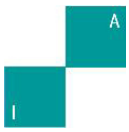
Inteligencia Artificial 16(52) (2013), 42-51



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Assesing The Performance of Different S-Metaheuristics To Solve Unrestricted Parallel Identical Machines Scheduling Problem

Claudia Ruth Gatica, Susana Cecilia Esquivel, Mario Guillermo Leguizamón

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)
Universidad Nacional de San Luis.
Ejército de Los Andes 950 - Local 106 (5700)- San Luis - Argentina.
Tel: (0266) 4420823 / Fax: (0266) 4430224
{crgatica,esquivel,legui}@unsl.edu.ar

Abstract In this paper we present a comparative study of four trajectory or single-solution based metaheuristics (S-metaheuristics): Iterated Local Search (ILS), Greedy Randomized Adaptive Search Procedure (GRASP), Variable Neighborhood Search (VNS), and Simulated Annealing (SA). These metaheuristics were considered to assess their respective performance to minimize the Maximum Tardiness (T_{max}) for the unrestricted parallel identical machine scheduling (P_m) problem, which is considered an NP-Hard problem. The results obtained through experimentation show that SA was the best performing metaheuristic.

Resumen En este trabajo se presenta un estudio comparativo de cuatro metaheurísticas basadas trayectoria o de solución única (S-metaheurísticas): Búsqueda Local Iterada (ILS), Procedimiento de Búsqueda Greedy Aleatorizado Adaptativo (GRASP), Búsqueda de Vecindario Variable (VNS), y Recocido Simulado (SA). Estas metaheurísticas fueron consideradas para evaluar su rendimiento respectivo de minimizar la Máxima Tardanza (T_{max}) para el problema de planificación de máquinas paralelas idénticas irrestrictas (P_m), el cual se considera un problema NP-duro. Los resultados obtenidos mediante la experimentación demuestran que SA fue la metaheurística de mejor desempeño.

Keywords: Iterated Local Search, Greedy Randomized Adaptive Search Procedure, Variable Neighborhood Search, Simulated Annealing, Parallel Machines Scheduling, Maximum Tardiness, Nonparametric Statistical Tests.
Palabras Clave: Búsqueda Local Iterada, Procedimiento de Búsqueda Greedy Aleatorizado Adaptativo, Búsqueda de Vecindario Variable, Recocido Simulado, Planificación de Máquinas Paralelas Idénticas Irrestrictas.

1 Introduction

The unrestricted parallel identical machines problem P_m considered in this paper consists of scheduling n jobs on m identical parallel machines P_m to minimize the Maximum Tardiness or T_{max}. There are no constraints in the assignment of jobs to machines, therefore the problem is described by $(P_m||T_{max})$. This problem belongs to more basic model of P_m which is NP-hard, even when $m = 2$ [15]. The P_m is representative of many real world problems. In such systems is usual to minimize objective functions based on the due dates, such as the T_{max}. Trajectory based metaheuristics have been successfully used to solve different scheduling problems. For instance in [14], [15] a set of dispatching rules and heuristics are presented. In a related work [17] the P_m was solved with SA and GRASP algorithms to minimize

ISSN: 1988-3064(on-line)

the maximum completion time (i.e., the makespan). In [1] SA was used to solve P_m with sequence-dependent setup times to minimize the makespan. The VNS algorithm was developed in [16] to minimize the weighted number of late jobs with release dates and in [11] to minimize the makespan. The ILS algorithm was presented in [4] to solve P_m with sequence-dependent setup times and unequal ready times to minimize total weighted number of tardy jobs.

This work is part of a preliminary study that has a double purpose: a) determine which S-metaheuristic shows better performance for solving the problem at hand, and b) find (if possible) new and better know optimal values for the instances used as benchmark. S-metaheuristics were chosen because they are the most classical optimization techniques and they have proven to be effective for finding good solutions on various types of scheduling problems. A statistical analysis is also conducted to determine the best parameter configuration for each metaheuristic and determine the best performing among them.

The remainder of this paper is organized as follows. Section 2 gives a brief description of the P_m problem. In Section 3, the studied trajectory based metaheuristics are presented. The experimental design is described in Section 4. In Section 5, the results are analyzed. Section 6 provides our conclusions and future work.

2 Unrestricted Parallel Machine Scheduling Problem

The formal notation used in the literature [15] for the scheduling problem that we are dealing is a triplet: $(P_m||T_{max})$. The first field describes the machine environment P_m, the second one contains the restrictions, we note that our problem is unrestricted, therefore this field is empty, and the third one provides the objective function T_{max} to be optimized.

This scheduling problem can be stated as follows: there are n jobs to be processed without interruption on some of the m identical machines belonging to the system P_m; each machine can process not more than one job at a time. Job j $(j = 1, 2, ...n)$ is made available for the processing at time zero. It requires an uninterrupted positive processing time p_j on a machine and it has a due date d_j by which it should ideally be finished. For a given processing order of the jobs (schedule), the earliest completion time C_j and the maximum delay time $T_j = \{C_j$ - $d_j, 0\}$ of the job j can be easily estimated. The problem consists in finding an optimum schedule objective value. The objective to be minimized is:

$$MaximumTardiness : T_{max} = max_j(T_j)$$

The problems related to the due dates have received considerable attention from a practical and theoretical point of view. Besides, they are considered as NP-Hard when $2 \leq m \leq n$ [15].

3 Description of Trajectory Based Metaheuristics

Single-solution based metaheuristics (S-metaheuristics) improve a single solution. They could be viewed as walks through neighborhoods or search trajectories through the search space of the problem at hand [18]. The walks (or trajectories) are performed by iterative procedures that move from the current solution to another one in the search space. S-metaheuristics show their efficiency in tackling various optimization problems in different domains. A short description of the basic local search as well as the studied S-metaheuristics are given in the following.

3.1 BLS

Basic Local Search is the oldest and simplest metaheuristic. It starts at a given initial solution. At each iteration, the search procedure replaces the current solution by a neighbor that improves the objective function. The search stops when all candidate neighbors are worse than the current solution, meaning that a local optimum is reached. Algorithm 1 describes this process [18]. BLS metaheuristic is included here because it is invoked by ILS, GRASP, and VNS.

Algorithm 1 BLS(s_0)

1: s=s_0 {s_0 is the initial solution}
2: **while** {not Termination Criterion} **do**
3: Generate($N(s)$) {Generation of candidate neighbors}
4: **if** {There is no better neighbor in $N(s)$} **then**
5: Stop
6: **else**
7: s'=s {Select a better neighbor in $N(s)$}
8: **end if**
9: **end while**
10: {**Output**: Best solution found}

3.2 ILS

Iterated Local Search (ILS) may be used to improve the quality of successive local optima. In multistart local search, the initial solution is always chosen randomly and then is unrelated to the generated local optima. ILS improves the classical multistart local search by perturbing the local optima and reconsidering them as initial solutions. Algorithm 2 shows the pseudocode of ILS [18].

Algorithm 2 ILS

1: s=BLS(s_0) {Apply a local search algorithm}
2: **repeat**
3: s'= Perturb(s) {Perturb the obtained local optima}
4: s''=BLS(s') {Apply a local search algorithm on the perturbed solution}
5: s= Accept(s, s'') {accepting criteria}
6: **until** Stopping criteria
7: {**Output**: Best solution found}

3.3 GRASP

GRASP metaheuristic is an iterative greedy heuristic to solve combinatorial optimization problems. Each iteration of the GRASP algorithm contains two steps: construction and local search. In the construction step, a feasible solution is built using a randomized greedy algorithm, then a local search heuristic is applied to the solution constructed in the previous step. GRASP pseudocode is displayed in Algorithm 3 [18].

Algorithm 3 GRASP

1: **repeat**
2: s=*Greedy*(*seed*) {Apply a randomized greedy heuristic}
3: s'=BLS(s) {Apply a local search algorithm}
4: **until** Stopping criteria
5: {**Output**: Best solution found}

The Greedy subalgorithm (line 2) uses either heuristics to improve the current solution, these are the minimum of the due date, or the minimum slack [14].

3.4 VNS

The basic idea of VNS is to successively explore a set of predefined neighborhoods to provide a better solution. It explores either at random or systematically a set of neighborhoods to get different local optima.

Algorithm 4 VNS

{**Input**: a set of neighborhood structures N_k for $k = 1, ..., k_{max}$}
s=s_0 {Generate the initial solution}
repeat
  $k = 1$
  repeat
    Shaking() {Pick a random solution s' from the N_k(s)}
    s''=BLS(s') {Apply a local search algorithm}
    if $s'' < s$ **then**
      $s = s''$
      $k = 1$ {Restart the search from $N_1(s)$}
    else
      $k = k + 1$
    end if
  until $k > k_{max}$
until Stopping Criteria
{**Output**: Best solution found.}

VNS exploits the fact that using various neighborhoods in local search may generate different local optima and that the global optima is a local optima for a given neighborhood. Indeed, different neighborhoods generate different landscapes. VNS pseudocode is given in Algorithm 4 [18].

3.5 SA

SA is based on the principles of statistical mechanics whereby the annealing process requires heating and then slowly cooling a substance to obtain a strong crystalline structure. The strength of the structure depends on the rate of cooling metals. If the initial temperature is not sufficiently high or a fast cooling is applied, imperfections (metastable states) are obtained. In this case, the cooling solid will not attain thermal equilibrium at each temperature. Strong crystals are grown from careful and slow cooling. The SA algorithm simulates the energy changes in a system subjected to a cooling process until it converges to an equilibrium state (steady frozen state). This scheme was developed in 1953 by Metropolis [14] and it is described in Algorithm 5 [18].

Algorithm 5 SA

$k = 0$ {Using for the Equilibrium condition}
$s = s_0$ {Generation of the initial solution}
$T = T_{max}$ {Starting temperature}
repeat
  repeat
    $k = k + 1$
    Generate a random neighbor s'
    $\Delta E = f(s') - f(s)$
    if $\Delta E \leq 0$ **then**
      $s = s'$
    else
      Acept s' with a probability $e^{\frac{-\Delta E}{T}}$
    end if
  until *mod*(k,*Markov-chain-length*) == 0 {Equilibrium condition}
  Update (T) {Temperature Update}
until Stopping Criteria
{**Output**: Best solution found.}

4 Description of Experiments

As it is not usual to find published benchmarks (known optimal values) for the unrestricted parallel identical machines scheduling problems, we work with some problems that were used by previous works such as [8], [7] and [6]. In those works, the problem instances were built based on selected data corresponding to weighted tardiness problems and they were taken from the OR-Library [12]. These data were created as follows: For each job j ($j = 1, 2, ..., n$), an integer processing time p_j was generated from the uniform distribution $[1, 100]$. Instance classes of varying hardness were generated by using different uniform distributions for obtain the due dates. For a given relative range of due dates RDD ($RDD = 0.2, 0.4, 0.6, 0.8, 1.0$) and a given average tardiness factor TF ($TF = 0.2, 0.4, 0.6, 0.8, 1.0$) an integer due date d_j for each job j was randomly generated from the uniform distribution $[P(1 - TF - RDD/2), P(1 - TF + RDD/2)]$, where $P = SUM(j = 1, 2, ..., n)p_j$. In this way five instances were generated for each of the 25 pairs of values of RDD and TF, yielding 125 instances for each value of n.

With the purpose of creating the benchmark values in the works previously cited, the authors extracted pairs (p_j, d_j) which were taken from problem size of 100 jobs. For the experiments were selected 20 instances, the number of instances are not consecutive because each one was chosen randomly from different groups. The problem is harder for those with a highest identification number. The pairs were the input for dispatching rules provided by PARSIFAL [14], a Software package provided by Morton and Pentico. In the present work, the values returned by these rules are taken as benchmarks.

In order to assess the respective performance of the studied algorithms, we use the following three performance metrics:

1. *Best:* it is the best solution found in each run.
2. *Mean best (MBest):* it is the mean value of Best metric throughout all runs.
3. *Ebest* = ((*best value*−*opt-val*)/*opt-val*) × 100 it is the percentage error of the best found solution when compared with the known or estimated (upper bound) optimum value opt-val. It gives a measure on how far the best solution is from that known opt-val. When this value is negative, it means that the opt-val has been improved.

Before the optimization runs are started, we drive the design for computer experiments to choose the parameter values for each metaheuristic. There are two different design techniques described in [3]. The samples can be placed either on the boundaries, or in the interior of the design space. The former technique is used in the classical design of experiments (DOE) and the second one is used by a more modern method called Design and Analysis of Computer Experiments (DACE). DACE assumes that the interesting features of the true model can be found in the whole sample space. Therefore space-filling or exploratory design, which places a set of samples in the interior of the design space, are commonly used. Mckay et al. (1979) [13] proposed Latin Hypercube Sampling (LHS) as an alternative to the first proposed Monte Carlo method. LHS can be used to generate the design points for algorithm design, this is a Latin Hypercube Design (LHD).

4.1 Parameter Settings

We used the statistical software Project R to generate design points for each metaheuristic. And thus, we obtained a LHD for each Algorithm. Each design point represents one configuration of parameters. Thus, for ILS, GRASP, VNS, and SA we obtained a LHD of 20 points, one for each metaheuristic, and we did 20 experiments separately, because each metaheuristic has their proper design space. The configuration of parameters for BLS was included in ILS, VNS, and GRASP. The ranges of the parameters are given to generate the LHD samples, and they represent the dimensions of the design points. Table 1 depicts the parameter where each one represents respectively: OP=Operator Perturbation or Movement, GH=Greedy Heuristic (1) Minimum due date and (2) Minimum slack, NE=Number of Evaluations, MCL=Markov Chain Length, CR=Cooling Rate, and IT=Initial Temperature. The operators used are: (1) n-swap, (2) 2-opt, (3) 3-opt, (4) 4-opt, (5) shift, and (6) scramble. A detailed description of these operators can be found in [2].

We applied the Friedman test [9], [10], and [5], (Friedman two-way analysis of variances by ranks) which is a nonparametric similar of the parametric two-way analysis of variance. This test can be used

Table 1: Parameter Ranges

Heuristic	OP	GH	NE	MCL	CR	IT
ILS	[1,6]	–	[10e3,15e3]	–	–	–
GRASP	[1,6]	[1,2]	[10e3,15e3]	–	–	–
VNS	–	–	[10e3,15e3]	–	–	–
SA	[1,6]	–	[10e3,15e3]	[10e3,10e4]	[0.5,1]	[10e4,10e5]

Table 2: Average Rankings (Friedman)

Config.	ILS	GRASP	VNS	SA
c1	10.450000000000001	12.350000000000001	17.425	10.325
c2	12.850000000000001	8.700000000000001	7.6499999999999995	6.8
c3	6.974999999999999	10.274999999999999	7.025000000000001	7.950000000000002
c4	4.9	10.4	9.249999999999998	6.874999999999998
c5	6.075	8.55	6.625000000000001	1.6250000000000004
c6	8.7	14.775	17.875	12.700000000000003
c7	10.325	7.449999999999999	10.174999999999999	13.549999999999999
c8	**1.5500000000000007**	11.650000000000002	7.475	9.1
c9	13.1	13.85	16.625	**1.4500000000000002**
c10	18.275	14.824999999999998	**1.1250000000000002**	7.525000000000001
c11	9.75	8.399999999999999	6.900000000000001	11.25
c12	13.524999999999999	10.725000000000001	7.525000000000002	9.799999999999999
c13	18.674999999999997	10.849999999999998	15.999999999999996	16.6
c14	1.7500000000000007	**6.749999999999999**	16.6	6.15
c15	11.5	13.849999999999998	10.000000000000004	13.6
c16	7.6000000000000005	8.65	17.1	17.499999999999996
c17	7.6000000000000005	12.725	16.15	16.249999999999996
c18	18.4	7.675	6.925000000000001	14.45
c19	9.350000000000001	10.450000000000001	1.9250000000000007	17.175000000000004
c20	18.65	7.1000000000000005	9.625	9.325

for answering the following question: In a set of k samples (where $k \geq 2$), do at least two of the samples which represent populations with different median values?. The Friedman test is a multiple comparison test that aims to detect significant differences between the behavior of two or more algorithms.

For each metaheuristic, we applied the Friedman test and its results indicate us which parameter configurations are better than others. Theses configurations are shown in Table 2, where the configuration that achieved the best ranking for each metaheuristic is marked in boldface.

Table 3 shows the values that belong to the better configurations. It must be noticed that for VNS, column OP indicates [1,6] as this metaheuristic uses in turn each operator to generate the different neighborhoods when necessary (in Algorithm 4, $k_{max} = 6$).

Table 3: Parameter Settings

Heuristic	Config.	OP	GH	NE	MCL	CR	IT
ILS	c_8	6	–	13139	–	–	–
GRASP	c_{14}	5	1	14546	–	–	–
VNS	c_{10}	[1,6]	–	14719	–	–	–
SA	c_9	1	–	12301	6727	0.65	15197

All the experiments reported in this work were run on a sub-cluster conformed by 1 CPUs of 64 bits, processor Intel Q9550 Quad Core 2.83GHz, with 4GB DDR3 1333Mz of memory, 500 Gb SATA and 2 TB SATA hard disks, Asus P5Q3 motherboard and 11 CPUs of 64 bits each with processor Intel Q9550 Quad Core 2.83GHz, 4GB DDR3 1333Mz memory, 160 Gb SATA hard disk and Asus P5Q3 motherboard.

5 Analysis of results

All the metaheuristics have been run 30 times for each problem instance. Each run stop when the maximal number of objective function evaluations (300000) is achieved. In Table 4 we display the *Best*

values found by each metaheuristic. Table entries in boldface indicates that the heuristic found better values than the benchmark while entries in italics shows that the heuristic obtained values very close or equal to the known optimal value.

Table 4: The Best achieved by each metaheuristic

Ins.	Bench.	ILS	GRASP	VNS	SA
1	**548**	587	597	567	**542**
6	**1594**	**1594**	**1581**	**1576**	**1567**
11	**2551**	2577	2626	*2552*	**2539**
19	**3703**	3756	3784	3737	3718
21	**5187**	5193	5232	**5184**	**5177**
26	**84**	148	407	121	**70**
31	**1134**	1160	1366	1191	*1135*
36	**2069**	2128	2360	2116	**2061**
41	**3651**	**3631**	3821	3658	**3607**
46	**4439**	4475	4599	4460	*4440*
56	**617**	725	1104	704	**609**
61	**1582**	1779	2453	1720	**1580**
66	**2360**	2483	2870	2453	**2359**
71	**3786**	3924	4413	3890	3791
86	**1194**	1455	2281	1408	**1194**
91	**2204**	2427	2953	2419	2222
96	**3185**	3256	3780	3217	*3187*
111	**1365**	1846	3216	1684	1458
116	**2222**	2537	3055	2515	2266
121	**2999**	3407	3890	3260	3099
avg	**2323,7**	**2454,40**	**2819,40**	**2421,60**	**2331,05**

As shown in Table 4, it can be observed that SA improved benchmark values in ten instances (1, 6, 11, 21, 26, 36, 41, 56, 61, 66), in three instances (31, 46, 96) the values obtained are greater than the benchmark in just one unit or two units, and finally in instance 86 the two values match. Also in the remaining instances the solutions reached by SA are better than the results achieved by other metaheuristics. In other hand, with respect to the other metaheuristics the order is VNS, ILS, and GRASP from best to worst performing.

The results indicate that SA achieved a better performance in all the problem instances, this is graphically illustrated in Figure 1 where the boxplots draw the values of the *Ebest* metric, which represents the percentage error of the best found solution when compared with the known or estimated optimum value. Clearly, the boxplot for SA shows values nearly zero for almost all found results. This indicates the superiority of this optimization technique for the benchmarks considered with respect to the rest of S-metaheuristics studied in this work.

To establish a ranking among the considered algorithms we use the Friedman test. In nonparametric statistics is common to use this test to determine the difference between more than two related samples. In this study the related samples are the performance of the metaheuristics measured across the same data sets. The null hypothesis being tested is that all methods obtain similar results with non significant differences. The first step in calculating the test statistic is to convert the original results to ranks. Thus, the best performing algorithm should have the rank of 1, the second best rank 2, and so on. In order to verify the null hypothesis, if the algorithms have similar behavior ranges should be equal. In our analysis the Friedman test takes as input for each algorithm/problem pair the values of the metric *Mbest* (it is the mean value of *Best* value found by each metaheuristic throughout all runs).

The test results are shown in Table 5. Based on the results given by the test we can reject the null hypothesis since the ranges differ, which indicates that there are significant differences between the algorithms. Also the S-metaheuristic that shows the best behavior is SA because it is the first in the ranking. Additionally we also apply the Friedman Aligned Ranks and Quade tests which offer a different

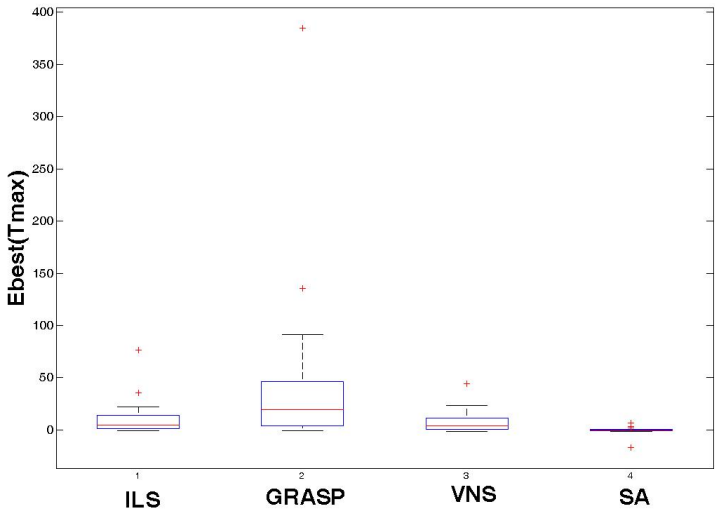


Figure 1: Boxplots of Trajectory Metaheuristics.

Table 5: Rankings of the algorithms

Heuristic	Friedman (R_j)	Friedman Aligned	Quade
ILS	3.049999999999999	51.275	3.142857142857143
GRASP	3.800000000000001	67.475	3.657142857142856
VNS	2.1500000000000004	28.75	2.1999999999999993
SA	1.0000000000000002	14.5	0.9999999999999999
statistic	52.380000000000024	15.196102517482714	68.34302921578677
P-value	5.608424835656933E-11	0.001656507570199528	6.985190855840604E-19

way of ranking computation, but all of them located SA in the first place. The main drawback of the previous tests is that they only detect significant differences over the whole multiple comparisons, being unable to establish proper comparison between some of the metaheuristics considered [5].

To compare each algorithm with each other S-metaheuristics and check if differences exist between them, we used a post-hoc procedure. The process is as follows: choose the S-metaheuristic that has shown better performance (in our case SA) and it is used as a control algorithm. Then each of the other S-metaheuristics used in the experimental study is compared with the control S-metaheuristic, and a family of hypotheses is built, all referred to the control method. The application of a post-hoc test allows us to obtain *p-values* that determines the degree of rejection of each hypothesis, depending on a certain level of significance. Table 6 displays the adjusted *p-values* obtained by the application of Holm Post-hoc test.

In the pairwise comparison we note that there are a significant difference between SA and VNS, ILS and, GRASP since all *p-values* are less than 0.05 (except en Frideman Aligned, but is almost equal).

Table 6: The adjusted *p-values* of Holm Post-hoc test

Heuristic	Friedman	Friedman Aligned	Quade
ILS	1.025635869766 96E-6	1.120472066462942E-6	3.4457480409525625E-4
GRASP	2.086578772558495E-11	1.6911216008002135E-12	9.574514552819139E-6
VNS	0.004848762721678958	0.05247949955924666	0.03540852150266215

Figure 2 illustrates the average number of evaluations of the objective function that each metaheuristic

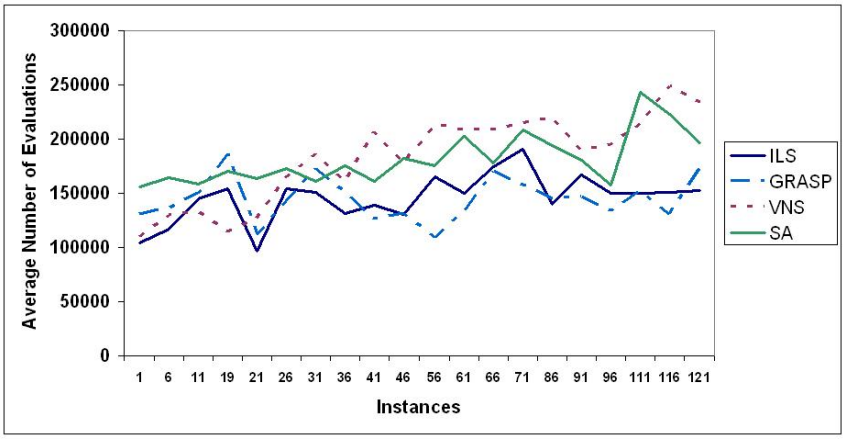


Figure 2: Average Number of Evaluations.

consuming to find the best solution. ILS and GRASP show a premature convergence (Used only half or less of the evaluations permitted) which leads to stagnation in distant solutions regarding the benchmark values. SA and VNS has a slower degree of convergence, which shows that they have greater capacity explorative, which allowed them to obtain better solutions in all instances tested, even in the hardest ones.

6 Conclusions

We compare four trajectory metaheuristics: ILS, GRASP, VNS, and SA to minimize the maximum tardiness for unrestricted parallel identical machine scheduling problem. Previously we realized a statistical study to determinate the best parameter values for each metaheuristic using the DACE method. Finally, we analyzed the results using different nonparametric statistical tests that allowed us to perform multiple comparisons between the algorithms to determine if any of the heuristics was better to solve the problem at hand. The tests shown that SA had a better performance. SA was then compared with each other metaheuristics, using the Holm Post-hoc test, in order to determine whether there were significant differences between them. The results showed that SA had indeed statistically significant differences with ILS, GRASP, and VNS. With respect to the goals outlined in the introduction we can conclude that: a) it has been determined that SA is the S-metaheuristic that showed the best performance, and b) the second purpose was also partially satisfied since improved best known values were obtained for 10 out of 20 instances of the problem analyzed. In order to improve the optimum values found so far (if possible) our future works will extend the present study by hybridizing the best performing S-metaheuristic found here with population-based metaheuristics (P-metaheuristics) as Genetic Algorithms or Ant Colony Optimization.

Acknowledgements

The authors would like to thanks to the University Nacional de San Luis for its continous support.

References

[1] Georgios C. Anagnostopoulos and Ghaith Rabadi. A simulated annealing algorithm for the unrelated parallel machine scheduling problem. *World Automation Congress Eight International Symposium on Manufacturing with Applications*, June 9-13 2002.

[2] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. Institute of Physics Publishing Bristol Philadelphia and Oxford University Press, New York, USA, 1997.

[3] T. Bartz-Beielstein. *Experimental Research in Evolutionary Computation - The New Experimentalism*. Springer, 2006.

[4] Chun-Lung Chen. An iterated local search for unrelated parallel machines problem with unequal ready times. *International Conference on Automation and Logistics Qingdao Proceedings of the IEEE*, September 2008.

[5] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 2011.

[6] E. Ferretti and S. Esquivel. A comparison of simple and multirecombinated evolutionary algorithms with and without problem specific knowledge insertion for parallel machines scheduling. *International Transaction on Computer Science and Engineering*, 3(1):207–221, 2005.

[7] E. Ferretti and S. Esquivel. An efficient approach of simple and multirecombinated genetic algorithms for parallel machine scheduling. *IEEE Congress on Evolutionary Computation*, 2:1340–1347, September 2005.

[8] E. Ferretti and S. Esquivel. Knowledge insertion: An efficient approach to simple genetic algorithms for unrestricted for parallel equal machines scheduling. *GECCO'05*, pages 1587–1588, 2005.

[9] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of American Statistical Association*, 3:674–701, 1937.

[10] M. Friedman. A comparison of alternative test of significance for the problem of the m rankings. *Annals of Mathematical Statistics*, 11:86–92, 1940.

[11] Kai Li and Ba-Yi Cheng. Variable neighborhood search for uniform parallel machine makespan scheduling problem with release dates. *International Symposium on Computational Intelligence and Design*, 2010.

[12] OR library Beasley J. `http://people.brunel.ac.uk/mastjjb/info.html`.

[13] M. D. Mckay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[14] T. Morton and D. Pentico. *Heuristic Scheduling Systems*. John Wiley and Sons, New York, 1993.

[15] M. Pinedo. *Scheduling: Theory, Algorithms and System*. Prentice Hall, 1995.

[16] Marc Sevaux and Kenneth Sörensen. Vns/ts for a parallel machine scheduling problem. *MEC-VNS: 18th Mini Euro Conference on VNS*, 2005.

[17] Panneerselvam Sivasankaran, Thambu Sornakumar, and Ramasamy Panneerselvam. Design and comparison of simulated annealing algorithm and grasp to minimize makespan in single machine scheduling with unrelated parallel machines. *Intelligent Information Management*, 2:406–416, Published Online July 2010. doi: 10.4236/iim.2010.27050 `http://www.SciRP.org/journal/iim`.

[18] E.G. Talbi. *Metaheuristics from design to implementation*. by John Wiley & Sons, Canada, 2009.