Genetic Algorithm for Sequencing in Mixed Model Non-Permutation Flowshops using Constrained Buffers.

Gerrit Färber *Anna M. Coves Moreno*

Instituto de Organización y Control de Sistemas Industriales (IOC)
Universidad Politécnica de Cataluña (UPC), Barcelona, Spain
gerrit_faerber@gmx.de, anna.maria.coves@upc.es

Resumen

Este trabajo presenta un Algoritmo Genético (GA) del problema de secuenciar unidades en una línea de producción. Se tiene en cuenta la posibilidad de cambiar la secuencia de piezas mediante estaciones con acceso a un almacén intermedio o centralizado. El acceso al almacén además está restringido, debido al tamaño de las piezas.

Abstract

This paper presents a Genetic Algorithm (GA) for the problem of sequencing in a mixed model non-permutation flowshop. Resequencing is permitted where stations have access to intermittent or centralized resequencing buffers. The access to a buffer is restricted by the number of available buffer places and the physical size of the products.

Keywords: Genetic Algorithm; Non-permutation Flowshop; Intermittent and Centralized Constrained Buffer; Mixed model assembly line

1. Introduction

In the classical production line, only products with the same options were processed at once. Products of different models, providing distinct options, were either processed on a different line or major equipment modifications were necessary. For today's production lines this is no longer desirable and more and more rise the necessity of manufacturing a variety of models on one line, motivated by offering a larger variety of products to the client. Furthermore, the stock for finished products is reduced considerably with respect to a production with batches, and so are the expenses derived from it. 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 are limited to solutions which determine the job sequence before the jobs enter the line and maintain it without interchanging jobs until the end of the production line, known as permutation flowshop. Exact approaches for makespan minimization can be found in [10], [15], and [4], among others. In two recent reviews [6], [7] heuristic methods for sequencing problems are presented.

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

In the case of more than three stations and with the objective function to minimize the makespan, a unique permutation is no longer optimal. In [16] a study of the benefit of using non-permutation flowshops is presented. Furthermore, there exist various designs of production lines which permit resequencing of jobs: using large buffers (Automatic-Storage-and-Retrieval-System) which decouple one part of the line from the rest of the line [12]; buffers which are located off-line [11]; hybrid or flexible lines [5]; and more seldomly, the interchange of job attributes instead of physically changing the position of a job within the sequence [17]. Resequencing of jobs on the line is even more relevant with the existence of an additional cost or time, occurring when at a station the succeeding job is of another model, known as setup-cost and setup-time [2].

This paper presents a Genetic Algorithm for a mixed model non-permu-tation flowshop with the possibility to resequence jobs between consecutive stations. The buffers are located off-line either accessible from a single station (intermittent 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, the setup-cost and the setup-time. Several genetic operators are used, among them inheritance, crossover and mutation. In order to further improve the Genetic Algorithm, it was partitioned into two cascades. In the first cascade, the possibility of resequencing jobs within the production line is ignored and reduced chromosomes are used. In the second cascade, the resequencing possibilities, provided by stations with access to resequencing buffers, are taken into account.

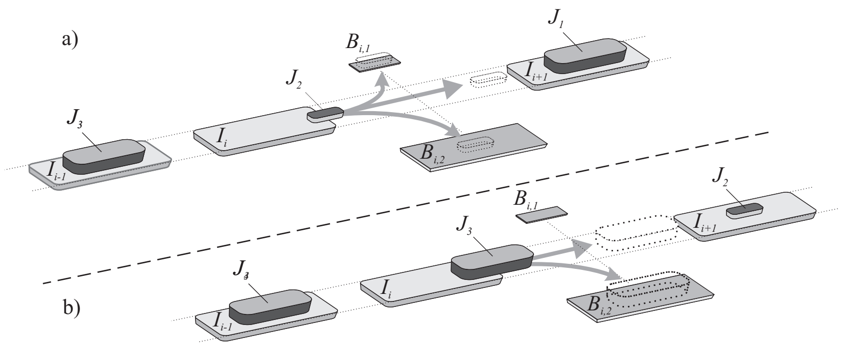


Figure 1: Scheme of the considered flowshop. The jobs J_j pass consecutively through the stations I_i. The buffer B_i permits to temporally store a job with the objective of reinserting it at a later position in the sequence. a) Job J_2 can pass through any of the two buffer places $B_{i,1}$ or $B_{i,2}$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

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. Also 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.

In what follows the problem is formulated in more detail and in continuation the applied Genetic Algorithm is explained. Then, a preliminary analysis and the adjustment of the genetic operators is performed, using the three steps *Rough-Adjustment*, *Repeatability*, *Fine-Adjustment*, ensuring flexibility and robustness. Thereafter promising results are presented, which demonstrate the relevance of the proposed concept, followed by the conclusions.

2. Problem definition

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 buffer place $B_{i,1}$, see figure 1b.

In a first step, the resequencing buffers are located intermittent, 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.

3. Genetic Algorithm

The concept of the Genetic Algorithm was first formulated by [8] and [9] and can be understood as the application of the principles of evolutionary biology, also known as the survival of the fittest. Genetic algorithms are typically implemented as a computer simulation in which a population of chromosomes, each of which represents a solution of the optimization problem, evolves toward better solutions. The evolution starts from an initial population which may be determined randomly. In each generation, the fitness of the whole population is evaluated and multiple individuals are stochastically selected from the current population, based on their fitness and modified to form a new population. The alterations are biologically-derived techniques, commonly achieved by inheritance, mutation and crossover. Multiple genetic algorithms were designed for mixed model assembly lines such as [3], [13], [19] and [18].

The heuristic used here is a variation of the Genetic Algorithm explained in [14]. The genes represent the jobs which are to be sequenced. The chromosomes υ, 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 fraction Π. In the non-permutation case, the chromosomes are $L+1$ times larger, resulting in the fractions $\Pi_1^{'}, ..., \Pi_{L+1}^{'}$, being L the number of resequencing possibilities. In both cases, special attention is required when forming the chromosomes, because of the fact that for each part of the production line every job has to be sequenced exactly one time.

The relevant information for each chromosome is its fitness value (objective function), the number of job changes and the indicator specifying if the chromosome represents a feasible solution. A chromosome is marked unfeasible 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, the one with fewer job changes is preferred.

3.1. Parameter and variable definition

The parameters and variables used in the Genetic Algorithm are as follows:

R	Population size		
s	Index of chromosomes	$s = 1, ..., R$	
υ_s	Chromosome s		
g	Index of generations		
G	Number of generations		
L	Number of Resequencing possibilities		
N	Number of Jobs		
MBS	Number of best solutions to maintain		
p_{b}	Probability to eliminate best solutions	p_{b}	$\in [0..1]$
$p_{\text{c-I}}$	Probability of crossover-I	$p_{\text{c-I}}$	$\in [0..1]$
$p_{\text{c-II}}$	Probability of crossover-II	$p_{\text{c-II}}$	$\in [0..1]$
$p_{\text{m-I(f)}}$	Probability of mutation I (forward)	$p_{\text{m-I(f)}}$	$\in [0..1]$
$p_{\text{m-I(b)}}$	Probability of mutation I (backward)	$p_{\text{m-I(b)}}$	$\in [0..1]$
$p_{\text{m-II}}$	Probability of mutation II	$p_{\text{m-II}}$	$\in [0..1]$
pos	Random position of a gene in the chromosome υ		
FP	Penalty for non-feasible solution		

3.2. Reduction of chromosome size

The computational effort necessary to solve the Genetic Algorithm is directly related to the size of the chromosomes. Therefore it is preferable to work with the minimum necessary size. The chromosome is basically formed by queuing the sequences $\Pi_1 ... \Pi_M$, M fractions of length N. Due to the fact that a change in the sequence can only occur in the case a station is provided with access to a resequencing buffer, the sequences for several consecutive stations is the same. The sequences of stations which are subsequent to a station with access to resequencing buffers, until the next stations with access to resequencing buffers, are not considered in the chromosome. This results in the reduction of the necessary chromosomes from $\Pi_1 ... \Pi_M$ to $\Pi_1^{'} ... \Pi_{L+1}^{'}$. L is the number of stations with access to a resequencing buffer.

Figure 2 shows an example of a plant with seven jobs to be processed on five stations. Two of the stations, I_2 and I_4, have access to a resequencing buffer. This results in the same sequence for the first two stations ($\Pi_1^{'} = \Pi_1 = \Pi_2$) and the next two stations ($\Pi_2^{'} = \Pi_3 = \Pi_4$). The final station processes the sequence $\Pi_3^{'} = \Pi_5$. The size of the resulting chromosome υ_s is reduced to a fraction of 2/5.

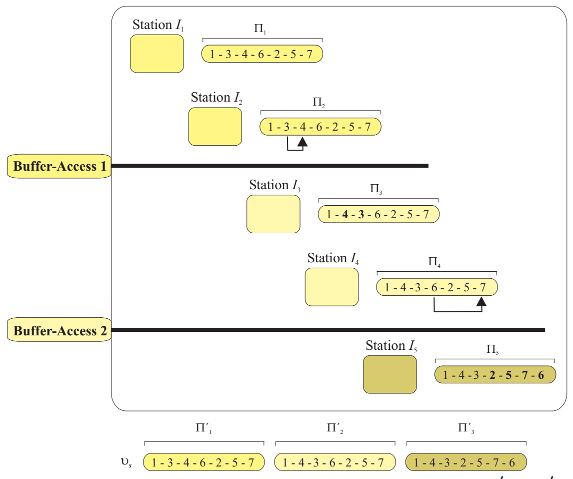


Figure 2: Reduction of chromosome υ_s from $\Pi_1 ... \Pi_5$ to $\Pi_1^{'} ... \Pi_3^{'}$ for the Genetic Algorithm due to the fact that resequencing is not possible for all stations.

3.3. Validation

Before applying the genetic operators, the actual population requires to be validated and the chromosomes for reproduction need to be selected.

Total fitness of population F: The total fitness of the population F is the sum of the individual fitness values of the individual chromosomes:

$$F = \sum_{s=1}^{R} eval(\upsilon_s) \tag{1}$$

Probability of individual chromosome p_s: The probability of the individual chromosomes p_s is a measure of how likely a chromosome is to be selected for reproduction. The sum of the over all chromosomes is equal to 1. The probability of the individual chromosomes is calculated as:

$$p_s = eval(\upsilon_s)/F \tag{2}$$

Sorting solutions by fitness: After the evaluation of the chromosomes with respect to their fitness, they are sorted in increasing order by their fitness. Then the fittest chromosome is the uppermost. The reason for sorting them is the far easier handling of the population when e.g. declaring taboo to overwrite a certain amount of best solutions, defined by MBS.

Cumulative Probability The cumulative probability is the sum of the probability of the individual chromosomes p_s, from the first chromosome until the one at position s in the list of chromosomes. The cumulative probability is calculated as:

$$q_s = \sum_{j=1}^{s} p_j \tag{3}$$

Storage of best solution: Apart from the initial population, the actual population does not necessarily contain a copy of the so far best feasible solution. If the actual population is the initial population, the best feasible solution is to be stored. Otherwise, the currently best solution gets overwritten if the actual population provides an enhanced, fitter, and feasible solution.

Deletion of duplicate solutions: In order to avoid occupation of chromosomes with duplicate samples, a duplicate chromosome is deleted. This is achieved after the chromosomes have been sorted by their fitness. The chromosomes which result in the same fitness are then evaluated regarding their genes' sequence. If two chromosomes are identical, one is deleted from the actual population and replaced by a new permutation chromosome, as in the initial population. An additional chromosome is not yet evaluated and therefore is assigned with the weakest fitness obtained so far, hence is not likely to reproduce. The added chromosome describes a permutation sequence, hence, is a feasible solution.

3.3.1. Selection of chromosomes

The selection process for chromosomes which are used for the crossover is based on spinning a roulette wheel $(R - 2 \cdot MBS)$-times. Reason for this reduced number is that, e.g., if p_C is set to 1.0, a total of 100% crossover has to be performed and only chromosomes can be overwritten which are not part of the "best solutions", determined by MBS or are part of the second copy of the best solutions which only get applied the genetic operator mutation. Each time a single chromosome is selected in the following way:

- For each chromosome in the population a random (float) number r is generated in the range [0..1].
- If $r < q_1$ then the first chromosome(υ_1) is select; otherwise the s-th chromosome υ_s $(2 \leq s \leq R)$ is selected such that $q_{s-1} < r \leq q_s$.

Clearly, some chromosomes are selected more than once. This results from the fact that more promising chromosomes are used for reproduction more often, the average stay even, and the worst are not reproduced and most probably will die off within one of the next generations.

The parameter p_C defines the overall probability of crossover. The expected number of chromosomes which undergo the crossover is $p_\text{C} \cdot R$. From the previously selected chromosomes some have to be discarded in order to meet the total number of chromosomes for the crossover:

- A random (float) number r is generated in the range [0..1].
- If $r < p_\text{C}$ the given chromosome is selected for crossover.

In the next step the selected chromosomes are mated depending on the type of crossover and the proportion with which the operator crossover-I and the operator crossover-II is performed, defined by $p_\text{C-I}$ and $p_\text{C-II}$, respectively. The sum of $p_\text{C-I}$ and $p_\text{C-II}$ is smaller or equal to 1.0:

$$(p_\text{C-I} + p_\text{C-II}) \leq 1.0 \tag{4}$$

The selection process for the operator mutation is similar, but only $(R - MBS)$ chromosomes are selected and instead of the p_C, the probability p_m is applied.

3.4. 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 of applying a genetic operator X to a chromosome. Figure 3 gives an overview on how the genetic operators are applied to the population of generation g in order to form the next generation $g + 1$.

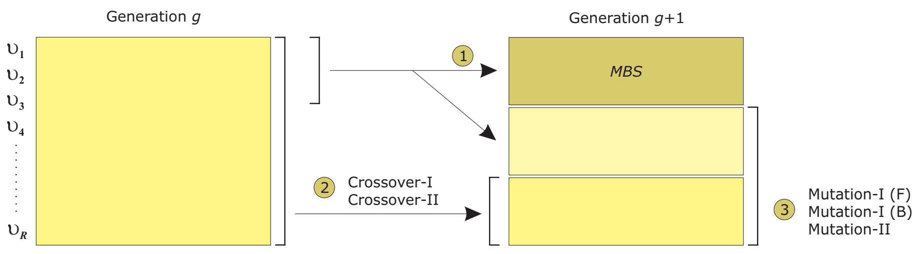


Figure 3: Application of the genetic operators to the population of generation g to form the next generation $g + 1$. The sequence in which the genetic operators are applied is (1) Inheritance, (2) Crossover and finally (3) Mutation.

Inheritance: The parameter MBS determines the percentage of the best solutions which will be copied directly to the next generation, called the cluster of promising chromosomes, ensuring that promising chromosomes are not extinct. In order to not remain in a local minimum, the parameter p_{b} determines the percentage of chromosomes which are removed from this cluster.

Mutation: This operator specifies the operation of relocating jobs at position pos_1 to position pos_2 within the same fraction of a chromosome. Two mutation operators are applied, mutation-I and mutation-II (figure 4). Furthermore, there exist two cases for mutation-I: forward mutation, where $pos_1 < pos_2$; and backward mutation, where $pos_1 > pos_2$. 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 $p_{\text{m-II}}$.

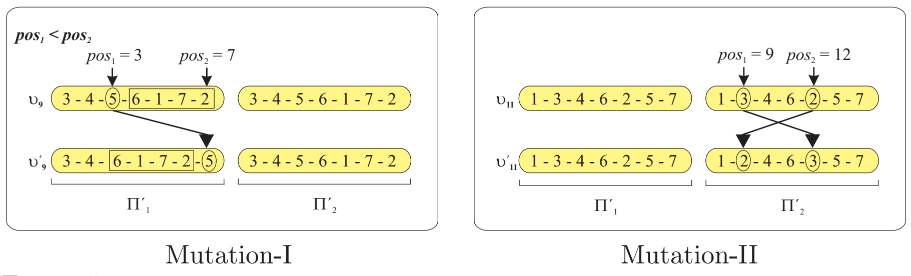


Figure 4: **Operators mutation-I and mutation-II.** a) The job at position pos_1 is taken off the line and reinserted to the line at position pos_2. b) The two jobs at position pos_1 and pos_2 are interchanged.

Crossover: This operator specifies the operation of crossing information of two chromosomes. Two crossover operations are applied, crossover-I (figure 5a,b) and crossover-II (figure 5c,d). The probabilities with which these operations are applied to a chromosome are $p_{\text{C-I}}$ and $p_{\text{C-II}}$, and the crossover points are defined by the random number pos, and the pair pos_1 and pos_2, respectively.

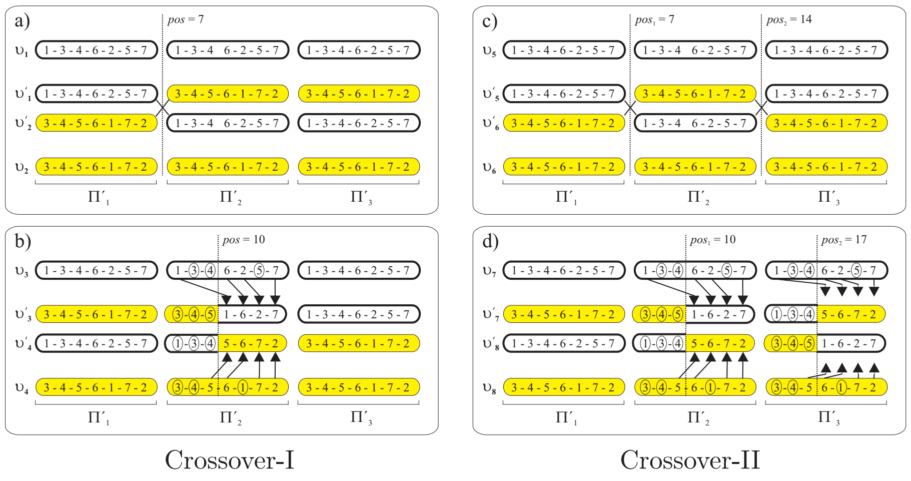


Figure 5: **Operators crossover-I and crossover-II.** a) and c) In the simple case the crossing takes place between two main fractions of the chromosome. After the crossover point the chromosomes are completely crossed over. b) and d) In the more complex case it has to be assured that each job is sequenced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

3.5. Overwrite-position for crossover

The new chromosomes, generated by the genetic operators Crossover-I and Crossover-II, can overwrite any of the chromosomes which are not part of the best solutions or the second copy of best solutions which are reserved for the genetic operators of mutation. Two possible strategies are implemented in the Genetic Algorithm:

- **Last position:** The chromosomes of the new population are overwriting the chromosomes of the previous population in increasing order of their fitness, starting with the weakest.
- **Random position**: The chromosomes of the new population are randomly overwriting the chromosomes of the previous population. Clearly one position is not overwritten two times in order to ensure the correct number of new chromosomes.

In both cases neither the best solutions nor the second copy of the best solutions, which are reserved for mutation only, are overwritten.

3.6. Cascading

In order to further improve the Genetic Algorithm, 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.

3.7. Condition for termination of Genetic Algorithm

Apart from the number of generations (G), which terminates the Genetic Algorithm when the maximum number of generations is reached, the algorithm can use a second condition which may result in an early termination, the Convergence-control. In the case in which the algorithm has not improved the sofar best solution for the last 300 generations, it is assumed that the the algorithm has converged and is interrupted.

4. Preliminary Analysis of Parameters

Preceding to the adjustment of the parameters of the genetic operators, a preliminary analysis is performed. This analysis intents to obtain a better understanding of the behavior of the parameters, as for example to estimate how likely it is for the preceding adjustment of the parameters to remain in local minima and consists in keeping all except two parameters constant and was performed on three pairs: $p_{\text{c-I}}/p_{\text{c-II}}$, $p_{\text{m-I(f)}}/p_{\text{m-I(b)}}$, and $p_{\text{m-I(f)}}/p_{\text{m-II}}$. Here only the pair $p_{\text{m-I(f)}}/p_{\text{m-I(b)}}$ is presented, the other two pairs showed similar results.

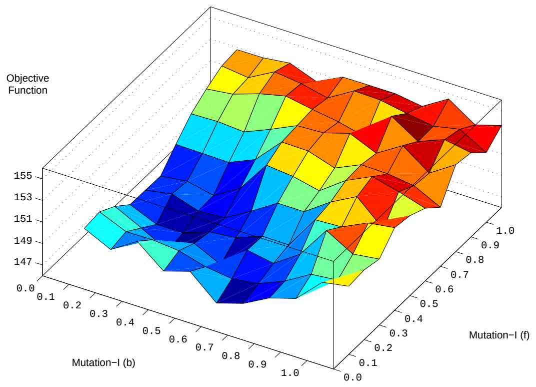


Figure 6: Study of the influence of $p_{\text{m-I(f)}}$, the probability for mutation-I (f) and $p_{\text{m-I(b)}}$, the probability for mutation-I (b), on the objective function.

4.1. Mutation-I (f) versus Mutation-I (b)

The two parameters $p_{\text{m-I(f)}}$ and $p_{\text{m-II}}$ in the range [0.0, 0.1, ..., 1.0]. The other three parameters are set to small values: $p_{\text{c-I}} = 0.3$, $p_{\text{c-II}} = 0.2$, $p_{\text{m-II}} = 0.1$.

Figure 6 shows the influence of the two parameters for mutation-I on the resulting value of the objective function. The plotted mesh shows that a continuous surface, indicating that local minima are not to be expected. Furthermore, the values of the objective function are better when both parameters for mutation-I are decreased. Except for the case in which both of them become too small.

4.2. Variability of solutions

The Genetic Algorithm is based on random numbers, giving the algorithm its strength. However, this also leads to the disadvantage that the algorithm on the other hand is not very predictable and in order to determine promising parameters, useful for a multitude of problems, the analysis of the parameters is to be repeated with various different seeds.

Figure 7a shows the variability of the Genetic Algorithm with respect to the obtained solutions. The same problem is solved 100 times, each time a different seed is used, for the permutation case as well as for the non-permutation case. The solutions, permitting non-permutation sequences, in general result in better solutions with a larger deviation. The average value of the objective function of the particular example, used in figure 7a, is 500.93 with a standard deviation of 2.96 for the permutation case and 490.21 with a standard deviation of 4.13 for the non-permutation case.

Analyzing the obtained data with respect to the number of job-changes with more detail, figure 7b shows that in order to obtain better solutions, the number of jobs, which have to be taken off the line for resequencing, tends to be higher.

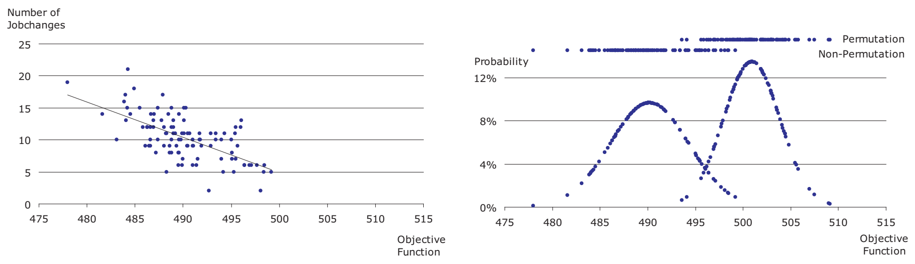


Figure 7: Variability of the solutions of the Genetic Algorithm. The same problem is solved 100 times, each time with a different seed, showing the following behavior: a) Probability of the solution for the permutation case as well as for the non-permutation case. b) Dependency of the number of job-changes on the objective function for the non-permutation case. Better solutions tend to have a larger number of jobs which have to be taken off the line in order to be resequenced.

5. Parameter adjustment

The adjustment of the values of the genetic operators, which are used for the two cascades of the Genetic Algorithm, is determined with an extended experimentation and consists of three steps:

Rough adjustment: In order to adjust the parameters in a robust manner, different sets of parameters are applied to a series of 14 differently sized problems, varying the number of jobs to be sequenced, the number of stations and the number of resequencing possibilities. During the rough adjustment only one unique seed is used for the random number generation in the Genetic Algorithm. The sets of parameters are summarized and the 300 most promising which show good performance on all problem sizes are used for further adjustment.

Repeatability: The use of only one seed in the rough adjustment requires to determine amongst the promising sets of parameters which set achieves good results for a multitude of seeds. The fact that a set of parameters achieves good results for different seeds indicates that the same set of parameters also performs well for different plant configurations. The promising sets of parameters are verified with 16 different seeds for the 14 differently sized problems. Once the sets of promising parameters are examined with respect to repeatability, one set is used for the fine adjustment, determined by grouping into clusters [1].

Fine adjustment: Due to the fact that in the previous analysis predetermined discrete values for the parameters are used, a fine adjustment succeeds. The genetic operators are subject to an adjustment of 0.1 for the previously determined sets of parameters and are revised with 16 seeds for the 14 differently sized problems, used for the repeatability.

Cascade	R	G	MBS	p_{b}	$p_{\text{c-I}}$	$p_{\text{c-II}}$	$p_{\text{m-I(f)}}$	$p_{\text{m-I(b)}}$	$p_{\text{m-II}}$
Step 1	100	1000	0.05	0.1	0.3	0.6	0.25	0.25	0.25
Step 2	100	10000	0.05	0.4	0.5	0.35	0.45	0.1	0.1

Table 1: Characteristic values of the Genetic Algorithm. The first cascade is applied to determine a generation with only permutation solutions, which is then used as an initial generation for the second cascade.

The experimentation is first performed on the permutation case (first cascade), and then on the non-permutation case (second cascade). Table 1 lists the results for the genetic operators used in the following performance study.

6. Performance Study

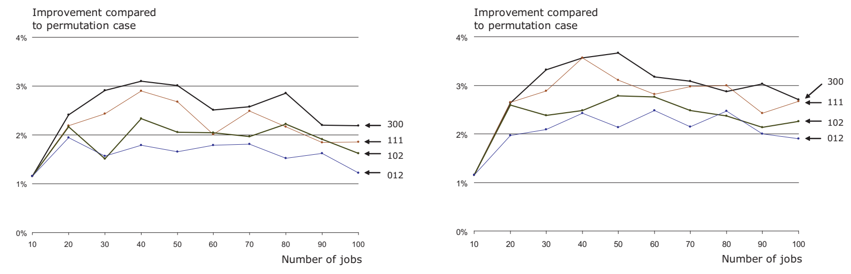
For the study of performance, a flowshop which consists of 5 stations is considered. The range of the production time is [0...20] such that for some jobs exists zero-processing time at some stations, for the setup-time [2...8] and for the setup-time [1...5]. The number of jobs is varied from 5 to 100 with increments of 5. The objective function, is the weighted sum of the makespan (factor of 1.0) and the setup-cost (factor of 0.3), where the setup-time has is not concerned with a weight but is indirectly included in the calculation of the makespan.

Case	Intermittent			Centralized		
Size	l	m	s	l	m	s
(300)	1/2	0/0	0/0	3	0	0
(111)	0/1	1/0	0/1	1	1	1
(102)	0/1	0/0	1/1	1	0	2
(012)	0/0	0/1	1/1	0	1	2

Table 2: Allocation of the buffer places to the buffers. In the intermittent case the allocation is done to two different buffers.

6.1. Difference in physical size of buffer places

Introducing limitations on the physical size of the buffer places on one side restricts possible solutions but on the other side minimizes the necessary buffer area. This limitation arises, for example, in a chemical production. The arrangement of two tanks which are located off the line, accessible after a certain station, equals an intermittent resequencing buffer with two buffer places. With tank capacities of 50 and 100 liters, a client order of 80 liters can be stored only in the larger of the two tanks which is capable of storing this volume. Whereas, a client order of 50 liters can be stored in either of the tanks. A close look at the local conditions may amortize an increase in the objective function compared to an investment reduction with respect to tank size and gained area.



a) Intermittent case b) Centralized case

Figure 8: Influence of the variation of the physical size of the buffer places. The code "102" represents 1 large, 0 medium and 2 small buffer places. In the intermittent case, the buffer places are divided to two buffers, each with access from a designated station. In the centralized case, the same two stations have simultaneously access to the buffer, containing all three buffer places. The ratio of jobs is $\frac{3}{10}$ large, $\frac{3}{10}$ medium and $\frac{4}{10}$ small.

As a concrete example, three differently sized buffer places (large, medium, small) are available and the ratio of jobs is $\frac{3}{10}$ large, $\frac{3}{10}$ medium and $\frac{4}{10}$ small. As in the previous section, the second and the third station have access to the resequencing buffers and table 2 shows the allocation of the buffer places to the buffers, considering eight scenarios. The code "300" represents 3 large, 0 medium and 0 small buffer places. In the intermittent case the first buffer is provided with 1 and the second buffer with 2 large buffer places. In the centralized case the same two stations have access to a single centralized buffer, containing the three buffer places. Figure 8 shows the influence of the limitation of the physical size. The variation of the size of the buffer places towards smaller buffer places on the one hand decreases the benefit achieved by the possibility of resequencing jobs. On the other hand, it may amortize when taking into account the reduction of investment with respect to tank size and gained area.

7. Conclusions

This paper has presented a Genetic Algorithm which was applied to a mixed model non-permutation flowshop using constrained buffers. The algorithm uses the genetic operators inheritance, crossover and mutation and is designed to consider intermittent or centralized resequencing buffers. Furthermore, the buffer access is restricted by the number of buffer places and the physical size of jobs. The reduction of the chromosome size benefits the performance of the algorithm, which is further improved by the use of a two step cascade, first seeking permutation sequences, then widening the solution space to non-permutation sequences.

The preliminary analysis of the behavior of the genetic operators appeared to be valuable, showing that the solution space, when varying two genetic operators, is a continuous surface without local minima. Furthermore the variability of the solutions was observed, pointing out that the solutions permitting non-permutation lead to better solutions but with a larger deviation. The parameter adjustment was designed such that both, flexibility and robustness were ensured.

Then, the study of performance demonstrated the effectiveness of resequencing jobs within the line. The results of the simulation experiments revealed the benefits that come with a centralized buffer location, compared to the intermittent buffer location. It either improves the solution or leads to the utilization of fewer resequencing buffer places. An increased number of large buffer places clearly improves the objective function and including buffers, constrained by the physical size of jobs to be stored, on one side limits the solutions but on the other side minimizes the necessary buffer area.

In order to take full advantage of the possibilities of resequencing jobs in a mixed model flowshop, additional installations may be necessary to mount, like buffers, but also extra efforts in terms of logistics complexity may arise. The additional effort is reasonable if it pays off the necessary investment. Due to the strong dependency on local conditions, a general validation is not simple and was not part of this work.

8. Bibliography

[1] B. Balasko, J. Abonyi, and B. Feil. Fuzzy clustering and data analysis toolbox; for use with matlab. 2005.

[2] A. Bolat. Sequencing jobs on an automobile assembly line: objectives and procedures. *International Journal of Production Research*, 32(5):1219–1236, 1994.

[3] A. Bolat, I. Al-Harkan, and B. Al-Harbi. Flow-shop scheduling for three serial stations with the last two duplicate. *Computers & Operations Research*, 32(3):647–667, 2005.

[4] J. Carlier and I. Rebai. Two branch and bound algorithms for the permutation flowshop problem. *European Journal of Operational Res.*, 90(2):238–251, 1996.

[5] T. Engstrőm, D. Jonsson, and B. Johansson. Alternatives to line assembly: Some swedish examples. *International Journal of Industrial Ergonomics*, 17(3):235–245, 1996.

[6] J.M. Framinan, J.N.D Gupta, and R. Leisten. A review and classification of heuristics for permuation flowshop scheduling with makespan objective. *Technical Report OI/PPC-2001/02*, 2002. Vers. 1.2.

[7] J.M. Framinan and R. Leisten. Comparison of heuristics for flowtime minimisation in permuation flowshops. *Technical Report IO-2003/01*, 2003. Vers. 0.5.

[8] J.H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM J. Comput.*, 2(2):88–105, 1973.

[9] J.H. Holland. Adaptation in natural and artificial systems. *University of Michigan Press, Ann Arbor*, 1975.

[10] E. Ignall and L.E. Schrage. Application of the branch and bound technique to some flow-shop problems. *Operations Research*, 13(3):400–412, 1965.

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

[12] H.F. Lee and S.K. Schaefer. Sequencing methods for automated storage and retrieval systems with dedicated storage. *Computers and Industrial Engineering*, 32(2):351–362, 1997.

[13] G. Levitin, J. Rubinovitz, and B. Shnits. A genetic algorithm for robotic assembly line balancing. *European Journal of Operational Res.*, 168:811–825, 2006.

[14] Z. Michaelewicz. *Gentic Algorithms + Data Structures = Evolution Programs.* Springer Verlag, 3rd edition, 1996.

[15] C.N. Potts. An adaptive branching rule for the permutation flowshop problem. *European Journal of Operational Research*, 5(2):19–25, 1980.

[16] C.N. Potts, D.B. Shmoys, and D.P. Williamson. Permutation vs. non-permutation flow shop schedules. *Operations Res. Letters*, 10(5):281–284, 1991.

[17] P. Rachakonda and S. Nagane. Simulation study of paint batching problem in automobile industry. *http://sweb.uky.edy/˜pkrach0/Projects/ MFS605Project.pdf*, 2000. consulted 14.07.2004.

[18] R. Rubén and M. Concepción. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, 169(3):781–800, 2006.

[19] L. Wang, L. Zhang, and DZ. Zheng. An effective hybrid genetic algorithm for flow shop scheduling with limited buffers. *Computers & Operations Research*, 2006. Article in Press.