J Intell Manuf (2012) 23:341–356
DOI 10.1007/s10845-010-0385-4

REVIEW ARTICLE

Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling

E. Pérez · M. Posada · F. Herrera

Received: 1 April 2009 / Accepted: 6 February 2010 / Published online: 7 March 2010

Abstract In this paper the performance of the most recent multi-modal genetic algorithms (*MMGAs*) on the Job Shop Scheduling Problem (*JSSP*) is compared in term of *efficacy, multi-solution based efficacy* (the algorithm's capability to find multiple optima), and *diversity* in the final set of solutions. The capability of Genetic Algorithms (*GAs*) to work on a set of solutions allows us to reach different optima in only one run. Nevertheless, simple *GAs* are not able to maintain different solutions in the last iteration, therefore reaching only one local or global optimum. Research based on the preservation of the diversity through *MMGAs* has provided very promising results. These techniques, known as niching methods or *MMGAs*, allow not only to obtain different multiple global optima, but also to preserve useful diversity against convergence to only one solution (the usual behaviour of classical *GAs*). In previous works, a significant difference in the performance among methods was found, as well as the importance of a suitable parametrization. In this work classic methods are compared to the most recent *MMGAs*, grouped in three classes (sharing, clearing and species competition), for *JSSP*. Our experimental study found that those new *MMGAs* which have a certain type of *replacement process* perform much better (in terms of highest *efficacy* and *multi-solution based efficacy*) than classical *MMGAs* which do not have this type of process.

Keywords Genetic algorithms · Multimodal problems · Job shop scheduling problem · Niching methods

E. Pérez (✉) · M. Posada
Department Organización de Empresas y C.I.M,
University of Valladolid, 47011 Valladolid, Spain
e-mail: elena@eis.uva.es
URL: www.eis.uva.es/elena

M. Posada
e-mail: posada@eis.uva.es
URL: www.insisoc.org/posada

F. Herrera
Department of Computer Science and Artificial Intelligence,
University of Granada, 18071 Granada, Spain
e-mail: herrera@decsai.ugr.es
URL: www.decsai.ugr.es/~herrera/

Introduction

There are three types of optimization problems. The first one tries to solve problems with only one optimization function and only one global optimum. The objective is to find this optimum. The second type tries to optimize problems with several optimization functions (called multi-objective problem). The solution of the problem is the Pareto-set. The third type focuses on problems with only one optimization function but with several global and local optima, an example of a function with five global optima is shown in Fig. 1a and another function with one optimum and four local optima in Fig. 1b (Beasley et al. 1993).

These problems are called multi-modal, and the objective of optimization is to find the set of different global optima. In the last few years, works not only have been focused on solving the problem with the highest efficacy (getting the best solution of the problem), but also on obtaining the highest number of different optima of the problem (Harik 1995). For example, in Fig. 1a, finding one of the five optima is efficacy because the problem has been solved, but if the five optima could be obtained, the search space would be completely known. This is *multi-solution based efficacy*.

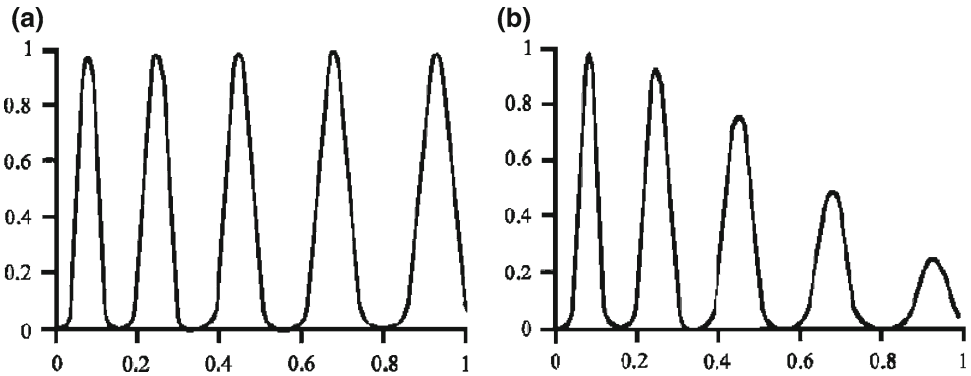


Fig. 1 Examples of different types of multi-modal problems (Beasley et al. 1993)

There are several strategies to solve multi-modal problems. One of them is to restart the optimization process many times as needed to obtain the number of desired optima. The main drawback is that the process could obtain several copies of the same optimum. This can be solved by implementing mechanisms to avoid the exploration of optima obtained in previous searches, a sort of memory from previous searches. There are few studies on finding multiple different solutions (Hoss and Stützle 2004). Another possibility is to use algorithms that are able to obtain multiple different solutions in each step of the search, like Genetic Algorithms (GAs) (Goldberg 1989). But if the problem has high difficulty like Job Shop Scheduling Problem (JSSP) simple *GAs* have premature convergence and obtain either poor solutions or only one optimum. To overcome this limitation, different types of Multi-Modal Genetic Algorithms (MMGAs) were developed in the last years.

The first MMGA was developed in 1987 but with poor results in high difficulty problems (Sareni and Krahenbuhl 1998). In the following 12 years, few attempts were done, but in the last 10 years the number of different approaches has dramatically increased. Pérez et al. (2003) compared the main MMGAs approaches applied to JSSP, proposals until 1999. They showed the ability of niching *GAs* to find multiple optima by analyzing the classical techniques: sharing, crowding and clearing. The analyzed methods were fitness sharing (Goldberg and Richardson 1987) and continuously updated sharing (Oei et al. 1991), deterministic crowding (Mahfoud 1992) and multi-niche crowding method (Cedeño and Vemuri 1999), and lastly clearing method (Pétrowski 1996, 1997). The conclusions of this work were that MMGAs are able to find multiple global optima in multi-modal problems, but there are a lot of differences in the performance between methods. Thus, the choice of the more appropriated MMGA is crucial. The fitness sharing, continuously updated sharing and multi-niche crowding have a very poor performance and can not find multiple global optima. The deterministic crowding only has good performance with easy problems (of little size). The clearing presents the best result to solve the JSSP for any problem size.

As the choice of the most appropriate MMGA is crucial, in this paper we extend this analysis to the last promising MMGAs published in the last years, comparing them with the results obtained in the study from 2003 (Pérez et al. 2003).

The perfect testing-ground for the MMGAs is the JSSP because of its high difficulty. First, it belongs to the *NP-hard* problems. That is, there are not algorithms that find an optimal solution in polynomial time (Garey and Johnson 1979). Secondly, the search space of the JSSP can be typified by hills, valleys, and mountainous areas, that is, the JSSP has multiple different optima. We call the algorithm's capability to find these multiple optima a *multi-solution based efficacy*. To compare the different MMGAs, we have selected eight instances (four which are easy to solve, and the other four which are difficult) based on our previous experience and knowledge about the distributions of optima.

We compare the best classical method [Classic Clearing Method (Pétrowski 1996, 1997)] according to Pérez et al. (2003), with the more recent promising proposals. So we extend previous works with the study of the following MMGAs:

- Fitness sharing:
  - Adaptive niche hierarchy genetic algorithm (Dunwey et al. 2002).
  - Niche identification technique (Lin and Wu 2002).
- Clearing:
  - Restricted competition selection method (Lee et al. 1999).
  - Restricted competition selection with pattern search method (Kim et al. 2002).

- Species competition:
  - Species conserving genetic algorithm, (Li et al. 2002).
  - Quick hierarchical fair competition, (Hu and Goodman 2004).

The analysis of the results is based on three different points of view: efficacy, *multi-solution based efficacy*, and diversity in the final set of solutions. The efficacy is to find the optimum of the problem. We define*multi-solution based efficacy* as the algorithm's capability to find multiple optima. Diversity can be understood as exploration or exploitation. We consider search space exploration when optima that belong to different areas are reached, and exploitation is to find very similar optimal solutions.

For each point of view, the sequence of the analysis is developed according to the following steps:

Step 1. Find the best suitable parametrization of each method (populations size, niche radius, etc).
Step 2. Determine the best technique for each point of view.

The rest of the paper is organized as follows. In the next section we present the scheduling theory, including the problem definition and the main techniques that have been used to solve it. In section "Genetic algorithms", we introduce GAs and their use to solve the JSSP. In section "Niching *GAs*", we describe briefly different niching methods analyzed in this paper. In section "Experimental study", we establish an experimental study to compare different niching GAs, considering efficacy, *multi-solution based efficacy*, and exploration or exploitation of search space. Finally, in section "Conclusions" we point out some concluding remarks. In the appendix a short explanation of analyzed niching methods is shown.

Scheduling theory

The job shop is a complex case where each one of the n job has its own movement within m machines, and each of the m machine has its own sequence of n jobs. For example, In Table 1 the dates of a job shop problem with three machines and three jobs are shown, *(a)* provides details about the processing time of operations, and *(b)* the sequence of jobs in the shop. The job shop has $(n!)^m$ possible solutions (being n jobs and m operations), i. e. the above example has 216 solutions with multiple global and local optima. One of them is shown in Fig. 2 and Table 2. The cost function can define termination times (makespan or C_{max}), delay times (T_{max}) or total flow times (F_{max}), among others (Brucker 1997).

Table 1 Dates of a job shop problem

	(a)			(b)		
	M1	M2	M3	Machines		
J1	3	4	5	2	1	3
J2	5	3	2	3	2	1
J3	1	2	3	1	3	2

At the beginning, attempts were targeted towards formulation of the easiest problems (French 1982) and their resolutions by mathematical methods (Greenberg 1968; Carlier and Pinson 1989). However, other approaches have emerged due to the limitation in the industry such as heuristic approaches (Panwalkar and Iskander 1977; Adams et al. 1988; Wenqi and Aihua 2004) or meta-heuristic methods like:

- Tabu search (Glover and Laguna 1997; Nowicki and Smutnicki 1996; Watson et al. 2003; Geyik and Cedimoglu 2004; Nowicki and Smutnicki 2005; Watson et al. 2006).
- Simulated annealing (Kirkpatrick et al. 1983, Van Laarhoven et al. 1992, Aydin and Fogarty 2002, 2004a; El-Bouri et al. 2007).
- GAs (Mattfeld 1995; Vazquez and Whitley 2000, Amirthagadeswaran and Arunachalam 2007).
- Neural networks (Yang and Wang 2000; Weckman et al. 2008).
- Fuzzy systems (Canbolat and Gundogar 2004).
- Iterated local search (Ramalhinho et al. 2003).
- Hybrid optimization strategies (Wang and Zheng 2001; Hasan et al. 2007).
- Multi-agent approach (Usher 2003; Aydin and Fogarty 2004b) among others.

In all these studies, the number of optima found is never detailed, because efficacy has been the only traditional objective pursued. There is a good review of the different algorithms to solve the JSSP in Jain and Meeran (1999).

We have selected eight instances to compare analyzed methods: four of which are easy to find multiple optima (*mt06, la01, la02* and*la05*), two of which are easy to find the optimum but difficult to find multiple optima (*la03* and *la04*). Lastly, *mt10* and *mt20* are difficult to find the optimum and very difficult to find multiple optima. From the *multi-solution based efficacy* point of view, the difficulty of the instance to find multiple optima is inversely proportional to its number of different global optima known. In Table 3 the number of different global optima known to date for each instances is shown. More information can be found at www.insisoc.org/elena/JSSP/optima.htm.

The easiest instance is *mt06* due to its small size, only 6 machines and 6 jobs (6×6). We have selected this example based on our previous experience and knowledge about the

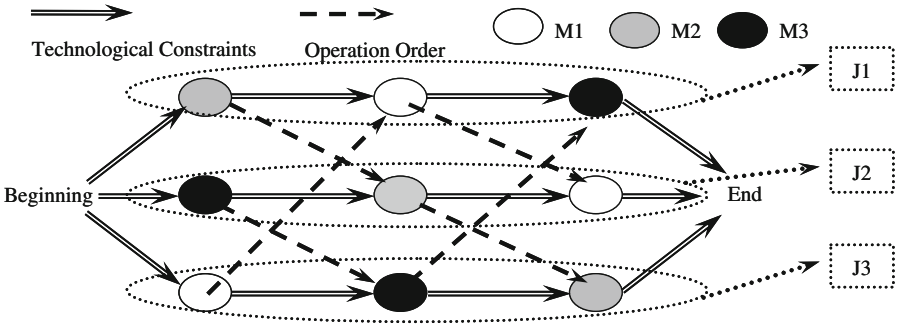


Fig. 2 One possible solution to the example of Table 1

Table 2 Details of the sequence

	Jobs		
M1	3	1	2
M2	1	2	3
M3	2	3	1

Table 3 Number of different global optima known to date

mt06	la01	la02	la03	la04	la05	mt10	mt20
42	26813	5321	706	143	48471	4	1

distributions of optima in the solutions space. For *mt06*, we have found three different types of optima: two big mountainous areas (*A* and *B*) and one isolated peak (*C*). The total number of global optima for this example found to date is 42. For a more detailed revision about this distribution see Pérez et al. (2003). The *la01, la02, la03, la04* and *la05* instances have the same size, 5 machines and 10 jobs. Both *mt10* (10 machines and 10 jobs) and *mt20* (5 machines and 20 jobs) instances are very difficult. More information about these instances and others is available at: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html.

Genetic algorithms

Gas are global search algorithms with a general purpose that use principles inspired by natural population genetics. The first studies can be found in the 1960s, in Fogel (1998) there is a good collection of the first proposals. But it was not until the 1970s when researchers began to use them as a useful optimization and search tool. In GAs, 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. New individuals are created using genetic operators by crossover and mutation, (Goldberg 1989, 2002; Michalewicz 1995; Eiben and Smith 2007; Sivanandam and Deepa 2007). The main parts of *GAs* are the following (see Fig. 3):

1. *Coding*. It allows us to handle the potential solutions in a simple manner.
2. *Evaluation.* It is the value of objective function for each solution. In this case C_{max}.
3. *Parent selection and genetic operators*. They allow the exploration and the exploitation in search areas. Parent selection gives us the mechanism for distinguishing and selecting the best individuals for its reproduction. Classical genetic operators are: crossover which exchanges the genetic material of the selected parent, and mutation which incorporates diversity into the search process.
4. *Replacement process*. The offspring population is included in the population for the next generation (iteration). Different proposals can be found in the literature, for example elitism (Goldberg 1989).

GAs have been widely used to solve scheduling problems. In Hart et al. (2005) there is a recent overview of the subject. In the following two subsections we briefly introduce the basic components which allow us to use GAs for the JSSP, coding approaches and genetic operators.

Coding for the *JSSP*

The most important decision we have to make when facing a problem is how to code solutions. There are different possibilities to code the solution of a *JSSP*: the direct (Bruns 1993; Kobayashi et al. 1995), the binary (Nakano and Yamada 1991), the circular (Fang et al. 1993), and the permutational with repetition (Mattfeld 1995). We have selected the latter because genetic operators always obtain valid children.

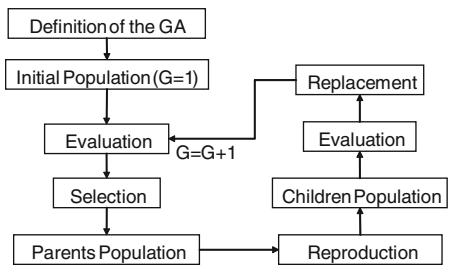


Fig. 3 *GA* procedure (*G* is the number of generations)

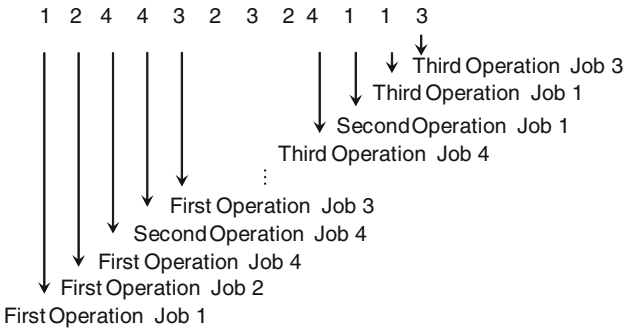


Fig. 4 Permutational with repetition coding applied to the job shop scheduling problem

In this coding process, if there are n jobs, there will be n amount of numbers, and these will be repeated during the coding process depending on how many operations are required. For instance, if we have 4 jobs and 3 operations are required for each of them, the string of coding has a size of 12 which is the result of 4×3 (see Fig. 4, it is coded from left to right).

Genetic operators for the job shop problem

Genetic operators must be adapted to the coding used. There are many crossover and mutation operators developed for the JSSP. In Gento and Pérez (2002) there is an extensive study on the most familiar operators. They concluded that adapted Order Crossover (OX) operator, initially developed for the travelling salesman problem (Davis 1989), and the Order Based Mutation (OBM) operator (Mattfeld 1995), provided a higher efficacy to the GAs for the JSSP. Readers who are not familiar with the subject can find further information at www.eis.uva.es/elena/JSSP. In Fig. 5 we show an example of operator OX for the adaptation developed for the permutation with repetition coding, and in Fig. 6 we show an example of OBM.

Definition of distance

Distance is the main concept GAs are based on. It is the measurement of proximity between solutions, *d(i,j)*. The concept

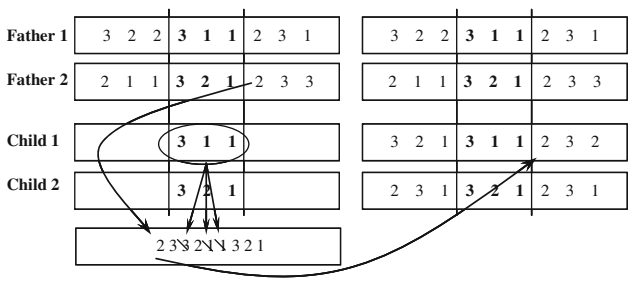


Fig. 5 *OX* crossover operator adapted to permutation coding with repetition

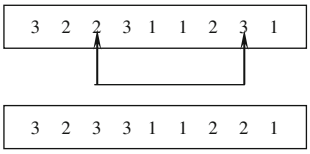


Fig. 6 *OBM* mutation operator

of closeness or remoteness (similarity) requires the calculation of the distance between solutions, which is problem-dependent. In our JSSP, we define the distance as the number of operations situated in different places for each machine. For example, the distance of theses two solutions (**1 3 2**) (**2** 3 **1**)(3 2 1) and (**3 2 1**)(**1** 3 **2**)(3 2 1) is five.

Niching *GAs*

In nature, the environment is divided into different zones due to some common characteristics, such as humid regions, high temperatures, great depths, etc. Each of these clearly-defined zones is known as a niche. Therefore, those individuals that are specially adapted to the conditions of the environment will survive. On the contrary, survival for other types of organisms will be clearly impossible because they are unable to adapt to the environment. Hence, the fight to survive among different species is implicitly limited.

Following these principles of nature, niching genetic algorithms try to split up populations into different niches. So solutions that occupy different areas in the search space, regardless of their quality, will be able to survive from generation to generation. The goal is to keep the necessary diversity to achieve a large search in different promising areas and reach different optima of the multi-modal problems.

In Table 4 we summarize niching methods analyzed in this paper. We have grouped them in fitness sharing, clearing and species competition. Although in the following subsection, www.eis.uva.es/elena/MMGAs, and appendices, we provide the main ideas of these methods, we recommend reading the original papers.

Fitness sharing methods

Methods included in this subsection are based on the distribution of solutions into niches by penalizing the quality of individuals according to their proximity to other solutions. Thus the quality of solutions that belong to densely populated areas is more penalized than those solutions that belong to less populated zones. The methods we have included in this study, as following:

- *Adaptive niche hierarchical genetic algorithm* (*ANHGA*) (Dunwey et al. 2002). The classical fitness sharing

Table 4 Summary of niching methods

Fitness sharing	Clearing	Species competition
Adaptive niche hierarchical genetic algorithm (*ANHGA*)	Clearing (*CM*)	Species conserving genetic algorithm (*SCGA*)
Niche identification techniques (*NIT*)	Restricted competition selection (*RCS*)	Quick hierarchical fair competition (*QHFC*)
	Restricted competition selection with pattern search (*RCS_PSM*)	

method is applied but with a modified sharing function. In addition, it uses an adaptive mutation and a substitution system based on quality as well as proximity to other solutions, by gathering parent and children populations (see Appendix A.1).

- *Niche identification techniques with fitness sharing* (*NIT*) (Lin and Wu 2002). First, the solutions of the population have to be divided into niches, using an identification technique of cluster kind (the process is focused on establishing the radius of each identified niche), and once niches are created the fitness sharing method is applied (see Appendix A.2).

Clearing based genetic algorithms

Methods included in this subsection also modify the quality of the solutions by the distance. The methods we have included in this study, as following:

- *Classical Clearing Method* (*CM*) (Pétrowski 1996, 1997). A certain number of solutions survive by niche, and the rest which do not survive, result in a quality of zero. They have a sense of elitism where not only one good solution is saved, but also the best k solutions of each niche (see Appendix A.3). It has been considered because it was the best performance in Pérez et al. (2003).
- *Restricted Competition Selection Method* (*RCS*) (Lee et al. 1999). The clearing philosophy (described in the *CM*) is applied in the stage of substitution instead of in the selection. Moreover, all the individuals of the initial population are obligated to enter in the matting pool, randomly grouping them in pairs for the reproduction process. This removes the pressure on the selection (see Appendix A.4)
- *Restricted Competition Selection with Pattern Search Method* (*RCS_PSM*) (Kim et al. 2002). It is a modification of the*RCS*. The best individuals of each niche of the initial population are improved by local search. In *JSSP* we have used Giffer and Thompson (1960)'s local search method. This method has been applied as an algorithm of improvement of performance of the *GAs* (Fang 1994; Kobayashi et al. 1995; Mattfeld 1995; Lin et al. 1997) (see Appendix A.5)

Species competition based genetic algorithms

Methods included in this subsection create different species inside each population, and make individuals of each species evolve and fight to survive in an independent way different from individuals in the rest of the existing species. The methods we have included in this study, as following:

- *Species Conserving Genetic Algorithm* (*SCGA*) (Li et al. 2002). This is based on dividing population into species and locating the best solution from each of them based of its quality (seed), and distributing the rest based on their proximity. Later, substitution depends on this distribution (a see Appendix A.6).
- *Quick Hierarchical Fair Competition* (*QHFC*) (Hu and Goodman 2004). The population is divided into as many subpopulations as the algorithm designer wishes and the solutions are distributed in them according to quality thresholds. Pseudo parallel genetic algorithms are performed by permitting the import–export of solutions between the different subpopulations (see Appendix A.7).

Experimental study

In this section we establish the bases of the experimental study for comparison among niching GAs, in terms of efficacy, the *multi-solution based efficacy* and diversity. First, we introduce algorithm parameters used to analyze the algorithms together with the measures. Finally, we present the results and their analysis.

Parameters and measures

General parameters of the *GA* are:

- Population size: $N = 100$ (except in *QHFC* where different N are checked)

- The genetic operators' probabilities are 0.8 for the crossover and 0.1 for the mutation per chromosome. (Gento and Pérez 2002).
- Stop conditions: Number of evaluations 100,000 for instances *mt06, la01* and*la05*), 300,000 for *la02, la03* and *la04* and 500,000 for *mt10* and *mt20*.
- Runs: 30 repetitions per combination of parameters to do the statistical study.

Parameters are different of those used in Pérez et al. (2003). The modifications on population size and the crossover and mutation probabilities are caused by posterior studies in which the exploration of the GAs was improved with these new values. On the other hand, most of the recent works in metaheuristics use the number of evaluations as stopping criterion instead of the number of generations. The evaluation process uses the most computational time in the optimization process, and the comparison between different optimization techniques is the easiest when the process is not based on generation as GAs. The computational time is not a good measure because different computers were used to run experiments.

Each niching GA needs to adapt to the JSSP by parameters such as niche radius or selection method. In Table 5 we summarize the nomenclature used in the experiment to simplify the linguistic treatment and the parameter values of each of the methods. Thus, values of the niche radius (σ_{share}) are adapted to the JSSP, and the rest of the parameters, both original (proposed by authors in their studies) and new values, have been selected and analyzed. For selection we have used the following: RWS (roulette-wheel selection), TS (tournament selection), SRS (stochastic remainder selection) and SUS (stochastic universal selection).

Results and analysis

Each MMGA has been run 30 times for each possible combination of parameter values. For instance, the ANHGA has 225 possible combinations (5 for niche radius, 3 for selection methods and 5 for replacement methods).

We separately analyze the efficacy, the *multi-solution based efficacy* and the connection between parameters values and diversity (exploitation vs. exploration). The following measures are used in the analysis:

1. V_m. We calculated the average of the best obtained solutions taken from each repetition (30). So we analyze the efficacy of the methods.
2. N_{mo}. Average of the number of global optima which is reached in each successful repetition (only with global optima). So we analyze the *multi-solution based efficacy* (the algorithm's capability to find multiple optima). The repetitions without optima (no global optima) do not count when calculating the average of the number of optima. For example, if 24 repetitions have global optima and the rest (6) have poor solutions (no global optima), the average of the number of optima is calculated with

Table 5 Parameters of the different analyzed methods

Method	Parameters	Values
Adaptive Niche Hierarchy Genetic Algorithm ($ANHGA$)	Niche radius (σ_{share})	$\sigma_{share} = \{0, 10, 20, 30, 40\}$
	Selection method (SM)	$SM : RWS$ and TS
	Replacement (B)	$B = \{0, 0.25, 0.5, 0.75, 1\}$
Niche Identification Technique (NIT)	Selection method (SM)	SM : RWS and TS
	Quality jump (B*)	$B^* = \{0, 0.1, 0.2\}$
	Niche size (N*)	$N^* = \{0, 0.1, 0.2\}$
	Study of Interference Niches (I)	$I = \{0$(no), 1(yes)$\}$
Clearing Method (CM)	Niche radius (σ_{share})	$\sigma_{share} = \{0, 10, 20, 30, 40\}$
	Selection method (SM)	$SM : SRS$ and SUS
	Number of winners (k)	$k = \{1, 5, 10\}$
Restricted Competition Selection (RCS)	Niche radius (σ_{share})	$\sigma_{share} = \{0, 10, 20, 30, 40\}$
	Size of Elite Set (M)	$M = \{10, 50, 100\}$
RCS with Pattern Search (RCS_PSM)	Niche radius (σ_{share})	$\sigma_{share} = \{0, 10, 20, 30, 40\}$
	Size of Elite Set (M)	$M = \{10, 50, 100\}$
Species Conserving Genetic Algorithm ($SCGA$)	Niche radius (σ_{share})	$\sigma_{share} = \{0, 20, 40, 60, 80\}$
	Selection method (SM)	$SM : RWS; SRS$; and TS
Quick Hierarchical Fair Competition ($QHFC$)	Size population (N)	$N = \{500, 1000, 1500, 2000\}$
	Number of species (L)	$L = \{3, 5, 8\}$
	Factor size subpopulations (γ)	$\gamma = \{0.6, 0.8, 1.0\}$

the number of optima reached in these 24 runs, and this is repeated for each combination of parameters.

3. D_m. Average distance between optima. With this measure we analyze diversity of the found optima.

Efficacy

In this section we analyze which methods allow us to find the best solution and the suitable parameters to reach this objective. The average of the best obtained solution of 30 repetitions (V_m) for each combination of parameters values has been calculated, and the best V_m and their *SD* (standard deviation) for each method and instances are summarized in Table 6. All analyzed methods found the optima of *la05* and *mt06* problems (593 and 55, respectively) so they have not been included in Table 6. The last row shows the optimum for each instance (*Opt*). In bold type we report the minimum obtained value for each instance (*la01, la02, la03*, and *la04*). The best method for all instances is ANHGA.

In Table 7 we show, for each instance, the methods whose efficacy is not statistically different (by *t*-test). ANHGA is the best method in *mt10* and *mt20*. But in smaller size instances (*la01, la02, la03* and *la04*), ANGHA's efficacy is not statistically different from SCGA or CM.

Although ANHGA is the best method, a statistical point of view shows that the efficacy of the ANHGA is not statistically different from SCGA or CM for all instances.

In Table 8 we show the most appropriate set of values of parameters to obtain the best performance of each method. For this, we carry out a statistical study (applying the *ANOVA* and *t*-test). As we can see, there is not only one combination of values for the different parameters, but rather a set of combinations, all of them equally suitable. Therefore, stressing that this set is the same for all instances, i. e. it is a reliable and robust set to different size problems. The main conclusion that we would like to point out from the results shown in Table 8 are:

- For ANHGA, surprisingly the replacement parameter was $1 (B = 1)$. The quality of the parents and children depends on its fitness (with a weight of B) and on its proximity between the rest of solutions (with a weight of $B - 1$). As $B = 1$, the replacement use only the fitness.
- For NIT the type of interference niches is $\boldsymbol{I} = \{\mathbf{0}\}$. Therefore the study of interference is not necessary because the solution space of the *JSSP* is not continuous and this process does not offer a good performance.
- For CM the niche radius is $\boldsymbol{\sigma_{share}} = \{\mathbf{0}\}$. This indicates that the clearing process is only done on identical solutions.
- For RCS and RCS_PSM the best value of size of the elite set is 100 ($\boldsymbol{M} = \mathbf{100}$). In this study, the population size is set as 100 solutions (equal to M). Therefore, in the elite set there is room for all initial solutions of each generation to compete with children population for survival.

In ANHGA, SCGA and CM, in the generation $t + 1$, there are niches of the generation t (from the surviving parents) and new niches (from new created solutions by crossover and mutation). Hence, this type of replacement process adds memory to the search which allows them to maintain good potential niches from generation to generation. We think that this type of replacement process is very good for multi-modal problems, and is the distinctive characteristic to improve the performance.

Multi-solution based efficacy

In this section we analyze which method allows us to find more number of optima per execution of GA. The average

Table 6 Best V_m obtained

		ANHGA	NIT	CM	RCS	RCS_PSM	SCGA	QHFD	Opt
*la*01	V_m	**666.00**	666.73	**666.00**	**666.00**	**666.00**	**666.00**	666.40	**666**
	SD	0.00	1.60	0.00	0.00	0.00	0.00	2.19	
*la*02	V_m	**660.63**	682.83	663.90	663.03	667.50	660.70	678.13	**655**
	SD	6.83	9.37	6.90	7.25	4.81	6.30	13.71	
*la*03	V_m	**595.97**	614.07	600.73	606.73	602.80	600.63	612.20	**590**
	SD	6.17	5.84	7.50	7.66	6.04	7.02	6.17	
*la*04	V_m	**608.97**	632.30	611.90	609.23	609.90	609.60	625.53	**597**
	SD	9.12	8.60	6.72	6.91	6.26	6.72	9.63	
*mt*10	V_m	**979.00**	1044.47	993.17	988.17	1006.07	989.70	1033.63	**930**
	SD	18.29	21.41	16.08	23.74	22.99	18.49	23.04	
*mt*20	V_m	**1213.03**	1320.63	1236.27	1235.47	1284.80	1237.97	1309.93	**1165**
	SD	24.72	21.17	30.23	22.24	24.45	27.46	30.89	

Table 7 Best methods according to *t*-test

la01	*la02*	*la03*	*la04*	*mt10*	*mt20*
ANHGA	ANHGA	ANHGA	ANHGA	ANHGA	ANHGA
CM	SCGA	RCS	SCGA		
RCS	RCS	SCGA	CM		
RCS_PSM	CM	RCS_PSM			
SCGA		CM			
QHFC					
NIT					

Table 8 Set of parameters for optimize the V_m

Method	Values of parameters for:			
ANHGA	RWS	$\sigma_{share} = \{20, 30\}$		$B = \{1\}$
	TS	$\sigma_{share} = \{20, 30, 40\}$		$B = \{1\}$
NIT	RWS	$B^* = \{0\}$	$N* = \{0.2\}$	$I = \{0\}$
	RWS	$B^* = \{0.1\}$	$N^* = \{0.2\}$	$I = \{0\}$
	RWS	$B^* = \{0.2\}$	$N^* = \{0\}$	$I = \{0\}$
	TS	$B^* = \{0.1\}$	$N^* = \{0.2\}$	$I = \{0\}$
	TS	$B^* = \{0.2\}$	$N^* = \{0.2\}$	$I = \{0\}$
CM	RWS	$\sigma_{share} = \{10\}$		$k = \{1\}$
RCS	$\sigma_{share} = \{10\}$		$M = \{100\}$	
RCS_PSM	$\sigma_{share} = \{10\}$		$M = \{100\}$	
SCGA	*SRS* with $\sigma_{share} = \{20\}$ for: $la01, la02, la03, la04$ and $la05$			
	TS with $\sigma_{share} = \{40\}$ for: $mt10$ and $mt20$			
QHFC	$N = 2000$	$L = 3$	$\gamma = \{0.6, 0.8\}$	
	$N = 2000$	$L = 5$	$\gamma = \{1\}$	
	$N = 1500$	$L = 5$	$\gamma = \{1\}$	
	$N = 1500$	$L = 3$	$\gamma = \{0.6, 1.0\}$	
	$N = 1000$	$L = 3$	$\gamma = \{0.8, 1.0\}$	
	$N = 500$	$L = 3$	$\gamma = \{1.0\}$	

number of optima of 30 repetitions for each combination of parameters values (N_{mo}) has been calculated, and the best N_{mo} for each method in different instances are summarized in Table 9. For *mt10* and *mt20*, we have to run 3 million iterations to obtain the optimum in some methods. In the rest of the instances, the experiments have been carried out with the same conditions as described in Sect. "Parameters and measures". In some experiments we have not obtained any run with optima and, therefore, there is not data to calculate the average (pointed out with a *). In other experiments we have obtained too few runs with optima and N_{mo} is calculated with few data for a suitable statistical study (pointed out with a **).

The maximum number of global optima obtained in each run is the highest value between the number of different global optima of the problem and the population size. As we can see, the population size limits the number of different optima to be obtained in every run (in this case this limit is 100 optima per run) for *la01, la02, la03, la04* and *la05* instances. For *mt06* instance, the number of optima to reach is limited to 52 because this is the number of different global optima known (42 obtained in Pérez et al. (2003) and 10 new optima found in this work). Thus, considering the *multi-solution based efficacy* the best methods are RCS, RCS-PSM and SCGA.

The parameter sets which reach the best performance for all instances (from the statistical point of view) are shown in Table 10.

The conclusions on this study are the following:

- In NIT the study of interferences ($I = 0$) and the elimination of niches ($N^* = 0$) should not be carried out.
- In CM, RCS, RCS_PSM and SCGA the best value of the niche radius is 0($\boldsymbol{\sigma_{share} = 0}$). Therefore, every solution is

Table 9 *Multi-solution based efficacy* N_{mo} (*) without data and (**) few values

	la01	*la02*	*la03*	*la04*	*la05*	*mt06*	*mt10*	*mt20*
ANHGA	30.54	16.86	12.40	19.50	39.73	11.70	2.22*	1**
NIT	8.76	7.00	*	*	13.03	8.22	*	*
CM	79.93	79.75	76**	32**	82.22	49.30	1**	1**
RCS	100.00	100.00	*	50**	100.00	49.67	1**	1**
RCS_PSM	99.90	99.95	*	50**	100.00	48.63	*	*
SCGA	93.37	99.00	95**	94**	98.93	48.07	*	*
QHFD	11.12	12**	4**	9**	75.10	13.10	*	*

Table 10 Set of parameters for N_{mo}

Method	Values of parameters
ANHGA	$\sigma_{share} = \{10\}$; $B = \{1\}$; RWS, TS
NIT	RWS; $B^* = \{0.1\}$; $N^* = \{0\}$; $I = \{0\}$
CM	$\sigma_{share} = \{0\}$; $k = \{1\}$; SRS and RWS
RCS	$\sigma_{share} = \{0\}$; $M = \{100\}$
RCS_PSM	$\sigma_{share} = \{0\}$; $M = \{100\}$
SCGA	$\sigma_{share} = \{0\}$; SRS and TS
QHFC	$N = \{1000, 1500, 2000\}$; $L = 3$; $\gamma = \{1\}$

a niche. For RCS and RCS_PSM the elite set is formed by all different solutions of the initial population. For SCGA all different solutions of the initial population of each generation are seeds of their own niche. In the replacement process, the common solutions of initial and children populations survive, and the worst solutions created in this iteration are substituted by seeds without copies in children population. In these methods, this radius causes slower convergence and higher diversification, i. e. poor performance but the highest *multi-solution based efficacy*.

- QHFC needs few populations with large sizes ($L = 3$ and $\gamma = 1$) to find the highest number of different optima but with slow convergence.

We point out that the best value of radius is 0, which is a surprising result. This radius causes slower convergence and higher diversification, i. e. poor performance but highest *multi-solution based efficacy*. On the other hand, again, the best methods have the same type of niches "memory". To obtain good final solutions and a high number of different optima the replacement process maintains in generation *t+1* the best niches of previous generations together with the new created niches in generation *t*.

A statistical study of the number of global optima is not possible in the instances *la02, la03* and *la04* because the analyzed methods obtain very few runs with optima (see * and ** in Table 9). For this reason, for the better methods: RCS, RCS_PSM, SCGA, CM and ANHGA, we have carried out all the necessary runs to achieve 30 runs with optima for each combination of parameters. The results of the best average values of N_{mo} are summarized in Table 11 but now, we have included a new column (P_E) to report the success percentage, i. e. the number of runs with optima divided by the number of total runs. The new results confirm the aforementioned conclusion, and the sets of values for the parameters are the same.

Diversity: exploration and exploitation

In this section we analyze the connection between the different parameter values and the distribution of solutions on the solution space. To do this, we have worked out the average distance between the found optima (D_m). Diversity can be understood as exploration or exploitation. On one hand, exploration of the search space is achieved with optima which belong to different areas (D_m is large). On the other hand,

Table 11 N_{mo} for large number of trials

	la02		la03		la04	
	N_{mo}	P_E	N_{mo}	P_E	N_{mo}	P_E
RCS	98.70	0.16	94.02	0.04	99.58	0.03
RCS_PSM	92.80	0.05	87.60	0.01	87.38	0.05
SCGA	86.09	0.09	91.5	0.01	95.00	0.01
CM	75.44	0.19	71.23	0.04	72.33	0.03
ANHGA	16.72	0.26	11.63	0.08	11.65	0.12

Table 12 Relationship between D_m and values of parameters

	Variable parameter	Connection	
ANHGA	σ_{share}	$\uparrow \sigma_{share}$	$\uparrow D_m$
CM	k, σ_{share}	$\uparrow \sigma_{share}$ with $\downarrow k$	$\uparrow D_m$
RCS	σ_{share}	$\uparrow \sigma_{share}$	$\uparrow D_m$
RCS_PSM	σ_{share}	$\uparrow \sigma_{share}$	$\uparrow D_m$
SCGA	σ_{share}	$\uparrow \sigma_{share}$	$\uparrow D_m$
QHFC	N, L, γ	$\uparrow N$ and $\downarrow L$ with $\gamma = \{1\}$	$\uparrow D_m$
		$\downarrow N$ and $\uparrow L$ with $\gamma = \{0.6\}$	$\downarrow D_m$

Table 13 Values for D_m (for *la05*)

	Parameters				
ANHGA	$\sigma_{share} = \{0\}$	$\sigma_{share} = \{10\}$	$\sigma_{share} = \{20\}$	$\sigma_{share} = \{30\}$	$\sigma_{share} = \{40\}$
	17.60	29.13	30.51	32.32	34.46
CM	$\sigma_{share} = \{0\}\mathbf{k} = \{10\}$		$\sigma_{share} = \{30\}\mathbf{k} = \{1\}$		$\sigma_{share} = \{40\}$
	24.08		36.24		erratic
RCS	$\sigma_{share} = \{0\}$	$\sigma_{share} = \{10\}$	$\sigma_{share} = \{20\}$	$\sigma_{share} = \{30\}$	$\sigma_{share} = \{40\}$
	28.11	31.59	34.55	37.38	erratic
RCS_PSM	$\sigma_{share} = \{0\}$	$\sigma_{share} = \{10\}$	$\sigma_{share} = \{20\}$	$\sigma_{share} = \{30\}$	$\sigma_{share} = \{40\}$
	24.23	30.88	34.19	37.09	41.33
SCGA	$\sigma_{share} = \{0\}$	$\sigma_{share} = \{20\}$	$\sigma_{share} = \{40\}$	$\sigma_{share} = \{60\}$	$\sigma_{share} = \{80\}$
	28.57	32.08	34.42	36.55	erratic
QHFC	$N = 500$ $L = 8$ $\gamma = 0.6$		$N = 2000$ $L = 3$ $\gamma = 1$		
	16.52		32.15		

in a search of exploitation the found optima belong to the same area (D_m is very short). The relationship between the distance and the parameter values are shown in Table 12.

Only the reported parameters have influence on the value of the distance, the rest do not have influence. In Table 13 we quantify these results (D_m) for different values of parameters (σ_{share}, N, L, and γ) for the *la05* instance. Note that *NIT* is not reported in both Tables 12 and 13 because the same distance is obtained (about 37) for all combination of the different parameter values (N^*, B^*).

As we expected, the radius niche (σ_{share}) has a great influence on diversity. If the radius is larger ($\uparrow \sigma_{share}$), the distance between solutions is greater ($\uparrow D_m$). Therefore, if we want to explore and obtain solutions distributed in the landscape, we have to use a larger radius. On the contrary, it is better to use a smaller radius for a more focused search. If the radius is too large, only one niche will exist and all solutions belong to this one. So, we can expect erratic behaviour (see CM, RCS, RCS-PSM and SCGA). In CM, the number of winners in each niche (k) and the niche radius are both important. If the number of winners is high and the radius is low there will be few near niches of big size in the population, and therefore, the search is focused on an area ($\downarrow D_m$). On the contrary, with the number of winner is $1 (k = 1)$ and high radius, there will be more distant niches of small size in the population, and the exploration will be large. In QHFC when the population is divided into few large subpopulations ($N = 2000$; $L = 3$; $\gamma = 1$), there will be exploration of the search space ($\uparrow D_m$). If there are many small size subpopulations ($N = 500$; $L = 8$; $\gamma = 0.6$) the optima will be closer (exploitation).

Conclusions

In this paper, we have compared the most recent MMGAs for the difficult real problem JSSP. Each MMGA has been adapted to the JSSP finding the best suitable parametrization of each method. The JSSP is a perfect testing-ground for MMGAs because it is NP-hard and its landscape has different global optima. This lets us appreciate the impact of the different parameter values on the results in terms of *efficacy, multi-solution based efficacy* and *diversity*. Very few works are focused on obtaining multiple different optima of the JSSP, but this can provide competitive advantages in this changing world.

From the point of view of efficacy, we have found that the most appropriate MMGAs is ANHGA in *mt10* and *mt20* (in general for instances of large size and high difficulty),

but ANHGA, CM, *and* SCGA in smaller size instances such as *mt06, la01, la02, la03, la04* and *la05*. We have statistically checked the importance of a suitable set of values for the different parameters. We have proved by *t*-test that these sets are robust and reliable, remaining the same for all analyzed instances. Particularly, we can see that the ANHGA with replacement value (B) equal to 1 has the best efficacy. With this value, the parent and children solutions are ordered by quality and *N* individuals with the highest quality survive to form the new population of next generation. In the case of RCS the best value of elite set (*M*) is 100 (the same as the population size) and the competition set for replacement is formed for parents and children. In the SCGA all niches (of parent and children population) are represented by a seed from each of them in the next population. The elitism process of the CM, the best parent niches survive along with new children niches.

On the other hand, in all these algorithms, in the generation *t+1* there are niches of the generation *t* (from the surviving parents) and new niches (from new created solutions by crossover and mutation). Hence, this type of *replacement process* adds memory to the search which allows us to maintain good potential niches from generation to generation. We think that this type of *replacement process* is very good for multi-modal problems, and is the distinctive characteristic to improve the performance. We think this is the key to improve the performance of MMGAs in JSSP in the future.

From the point of view of *multi-solution based efficacy*, we point out that the best value of radius is 0, which is a surprising result. This radius causes slower convergence and higher diversification, poor performance and the highest *multi-solution based efficacy*.

And from the point of view of diversity, radius (which defines the size of the niches) has a great influence. If the value of the radius is small, there are many niches of small sizes. Then, the number of optima is high (high *multi-solution based efficacy*) and they are near each other. On the contrary, if the value of the radius is large, there are less but broad niches, and few different optima (low *multi-solution based efficacy*) which are far from each other.

Acknowledgments This work was supported partly by the Spanish Ministry of Education and Science (MEC) under Project TIN2005-08386-C05-01 and by the Junta of Castilla y Leon under Project VA003B08.

Appendix

A.1. Flowchart of adaptive niche hierarchy genetic algorithm

See Fig. 7

A.2. Process for the niche identification technique (*NIT*)

1. The best solution is the centre of the first niche ($k = 1$).
2. Calculating the distance of the rest of the solutions of the population regarding this centre and sorting from lowest to highest with respect to this distance.
3. Calculating value $\beta_{j+1} = \frac{F_{j+1} - F_j}{F_{\max} - F_{\min}}$, being F_j the quality of the individual j, F_{j+1} the quality of the individual whose distance is the next highest to individual j in the sorted list, F_{max} and F_{min} the maximum and minimum quality of the population respectively.
4. Comparing β_{j+1} with a maximum permitted value called β^*.
   a. If $\beta_{j+1} < \beta^*$ individual *j+1* will be marked as belonging to niche *k*. Go to step 3 with the following individual in the list.
   b. If $\beta_{j+1} > \beta^*$ individual *j* will be the border of the niche establishing the radius of this niche (the distance between individual *j* and the centre *k*). Individual *j+1* will belong to different niche. Go to step 5.
5. Sorting out according to the quality of the population of unmarked individuals. The best unmarked individual will be the centre of the following niche (*k+1*). Go to step 2.
6. Once niches are identified, those than have a size less than parameter *N** are deleted.
7. And finally, a study on niche interferences is performed. Graphically, both possible interferences are represented in Fig. 8.

$$R_{new} = R_1 \times \sqrt[3]{\frac{R_2}{R_1}}, \quad L^* = \frac{R_2}{R_1 + R_2} \times L \text{ and}$$

$$R_{i-new} = \frac{R_i}{R_1 + R_2} \times L \quad \text{with} \quad i = 1, 2$$

8. The following sharing fitness is applied in selection:

$$f_{ik} = \frac{f_i}{N_k} \text{ with } N_k \text{ the size of niche } k,$$

or if the individual *i* does not belong to any niche:

$$f_j = \frac{f_j}{\frac{1}{m} \times \sum_{k=1}^{m} N_k} \quad \text{with } m \text{ the number of niches}$$

9. In the reproduction process a mating restriction strategy is implemented so that only individuals of the same niche can reproduce.

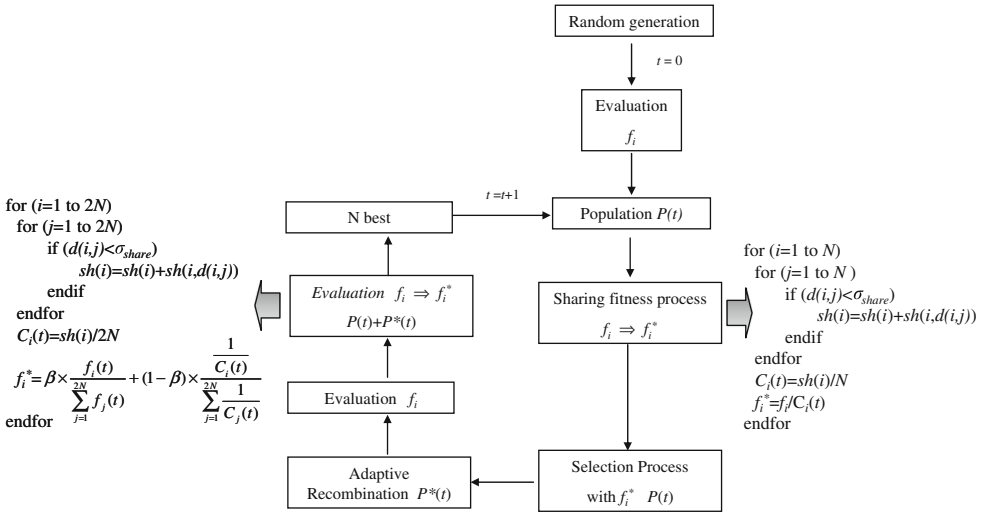


Fig. 7 Adaptive niche hierarchy genetic algorithm. (B = beta in the text)

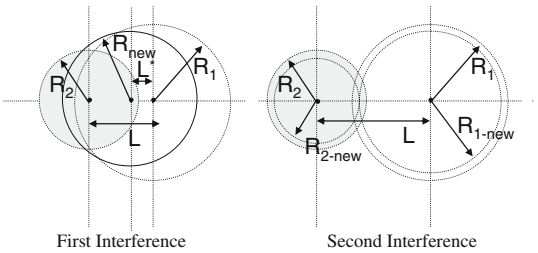


Fig. 8 Analysis of interferences. Source (Lin and Wu 2002)

10. Lastly, a modified elitism is applied in the substitution. All the centres of the identified niches are reserved for the next generation.

A.3. Flowchart of clearing method (*CM*)

See Fig. 9

A.4. Flowchart of restricted competition selection (*RCS*)

See Fig. 10

A.5. Process of local search in restricted competition selection with pattern search method (*RCS-PSM*)

For each solution of the Elite Set, the next local search method is applied (Giffer and Thompson 1960):

1. Build the set of sequential operations R that is initially going to contain the first operation of each job.
2. Estimate the shorter due date for operations that make up R, calculating $C^* = \min(C_r) \forall r \in R$. The operation with the C^* determines the machine M^*, in case of draw between machines it is randomly selected among them.
3. Make a set of conflict S with the necessary pieces of machine M^*.
4. Select one of the operations of S to arrange in sequences. In case of draw the sequence obtained by the *GAs* is selected.
5. Return to step 1.

A.6. Process of species conserving genetic algorithm (*SCGA*)

The general structure of the *SCGA* is very similar to a simple *GA*, and can be summarized in the following steps:

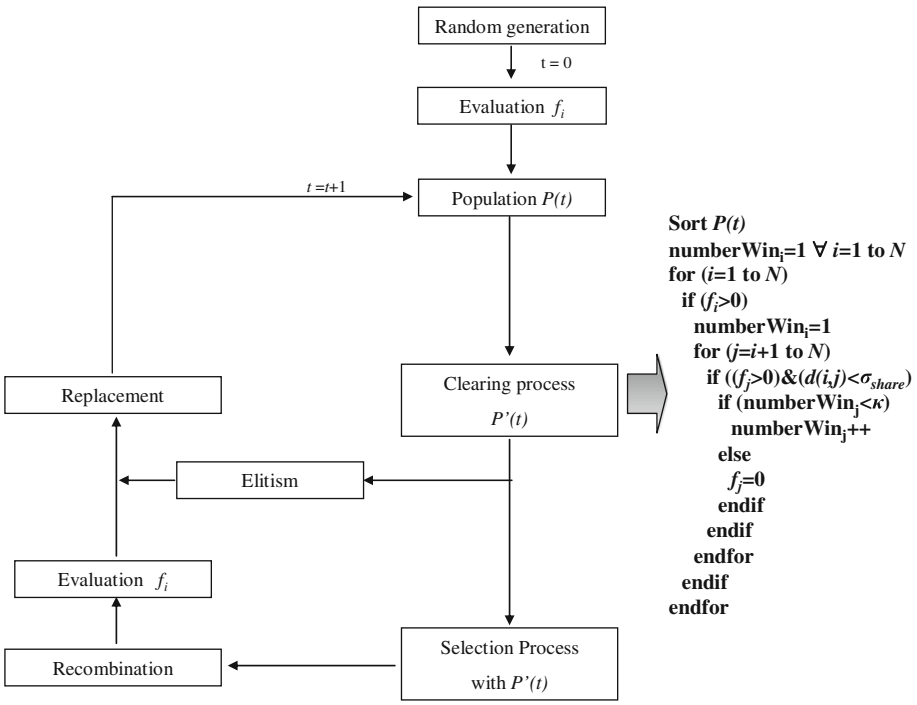
1. $P(t)$
2. Evaluate $P(t)$
3. Identify species seeds X_s
4. Select $P(t+1)$
5. Crossover and Mutate $P(t+1)$
6. Evaluate $P(t+1)$
7. Conserve species form X_s in $P(t+1)$
8. $t = t+1$, go back to step 3.

Next we will explain steps 3 and 7 more thoroughly.

Steps 3:

1. $P(t)$ is arranged from highest to lowest.
2. The best individual is the seed of the first species, and therefore the first component of set X_s.
3. If the distance of the rest of individuals in the arrangement is longer than $\sigma_{share}/2$ from all seeds contained to the moment in X_s, this individual will be a new seed and

Fig. 9 Clearing method



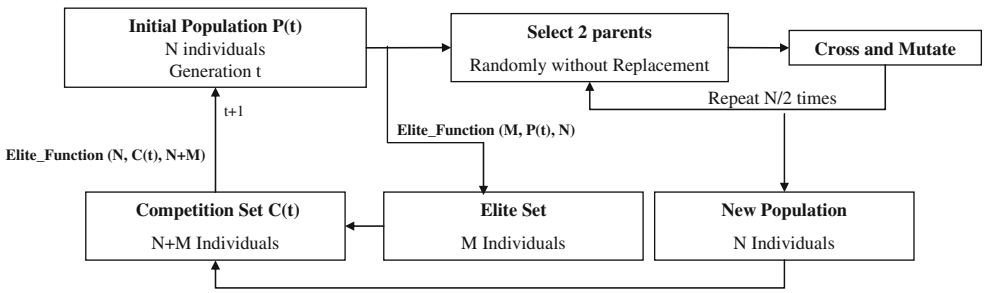


Fig. 10 Restricted competition selection

therefore will be joined to X_s. If not, the solution is not included.

4. With X_s population is classified into different identified species. An individual j will belong to a species which seed is x_i if its distance $d(j, x_i)$ is smaller than $\sigma_{share}/2$, if not this individual does not belong to any specie.

Step 7:

1. The individuals of $P(t+1)$, with a distance to any seed of X_s longer or equal to $\sigma_{share}/2$, will be marked. In the end, the marked individuals will belong to some species and the unmarked individuals will not belong to any of the species detected. Also, in X_s there will be seeds with species in $P(t+1)$ and others where this correspondence has not been produced.
2. In each species the seed will replace the solution with the worst quality.
3. Seeds without species will replace the individuals with the worst quality among the unmarked ones.

A.7. Process of quick hierarchical fair competition (*QHFC*)

The structure of quick hierarchical fair competition is:

1. Random generation.
2. The population *P(t) is* divided into *N* subpopulation by the size factor γ.
3. The threshold of minimal quality for each subpopulation is established.

The next steps are to be repeated:

4. *BreedTopFreq. BTF* generations are produced using deterministic crowding in the top level group. If during these generations there are *noprogressGen* (*NPG*)

generations without improvement the process is called *import_from_below* with parameters (*percentRefill, 0*) (step 8). Whereas, step 5 will be performed if improvement is produced.

5. *Potency testing*. Two individuals are randomly selected, then crossed, mutated and evaluated, repeating *catchup-Gen* (*CG*) evaluations. This is done in all the subsets, except in the top level. If there are *detectExportNo* (*DEN*) candidate individuals in all levels to be exported, then it will be considered that this process has been carried out with success. The exportation of individuals is performed by checking their quality and the ranges of minimum quality thresholds, each individual will be moved to its own specie.
6. In the case of success, if there is a gap in any level that is not filled after performing the potency test, let's assume in level l with g gaps, we will have to turn to the process *import_from_below* with parameters (g, l), step 8. If there are no gaps and the test has been performed with success then go back to step 4.
7. If in some level (l) the test has not been successfully performed go to step 8 with parameters (*percentRefill, l*). After a complete generation of deterministic crowding is carried out with all of the population in this level and go back to step 4.
8. *Import_from_below* with parameters (m, n). m will indicate the number of individuals that need to be randomly selected from level $n+1$ and be exported to level n. If during the procedure a gap is required to be filled, no individual has to be replaced. On the contrary, individuals that need to be replaced will be randomly selected (except the best one and those that have just been imported). Repeat step 8 with parameter $(m, n+1)$ until the lowest level is reached, where the gaps will be filled with individuals randomly generated.

References

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science*. doi:10.1287/mnsc.34.3.391.

Amirthagadeswaran, K. S., & Arunachalam, V. P. (2007). Enhancement of performance of genetic algorithm for job shop scheduling problems through inversion operator. *International journal of advanced manufacturing technology*. doi:10.1007/s00170-005-0392-3.

Aydin, M. E., & Fogarty, T. C. (2002). Simulated annealing with evolutionary process for job-shop scheduling problems. In K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou, & T. C. Fogarty, *Evolutionary methods for design, optimisation and control*. Barcelona: CIMNE.

Aydin, M. E., & Fogarty, T. C. (2004a). A simulated annealing algorithm for multi-agent systems: A job-shop scheduling application. *Journal of Intelligent Manufacturing*. doi:10.1023/B:JIMS.0000042665.10086.cf.

Aydin, M. E., & Fogarty, T. C. (2004b). Teams of autonomous agents for job-shop scheduling problems: An experimental study. *Journal of Intelligent Manufacturing*. doi:10.1023/B:JIMS.0000034108.66105.59.

Beasley, D., Bull, D., & Marti, R. (1993). A sequential niche technique for multimodal function optimization. *Evolutionary Computation*. doi:10.1162/evco.1993.1.2.101.

Brucker, P. (1997). *Scheduling algorithms* (2nd ed.). Berlin, Germany: Springer.

Bruns, R. (1993). Direct chromosome representation and advanced genetic operators for production scheduling. In S. Forrest (Ed.), *Proc. of the 5*th *International Conference on Genetic Algorithms* (pp. 352–359). San Mateo: Kaufmann.

Canbolat, Y., & Gundogar, E. (2004). Fuzzy priority rule for job shop scheduling. *Journal of Intelligent Manufacturing*. doi:10.1023/B:JIMS.0000034116.50789.df.

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

Cavichio, D. (1970). *Adaptive search using simulated evolution*. PhD Thesis. University of Michigan.

Cedeño, W., & Vemuri, V. R. (1999). Analysis of speciation and niching in the multi-niche crowding GA. *Theoretical Computers Science*, (229) (pp. 177–197). Elsevier.

Davis, L. (1989). Adapting operators probabilities in genetic algorithms. In J. D. Schaffer (Ed.), *Proc. of the 3rd International Conference on Genetic Algorithms* (pp. 375–378). San Mateo: Kaufmann.

Dunwey, G., Fengping, P., & Shifan, X. (2002). Adaptive niche hierarchy genetic algorithm. *In Proc. of IEEE TENCON* (pp. 39–42).

Eiben, A. E., & Smith, J. E. (2007). *Introduction to evolutionary computing (Natural Computing Series)*. Berlin, Germany: Springer.

El-Bouri, A., Azizi, A., & Zolfaghari, S. (2007). A comparative study of a New Heuristic based on adaptive memory programming and simulated annealing: The case of job shop scheduling. *European Journal of Operational Research*. doi:10.1016/j.ejor.2005.12.013.

Fang, H. (1994). *Genetic algorithms in timetabling and scheduling. Doctoral dissertation*. Department of Artificial Intelligence. University of Edinburgh.

Fang, H., Ross, P., & Corne, D. (1993). A promising genetic algorithm approach to job shop scheduling, rescheduling and open shop scheduling problem. In S. Forrest (Ed.), *Proc. of the 5th International Conference on Genetic Algorithms* (pp. 375–382). San Mateo: Kaufmann.

Fogel, D. B. (Ed.). (1998). *Evolutionary computation. The fossil record (Selected readings on the history of evolutionary computation)*. New York: IEEE press.

French, S. (1982). *Sequencing and scheduling: An introduction to the mathematics of the job shop*. Chichester, USA: Ellis Horwood.

Garey, M., & Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-Completeness*. New York: Freeman.

Gento, A. M., & Pérez, M. E. (2002). Study on the genetic operators for the job shop problem. *In Proc. of the First Spanish Conference on Evolutionary and Bioinspired Algorithms* (pp. 523–530). Mérida, Spain, (in Spanish).

Geyik, F., & Cedimoglu, I. (2004). The strategies and parameters of tabu search for job-shop scheduling. *Journal of Intelligent Manufacturing*. doi:10.1023/B:JIMS.0000034106.86434.46.

Giffer, B., & Thompson, G. L. (1960). Algorithms for solving production scheduling problems. *Operations Research*. doi:10.1287/opre.8.4.487.

Glover, F., & Laguna, M. (1997). *Tabu search*. Boston: Kluwer.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. MA: Addison-Wesley.

Goldberg, D. E. (2002). *The design of innovation: Lessons from and for competent genetic algorithms*. Boston, MA: Kluwer.

Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In *Proc. of the 2nd International Conference on Genetic Algorithms* (pp. 41–49).

Greenberg, H. (1968). A branch-bound solutions to the general scheduling problem. *Operations Research*. doi:10.1287/opre.16.2.353.

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

Hart, E., Ross, P., & Corne, D. (2005). Evolutionary scheduling: A review. *Genetic Programming and Evolvable Machines, 6*, 191–220.

Hasan, S. M. K., Sarker, R., & Cornforth, D. J. (2007). Hybrid genetic algorithm for solving job-shop scheduling problem. In R. Lee, M. U. Chowdhury, S. Ray, & T. Lee (Eds.), *Proceedings of the 6th IEEE International Conference on Computer and Information Science* (pp. 519–524). July 2007, Melbourne.

Hoss, H., & Stützle, T. (2004). *Stochastic local search-foundations and applications*. San Francisco: Morgan Kaufmann.

Hu, J. J., & Goodman, E. D. (2004). Robust and efficient genetic algorithms with hierarchical niching and a sustainable evolutionary computation model. In K. Deb, et al. (Eds.), *GECCO* (pp. 1220–1232).

Jain, A. S., & Meeran, S. (1999). Theory and methodology deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research, 113*, 390–434.

Kim, J., Cho, D., Jung, H., & Lee, C. (2002). Niching genetic algorithm adopting restricted competition selection combined with pattern search method. *IEEE Transactions on magnetic*. doi:10.1109/20.996257.

Kirkpatrick, S., Gelatt, C. D., Jr., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*. doi:10.1126/science.220.4598.671.

Kobayashi, S., Ono, I., & Yamamura, M. (1995). An efficient genetic algorithm for the job shop scheduling problem. In L. Eschelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms* (pp. 506–511). San Francisco: Kaufmann.

Lee, Ch., Cho, D., & Jung, H. (1999). Niching genetic algorithm with restricted competition selection for multimodal function optimization. *IEEE transactions on magnetics*. doi:10.1109/20.767361.

Li, J., Balazs, M., Parks, G. T., & Clarkson, P. J. (2002). A species conserving genetic algorithm for multimodal function optimization. *Evolutionary computation*. doi:10.1162/106365602760234081.

Lin, S., Goodman, E.D., & Punch, W. P. (1997). A genetic approach to dynamic job shop scheduling problems. In T. Bäck (Ed.), Proceedings *of the Seventh International Conference on Genetic Algorithms* (pp. 481–488). San Francisco: Kaufmann.

Lin, C., & Wu, W. (2002). Niche identification techniques in multimodal genetic search with sharing scheme. *Advances in Engineering Software* (33), 779–791.

Mahfoud, S. W. (1992). Crowding and preservation revisited. In R. Männer, & B. Manderick (Eds.), *Parallel problem solving form nature II* (pp. 27–36). New York: Elsevier.

Mattfeld, D. C. (1995). *Evolutionary search and the job shop. Investigations on genetic algorithms for production scheduling*. Berlin: Springer.

Michalewicz, Z. (1995). *Genetic algorithms + Data structures + Evolutions programs*. Berlin, Germany: Springer.

Nakano, R., & Yamada, T. (1991). Convencional genetic algorithms for job shop problems. In R. Belew, & L. B. Booker (Eds.), *Proc. of the 4th International Conference on Genetic Algorithms* (pp. 474–479). California: Kaufmann.

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

Nowicki, E., & Smutnicki, C. (2005). An advanced tabu algorithm for the job shop problem. *Journal of Scheduling, 8*, 145–159.

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

Panwalkar, S. S., & Iskander, W. (1977). A survey of scheduling rules. *Operations Research*. doi:10.1287/opre.25.1.45.

Pétrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. In *Proc. IEEE International Conference on Evolutionary Computation* (pp. 798–803). Japan.

Pétrowski, A. (1997). A new selection operator dedicated to speciatin. In T. Bäck (Ed.), *Proc. of the 7th International Conference on Genetic Algorithms* (pp. 144–151). San Mateo: Kaumann.

Pérez, E., Herrera, F., Hernández, C. (2003). Finding multiple solutions in job shop scheduling by niching genetic algorithms. *Journal of Intelligent Manufacturing*. doi:10.1023/A:1024649709582.

Ramalhinho, H., Marti, O., & Stützle, T. (2003). Iterated local search. In F. Glover & G. A. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 321–354). MA: Kluwer.

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

Sivanandam, S. N., & Deepa, S. N. (2007). *Introduction to genetic algorithms*. Berlin, Germany: Springer.

Usher, J. (2003). Negotiation-based routing in job shops via collaborative agents. *Journal of Intelligent Manufacturing*. doi:10.1023/A:1025705426184.

Van Laarhoven, P. J. M., Aarts, E. H. L., & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. Operations Research, doi:10.1287/opre.40.1.113.

Vazquez, M., & Whitley, L. D. (2000). *A comparison of genetic algorithms for the static job shop scheduling problem*. In *Parallel Problem Solving from Nature Conference 2000 (PPSN VI)* (pp. 303–312).

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

Watson, J. P., Beck, C., Howe, A. E., & Whitley, L. D. (2003). Problem difficulty for Tabu search in job-shop scheduling. *Artificial Intelligence, 143*(2), 189–217.

Watson, J. P., Howe, A. E., & Whitley, L. D. (2006). Deconstructing Nowicki and Smutnicki's *i*-TSAB tabu search algorithm for the job-shop scheduling problem. *Computers and Operations Research*. doi:10.1016/j.cor.2005.07.016.

Weckman, G., Ganduri, C., & Koonce, D. (2008). A neural network job-shop scheduler. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-008-0073-9.

Wenqi, H., & Aihua, Y. (2004). An improved shifting bottleneck procedure for the job shop scheduling problem. *Computers and Operations Research*. doi:10.1016/S0305-0548(03)00243-0.

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

URL: Further explanations about JSSP; October 2009; www.eis.uva.es/elena/JSSP.

URL: Further explanations about MMGAs; October 2009; www.eis.uva.es/elena/MMGAs.

URL: Optima solutions of la01-la05, mt06, mt10 ad mt20;October 2009; www.eis.uva.es/elena/JSSP/optima.htm.

URL: OR-Library; October 2009; http://people.brunel.ac.uk/~mastjjb/jeb/info.html.