Journal of Intelligent Manufacturing, 14, 323–339, 2003

Finding multiple solutions in job shop scheduling by niching genetic algorithms

E. PÉREZ,[1] F. HERRERA[2] and C. HERNÁNDEZ[1]

[1]*Industrial Engineering Group, School of Industrial Engineering, University of Valladolid, 47011–Valladolid, Spain*
E-mail: elena, cesareo@eis.uva.es
[2]*Department of Computer Science and Artificial Intelligence, University of Granada, 18071–Granada, Spain*
E-mail: herrera@decsai.ugr.es

Received May 2001 and accepted February 2002

The interest in multimodal optimization methods is increasing in the last years. The objective is to find multiple solutions that allow the expert to choose the solution that better adapts to the actual conditions.

Niching methods extend genetic algorithms to domains that require the identification of multiple solutions. There are different niching genetic algorithms: sharing, clearing, crowding and sequential, etc.

The aim of this study is to study the applicability and the behavior of several niching genetic algorithms in solving job shop scheduling problems, by establishing a criterion in the use of different methods according to the needs of the expert. We will experiment with different instances of this problem, analyzing the behavior of the algorithms from the efficacy and diversity points of view.

Keywords: Job shop scheduling problem, multimodal optimization, genetic algorithms, niching methods

1. Introduction

The objective of the scheduling problem is to settle the sequence of jobs for each machine, by defining the time intervals in which the operations have to be processed. It has to be accomplished in such a way that each machine can only perform one operation at a time, and also the technological constraints must be respected.

The complexity of this problem comes from the large number of constraints. This problem belongs to the NP-Hard problems (Garey and Johnson, 1979), for which, are no known algorithms that assure to find an optimal solution in polynomial time. This is the main reason for the great interest in this topic as well as its high applicability in the industry.

In the last few years, studies have not only been focused on solving the problem with the highest efficacy, but also to sort it out with the highest efficiency to adapt the studies to the new needs in the optimization process. A common optimization problem is the simplification of a real world problem because of its high complexity or the impossibility of defining all parameters (unknown, stochastic or non-quantifiable). Thus, it is desirable to offer different optimal solutions to be judged later by the expert, or to allow him/her to know some characteristics of the search space by exploitation or exploration (Harik, 1995).

Optimization methods that work with one solution at a time (tabu search, simulated annealing or iterated local search) need to restart the process to find multiple final solutions, but there are no mechanisms to guarantee multiple different solutions. Although, there is a modification in the iterated local search which uses a population of solutions (Stützle, 1998),

there are very few studies about finding multiple different solutions.

The capability of genetic algorithms (GAs) to work on a set of solutions allows that the evolution process obtains different optimal solutions (Goldberg, 1989). Nevertheless, the simple GA is not able to maintain different solutions. The research studies based on the preservation of the diversity by niching techniques in GAs have provided very promising results. These techniques permit not only to obtain multiple different solutions, but also to preserve useful diversity against a premature convergence that guides us to poor (local optimal) solutions (Sareni and Krahenbuhl, 1998).

The aim of this work is to study the applicability and the behavior of niching techniques in solving job shop scheduling problems, by establishing a criterion in the use of different methods according to the needs of the expert. This study is developed according to the following perspectives:

(1) We compare four different systems of niching methods (sharing, clearing, crowding and sequential) with some modifications for a better adaption to the needs of the search.

(2) We establish the most appropriate system parameters for different search needs.

(3) We determine which technique is the most efficient and generates the maximum number of different solutions.

(4) We analyze what method performs a highest exploration in the search space by finding optimal that belong to different areas, or a highest exploitation of some characteristic of the problem with optimal very similar.

We must remark that the job shop scheduling problem is strongly multimodal, that is, it has different global and local optima. This characteristic can be considered as a search space typified by hills, valleys and mountainous areas, which makes it the perfect testing-ground.

This paper starts with a description of the scheduling theory, including the problem definition and the main techniques that have been used to solve it. In Section 3 we introduce GAs and review the different niching methods that will be analyzed. In Section 4 we establish a comparison among niching GAs from the efficacy, number of different found solutions, and the exploration or exploitation of search space points of view. Finally, some concluding results will be pointed out.

2. Scheduling theory

The scheduling theory is characterized by a large number of different types of problems. However, each of these scheduling problems can be characterized by four-dimensions (a, b, A, B) (Conway *et al.*, 1967): where a is the number of jobs (J_i), b the number of machines (M_j), A the type of technological constraints and B the cost function.

Each job (J_i) consists of a set of operations (or tasks) (O_{ij}), each of which has to be processed by a machine (M_j) for a certain period of time (T_{ij}). The order of operations is defined by the technological constraints (A), where $A = \{P, F, J\}$:

(1) P (permutational flow shop) represents that the jobs have the same movement on all machines in the shop, and the machines have the same sequence of jobs (Fig. 1(a)).

(2) F (flow shop) is similar to the permutational one, but in this case each machine has its own sequence of jobs (Fig. 1(b)).

(3) J (job shop) is the most general and complex case where each job has its own movement on the machines, and each machine has its own sequence of jobs (Fig. 1(c)).

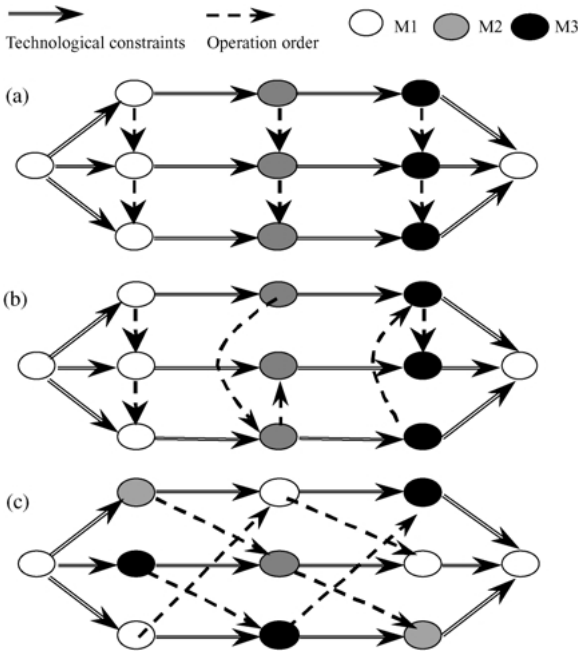

Fig. 1. Types of scheduling problems.

The cost function can define termination times (makespan or *Cmax*), delay times (*Tmax*) or total flow time (*Fmax*), among others. An extensive review of this classification can be obtained in Brucker (1997).

At the beginning, attempts were directed towards the definition of the easiest problems (French, 1982) and their resolution by mathematical methods (Greenberg, 1968; Carlier and Pinson, 1989). However, due to the limitation of these methods that are computationally prohibitive for real-word problems, the heuristic approaches emerged (Panwalkar and Iskander, 1977; Adams *et al*., 1988). Using them, it is possible to solve problems of larger size but with a loss of precision. In this case, to solve the problem is more important than to find the optimal solution.

The academic community was slow to accept this approach. But the best results obtained in high complexity problems soon permitted its evolution towards meta-heuristic methods like tabu search (TS) (Glover and Laguna, 1997; Nowicki and Smutnicki, 1996), simulated annealing (SA) (Kirkpatrick *et al*., 1983; Van *et al*., 1992), GAs (Mattfeld, 1995), neural networks (Yang and Wang, 2000), ILS (Stützle, 1998; Ramalhinho *et al*., 2000), and hybrid optimization strategies (Wang and Zheng, 2001), among others.

Because this is a very well studied problem by different techniques, it has a large number of benchmarks for which the optimal value or an upper bound are known. The most important benchmarks for a lot of different problems are available in the web site *http://mscmga.ms.ic.ac.uk/jeb/jeb.html*. In Table 1, some of these results are showed for different techniques for the job shop scheduling problem. Each entry indicates the result found in the corresponding paper for every instance: *mt06*, *la01*, *mt10* and *mt20*.

We have selected the *mt06* and *la01* benchmarks because of their size, since they are sufficiently small to know the landscape of the search space. Furthermore, the *mt10* and *mt20* instances will allow us to know the efficacy of the methods to find the optima. The definition of these instances is shown in Appendix A.

As we can observe, the number of optima found is never detailed in this type of tables, because efficacy has only been the traditional objective pursued. However, the new production needs involve having different possibilities from which we can choose the most adequate for a variable manufacturing environment.

3. Genetic algorithms

3.1. *Introduction*

GAs are global search algorithms with a general purpose that use principles inspired by natural population genetics. The GAs appeared in the 1960s (see a good collection of the first proposals in Fogel,

Table 1. Optima, methods and authors for the job shop problem (the optima are in bold type)

Method	*Reference*	*mt06*	*la01*	*mt10*	*mt20*
Branch & Bound	(Balas, 1969)	**55**		1177	1231
	(McMahon and Florian, 1975)	**55**		972	**1165**
	(Baker and McMahon, 1985)	**55**		960	1303
	(Carlier and Pinson, 1989)	**55**		**930**	**1165**
Shifting Bottleneck	(Adams *et al*., 1988)	**55**	**666**	**930**	1178
GA	(Nakano and Yamada, 1991)	**55**		965	1215
	(Yamada and Nakano, 1992)	**55**		**930**	1184
	(Fang *et al*., 1993)	**55**		939	**1165**
	(Della *et al*., 1995)	**55**	**666**	946	1178
	(Mattfeld, 1995)	**55**	**666**	**930**	**1165**
TS	(Dell'Amico and Trubian, 1993)	**55**	**666**	**930**	**1165**
Hybrid GA + SA	(Wang and Zheng, 2001)	**55**	**666**	**930**	**1165**
SA	(Van *et al*., 1992)	**55**	**666**	**930**	**1165**

1998), but it is not until the 1970s when researchers began to use them as a useful optimization and search tool. In a GA, each individual in the population represents a candidate solution to the problem and has an associated fitness to determine which individuals are used to form new ones in the process of competition. The new individuals are created by using genetic operators, such as crossover and mutation (Goldberg, 1989; Michalewicz, 1995).

The main parts inside of a GA are the following ones:

- *Evaluation*. Value of objective function for each solution.
- *Coding*. Critical decision in the design of the algorithm. It allows us to handle the potential solutions in a simple manner.
- *Genetic operators*. The heart of the algorithm. They allow us to explore and exploit search areas. The classical operators in the GA are:
  - *Crossover operator*. It allows the exchange of the genetic material of the parents selected for reproduction.
  - *Mutation operator*. It incorporates diversity to the search process.
- *The replacement process*. The offspring population will be the initial population for the next generation.

3.2. *Niching genetic algorithms*

Before presenting different niching techniques, we will explain the main concept on which niching GAs are based, i.e., the distance as a measure of proximity between individuals, $d(i,j)$.

The concept of closeness or remoteness (similarity) requires the calculation of the distance between solutions, which is problem-dependant. For example, in the real coding it is possible to define the metric distance. In our job shop problem, we can define the distance as the number of operations situated in different places for each machine. An example is shown in Table 2, with three machines and three jobs.

Table 2. Example of distances for scheduling problem

Solutions	*Phenotypic distance*
(1 3 2) (2 3 1) (3 2 1) (3 2 1) (1 3 2) (3 2 1)	5

In the following subsections we introduce the different niching techniques analized in this paper.

3.2.1. *Sharing fitness methods*

The classical sharing method is based on the sharing fitness function, which decreases the fitness of individuals in accordance with the number of similar individuals in the population. The sharing fitness of individual j (f_j^*) is the original fitness (f_j) divided by the sharing function $(Sh(d(i,j)))$ (see Equation 1):

$$f_j^* = \frac{f_j}{\sum_{i=1}^{N} Sh(d(i,j))} \tag{1}$$

where the sharing function depends on the distance between the individuals i and $j(d(i,j))$ in line with Equation 2.

$$Sh(d(i,j)) = \begin{cases} 1 - \left(\frac{d(i,j)}{\sigma_{share}}\right)^{\alpha} & \text{If} \quad d(i,j) < \sigma_{\text{share}} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The necessary parameters for this method are the maximum distance that defines a niche (σ_{share}), and the slope of the sharing fitness function (α) whose most frequently used value is 1.

Nevertheless, later studies have shown some limitations (Sareni and Krahenbuhl, 1998). The fitness sharing must be implemented with the least biased selection methods. For this, a possible improvement is to combine the tournament with a modified fitness sharing called continuously updated sharing (Oei *et al*., 1991), as it is described below:

(1) The shared fitness is calculated by considering each selected individual as a father. That is, the feedback is used immediately in the next shared fitness calculation.

(2) To implement this method, authors proposed to introduce a niche size parameter (n^*), which is used in the tournament process. When both individuals belong to niches whose members are less than n^*, then the individual with higher fitness will gain, otherwise the individual that belongs to the niche with less members will gain.

Figure 2 shows the flowchart of the developed algorithm for this method, where the sharing value of a chromosome j is updated when its distance with the selected parent is less than σ_{share}.

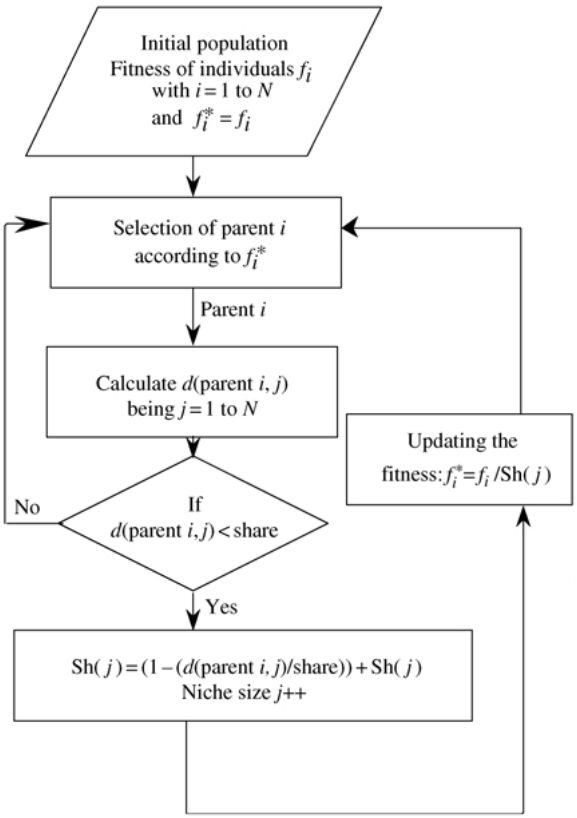

Fig. 2. Flowchart of continuously updated sharing.

3.2.2. *Clearing method*

The clearing method is very similar to fitness sharing but is based on the concept of limited resources of the environment (Pétrowski, 1996). This process is applied after the evaluation process and before the selection. The population is ordered according to their fitness, from the best to the worst. The first individual is called dominant (there are no individuals better than it), and it is compared with the $(n-1)$ remaining individuals of the population. By this comparison, we will obtain those individuals belonging to the same niche. Only the k best individuals of each niche will survive. The fitness of the rest of individuals will be reset to zero. The process will be repeated, but only with individuals whose fitness is greater than zero.

3.2.3. *Crowding methods*

In this group of methods the process of replacement is modified to allow the formation of niches in the population. In the traditional (generational) GA, the new individuals created in the reproduction process replace the whole population in each generation. However, in a steady state GA, each new created individual replaces one in the population (generally the worst). When this replacement is performed considering the distance, then we are dealing with crowding methods.

Mahfoud randomly grouped the individuals of the population in pairs in order to apply the crossover and mutation operators. Then, each child competes in a tournament versus the father most similar to it (Fig. 3). The winner, i.e., the survivor, is the one with the best quality (Mahfoud, 1992).

Since this method was developed, different versions have arisen (Cedeño *et al*., 1995; Harik, 1994). In the present study, we have also analyzed the modification made by the Cedeo *et al*. (1995). In order to form couples for the reproduction process, an individual is selected by its quality and the other one is randomly selected. On the replacement process, *CF* groups with *CS* individuals are randomly chosen. The individual with the smallest quality among the most similar of each group is replaced by this new created individual (Cedeo *et al*., 1995).

3.2.4. *Sequential niche method*

This method is based on multiple independent runs, but trying to eliminate the problem of searching in space zones that have already been explored in previous runs.

With this idea Beasley *et al*. (1993) created a method by which when the GA has explored a zone, the search never returns there again. With a very similar system to the sharing fitness method the already explored peaks or zones are eliminated. Thereby, the process incorporates the obtained experience in previous runs by storing the found optima.

In each run, the GA obtains a solution to the problem. This solution will be considered as the representative peak of the niche to which it belongs. In the next runs, the GA uses this information to avoid searching again and finding the same optimum. To do so, the fitness of the each of the new individuals generated by genetic operators will be diminished according to the proximity to the optima found in previous runs (see Equation 3).

$$\begin{aligned} M_0(x) &= f(x) \\ M_{n+1}(x) &= M_n(x) \times G(x, S_n) \end{aligned} \tag{3}$$

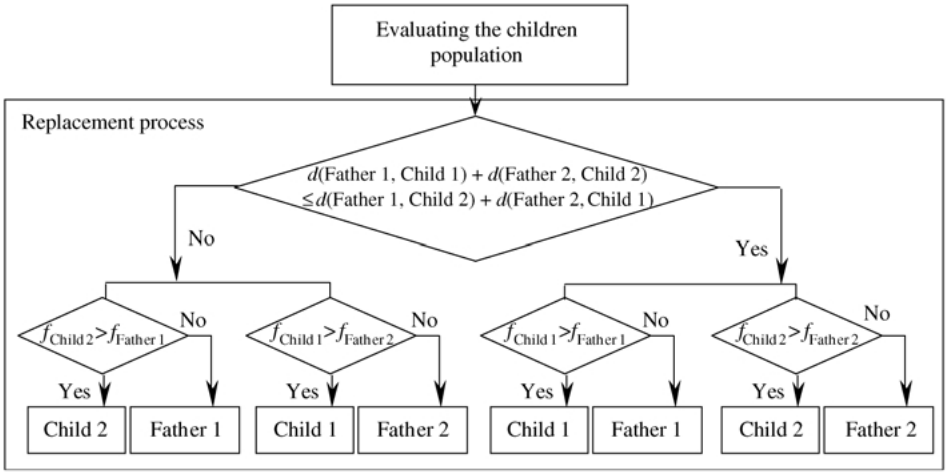

Fig. 3. Replacement process in deterministic crowding method.

where S_n is the optimal solution obtained in the previous run, x is the new individual and $G(x, S_n)$ is the function of the discount (or sharing). Considering n iterations, it can be:

$$G(x, S_n) = \begin{cases} \left(\frac{d(x,S_n)}{\sigma_{\text{share}}}\right)^{\alpha} & \text{if } d(x, S_n) < \sigma_{\text{share}} \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

3.3. *Algorithm components and parameters*

The algorithm that has been used in this study starts with a randomly created initial population of 50 individuals and runs during 600 generations (except for the largest problems in size where the number of generations has been increased to 5000).

There are different possibilities to code the solution of the job shop scheduling problem. There are the direct (Bruns, 1993; Kobayashi *et al.*, 1995), the binary (Nakano and Yamada, 1991), the circular (Fang *et al.*, 1993), and the permutation with repetition (Mattfeld, 1995). We have selected the latter because with it the genetic operators always obtain valid children.

The genetic operators must be adapted to the problem. We have used the classical order crossover but adapted it to the job shop (Fig. 4), and the order based mutation (OBM) developed by Mattfeld (1995), (Fig. 5).

The probabilities are 0.8 for crossover and 0.2 for mutation. The latter is realized on the string not on each element of the string. This is why its value is so high.

To simplify the tables of results we number each method as follows:

(1) Classical sharing method (Goldberg and Richardson, 1987)
(2) Continously updated sharing (Oei *et al.*, 1991)
(3) Clearing (Pétrowski, 1996)

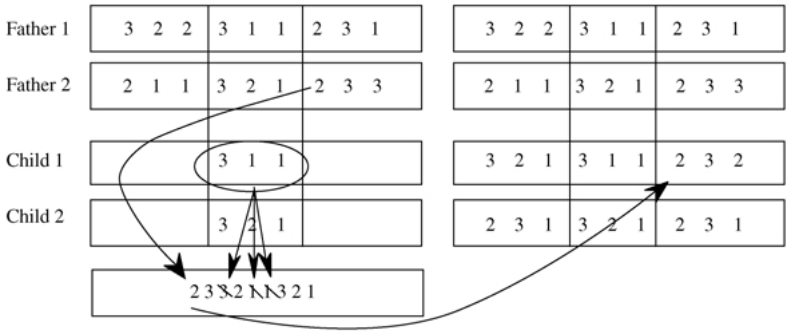

Fig. 4. Order crossover for the job shop.

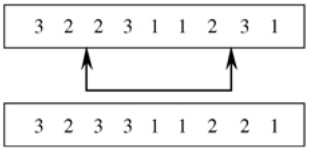

Fig. 5. Order based mutation.

(4) Deterministic crowding (Mahfoud, 1992)
(5) Multiniche crowding (Cedeo *et al*., 1995)
(6) Sequential niche method (Beasley *et al*., 1993)

The parameters for each method are detailed in Table 3:

4. Experimental study

We study how the different niching schemes behave by considering:

(1) The efficacy: The first objective for all optimization methods,
(2) The diversity in convergence: The number of different optima found, very important in multimodal problems, and
(3) The exploration: The percentage of runs with optima belonging to different areas of search space, and the exploitation, percentage of runs with optima in the same area.

To draw a comparison among them we have made 40 repetitions of each experiment. Before presenting the results, we will previously describe the instances and their landscapes.

4.1. *Description of instances and their landscapes*

We have selected four instances to compare the different developed methods: two of small size (*mt06* and *la01*), and other two more difficult (*mt10* and *mt20*). The smaller size of the former allows us to know their landscapes. Their description is presented in Appendix A.

For *mt06* we have found three different types of optima: two big mountainous areas, and one isolated peak:

(1) The first (A) is formed by at least 19 different optima. They have the same sequence of jobs in the first machine $(1\ 4\ 3\ 6\ 2\ 5)$ (first of the 19 solutions in Table 4). The average distance among them is 5, being the range values $[2, 11]$.

(2) Within the second area (B) we have found 18 different optima. The sequence of jobs in the first machine is $(1\ 4\ 6\ 3\ 2\ 5)$ (the next 19 rows in Table 4). The average distance is also 5, and the range values $[2, 10]$.

(3) Likewise, there is another type of optima (C) defined by the sequence in the first machine $(1\ 6\ 4\ 3\ 2\ 5)$. There are four different optima of this type and the average distance among them is 2.7 (one of them is the last solution in Table 4).

The average distances between the different types are shown in Table 5.

In accordance with these distances, two mountainous areas (A and B) form the search space of the *mt06* benchmark with big attraction basins. Because of this, the optima found in most runs belong to one of them. Furthermore, there also is an isolated peak (C) with a little attraction basin. These characteristics make it

Table 3. Characteristic of the niching methods

Method	*Parameters*		*Selection*	*Replacement*
Sharing	$\alpha = 1$	$R = 2, 5$ and 15 *mt06* $R = 2, 5, 15$ and 25 *la01*	Binary tournament	Elitist
Continuously updated sharing	$\alpha = 1$	$R = 2, 5$ and 15 *mt06* $R = 2, 5, 15$ and 25 *la01*	$n^* = 5$ Binary tournament	Elitist
Clearing	$\alpha = 1$	$R = 2, 5$ and 15 *mt06* $R = 2, 5, 15$ and 25 *la01*	$k = 1$ or 5 RWS	Elitist for clearing
Deterministic crowding	Non-parametric	Non-parametric	Own (Section 3)	Own (Section 3)
Crowding		$G = 25, 10, 5$ or 2	Binary tournament	Own
Sequential	$\alpha = 1$ (linear)	$R = 2$	Binary tournament	Elitist

Table 4. Distances between some optima of *mt06* instance

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
2	6																																					
3	8	2																																				
4	2	7	9																																			
5	2	4	6	3																																		
6	5	8	10	4	5																																	
7	6	4	2	8	8	11																																
8	4	5	7	6	5	8	6																															
9	6	6	8	4	6	7	8	2																														
10	4	6	8	5	2	2	10	7	8																													
11	2	8	10	4	4	3	8	6	8	2																												
12	3	4	6	4	4	7	4	2	4	6	4																											
13	4	2	4	5	2	6	6	3	4	4	6	2																										
14	4	2	2	6	8	9	2	4	6	8	6	2	4																									
15	6	4	6	7	4	4	8	5	6	2	4	4	2	6																								
16	6	4	6	8	8	7	4	6	8	6	4	4	6	2	4																							
17	3	6	8	2	2	2	9	6	5	4	5	5	4	7	6	9																						
18	2	7	9	4	3	6	8	2	4	5	4	4	5	6	7	8	4																					
19	8	2	4	9	6	6	6	7	8	4	6	6	4	4	2	2	8	9																				
20	15	14	16	16	14	14	17	12	13	12	13	13	12	15	10	13	16	15	12																			
21	14	18	20	13	14	10	20	15	15	12	12	16	16	18	14	16	12	13	16	4																		
22	8	12	12	11	8	8	16	13	14	6	8	12	10	14	8	12	10	11	10	6	6																	
23	9	14	16	12	10	10	17	12	13	8	9	13	12	15	10	13	12	10	12	4	4	2																
24	14	10	14	16	16	15	12	14	16	14	12	12	14	10	12	8	17	16	10	5	8	8	9															
25	11	14	16	11	10	6	17	14	13	8	9	13	12	15	10	13	8	12	12	8	4	2	4	7														
26	8	14	16	10	10	9	14	12	14	8	8	10	12	12	10	10	11	10	12	7	6	2	3	6	3													
27	14	12	14	14	12	12	16	13	14	10	12	12	10	14	8	12	12	15	10	2	6	4	6	4	6	6												
28	13	10	14	14	12	12	15	10	11	10	11	11	10	13	8	11	14	12	10	2	6	4	2	7	6	5	4											
29	12	10	12	13	10	10	14	11	12	8	10	10	8	12	6	10	12	13	8	4	8	2	4	6	4	4	2	2										
30	12	10	12	14	14	13	10	12	14	12	10	10	12	8	10	6	15	14	8	7	10	6	7	2	7	4	6	5	4									
31	16	10	12	17	14	14	14	15	16	12	14	14	12	12	10	10	16	15	8	5	8	6	8	2	8	8	2	6	4	4								
32	12	16	18	12	12	8	18	13	13	10	10	14	14	16	12	14	10	11	14	6	2	4	2	10	2	4	8	4	6	8	10							
33	15	10	12	16	14	14	13	12	13	12	13	13	12	11	10	9	16	14	8	4	8	6	4	5	8	7	6	2	4	3	4	6						
34	12	14	16	13	10	10	18	15	16	8	10	14	12	16	10	14	12	13	12	4	4	2	4	6	4	4	2	6	4	8	4	6	8					
35	13	17	18	14	12	12	19	14	15	10	11	15	14	17	12	15	14	12	14	2	2	4	2	7	6	5	4	4	6	9	6	4	6	2				
36	17	12	14	18	16	16	15	14	15	14	15	15	14	13	12	11	18	16	10	2	6	8	6	3	10	9	6	4	6	5	2	8	2	6	4			
37	10	12	14	12	12	11	12	10	12	10	8	8	10	10	8	8	13	12	10	5	8	4	5	4	5	2	4	3	2	2	6	6	5	6	7	7		
38	12	14	16	14	14	13	14	12	14	12	10	10	12	12	10	10	15	14	12	5	6	6	7	2	7	4	2	5	6	4	4	8	7	4	5	5	2	
39	18	21	22	18	18	15	21	19	19	16	16	19	19	21	17	19	17	18	19	14	12	11	11	16	10	11	14	12	12	14	16	11	14	14	13	16	12	14

hard to find them. Figure 6 shows the four optima of type C.

Regarding the *la01*, the landscape is similar. It has two big mountainous areas (A, B), and two isolated peaks (C, D).

The landscapes of *mt06* and *la01* benchmarks make them perfect to be studied:

(1) The number of different optima found for the method.

(2) The possibility of directing the search by varying the system parameters to explore the search space by finding multiple optima of different types, or exploit some characteristics by finding optima in the same area.

Table 5. Average distances between different optima types

	A	*B*	*C*
A	5		
B	12	5	
C	19	13	2.7

4.2. *Efficacy of niching methods*

The efficacy can be measured by the percentage of runs required to obtain the known optima. It is also important to measure the average of the finally found solutions and their variance.

To study this efficacy we will need first to set up the following parameters:

(1) Niche radius for the sharing fitness techniques: *mt06* $R = 2$, 5 and 15; *la01*, *mt10* and *mt20* $R = 2$, 5, 15 and 25.

(2) Number of dominant solutions for the clearing: $k = 1$ or 5.

(3) Number of groups for the crowding: $G = 25$, 10, 5, 2.

In Appendix B, we show the analysis of variance (ANOVA) results for each method while in Table 6 the results are summarized. The symbol $\approx$ indicates that the radius does not affect on the optimal value found. In each case, we show the parameter that obtains the best average value.

We have run the sequential technique enough times to find a certain number of different solutions fixed by the user. Consequently, its result is not the average of 40 runs made. Thus, the comparison is not possible.

If the ANOVA shows that parameters do not influence, then this indicates that they produces the same response to the system. In this case, all analyzed methods have a robust behavior regardless the niche radius or number of groups.

In the largest size instances (i.e., *mt*10 and *mt*20), we have used more generations. In Table 7, the average results are shown for 5000 generations and 10 runs. In the sequential method, the process was stopped in accordance with a time criterion without any fitness value being less than 1050 and 1250 for *mt10* and *mt20*, respectively. In this case, we have used a niche radius of 2, $k = 1$ and $G = 2$ to ease the study, since the methods are robust to these parameters.

In this table we can see that the poor average results correspond to deterministic crowding. Nevertheless,

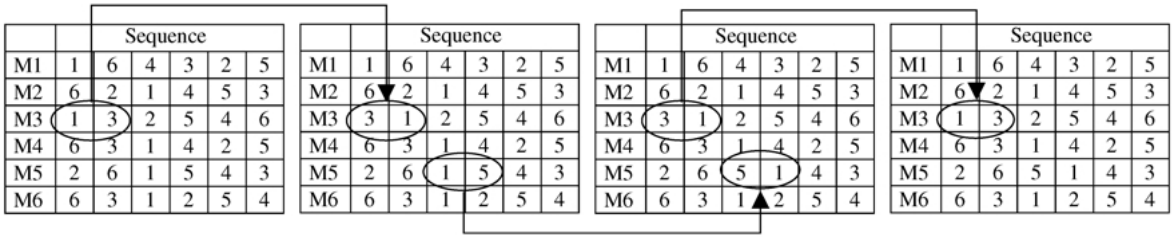

Fig. 6. Optima of type C.

Table 6. Influence of niche radius on the optimal value found

Methods	*(1)*			*(2)*			*(3)*		
Influence	$\approx$	*mt06* $R = 2$ 56.3	*la01* $R = 5$ 702.1	$\approx$	*mt06* $R = 5$ 55.5	*la01* $R = 2$ 684.5	$\approx$	*mt06* $R = 5$ $k = 1$ 55.1	*la01* $R = 15$ $k = 5$ 670.5
Methods	*(4)*			*(5)*			*(6)*		
Influence	$\approx$	*mt06* 55.0	*la01* 666.9	$\approx$	*mt06* $G = 2$ 55.3	*la01* $G = 5$ 671.6		*mt06* —	*la01* —

Table 7. Average optima for 5000 generations

Generations	(1) $R=2$	(2) $R=2$	(3) $R=2$, $k=1$	(4)	(5) $G=2$	(6)
mt10	1042.1	1095.1	1076.7	1114.5	1045.6	—
mt20	1269.3	1340.1	1353.1	1425.5	1310.8	—

Table 8. Final obtained solution and number of needed generations

mt10	Generation	3.10^5	3.10^5	2.10^5	3.10^5	2.10^5	5.10^5	1.10^5
	Solutions	940	951	940	951	951	937	951
mt20	Generation	4.10^5	4.10^5	4.10^5	4.10^5	2.10^5	4.10^5	5.10^5
	Solutions	1178	1178	**1165**	1178	1178	**1165**	1178

Table 9. Comparison of methods for optimal values

	mt06	*la01*					
(1) Sharing	$R=2$	$R=5$	2	3			
(2) Continuously updated sharing	$R=5$	$R=2$		3			
(3) Clearing	$R=5$ $k=1$	$R=15$ $k=5$			$\approx$ (*mt06*)	4 (*la01*)	$\approx$
(4) Deterministic crowding	Non-param.						4
(5) Multiniche crowding	$G=5$	$G=2$	(2)	(3)	(4)		(5)

this is the method that presents the highest diversity after 5000 generations and it is the most efficient one. For further, we investigate to extend the stop criterion (200,000 generations without improvement). Then, the results obtained are shown in Table 8, as well as the number of generations and the best optima when the process stopped.

In the case of the *mt10*, the algorithm was not able to find the optimal value (930) but for the *mt20* it did so twice (1165).

Once the better-adapted niche radius to each method is known (*mt06* and *la01* instances), we can compare the different methods to know which one is the best.

In Appendix B it is shown the Student's tests (*t*-tests) obtained. The results are summarized in Table 9. Each applied *t*-test is indicated by a filled in cell in the table. Each entry indicates the number of the best method found, or the symbol $\approx$ if the different is not significant.

Table 10 shows the percentage of success in each method, ordered from the largest to the smallest.

Thus, we can see that:

- between the two sharing methods, (1) and (2), and the similar one, clearing (3), the best is the clearing and the worst is the classic sharing;
- in the two crowding methods, (4) and (5), the difference is clear.

Therefore, we can conclude that the deterministic crowding method (4) has the higher efficacy, for all sizes of instances.

The study for the sequential method is different because of its characteristics. The process is iterative and its stop criterion is based on finding a number of different optima. In Table 11, we can study the stop criterion (10 or 15 different solutions), the percentage of success and the total needed runs for finding these optima. As we can see for *la01*, the algorithm was not able to find 15 different optima. Also in this case the niche radius used was equal to 2.

Considering these results, we can affirm that the best method, in terms of percentage of success, is the deterministic crowding for all problem sizes.

4.3. *Number of different optima found*

When we talk about multimodal problems, one of the most interesting measures for a successful run is the number of different optima found in the last generation. The results are shown in Tables 12 and 13, in which:

Table 10. Percentage of success

	(4)	*(3)*	*(5)*	*(2)*	*(1)*
*mt*06	Non-param.	$R=5$ $k=5$	$G=2$	$R=15$	$R=2$
	100%	92.5%	85.0%	72.5%	35.0%
	(4)	*(3)*	*(5)*	*(2)*	*(1)*
*la*01	Non-param.	$R=25$ $k=1$	$G=5$	$R=15$	$R=5$
	85%	75.0%	70.0%	17.5%	5.0%

Table 11. Results for the sequential method

Instances	*Number of solutions*	*% Success*	*Number of runs*
*mt*06	10	86.8%	11.52
	15	82.9%	18.1
*la*01	10	27.3%	36.7
	15	—	—

Table 12. Number of optima for successful run (*mt06*)

mt06	*(1)*			*(2)*			*(3)* $k=1$		
R	**2**	**5**	**15**	**2**	**5**	**15**	**2**	**5**	**15**
#	14	11	9	30	24	18	**194**	125	164
Av.	1	1	**1**	1	1	1	**5.4**	3.6	4.7

	(4) *Non-param.*	*(5)* *Groups*			*(6)* *Stop criterion*	
		25	**10**	**5**	**2**	**10**	**15**
#	**383**	**78**	67	53	60	400	**600**
Av.	**9.6**	**2.4**	2.2	1.8	1.8	10	**15**

Table 13. Number of optima for successful run (*la01*)

la01	*(1)*				*(2)*				*(3)* $k=1$			
R	**2**	**5**	**15**	**25**	**2**	**5**	**15**	**25**	**2**	**5**	**15**	**25**
#	1	2	0	0	4	5	7	2	**428**	414	390	389
Av.	1	1	0	0	1	1	1	1	**15.3**	15.3	13.5	13.0

	(4) *Non-param.*	*(5)* *Groups*				*(6)* *Stop criterion*	
		25	**10**	**5**	**2**	**10**	**15**
#	**64**	**95**	74	85	71	**400**	—
Av.	**1.9**	**5.2**	3.1	3.1	2.8	**10**	—

Table 14. Percentage of mixed runs (*mt06*)

mt06	*(3) K = 1*			*(3) K = 5*			*(4)*
R	*2*	*5*	*15*	*2*	*5*	*15*	*Non-param.*
%M	5.9%	5.9%	5.7%	5.7%	15.6%	3%	57.5%
	(5) Groups				*(6) Stop criterion*		
	25	*10*	*5*	*2*	*10*	*15*	
%M	0%	13.3%	20%	14.7%	97.5%	95.0%	

Table 15. Percentage of mixed runs (*la01*)

*la*01	*(3) K = 1*				*(3) K = 5*				*(4)*
R	2	5	15	25	2	5	15	25	*Non-param.*
%M	17.8%	3.7%	10.3%	16.7%	3.7%	0%	14.3%	13.1%	17.6%
	(5) Groups				*(6) Stop criterion*				
	25	*10*	*5*	*2*	*10*				
%M	0%	4.2%	10.7%	8%	95%				

- # shows the total number of different optima in the 40 realized runs, and
- Av. is the average number of different solutions for a successful run.

We only give the results for the *mt06* and *la01* instances, since for larger sizes (*mt10* and *mt20*) the efficacy has been very poor, and only in two cases the deterministic crowding method found an optimum.

As we can observe, the classical and continuous sharing methods (1 and 2), only obtain one optimum during a successful run. That is, only one of the formed niches has been able to evolve to optima whereas the rest have been trapped into local optima. However, the rest of methods achieves a desired effort, that a lot of their niches evolve to optima, offering different multiple optima.

In the case of the *mt06* instance, the method that offers the largest number of different optima for successful runs is the sequential scheme. However, in the *la01* instance the best is the clearing method followed by the sequential one.

In general, we suggest the use of the sequential method to find the largest possible number of different optima, since this method is relatively fast. The clearing method needs five times more computational time than the sequential one.

4.4. *Exploration versus exploitation*

In this section, we study what the distribution of the final optima is. There would be an explorative behavior when the final optima belong to different areas in the search space. On the other hand, there will be exploitation when the optima are from the same area.

In Tables 14 and 15 we are showing the distribution of solutions, where%M indicates the percentage of runs with optima belonging to different areas.

The method that produces more exploration with the larger number of optima of different areas is the sequential method. It finds optima belonging to different areas for the *mt06* and *la01* in a 95% of the runs.

In the same way, the largest exploitation is obtained with the crowding scheme, with a 100% of the solutions belonging to the same area in both instances.

5. Concluding remarks

This study about the use of different niching GA methods to get multiple solutions in job shop scheduling problems has proved that they have different behavior. This behavior depends on the

efficacy, the number of different final optimal and the exploration or exploitation.

In the following we present the most important conclusions obtained in our study:

Efficacy. The highest efficacy for all size problems is under the deterministic crowding. Moreover, it is a non-parametric method, very fast and very simple to be implemented. The elimination in the replacement of the weakest and the most similar individual to the survivor limits the quantity of the existing solutions in each of the zones, benefiting diversity that could produce an optimal convergence.

The second best method is the clearing method. This method eliminates before the selection, the accumulation of individuals in specific zones, which benefits diversity. Moreover, it maintains stable the efficacy when the problem size varies.

We must also point out that the analyzed methods are robust to the variation of the parameters (niche radius or number of groups). This result is very important to simplify the implementation process and the algorithm use.

Number of final optima. The method that allows us to obtain the largest number of final optima is the sequential or the clearing (for larger size problems), with an average number of optima of 15 for successful runs respectively. Nevertheless, the clearing method has a poor performance for small size problems that must be studied in the future.

The sequential method compels every valid run of the algorithm to seek for a new optimum to the problem. On the other hand, the clearing method is compelled to create different niches in the same run that, because of diversity, evolve towards multiple final solutions.

Exploration versus exploitation. The highest exploration is obtained by the sequential method. In our test it produces solutions that belong to different areas in 95% of the runs. This is accomplished by eliminating the already studied zones, and benefiting the exploration among runs which causes a larger dispersion of the solutions.

The highest exploitation is achieved by the crowding method, with a number of groups of 25 (2 individuals each group) with 100% of runs producing solutions of the same type. The low selection pressure (in this case present in the replacement) benefits the high exploration in the evolutionary process.

According to these conclusions, a user can select the appropriate niching method according to his/her needs on these above three points.

Appendix A

Table A.1. Dates of *mt06* instance

Processing time

	M1	*M2*	*M3*	*M4*	*M5*	*M6*
*J*1	3	6	1	7	3	6
*J*2	10	8	5	4	10	10
*J*3	9	1	5	4	7	8
*J*4	5	5	5	3	8	9
*J*5	3	3	9	1	5	4
*J*6	10	3	1	3	4	9

Technological constraints

	Sequence					
*J*1	3	1	2	4	6	5
*J*2	2	3	5	6	1	4
*J*3	3	4	6	1	2	5
*J*4	2	1	3	4	5	6
*J*5	3	2	5	6	1	4
*J*6	2	4	6	1	5	3

Table A.2. Dates of *la01* instance

Processing time

	M1	*M2*	*M3*	*M4*	*M5*
*J*1	53	21	34	55	95
*J*2	21	71	26	52	16
*J*3	12	42	31	39	98
*J*4	55	77	66	77	79
*J*5	83	19	64	34	37
*J*6	92	54	43	62	79
*J*7	93	87	87	69	77
*J*8	60	40	38	24	83
*J*9	44	49	98	17	25
*J*10	96	75	43	79	77

Technological constraints

	Sequence				
*J*1	2	1	5	4	3
*J*2	1	4	5	3	2
*J*3	4	5	2	3	1
*J*4	2	1	5	3	4
*J*5	1	4	3	2	5
*J*6	2	3	5	1	4
*J*7	4	5	2	3	1
*J*8	3	1	2	4	5
*J*9	4	2	5	1	3
*J*10	5	4	3	2	1

Table A.3. Dates of *mt10* instance

Processing time

	M1	*M2*	*M3*	*M4*	*M5*	*M6*	*M7*	*M8*	*M9*	*M10*
*J*1	29	78	9	36	49	11	62	56	44	21
*J*2	43	28	90	69	75	46	46	72	30	11
*J*3	85	91	74	39	33	10	89	12	90	45
*J*4	71	81	95	98	99	43	9	85	52	22
*J*5	62	2	14	26	69	61	53	49	21	72
*J*6	47	2	84	95	65	26	52	54	87	2
*J*7	37	46	13	61	55	21	32	30	89	32
*J*8	86	46	31	79	32	74	88	36	19	48
*J*9	76	69	85	76	26	51	40	89	74	11
*J*10	13	85	61	52	90	47	7	45	64	76

Technological constraints

	Sequence									
*J*1	1	2	3	4	5	6	7	8	9	10
*J*2	1	3	5	10	4	2	7	6	8	9
*J*3	2	1	4	3	9	6	8	7	10	5
*J*4	2	3	1	5	7	9	8	4	10	6
*J*5	3	1	2	6	4	5	9	8	10	7
*J*6	3	2	6	4	9	10	1	7	5	8
*J*7	2	1	4	3	7	6	10	9	8	5
*J*8	3	1	2	6	5	7	9	10	8	4
*J*9	1	2	4	6	3	10	7	8	5	9
*J*10	2	1	3	7	9	10	6	4	5	8

Table A.4. Dates of *mt20* instance

Processing time

	M1	*M2*	*M3*	*M4*	*M5*
*J*1	29	9	49	62	44
*J*2	43	75	46	69	72
*J*3	39	91	90	45	12
*J*4	71	81	85	22	9
*J*5	26	22	14	21	72
*J*6	47	52	84	6	48
*J*7	61	46	32	32	30
*J*8	32	46	31	19	36
*J*9	76	40	85	76	26
*J*10	64	85	61	47	90
*J*11	11	78	21	36	56
*J*12	11	28	90	46	30
*J*13	85	10	74	89	33
*J*14	99	52	95	98	43
*J*15	6	61	49	53	69
*J*16	95	2	25	72	65
*J*17	37	21	13	89	55
*J*18	86	74	48	79	88
*J*19	11	69	51	89	74
*J*20	13	7	76	52	45

Technological constraints

	Sequence				
*J*1	1	2	3	4	5
*J*2	1	2	4	3	5
*J*3	2	1	3	5	4
*J*4	2	1	5	3	4
*J*5	3	2	1	4	5
*J*6	3	2	5	1	4
*J*7	2	1	3	4	5
*J*8	3	2	1	4	5
*J*9	1	4	3	2	5
*J*10	2	3	1	4	5
*J*11	2	4	1	5	3
*J*12	3	1	2	4	5
*J*13	1	3	2	4	5
*J*14	3	1	2	4	5
*J*15	1	2	5	3	4
*J*16	2	1	4	5	3
*J*17	1	3	2	4	5
*J*18	1	2	5	3	4
*J*19	2	3	1	4	5
*J*20	1	2	3	4	5

Appendix B

ANOVA for *mt06*

Table B.1. ANOVA for classic sharing method

Source of variation	*Sum of square*	*Degrees of freedom*	*Mean square*	F_0	$F_{0.05,2,117}$
SSA (niche radius)	6.65	2	3.32	1.82	2.99
SSE	214.15	117	1.83		
SSY	220.80	119			

Table B.2. ANOVA for continously updated sharing

Source of variation	*Sum of square*	*Degrees of freedom*	*Mean square*	F_0	$F_{0.05,2,117}$
SSA (niche radius)	0.32	2	0.16	0.25	2.99
SSE	73.47	117	0.63		
SSY	73.79	119			

Table B.3. ANOVA for clearing $K = 1$

Source of variation	*Sum of square*	*Degrees of freedom*	*Mean square*	F_0	$F_{0.05,2,117}$
SSA (niche radius)	0.02	2	0.01	0.05	2.99
SSE	19.85	117	0.17		
SSY	19.87	119			

Table B.4. ANOVA for crowding of C. and V.

Source of variation	*Sum of square*	*Degrees of freedom*	*Mean square*	F_0	$F_{0.05,3,156}$
SSA (number of groups)	0.65	3	0.22	0.34	2.68
SSE	99.75	156	0.64		
SSY	100.40	159			

ANOVA for *la01*

Table B.5. ANOVA for classic sharing method

Source of variation	*Sum of square*	*Degrees of freedom*	*Mean square*	F_0	$F_{0.05,3,156}$
SSA (niche radius)	2031.15	3	677.05	0.88	2.6
SSE	120266.85	156	770.94		
SSY	122298.00	159			

Table B.6. ANOVA for continuously updated sharing

Source of variation	*Sum of square*	*Degrees of freedom*	*Mean square*	F_0	$F_{0.05,3,156}$
SSA (niche radius)	721.22	3	240.41	1.04	2.6
SSE	36113.78	156	231.50		
SSY	36834.99	159			

Table B.7. ANOVA for clearing $K = 5$

Source of variation	*Sum of square*	*Degrees of freedom*	*Mean square*	F_0	$F_{0.05,3,156}$
SSA (niche radius)	245.23	3	81.74	0.82	2.68
SSE	15475.6	156	99.20		
SSY	15720.8	159			

Table B.8. ANOVA for crowding of C. and V.

Source of variation	*Sum of square*	*Degrees of freedom*	*Mean square*	F_0	$F_{0.05,3,156}$
SSA (number of groups)	457.87	3	152.62	1.01	2.68
SSE	23598.77	156	151.27		
SSY	24056.24	159			

t-test for *la01*

Table B.9. *t*-test (1) vs (2)

Average	*Variance*	F_0	$F_{0.05,39,39}$	ν^*	t_1	$t_{0.025,57}$
56.32	1.71	4.28	1.69	57	4.03	2.005
55.40	0.40					

Table B.10. *t*-test (1) vs (3)

Average	*Variance*	F_0	$F_{0.05,39,39}$	ν^*	t_1	$t_{0.025,46}$
56.32	1.71	10.47	1.69	46	5.54	2.015
55.12	0.16					

Table B.11. *t*-test (2) vs (3)

Average	*Variance*	F_0	$F_{0.05,39,39}$	ν^*	t_1	$t_{0.025,67}$
55.40	0.40	2.45	1.69	67	2.32	1.999
55.12	0.16					

Table B.12. *t*-test (3) vs (4)

Average	*Variance*	F_0	$F_{0.05,39,39}$	ν^*	t_1	$t_{0.025,39}$
55.12	0.16	—	1.69	39	1.96	2.021
55.00	0					

Table B.13. *t*-test (3) vs (5)

Average	*Variance*	F_0	$F_{0.05,39,39}$	ν^*	t_1	$t_{0.025,57}$
55.12	0.16	4.07	1.69	57	1.04	2.005
55.27	0.67					

Table B.14. *t*-test (4) vs (5)

Average	*Variance*	F_0	$F_{0.05,39,39}$	ν^*	t_1	$t_{0.025,39}$
55.00	0	—	1.69	39	2.13	2.021
55.27	0.67					

t-test for *la01*

Table B.15. *t*-test (1) vs (2)

Average	*Variance*	F_0	$F_{0.05,39,39}$	t_0	$t_{0.025,78}$
702.15	367.72	1.56	1.69	4.41	1.993
685.00	236.10				

Table B.16. *t*-test S(1) vs (3)

Average	*Variance*	F_0	$F_{0.05,39,39}$	ν^*	t_1	$t_{0.025,53}$
702.15	367.72	5.4	1.69	53	9.58	2.010
670.55	68.05					

Table B.17. *t*-test (2) vs (3)

Average	*Variance*	F_0	$F_{0.05,39,39}$	ν^*	t_1	$t_{0.025,60}$
685.00	236.10	3.47	1.69	60	5.24	1.996
670.55	68.05					

Table B.18. *t*-ring test (3) vs (4)

Average	*Variance*	F_0	$F_{0.05,39,39}$	ν^*	t_1	$t_{0.025,46}$
670.55	68.05	10.17	1.69	46	2.61	2.016
666.97	6.69					

Table B.19. *t*-test (3) vs (5)

Average	*Variance*	F_0	$F_{0.05,39,39}$	t_0	$t_{0.025,78}$
670.55	68.05	1.68	1.69	0.51	1.993
671.65	114.64				

Table B.20. *t*-test (4) vs (5)

Average	*Variance*	F_0	$F_{0.05,39,39}$	ν^*	t_1	$t_{0.025,43}$
666.97	6.69	17.13	1.69	43	2.68	2.021
671.65	114.64					

References

Adams, J., Balas, E. and Zawack, D. (1988) The shifting bottleneck procedure for job shop scheduling. *Management Science*, **34**, 391–401.

Balas, E. (1969) Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. *Operations Research*, **17**, 941–957.

Baker, J. R. and McMahon, G. B. (1985) Scheduling the general job shop. *Management Science*, **31**, 594–598.

Beasley, D., Bull, D. R. and Martin, R. R. (1993) A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, **1**, 101–125.

Brucker, P. (1997) *Scheduling Algorithms*, 2nd Edn, Springer-Verlag, Berlin, Germany.

Bruns, R. (1993) Direct chromosome representation and advanced genetic operators for production scheduling. *Proceedings of the Fifth International Conference on Genetic Algorithms*, Stephanie Forrest (ed.), Kaufmann, San Mateo, California, USA, pp. 352–359.

Carlier, J. and Pinson, E. (1989) An algorithm for solving the job shop problem. *Management Science*, **35**, 164–176.

Cedeño, W., Vemuri, V. R. and Slezak, T. (1995) Multiniche crowding in genetic algorithms and its application to the assembly of DNA restriction-fragments. *Evolutionary Computation*, **2**, 321–345.

Conway, R. W., Maxwell, W. L. and Miller, L. W. (1967) *Theory of Scheduling*, Addison-Wesley, Massachusetts, USA.

Dell'Amico, M. and Trubian, M. (1993) Applying Tabu search to the job shop scheduling problem. *Annals of Operations Research*, **41**, 231–252.

Della, F., Tadeis, R. and Volta, G. (1995) A genetic algorithm for the job shop problem. *Computers and Operations Research*, **22**, 15–24.

Fang, H., Ross, P. and Corne, D. (1993) A promising genetic algorithm aproach to job shop scheduling, rescheduling, and open shop scheduling problems, in *Proceedings of the Fifth International Conference on Genetic Algorithms*, Stephanie Forrest (ed.), Kaufmann, San Mateo, California, USA, pp. 375–382.

Fogel, D. B. (ed.) (1998) *Evolutionary Computation, The Fossil Record (Selected readings on the history of evolutionary computation)*, IEEE Press, NY, USA.

French, S. (1982) *Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop*, Ellis Horwood, Chichester, USA.

Garey, M. and Johnson, D. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, NY, USA.

Glover, F. and Laguna, M. (1997) *Tabu Search*, Kluwer Academic, Boston, USA.

Goldberg, D. E. and Richardson, J. (1987) Genetic algorithms with sharing for multimodal function optimization, in *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 41–49.

Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Massachusetts, USA.

Greenberg, H. (1968) A branch and bound solution to the general scheduling problem. *Operations Research*, **16**, 353–361.

Harik, G. (1995) Finding multiple solutions using restricted tournament selection, in *Proceedings of the Sixth International Conference on Genetic Algorithms*, Eschelman, L. (ed.), Kaufmann, USA, pp. 24–31.

Kirkpatrick, S., Gelatt, C. D. Jr. and Vecchi, M. P. (1983) Optimization by simulated annealing. *Science*, **220**, 671–680.

Kobayashi, S., Ono, I. and Yamamura, M. (1995) An efficient genetic algorithm for job shop scheduling problems, in *Proceedings of the Sixth International Conference on Genetic Algorithms*, Eschelman, L. (ed.), Kaufmann, San Francisco, California, USA, pp. 506–511.

Mahfoud, S. W. (1992) Crowding and preselection revisited. *Parallel Problem Solving from Nature II*, Männer, R. and Manderick, B. (eds), Elsevier, pp. 27–36.

Mattfeld, D. C. (1995) *Evolutionary Search and the Job Shop, Investigations on Genetic Algorithms for Production Scheduling*, Springer, Berlin, Germany.

McMahon, G. and Florian, M. (1975) On scheduling with ready times and due date to minimize maximum lateness. *Operations Research*, **23**, 475–482.

Michalewicz, Z. (1995) *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, Germany.

Nakano, R. and Yamada, T. (1991) Conventional genetic algorithms for job shop problems, in *Proceedings of the Fourth International Conference on Genetic Algorithms*, Belew, R. and Booker, L. B. (co-ed.), Kaufmann, San Mateo, California, USA, pp. 474–479.

Nowicki, E. and Smutnicki, C. (1996) A fast tabu search algorithm for the job shop problem. *Management Science*, **42**, 797–813.

Oei, C. K., Goldberg, D. E. and Chang, S. J. (1991) Tournament selection, niching and the preservation of diversity. *IlliGAL Report* No. 91011, University of Illinois, USA.

Panwalkar, S. S. and Iskander, W. (1977) A survey of scheduling rules. *Operations Research*, **25**, 45–61.

Pétrowski, A. (1996) Clearing procedure as a niching method for genetic algorithms, in *Proc. 1996 IEEE Int. Conf. Evolutionary Computation*, Nagoya, Japan, pp. 798–803.

Ramalhinho, H., Marti, O. and Stützle, T. (2000) Iterated local search. *Economic Working Papers Series*, Universitat Pompeu Fabra (To appear in the Metaheuritics book, Glover, F. and Kochenberger, G. eds.).

Sareni, B. and Krahenbuhl, L. (1998) Fitness sharing and niching methods revised. *IEEE Transations on Evolutionary Computation*, **2**, 97–106.

Stützle, T. (1998) *Local Search Algorithms for Combinatorial Problems—Analysis, Improvements, and New Applications*, PhD. thesis, Computer Science Departmentm Darmstadt University of Technology, Darmstadt, Germany.

Van Laarhoven, P. J. M., Aarts, E. H. L. and Lenstra, J. K. (1992) Job shop scheduling by simulated annealing. *Operations Research*, **40**, 112–129.

Wang, L. and Zheng, D.-Z. (2001) An effective hybrid optimization strategy for job-shop scheduling problems. *Computers & Operations Research*, **28**, 585–596.

Yamada, T. and Nakano, R. (1992) A genetic algorithm applicable to large-scale job shop problems. *Parallel Problem Solving from Nature II*, Männer, R. and Manderick, B. (eds.), Elsevier Science, Amsterdam, The Netherlands, pp. 281–290.

Yang, S. and Wang, D. (2000) Constraint satisfaction adaptive neural network and heuristics combined approach for generalized job-shop scheduling. *IEEE Trans. on Neural Networks* **11**, pp. 474–486.