KNOWLEDGE INSERTION: AN EFFICIENT APPROACH TO REDUCE SEARCH EFFORT IN EVOLUTIONARY SCHEDULING

Pandolfi D., Lasso M., De San Pedro M., Villagra A.
Proyecto UNPA-29/B032[1]
División Tecnología
Unidad Académica Caleta Olivia
Universidad Nacional de La Patagonia Austral
Ruta 3 Acceso Norte s/n
(9011) Caleta Olivia – Santa Cruz - Argentina
e-mail: {mlasso,dpandolfi,edesanpedro,avillagra}@uaco.unpa.edu.ar
Phone/Fax : +54 0297 4854888

Gallard R.
Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)[2]
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 - Local 106
(5700) - San Luis -Argentina
e-mail: rgallard@unsl.edu.ar
Phone: +54 2652 420823
Fax : +54 2652 430224

Abstract

Evolutionary algorithms (EAs) are merely blind search algorithms, which only make use of the relative fitness of solutions, but completely ignore the nature of the problem. Their performance can be improved by using new multirecombinative approaches, which provide a good balance between exploration and exploitation. Even though in difficult problems with large search spaces a considerable number of evaluations are required to arrive to near-optimal solutions.

On the other hand specialized heuristics are based on some specific features of the problem, and the solution obtained can include some features of optimal solutions. If we insert in the evolutionary algorithm the problem specific knowledge embedded in good solutions (seeds), coming from some other heuristic or from the evolutionary process itself, we can expect that the algorithm will be guided to promising sub-spaces avoiding a large search.

This work shows alternative ways to insert knowledge in the search process by means of the inherent information carried by solutions coming from that specialised heuristic or gathered by the evolutionary process itself. To show the efficiency of this approach, the present paper compares the performance of multirecombined evolutionary algorithms with and without knowledge insertion when applied to selected instances of the Average Tardiness Problem in a single machine environment.

Keywords: Average tardiness scheduling problem, Evolutionary scheduling, conventional heuristics, problem-specific knowledge.

---

[1] The Research Group is supported by the Universidad Nacional de La Patagonia Austral.
[2] The LIDIC is supported by the Universidad Nacional de San Luis and the ANPCYT (National Agency to Promote Science and Technology).

Introduction

In a production system it is usual to stress minimum average tardiness to achieve higher client satisfaction on the average. The Average Tardiness problem ($1 \mid\mid 1/n \sum T_j$) [3, 15], is an important NP-hard scheduling problem which measures the adaptation of the system to client requirements. Its minimization leads to a situation where it is less likely that the waiting time of any given job will be unacceptably long.

Branch and Bound and other partial enumeration based methods, which guarantee exact solutions, are prohibitively time consuming even with only 20 jobs. To provide reasonably good solutions in very short time the scheduling literature offers a set of dispatching rules and heuristics. Depending on the particular instance of the problem we are facing, some of them behave better than others. Among others heuristics [11], evolutionary algorithms (EAs) have been successfully applied to solve scheduling problems [17, 18]. Current trends in evolutionary algorithms make use of multiparent [4, 5, 6] and multirecombinative approaches [7, 8, 9]. The latter we called, *multiple-crossovers-on-multiple-parents* (MCMP). Instead of applying crossover once on a pair of parents this feature applies n_1 crossover operations on a set of n_2 parents. In order to improve the balance between exploration and exploitation in the search process a variant called MCMP-SRI [12, 13] recombines a breeding individual (stud) by repeatedly mating individuals that randomly immigrates to a mating pool. Under this approach the random immigrants incorporate exploration and the multi-mating operation with the stud incorporates exploitation to the search process.

If we are trying to incorporate knowledge to the blind evolutionary search process, the issue here is how to introduce problem-specific knowledge? If optimality conditions for the solutions are known in advance we can restrict the search operating only on solutions which hold these conditions. When optimality conditions are unknown, which is the case, the answer is to provide information which is gathered by the evolution process itself and resides in the elitist individual, or to import this knowledge from solutions that come out from heuristics specifically designed for the problem under consideration. Both kinds of knowledge-based intermediate solutions contain some of the features, which are present in the best (optimal or quasi-optimal) solution at the end of the evolutionary process.

Consequently, MCMP-SRSI, a latest variant, considers the inclusion of a stud-breeding individual in a pool of random and seed-immigrant parents. Here, the seeds generated by conventional heuristics or by the evolutionary process itself, introduce the problem-specific knowledge. Next sections describe the average tardiness-scheduling problem, alternative ways to insert *problem-specific knowledge* and discuss the results obtained.

1. The average tardiness scheduling problem

The problem [15] can be stated as follows: n jobs are to be processed without interruption on a single machine that can handle no more than one job at a time. Job j (j=1,...,n) becomes available for processing at time zero, requires an uninterrupted positive processing time p_j on the machine, and has a due date d_j by which it should ideally be finished. For a given processing order of the jobs, the earliest completion time C_j and the tardiness $T_j = \max\{C_j - d_j, 0\}$ of job j can readily be computed. The problem is to find a processing order of the jobs with minimum average tardiness

$$\frac{1}{n}\sum_{j=1}^{n} T_j$$

The problem has received considerable attention by different researchers. For many years its computational complexity remained open until established as NP-Hard in 1989 [15].

2. Conventional approaches to the average tardiness problem

Dispatching heuristics assign a priority index to every job in a waiting queue and the one with the highest priority is selected to be processed next. There are different heuristics [11] for the Average Tardiness problem whose principal property is not only the quality of the results, but also to give an ordering of the jobs (schedule) close to the optimal sequence. The following dispatching rules and heuristics were selected to determine priorities, build schedules and contrast their outcomes with those obtained by the evolutionary algorithms.

SPT (Shortest Processing Time first) the job with the shortest processing time is selected first and in the final schedule jobs are ordered satisfying: $p_1 \leq p_2 \leq \ \ldots \leq p_n$.

EDD (Earliest Due Date first) the job with earliest due date is selected first and in the final schedule jobs are ordered satisfying: $d_1 \leq d_2 \leq \ \ldots \leq d_n$.

SLACK (Least slack) he job with smallest difference between due date and processing time is selected first first and in the final schedule jobs are ordered satisfying: $d_1\text{-}p_1 \leq d_2\text{-}p_2 \leq \ \ldots \leq d_n\text{-}p_n$.

Hodgson Algorithm: This heuristic provides a schedule according to the following procedure,
Step 1: Order the activities in EDD order.
Step 2: If there are no tardy jobs, stop; this is the optimal solution.
Step 3: Find the first tardy job, say *k*, in the sequence.
Step 4: Move the single job *j* ($1 \leq j \leq k$) with the longest processing time to the end of the sequence.
Step 5: Revise the completion times and return to step 2.

The algorithm is optimal for a related objective (unweighted number of tardy jobs) and can behave well for some instances of average tardiness.

Rachamadagu and Morton Heuristic (R&M). This heuristic provides a schedule according to the following priority,

$$\pi_j = (w_j / p_j)[\exp\{-(S_j)^+ / kp_{av}\}]$$

with $Sj = [dj - (pj + Ch)]$ is the slack of job *j* at time *Ch*, where *Ch* is the total processing time of the jobs already scheduled, *k* is a parameter of the method (usually k =2.0) and p_{av} is the average processing time of jobs competing for top priority. In the *R&M* heuristic, also called the *Apparent Tardiness Cost* heuristic, jobs are scheduled one at a time and every time a machine becomes free a ranking index is computed for each remaining job. The job with the highest-ranking index, is then selected to be processed next.

3. Multirecombination of random and seed immigrants with the stud

Multiple Crossovers per Couple (MCPC) [7, 8] and Multiple Crossovers on Multiple Parents (MCMP) [9] are multirecombination methods, which improve EAs performance by reinforcing and balancing exploration and exploitation in the search process. In particular, MCMP is an extension of MCPC where the multiparent approach of Eiben [4, 5, 6] is included. Results obtained in diverse

single and multiobjective optimization problems indicated that the searching space is efficiently exploited by the multiple application of crossovers and efficiently explored by the greater number of samples provided by the multiple parents. A further extension of MCMP is known as MCMP-SRI [12, 13]. This approach considered the mating of an evolved individual (the stud) with random immigrants. The process for creating offspring is performed as follows. From the old population, the stud is selected by means of proportional selection and inserted in the mating pool. The number of n_2 parents in the mating pool is completed with randomly created individuals (random immigrants). The stud mates every other parent, the couples undergo partial mapped crossover (PMX) and $2n_2$ offspring are created. The best of these 2 n_2 offspring is stored in a temporary children pool. The crossover operation is repeated n_1 times, for different cut points each time, until the children pool is completed. Finally, the best offspring created from n_2 parents and n_1 crossover is inserted in the new population. MCMP-SRSI [14, proposes to insert problem-specific-knowledge by recombining potential solutions (individuals of the evolving population) with seeds, which are solutions provided by other heuristics specifically intended to solve the scheduling problem under study. In MCMP-SRSI, the process for creating offspring is similar to that of MCMP-SRI, except that the mating pool contains also seed immigrants. In this way the evolutionary algorithm incorporates problem-specific-knowledge supplied by the specific heuristic.

In the present work we propose different versions of the MCMP family:

a) Without knowledge insertion

   MCMP-SRI, it works as above described.

b) With knowledge insertion, including the following MCMP-SRSI variants,

   b1) internal knowledge insertion. Here the knowledge acquired during the evolutionary process itself is inserted as a seed.

   SRSI-E. After the second generation, the best individual found so far (the elitist individual), is inserted in the mating pool as a unique seed individual along with the stud and random immigrants.

   SRSI-E-N. The mating pool is built as in SRSI-E variant but here a neighbourhood operator is added to eliminate possible copies of the best individual. After a search for copies, this operator replaces each copy by a neighbour created by random interchange of allele values. If more than one copy exists then, a neighbour created by a single interchange replaces the first copy, another neighbour created by two interchanges replaces the second copy, and so on. The idea is that copies will be replaced by individuals that retain certain genetic characteristics of the best individual, but differ more and more from this best individual as the number of copies augments.

   b2) external knowledge insertion. Here, solutions provided by other heuristics specifically intended to solve the scheduling problem are inserted as seeds in the mating pool.

   SRSI-H. Here, only one immigrant seed is inserted in the mating pool along with the stud and random immigrants. This seed is selected as the schedule built by the best heuristic in the corresponding instance. The seed belongs to every mating pool.

b3) internal and external knowledge insertion. Here, both the best individual found so far (elitist) and a solution provided by the best heuristic, are inserted as seeds in the mating pool.

SRSI-H-E. During the first two generations the schedule built by the best heuristic, in the corresponding instance, is inserted as a unique seed in the mating pool. In successive generations, the elitist individual remains as the unique seed. The seed shares the mating pool with the stud and random immigrants.

SRSI-H-E-N. The mating pool is built as in SRSI-H-E variant but here a neighbourhood operator is added to eliminate possible copies of the best individual. The neighbourhood operator works in the same way above described.

4. Experimental tests and results

As it is not usual to find published benchmarks for the Average Tardiness problem we built our own test suite with data (p_j ,d_j) extracted from 25 selected instances of the OR-library benchmarks for the weighted tardiness problem, with 40-jobs problem size, [1,2]. This data was the input for dispatching rules, conventional heuristics and our proposed multi-recombined EAs, MCMP-SRI and MCMP-SRSI.

Instance	SPT	LPT	EDD	SLACK	HODGSON	R&M
1	40.23	106.68	**13.05**	19.85	22.18	25.30
6	94.60	314.38	116.68	132.35	**88.55**	95.70
11	214.28	676.25	292.65	350.78	**203.08**	228.00
19	**542.85**	991.68	773.78	808.05	557.45	610.35
21	**525.43**	1236.13	879.48	1036.38	805.40	616.03
26	51.33	97.23	**0.40**	0.90	**0.40**	5.58
31	150.08	427.75	93.95	99.53	100.83	**84.95**
36	233.10	617.88	353.78	377.30	**199.88**	232.63
41	413.65	980.10	667.68	709.78	**400.70**	471.83
46	**375.33**	1001.25	628.53	748.40	704.88	439.63
51	121.38	211.35	**0.00**	**0.00**	**0.00**	1.05
56	107.03	222.58	30.93	40.33	56.50	**30.03**
61	241.20	688.00	263.43	292.68	**171.33**	199.75
66	**455.78**	987.53	608.65	631.00	566.78	530.93
71	**469.43**	1059.28	670.50	789.10	800.25	538.43
76	42.38	171.30	**0.00**	**0.00**	**0.00**	**0.00**
81	162.88	348.90	**4.85**	7.10	28.58	7.20
86	207.70	599.63	**127.35**	139.23	157.03	128.70
91	404.28	903.35	558.25	583.23	477.90	**383.68**
96	**658.40**	1210.18	920.70	1009.35	872.85	767.30
101	76.70	112.45	**0.00**	**0.00**	**0.00**	**0.00**
106	180.25	412.40	**0.00**	**0.00**	**0.00**	**0.00**
111	397.63	727.25	333.85	350.55	441.70	**275.23**
116	325.78	894.18	412.00	457.85	509.23	**319.13**
121	**598.78**	1322.30	904.50	1030.98	941.65	654.80

Table 1. Average Tardiness values found by each heuristic

To evaluate the dispatching rules and the conventional heuristics (SPT, EDD, SLACK, Hodgson and R&M) we used PARSIFAL [11] which is a software package provided by Morton and Pentico, to solve different scheduling problems by means of different heuristics.

The initial phase of the experiments consisted in to establish the best results from dispatching rules and conventional heuristics to use them as upper bounds for this scheduling objective. These results can be observed in table 1, where the first column identifies the instance number and the rest indicate the Average Tardiness achieved by each approach. Boldfaced values indicate the best (minimum) objective value, which will be used as an upper bound.
Results in table 1, shows that EDD is the best in 32% of the cases (8 instances), then SPT follows being the best in 28% of the cases (7 instances). Both best performers are followed by Hodgson and R&M which are he best in 20% of the cases (5 instances).

The second phase of the experiments consisted in to establish adequate parameter settings for MCMP-SRI and MCMP-SRSI and then to run a number of experimental series. After a series of initial trials the best parameter setting was determined under each algorithm, as follows:
The maximum number of generations was fixed to 500 and 200 for MCMP-SRI and MCMP-SRSI, respectively. Both algorithms run with a population size of 15 individuals, with $n_1 = 20$, $n_2 = 18$, and crossover probability fixed at 0.65. Mutation probability was set to 0.0 and 0.05 for MCMP-SRI and MCMP-SRSI, respectively. Series of ten runs where performed for each instance. To compare the algorithms, the following relevant performance variables were chosen:

Ebest = ((best value - opt_val)/opt_val)100
It is the percentile error of the best-found individual when compared with the known, or estimated (upper bound) optimum value opt_val. It gives us a measure on how far the best individual is from that opt_val.

Mean Ebest: It is the mean value of Ebest throughout all runs.

Best: It is the minimum objective value corresponding to some of the best-found individuals throughout all runs.

Max Best: It is the maximum objective value corresponding to some of the best-found individuals throughout all runs.

Mean Best: It is the mean objective value obtained from the best-found individuals throughout all runs.

Gbest:It is the generation where the best individual was found.

Mean Gbest: It is the mean generation number where the best individual was found, throughout all runs.

Hit Ratio: Denotes the percentage of runs where the algorithm reaches the upper bound or improves it. Its value is 1 (a 100% of success) when the upper bound is reached or improved in every run.

Evals: Is the number of evaluations necessary to obtain the best-found individual in a run.

Mean Evals: Is the mean number of evaluations necessary to obtain the best-found individual throughout all runs.

Values of some of these performance variables, obtained under MCMP-SRI, are listed in the following tables. Columns one to three indicate the instance number, the upper bound and the heuristic providing that upper bound, respectively, the remaining columns indicate the performance variable values. At the bottom of the tables, average, minimum and maximum *Hit ratio*, *Avg Ebest* and *Avg Evals* values, are indicated.

Inst	Upper Bound	Provided by	Best	Max Best	Mean Best	Mean Gbest	Hit Ratio	Mean Ebest	Mean Evals
1	13.05	**EDD**	11.98	11.98	11.98	34.00	1.00	-8.20	174420
6	88.55	**HDS**	73.15	74.08	73.34	117.90	1.00	-17.18	604827
11	203.08	**HDS**	191.30	192.43	191.56	239.80	1.00	-5.67	1230174
19	542.85	**SPT**	509.25	514.07	511.25	377.60	1.00	-5.82	1937088
21	525.43	**SPT**	522.50	522.60	522.53	386.60	1.00	-0.55	1983258
26	0.40	**EDD**	0.40	0.40	0.40	22.60	1.00	0.00	115938
31	84.95	**RM**	71.32	71.88	71.43	230.70	1.00	-15.91	1183491
36	199.88	**HDS**	183.18	185.90	184.15	348.50	1.00	-7.87	1787805
41	400.70	**HDS**	374.50	378.10	376.76	359.20	1.00	-5.98	1842696
46	375.33	**SPT**	369.35	369.58	369.38	397.60	1.00	-1.58	2039688
51	0.00	**EDD**	0.00	0.00	0.00	34.10	1.00	0.00	174933
56	30.03	**RM**	16.17	17.00	16.28	198.00	1.00	-45.78	1015740
61	171.33	**HDS**	150.63	153.95	152.27	380.90	1.00	-11.12	1954017
66	455.78	**SPT**	395.90	396.00	395.95	431.90	1.00	-13.13	2215647
71	469.43	**SPT**	449.23	449.30	449.24	380.80	1.00	-4.30	1953504
76	0.00	**EDD**	0.00	0.00	0.00	5.20	1.00	0.00	26676
81	4.85	**EDD**	3.20	3.45	3.27	217.00	1.00	-32.52	1113210
86	127.35	**EDD**	82.55	85.07	83.29	388.10	1.00	-34.60	1990953
91	383.68	**RM**	329.98	330.63	330.14	407.30	1.00	-13.95	2089449
96	658.40	**SPT**	639.65	639.83	639.67	412.90	1.00	-2.84	2118177
101	0.00	**EDD**	0.00	0.00	0.00	5.50	1.00	0.00	28215
106	0.00	**EDD**	0.00	0.00	0.00	127.80	1.00	0.00	655614
111	275.23	**RM**	210.80	213.25	211.69	402.00	1.00	-23.09	2062260
116	319.13	**RM**	242.90	244.85	243.85	463.70	1.00	-23.59	2378781
121	598.78	**SPT**	576.57	576.68	576.60	448.40	1.00	-3.70	2300292
					Avg	**272.72**	**1.00**	**-11.10**	**1399074**
					Min	**5.20**	**1.00**	**-45.78**	**26676**
					Max	**463.70**	**1.00**	**0.00**	**2378781**

Table 2. MCMP-SRI. Values of the performance variables for the Average Tardiness problem

From table 2 the following observations can be done:

MCMP-SRI outperforms all other heuristics improving the upper bounds except when EDD is optimal (reaching the same optimal value), which is the case for instances 26, 51, 76, 101 and 106. Recall that for this problem EDD provides an optimal schedule when the total tardiness is zero (and consequently average tardiness is also zero) or when EDD produces one tardy job [11]. *Mean Ebest* ranks from 0.0 to –45.78% with a global average value of –11.10%. Consequently, the *Hit Ratio* is 1 for each instance. These best values are obtained through a number of generations *Gbest* that goes

from 5 to 474 (273 in average) which requires a number of evaluations *Mean Evals* ranking from 26,676 to 2,378,781 (1,399,074 in average).

A similar study was done for each MCMP-SRSI variant. In what follows we present tables summarizing results of mean Ebest and mean Evals for each method.

Instance	MCMP					
	SRI	**SRSI-E**	**SRSI-E-N**	**SRSI-H**	**SRSI-H-E**	**SRSI-H-E-N**
1	-8.20	-8.20	-8.20	-8.20	-8.20	-8.20
6	**-17.18**	-16.72	-16.68	-17.09	-16.34	-17.14
11	-5.67	-5.42	-5.47	-5.74	**-5.80**	-5.24
19	-5.82	-5.94	-6.03	-5.74	**-6.19**	**-6.19**
21	-0.55	**-0.56**	**-0.56**	**-0.56**	**-0.56**	**-0.56**
26	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
31	-15.91	-15.72	**-16.04**	-15.98	-15.91	**-16.04**
36	-7.87	-8.23	-8.28	-8.08	**-8.36**	**-8.36**
41	-5.98	-5.75	-5.81	-5.81	**-6.55**	-5.99
46	-1.58	**-1.59**	**-1.59**	**-1.59**	**-1.59**	**-1.59**
51	0.00	0.00	0.00	0.00	0.00	0.00
56	-45.78	-37.95	**-46.14**	-46.05	**-46.14**	**-46.14**
61	-11.12	-11.71	-11.61	-11.83	**-12.16**	-12.15
66	**-13.13**	**-13.14**	**-13.14**	-13.08	**-13.14**	**-13.14**
71	**-4.30**	**-4.30**	**-4.30**	**-4.30**	**-4.30**	**-4.30**
76	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
81	-32.52	**-34.02**	**-34.02**	**-34.02**	**-34.02**	**-34.02**
86	-34.60	**-35.40**	-35.38	-35.03	-35.30	-35.21
91	-13.95	**-14.00**	**-14.00**	-13.78	**-14.00**	**-14.00**
96	-2.84	**-2.85**	**-2.85**	**-2.85**	**-2.85**	**-2.85**
101	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
106	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
111	-23.09	-23.27	-23.25	-22.73	**-23.41**	**-23.41**
116	-23.59	**-23.89**	-23.87	-23.41	**-23.89**	-23.87
121	-3.70	**-3.71**	**-3.71**	**-3.71**	**-3.71**	**-3.71**
Avg	**-11.10**	**-10.89**	**-11.24**	**-11.18**	**-11.30**	**-11.28**
Min	**-45.78**	**-37.95**	**-46.14**	**-46.05**	**-46.14**	**-46.14**
Max	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

Table3: Mean Ebest values

Results in table 3 indicate that all MCMP evolutionary algorithms outperform conventional heuristics showing improvements that range from 0 to 46% and an average value of about 11%. Although their performance is similar, SRSI-E shows the lower and SRSI-H-E-N the higher performance, respectively.

Instance	MCMP					
	SRI	SRSI-E	SRSI-E-N	SRSI-H	SRSI-H-E	SRSI-H-E-N
1	174420	170829	127737	141588	8208	7695
6	604827	234954	230337	349353	15903	80028
11	1230174	574560	451953	540189	26163	47709
19	1937088	533520	563274	795150	75924	97983
21	1983258	409887	354996	399627	24111	90801
26	115938	127224	41040	12312	39501	6669
31	1183491	349866	339093	370899	62073	32832
36	1787805	557118	538650	646380	42066	47196
41	1842696	407835	376542	559683	54378	224694
46	2039688	477090	418608	451953	28215	83619
51	174933	212382	180576	5130	5130	5130
56	1015740	279072	430920	377568	12312	32319
61	1954017	675108	739746	807975	68742	88236
66	2215647	552501	573534	903906	71820	88749
71	1953504	407322	381159	546858	37449	52839
76	26676	33345	29241	5130	5130	5130
81	1113210	471447	384237	28728	6669	35397
86	1990953	630990	601749	698193	84132	251370
91	2089449	619704	730512	933660	69255	111834
96	2118177	444258	428355	738720	38988	55404
101	28215	29241	31806	5130	5130	6156
106	655614	286767	310365	5130	6669	6669
111	2062260	441180	505818	938277	44118	97983
116	2378781	641250	587898	943920	83106	86184
121	2300292	471447	435024	840294	38475	39501
Avg	**1399074**	**401556**	**391727**	**481830**	**38147**	**67285**
Min	**26676**	**29241**	**29241**	**5130**	**5130**	**5130**
Max	**2378781**	**675108**	**739746**	**943920**	**84132**	**251370**

Table4: Mean Eval values

Table 4 indicates the mean number of evaluations necessary to obtain the results shown in table 3. Here, we can observe that MCMP-SRI (without knowledge insertion) is the most costly algorithm needing 1,400,000 evaluations in average. All other variants including some kind of knowledge reduce the number of evaluation in a range that goes from 65.5% (SRSI-H) to 97.3% (SRSI-H-E).

5. Conclusions

The scheduling problem of minimizing Average Tardiness in a single machine environment, is a difficult problem by itself and some conventional heuristics were developed to provide optimal or quasi-optimal solutions. In this work two of the latest multirecombined evolutionary algorithms were contrasted against the most common, rapid, and good heuristics for the problem.

Evolutionary algorithms are robust search algorithms in the sense that they provide good solutions to a broad class of problems which otherwise are computationally intractable. To improve EAs performance, multi-recombined EAs allow multiple interchange of genetic material among multiple parents (MCMP). To ameliorate the search process, by means of a better balance between

exploration and exploitation, the concept of the stud and the random immigrants was inserted in MCMP-SRI.

Nevertheless the robustness of EAs has as a drawback the kind of search process they perform: a blind search that slightly addressed by the relative fitness of the solutions, completely ignores the nature of the problem. In order to improve their performance we decided to insert problem-specific-knowledge by recombining internal or external seeds in the evolutionary process.

Results indicate that:

- Both multirecombined EAs produced solutions of higher quality (11% in average) than those achieved by typical heuristics.

- MCMP-SRSI variants outperform the former MCMP-SRI. In particular their superiority is strongly related to speed of convergence, perceptible in a reduction of the number of evaluations that ranges from 65.5% to 97.3%.

Further work will be dedicated to find alternative ways to guide the evolutionary search for different scheduling problems.

6. Acknowledgements

We acknowledge the co-operation of the LIDIC for providing new ideas and constructive criticisms. Also to the Universidad Nacional de San Luis, the Universidad Nacional de La Patagonia Austral, and the ANPCYT from which we receive continuous support.

7. References

[1] Beasley J.E. “Common Due Date Scheduling”, OR Library, http://mscmga.ms.ic.ac.uk/

[2] Crauwels H.A.J., Potts C.N. and Van Wassenhove L.N. “Local search heuristics for the single machine total weighted tardiness scheduling problem”, *Informs Journal on Computing* 10, 341-350. 1998.

[3] Chen T. and Gupta M., “Survey of scheduling research involving due date determination decision”, *European Journal of Operational Research*, vol 38, pp. 156-166, 1989.

[4] Eiben A.E., Raué P.E., and Ruttkay Z., “Genetic algorithms with multi-parent recombination”, *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, Springer-Verlag, 1994, number 866 in LNCS, pp. 78-87.

[5] Eiben A.E., Van Kemenade C.H.M., and Kok J.N., “Orgy in the computer: Multi-parent reproduction in genetic algorithms”. *Proceedings of* the *3rd European Conference on Artificial Life*, Springer-Verlag, 1995, number 929 in LNAI, pages 934-945.

[6] Eiben A.E. and. Bäck Th., “An empirical investigation of multi-parent recombination operators in evolution strategies”. *Evolutionary Computation*, 5(3):347-365, 1997.

[7] Esquivel S., Leiva A., Gallard R., “Multiple Crossover per Couple in Genetic Algorithms”, *Proceedings of* the *Fourth IEEE Conference on Evolutionary Computation (ICEC'97)*, Indianapolis, USA, April 1997, pp 103-106.

[8] Esquivel S., Leiva A., Gallard R., “Couple Fitness Based Selection with Multiple Crossover per Couple in Genetic Algorithms“. *Proceedings of* the *International Symposium on Engineering of Intelligent Systems (EIS´98)*, La Laguna, Tenerife, Spain, February 1998, pp 235-241.

[9] Esquivel S., Leiva H., Gallard R., "Multiple crossovers between multiple parents to improve search *in evolutionary algorithms", Proceedings* of the *Congress on Evolutionary Computation (IEEE)*. Washington DC, 1999, pp 1589-1594.
[10] Michalewicz M., "*Genetic Algorithms + Data Structures = Evolution Programs"*. Third revised edition, Springer, 1996.
[11] Morton T., Pentico D.*, "Heuristic scheduling systems"*, Wiley series in Engineering and technology management. John Wiley and Sons, INC, 1993.
[12] Pandolfi D., Vilanova G., M. De San Pedro, A. Villagra, "Multirecombining studs and immigrants in evolutionary algorithm to face earliness-tardiness scheduling problems". *Proceedings of the International Conference in Soft Computing*. University of Paisley, Scotland, U.K., June2001, pp.138
[13] Pandolfi D., De San Pedro M., Villagra A., Vilanova G., Gallard R.- "Studs mating immigrants in evolutionary algorithm to solve the earliness-tardiness scheduling problem" . *In Cybernetics and Systems of Taylor and Francis Journal*, Vol. 33 Nro. 4, pp 391-400 (U.K.) June 2002.
[14] Pandolfi D. De San Pedro M. M., Villagra A., Vilanova G., R. Gallard – "Multirecombining random and seed immigrants in evolutionary algorithms to solve W-T scheduling problems"- *In proceedings of CSITeA02*, pp 133-138, Iguazu Falls, June 2002, Brazil.
[15] Pinedo M., "*Scheduling: Theory, Algorithms and System*." First edition Prentice Hall, 1995.
[16] Rachamadugu R.V., Morton T.E., "Myopic heuristics for the single machine weighted tardiness problem". *GSIA*, Carnigie Mellon University, Pittsburgh, PA. 1982., Working paper 30-82-83.
[17] Reeves C., "A genetic algorithm for flow shop sequencing", *Computers and Operations Research*, vol 22, pp5-13, 1995.
[18] Tsujimura Y., Gen M., Kubota E.: "Flow shop scheduling with fuzzy processing time using genetic algorithms". *The 11th Fuzzy Systems Symposium,* Okinawa,. 1995,.pp 248-252.