison of the Hybrid-CLP with the GA demonstrates that the GA outperforms the Hybrid-CLP in the vast majority of the cases with static demand. In the case of the semi dynamic demand, with a fixed job sequence for the first station, the improvements for both approaches are considerably larger, even for larger problem sizes. The Hybrid-CLP outperforms the GA when the number of buffer places at each resequencing possibility is limited to a small number, whereas, the GA gives better results when larger buffer places are used. Concerning future work, on one hand it would be interesting to consider a cyclic product flow rather than a cluster of jobs, and on the other hand, to study a dynamic demand for the incoming jobs. Secondly, and based on the experience from the Hybrid-CLP, the use of a Hybrid-GA, which is the use of the CLP within the GA, could give competitive and promising results.

References

Apt, K. (2003). Principles of constraint programming. Cambridge University Press.

Brucker, P., Heitmann, S., and Hurink, J. (2003). Flow-shop problems with intermediate buffers. OR Spektrum, 25:549–574.

Brucker, P., Heitmann, S., Hurink, J., and Nieberg, T. (2006). Job-shop scheduling with limited capacity buffers. OR Spektrum, 28:151–176.

Caseau, Y. and Laburthe, F. (1994). Improving clp scheduling task intervals. Proc. of the 11th International Conference on Logic Programming, pages 369–383.

Cohen, J. (1990). Constraint logic programming languages. Communications of the ACM, 33(7):52–68.

Dutta, S. and Cunningham, A. (1975). Sequencing two-machine flow-shops with finite intermediate storage. Management Science, 21:989–996.

Färber, G. (2006). Sequencing in mixed model non-permutation flowshop production line using constrained buffers. PhD Thesis, Universitat Politècnica de Catalunya, Spain.

Färber, G. and Coves, A. (2006). Performance study of a genetic algorithm for sequencing in mixed model non-permutation flowshops using constrained buffers. Springer LNCS, ISSN: 0302-9743, 3982/2006:638–648.

Filipe, N. (1997). A clp model to the job sequencing problem. Proceedings of the 8th Portuguese Conference on Artificial Intelligence, Lecture Notes in Computer Science, 1323:291–296.

Hentenryck, P. (2002). Constraint and integer programming in opl. INFORMS Journal of Comp., 14(4):345–372.

Holland, J. (1973). Genetic algorithms and the optimal allocation of trials. SIAM Journal of Comp., 2(2):88–105.

Jaffar, J. and Lassez, J. (1987). Constraint logic programming. 14th ACM Symposium on Principles of Programming Languages POPL87, ACM Press, Munich Germany, pages 111–119.

Lahmar, M., Ergan, H., and Benjaafar, S. (2003). Resequencing and feature assignment on an automated assembly line. IEEE Transaction on Robotics and Automation, 19(1):89–102.

Leisten, R. (1990). Flowshop sequencing problems with limited buffer storage. IJPR, 28(11):2085–2100.

Levitin, G., Rubinovitz, J., and Shnits, B. (2006). A genetic algorithm for robotic assembly line balancing. European Journal of Operational Research, 168:811–825.

Liesegang, G. and Schirmer, A. (1975). Heuristische verfahren zur maschinenbelegungsplanun g bei reihenfertigung. Zeitschrift fr Operations Research, 19:195–211.

Marriot, K. and Stuckey, P. (1998). Programming with constraints. The MIT Press. Cambridge, Massachusetts.

Mascic, A. and Pacciarelli, D. (2002). Job-shop scheduling with blocking and no-wait constraints. EJOR, 143:498–517.

Michaelewicz, Z. (1996). Gas + Data Structures = Evolution Programs. Springer Verlag, 3rd edition.

Nowicki, E. (1999). The permutation flow shop with buffers: A tabu search approach. EJOR, 116:205–219.

Papadimitriou, C. and Kanellakis, P. (1980). Flowshop scheduling with limited temporary storage. Journal of the Association for Computing Machinery, 27:533–554.

Park, Y., Pegden, C., and Enscore, E. (1984). A survey and evaluation of static flowshop scheduling heuristics. IJPR, 22:127–141.

Pinedo, M. (1995). Scheduling. Theory, Algorithms and Systems. Prentice Hall Inc., Englewood Cliffs.

Potts, C. (2003). A ga-sa multiobjective hybrid search algorithm for integrating lot sizing and sequencing in a flow-line scheduling. International Journal of Advanced Manufacturing.

Potts, C., Shmoys, D., and Williamson, D. (1991). Permutation vs. non-permutation flow shop schedules. Operations Research Letters, 10(5):281–284.

Reddi, S. (1976). Sequencing with finite intermediate storage. Management Science, 23:19–25.

Rossi, F., Van Beek, P., and Walsh, T. (2006). Handbook of constraint programming. Elsevier.

Roy, B. (1962). Cheminement et connexité dans les graphes-applications aux problémes d'ordonnancement. Revue METRA, série spéciale, (1). 130 pages.

Ruiz, R. and Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. EJOR, 169(3):781–800.

Smutnicki, C. (1998). A two-machine permutation flow shop scheduling problem with buffers. OR Spektrum, 20:229–235.

Taillard, E. (1993). Benchmarks for basic scheduling problems. EJOR, 64(2):278–285.

Wang, L., Zhang, L., and Zheng, D. (2006). An effective hybrid genetic algorithm for flow shop scheduling with limited buffers. Computers and Operations Research. Article in Press.

Witt, A. and Voss, S. (2007). Simple heuristics for scheduling with limited intermediate storage. Computers & Operations Research, 34(8):2293–2309.