A HYBRID EVOLUTIONARY ALGORITHM: MULTIRECOMBINATION WITH PRIORITY RULE BASE REPRESENTATION FOR THE JOB SHOP SCHEDULING PROBLEM

Salto C., Minetti G., Alfonso H.
Proyecto UNLPAM-09/F009[1]
Departamento de Informática - Facultad de Ingeniería
Universidad Nacional de La Pampa
Calle 110 esq. 9
(6360) General Pico – La Pampa – Rep. Argentina
e-mail: {minettig,saltoc,alfonsoh }@ing.unlpam.edu.ar
Phone: (02302)422780/422372, Ext. 6302

Gallard R.
Proyecto UNSL-338403[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

A variety of optimization problems in fields such as production operations in manufacturing industry, parallel and distributed systems, logistics and traffic can be summarized within the general class of scheduling problems. A common feature of this problems is that they belong to the class of NP-complete problems, which means that no deterministic algorithm is known yet for solving them in polynomial time.

The major advantage of evolutionary techniques resides in their ability of providing good solutions to extremely complex problems in reasonable time.
This work introduces MCMP-PRB to face the Job Shop Scheduling Problem (JSSP). Enhancements include a multiplicity feature (MCMP) and a further hybridization with a conventional heuristic know as the priority dispatching rule.

Keywords: evolutionary algorithms, chromosome representation, multiplicity, scheduling, optimization.

---

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

1. INTRODUCTION

Due to its complexity [12] and reflecting the industrial relevance of this application domain, a variety of evolutionary schedulers based on genetic algorithms have been reported in the literature in the past [1, 3, 10, 13, 14, 15, 16, 17, 20, 21]. In general, the task of scheduling is the allocation of jobs over time when limited resources are available, where a number of objectives should be optimized, and several constraints must be satisfied. A job shop can be seen as a multi-operation model where jobs follow fixed routes, but not necessarily the same for each job. Here there exists a facility that produces goods according to specified production plans under several domain-dependent constraints. Job Shop Scheduling (JSS) attempts to provide optimal schedules. Common variables to optimize are total completion time (makespan), machine idleness, lateness, earliness and total weighted completion time. According to this variables different objectives can be devised.

The model considered here assumes that the system consists of m different machines and n jobs. Only one job may execute on a machine at a time. All schedules and jobs are non-preemptive. Jobs can have distinct priorities and all of them are available at production initiating time. Each job visits all machines, only once, following a predetermined sequence of machines, called a route. Consequently a job can be seen as composed by various steps, called operations. Our problem is to find a schedule of minimum makespan.

An appropriate representation associated with a set of genetic operators is of utmost importance when designing an evolutionary algorithm for the JSSP. In this way the individuals created during the evolutionary process will produce feasible schedules.

In [2] a genetic algorithm with priority rule based representation is proposed. Here a chromosome is encoded as a sequence of dispatching rules for job assignments and a schedule is built with a priority dispatching heuristic based on the sequence of dispatching rules. The genetic algorithms are used here to evolve those chromosomes improving the sequences of dispatching rules.

One of the latter contributions in the theoretical field of Evolutionary Computation known as *the multiplicity feature* is related to new proposed multi-recombination methods:

- MCPC: Multiple Crossovers per Couple which reinforces the exploitation of features of previously found (good) solutions.
- MCMP: Multiple Crossovers on Multiple Parents [6,7,8,9] which provides a balance in exploitation and exploration because the searching space is efficiently exploited (by the multiple application of crossovers) and explored (by a greater number of samples provided by multiple parents).

This novel approach was successfully applied to single and multicriteria optimization. Recently it was also applied to JSSP with decoder representation [18]. It was shown that a greater number of crossovers for a given number of parents provides better results.

This presentation discusses the use of multiple crossovers on multiple parents on a priority rule based representation (MCMP-PRB), details of implementation and results are also shown.

2. PRIORITY DISPATCHING RULE HEURISTIC AND MCMP-PRB

Priority dispatching rule heuristics are frequently applied to solve scheduling problems due to their ease of implementation and low time complexity. Giffler and Thompson´s algorithms can be considered as the basis of priority rule based heuristics [12, 19]. The main problem is to determine an effective priority rule. These heuristics are greedy ones in the sense that at each iteration an operation is selected between the available ones and scheduled as soon as possible in the respective machine. The choice of the next operation to be scheduled is done following a priority function (rule). Several priority rules have been proposed for the job shop scheduling:

- ✓ **SPT,** *Shortest Processing Time,* select an operation with the shortest processing time
- ✓ **LPT,** *Longest Processing Time,* select an operation with the longest processing time
- ✓ **MWR,** *Most Work Remaining,* select an operation for the job with the most total remaining processing time
- ✓ **LWR,** *Least Work Remaining,* select an operation for the job with the least total remaining processing time
- ✓ **MOR,** *Most Operations Remaining,* select an operation for the job with the greatest number of operations remaining.
- ✓ **LOR,** *Least Operations Remaining,* select an operation for the job with the smallest number of operations remaining
- ✓ **EDD,** *Earliest Due Date,* select a job with the earliest due date
- ✓ **FCSF,** *First Come First Served,* select the first operation in the queue of jobs for the same machines
- ✓ **RND,** *Random*, select an operation at random

Dorndorf and Pesch [2], proposed an evolutionary algorithms for the job-shop scheduling problem. That algorithm uses the priority-rule-based representation, and the well-known Giffler and Thompson algorithm was incorporated into the evolutionary algorithm. Here, the evolutionary algorithm is used to evolve a sequence of priority dispatching rules and the Giffler and Thompson algorithm is used to deduce a schedule from the encoding of priority dispatching rules.

For an n job and m machine problem a chromosome is a string of $n \times m$ entries ($p_1, p_2, \ldots, p_{n\,m}$). Each entry represents one of the prespecified rules. The entry in the i^{th} position indicates that a conflict in the i^{th} iteration of the Giffler and Thompson algorithm should be resolved using the priority rule p_i. Ties are broken by a random choice.

Let

PS_t	=	a partial schedule containing t scheduled operations.
S_t	=	the set of schedulable operations at iteration t, corresponding to PS_t.
σ_i	=	the earliest time at which operation $i \in S_t$ could be started.
ϕ_i	=	the earliest time at which operation $i \in S_t$ could be completed.
C_t	=	the set of conflicting operations in iteration t.

The procedure *builder* deduces a schedule for each chromosome ($p_1, p_2, \ldots, p_{n\,m}$):

Procedure: builder (Deduce a schedule for Priority-Rule-Based Encoding)

1. Let $t = 1$ and begin with PS_t as the null partial schedule and let S_t include all operations with no predecessors.
2. Determine $\phi^*_t = \min_{i \in S_t} \{\phi_i\}$ and the machine m^* on which ϕ^*_t could be realized. If more than one such machine exists, the tie is broken by a random choice.
3. Form a conflicting set C_t which includes all operations $i \in S_t$ with $\sigma_i < \phi^*_t$ that require machine m^*. Select one operation from C_t by the priority rule p_t and add this operation to PS_t as early as possible, thus creating a new partial schedule PS_{t+1}. If more than one operation exists according to the priority rule p_t, the tie is broken by a random choice.
4. Update PS_{t+1} by removing the selected operation from S_t and adding the direct successor of the operation to S_t. Increment t by one.
5. Return to step 2 until a complete schedule is generated.

Figure 1 depicts the structure of the proposed algorithm.

```
Procedure: MCMP-PRB
  begin
      k ← 0;
      initialize P(k);
      builder(P(k)); //generate a schedule and assign a fitness value
      while not (termination condition) do
          recombine using MCMP on P(k)  and mutate to produce C(k);
          builder(C(k)); // generate a schedule and assign a fitness value
          select P(k+1) from P(k) and C(k);
          k ← k + 1;
      end
end
```

Figure 1. Evolutionary Algorithm with the *builder* procedure.

where builder(*P(k)*) is the procedure to deduce a schedule for Priority-Rule-Based Encoding.

To clarify an example of a three-job three-machine problem is presented in the table 1.

Processing Time				**Machine Sequence**			
	Operations				Operations		
Job	1	2	3	Job	1	2	3
j_1	1	5	3	j_1	m_1	m_2	m_3
j_2	4	6	1	j_2	m_1	m_3	m_2
j_3	3	2	3	j_3	m_2	m_1	m_3

Table 1. Example of Three-Job Three-Machine Problem

Assume the following rules: SPT, LPT, MWR and LWR. Consider the following chromosome [2 1 3 3 4 4 1 1 3], where 1 stands for rule SPT, 2 for rule LPT, 3 for rule MWR and 4 for LWR. The operations are indicated by o_{jom} (where j indicates the job, o indicates the operation of job j, and m indicates the machine to perform this operation). At the initial step, we have

$$S_1 = \{ o_{111}, o_{211}, o_{312} \}$$
$$\phi_1^* = \min \{1, 4, 3\} = 1$$
$$m^* = 1$$
$$C_1 = \{ o_{111}, o_{211} \}$$

Now operations o_{111} and o_{211} compete for machine m_1. Because the first gene in the given chromosome is 2 (which means LPT priority rule), operation 221 is scheduled on machine m_1. After updating the data, we have

$$S_2 = \{ o_{111}, o_{223}, o_{312} \}$$
$$\phi_2^* = \min \{5, 10, 3\} = 3$$
$$m^* = 2$$
$$C_2 = \{ o_{312} \}$$

Operation o_{312} is scheduled on m_2. After updating the data, we have

$S_3 = \{ o_{111}, o_{223}, o_{321} \}$
$\phi_3^* = \min \{5,10, 9\} = 5$
$m^* = 1$
$C_3 = \{ o_{111}, o_{321} \}$

Operations o_{111} y o_{321} compete for machine m_1. Because the third gene in the given chromosome is 3 (which means MWR priority rule), operation o_{111} is scheduled on machine m_1. These steps are repeated until a complete sechedule is deduced from the given chromosome.

3. MULTIPLICITY OF CROSSOVERS AND PARENTS (MCMP)

In his multiparent approach Eiben [5] used, initially, three *scanning crossover* (SX) methods, which essentially take genes from parents to contribute in building the offspring. The SX general mechanism assigns a marker to each parent and to the offspring. The offspring's marker traverses all of its positions from left to right. At each step parent's markers are updated each time a gene is selected. The main characteristics of all gene-scanning procedures are the update marker mechanisms and the way to choose a marked gene from the parents. Only one offspring is generated. In our work we choose *uniform scanning crossover* (USX). This method is a natural extension of *uniform crossover*. Consequently, each gene in the child is provided from any of the corresponding genes in the parents with equal probability.

Multiple crossovers on multiple parents (MCMP), the method used here, allows multiple recombination of multiple parents under any of the SX variants, expecting that exploitation and exploration of the problem space be adequately balanced. MCMP provides a means to exploit good features of more than two parents selected according to their fitness by repeatedly applying one of the SX variants: a number n_1 of crossovers is applied on a number n_2 of selected parents. From the n_1 produced offspring a number n_3 of them are selected, according to some criterion, to be inserted in the next generation.

4. EXPERIMENTS AND RESULTS

An EA with rule based representation and MCMP was applied on a set of selected JSSP instances. For each instance a series of ten runs was performed. Experiments corresponded to different (n_1, n_2) combinations, SX and multi-recombination methods. The evolutionary algorithms used proportional selection for mating. Elitism to retain the best valued individual was implemented. Population size was fixed at 50 individuals. For insertion in the next generation the best child was chosen ($n_3 = 1$). Number of crossovers and parents were set to: $1 \leq n_1 \leq 4$ and $3 \leq n_2 \leq 7$, respectively. For mutation a *big-creep* operator, replacing the gene value by another in the permitted range was used. The maximum number of generations was fixed at 500 and probabilities for crossover and mutation were fixed at 0.8 and 0.01, respectively. These values were determined as the best combination of probabilities after many initial trials. For those experiments the following priority rules were used: SPT, LPT, MWR, LWR, MOR, LOR, FCSF and RND. Seven instances [14], with known optimal makespan were used. They were:

Instance	Size	Optimum
abz6	10×10	943
la01	10×5	666
la06	15×5	926
la12	20×5	1039
la15	20×5	1207
ft06	6×6	55
ft10	10×10	930

Tabla 2: Instances

The following relevant performance variables were chosen:

Ebest = (Abs(*opt_val* - best value)/*opt_val*)100

It is the percentile error of the best found individual when compared with the known, or estimated, optimum value *opt_val*. It gives us a measure of how far we are from that *opt_val*.

Epop = (Abs(*opt_val*- pop mean fitness)/*opt_val*)100

It is the percentile error of the average individual (population mean fitness) when compared with *opt_val*. It tell us how far the mean fitness is from that *opt_val*.

Tables 3, 4, 5, and 6 show in detail results for the *la06* instance.

	3 Parents	4 Parents	5 Parents	6 Parents	7 Parents
1 Cross	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
2 Cross	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
3 Cross	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
4 Cross	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000

Table 3. *Ebest* values for each (n_1, n_2) combination

	3 Parents	4 Parents	5 Parents	6 Parents	7 Parents
1 Cross	0.032397408	0.05399568	0.00000000	0.00000000	0.00000000
2 Cross	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
3 Cross	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
4 Cross	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000

Table 4. Mean *Ebest* values for each (n_1, n_2) combination.

In table 3 we can observe that the optimum value was found in every case. Moreover, table 4 shows that except for $n_1 = 1$ and n_2 between 3 and 4, all runs found the minimum makespan. Even in these worst cases mean percentile error values are very low. This fact demonstrates that for this instance the algorithm has high performance.

	3 Parents	4 Parents	5 Parents	6 Parents	7 Parents
1 Cross	13.79490062	14.88442626	17.38191379	15.12938239	14.76233867
2 Cross	5.35742506	5.87388303	5.66669343	5.15224994	5.56340243
3 Cross	0.92650531	0.83226955	0.45707971	0.64432496	0.55061516
4 Cross	0.17751455	0.08467163	0.08467163	0.17751455	0.08467163

Table 5. *Epop* values for each (n_1, n_2) combination.

	3 Parents	4 Parents	5 Parents	6 Parents	7 Parents
1 Cross	16.67952857	17.9265597	18.20755521	18.05584662	17.31246756
2 Cross	6.95102552	6.81043389	7.15795444	7.63706185	8.39825058
3 Cross	3.41570357	2.96842072	1.79691919	2.54193253	4.36856278
4 Cross	0.60771615	0.83697745	0.85830443	0.61189655	0.74552255

Table 6. Mean *Epop* values for each (n_1, n_2) combination.

Tables 5 and 6 show a regular behaviour when the multiplicity feature is applied. Observe that when we pass from the application of a single crossover to multiple crossovers *Epop* is drastically reduced, indicating that the final population is concentrated around the best found individual. This effect is incremented as long as n_1 is augmented independently of the n_2 value. A similar behaviour was detected on instances *la01*, *la02* and *ft06* were every (n_1, n_2) combination have found the optimal value in at least one opportunity. In particular for *la06* and *la12* when $2 \leq n_1 \leq 4$, always the best value is obtained independently of the number of parents used. Figure 2 summarizes minimum, mean and maximum *Ebest* values for each instance. Here, this behaviour is confirmed for instances *la01*, *la06*, *la12* and *ft06*, while for the remaining instances the optimal value is not reached.

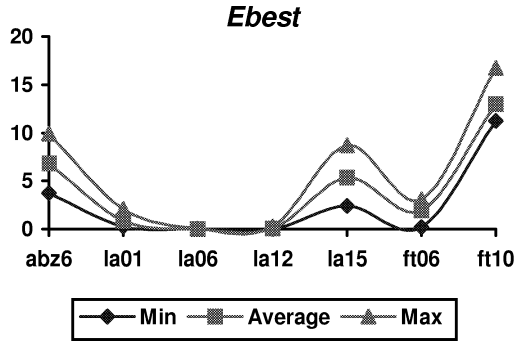

Figure 2. Minimum, Average and Maximum Mean *Ebest* values for each instance of the JSSP.

Figues 3 to 8 summarize *Ebest* and *Epop* mean values for different n_1 values through all instances.

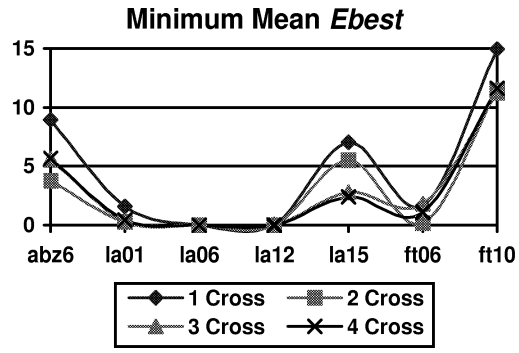

Figure 3. Minimum Mean *Ebest* for different n_1 values.

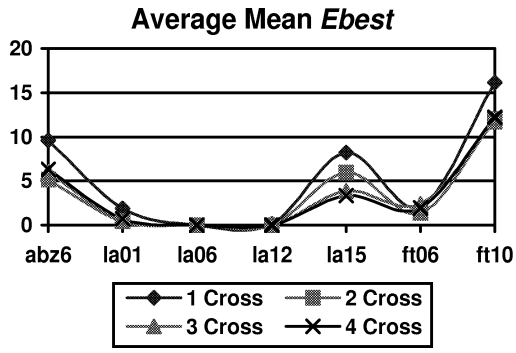

Figure 4. Average Mean *Ebest* for different n_1 values

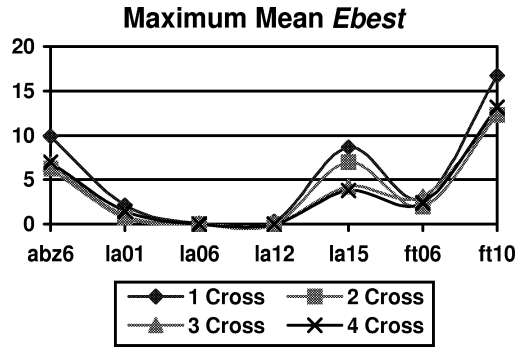

Figure 5. Maximum Mean *Ebest* for different n_1 values

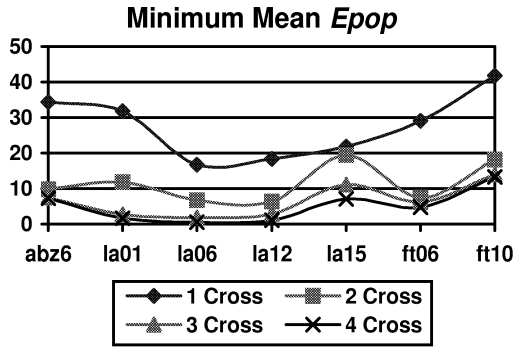

Figure 6. Minimum Mean *Epop* for different n_1 values

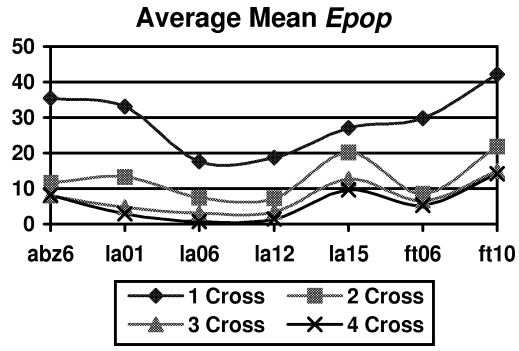

Figure 7. Average Mean *Epop* for different n_1 values

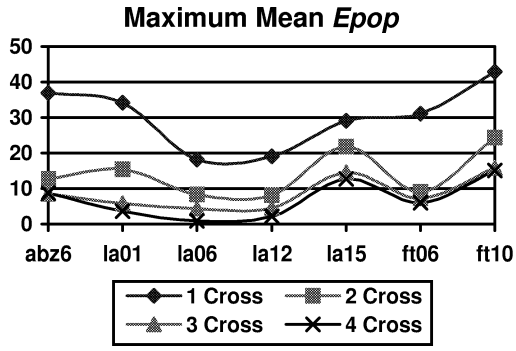

Figure 8. Maximum Mean *Epop* for different n_1 values

In figures 3, 4 and 5 it can be in general observed that percentile error of the best found individuals decrease as long as the number n_1 of crossovers is increased.

Figures 6, 7 and 8 indicate a similar behaviour in all instances; a reduction of the percentile error of the average individual in the final population as long as the number n_1 of crossovers is increased. It also can be seen an abrupt decay in this error when we pass from 1 to 2 crossovers and then the reduction persists gradually when n_1 is augmented. It worth to remark that for $n_1 = 1$ *Epop* ranges within very high values, from 17% to 42%. For instances *la01*, *la02* and *ft06* the individuals in the final population are centred around the optimum differing by a negligible value (see figures 3 and 6).

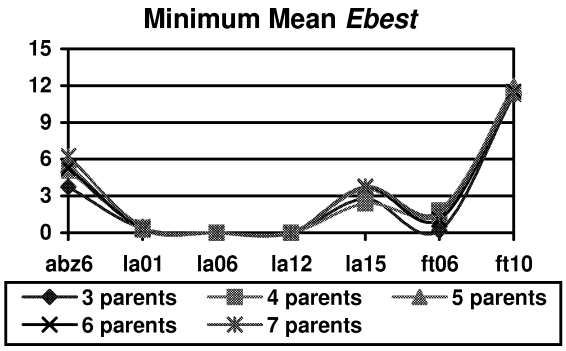

Figure 9. Minimum Mean *Ebest* for different n_2 values

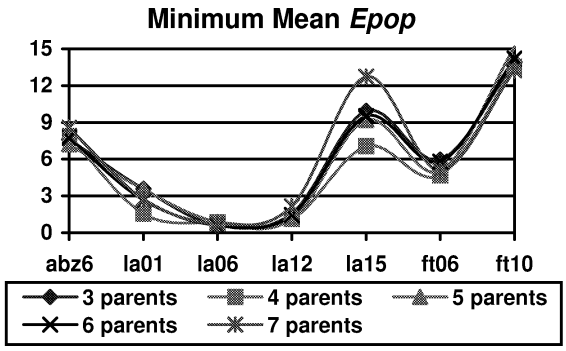

Figure 10. Minimum Mean *Epop* for different n_2 values

From figures 9 and 10 average minimum values of *Ebest* and *Epop* suggest that better performance is achieved when $3 \leq n_2 \leq 5$.

5. CONCLUSIONS

This contribution introduces MCMP-PRB a hybrid multirecombinative evolutionary algorithm using both multiple crossovers on multiple parents and priority dispatching rules, a conventional heuristic, to solve the JSSP. By means of the rule based representation MCMP can be applied creating valid offspring after each recombination operation. Repair algorithms or penalty functions are no more needed, as under other evolutionary approaches. The novel approach was tested on a set of instances of varied complexity. After this preliminary experiments we can conclude that MCMP-PRB provide optimal solutions on most of them. It is remarkable also the fact that the composition of the final population guarantees the supply of many near optimal solutions which can help facing changes in system dynamics. As a consequence of this promising results future work is addressed to improve the algorithm by self-adaptation to automatically apply the suitable (n_1, n_2) combination on each step of the evolutionary process.

6. ACKNOWLEDGEMENTS

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

7. BIBLIOGRAPHY

[1]. Bruns, R., *Direct chromosome representation and advanced genetic operators for production scheduling*. S. Forrest, editor. Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA93). Morgan Kaufmann Publishers, San Mateo, CA, pp. 352-359, 1993.

[2]. Dorndorf, U., and Pesch, E., *Evolution based learning in a job shop scheduling environment*, Computers and Operations Research, vol. 22, pp. 25-40, 1995.

[3]. Easton F. F and Mansour N. *A distributed genetic algorithm for employee staffing and scheduling problems*. S. Forrest, editor. Proceedings of the Fifth International Conference on

Genetic Algorithms (ICGA93). Morgan Kaufmann Publishers, San Mateo, CA, pp. 360-367, 1993.

[4]. Eiben, A.E., *A Method for Designing Decision Support Systems for Operational Planning*, PhD Thesis, Eindohoven University of Technology, 1991.

[5]. Eiben, A.E and van Kemenade, C.H.M, *Diagonal crossover in Genetic Algorithms for numerical Optimization*, Journal of Control and Cybernetics, vol. 26, no. 3, pp. 447-465, 1995.

[6]. Esquivel S., Leiva A., and Gallard R., *Multiple crossover per couple in genetic algorithms*. Proceedings of the 4th IEEE International Conference on Evolutionary Computation (ICEC'97), pp. 103-106, Indianapolis, USA, 1997.

[7]. Esquivel S., Leiva A., and 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, vol. 1, pp. 235-241, ed. E.Alpaydin. Publishesd by ICSC Academic Press, Canada/Switzerland, 1998.

[8]. Esquivel S., Leiva H., and.Gallard R., *Multiplicity in genetic algorithms to face multicriteria optimization,* Proceedings of the 1999 Congress on Evolutionary Computation (IEEE). Washington DC, 1999.

[9]. Esquivel S., Leiva H.,.and Gallard R., *Multiple crossovers between multiple parents to improve search in evolutionary algorithms,* Proceedings of the 1999 Congress on Evolutionary Computation (IEEE). Washington DC, pp. 1589-1594, 1999.

[10]. Fang H.-L., Ross P., and Corne D. *A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems*. S. Forrest, editor. Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA93). Morgan Kaufmann Publishers, San Mateo, CA, pp. 375-382, 1993.

[11]. Gen, M., Runwei, C., *Genetic Algorithms and Engineering Design*, Wiley-Interscience Publication John Wiley & Sons, Inc, 1997

[12]. Giffler, B., and Thompson, G., *Algorithms for solving production scheduling problems*, Operations Research, vol. 8, no. 4, pp. 487-503, 1960.

[13]. Husbands P., Mill F., and Warrington S. *Genetic algorithms, production plan optimisation, and scheduling*. In Schwefel and Maenner PPSN91, pp. 80-84, 1991.

[14]. Lawrence, S., *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques*, (Supplement), Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

[15]. Lee I., Sikora R., and Shaw M. J. *Joint lot sizing and sequencing with genetic algorithms for scheduling: Evolving the chromosome structure*. S. Forrest, editor. Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA93). Morgan Kaufmann Publishers, San Mateo, CA, pp. 383-389, 1993.

[16]. Michalewicz, M., *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, third revised edition, 1996.

[17]. Nakano R. and Yamada T. *Conventional genetic algorithms for job shop problems*. In R. K. Belew and L. B. Booker, editors, Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, CA, pp. 474-479, 1991.

[18]. Salto, C., Alfonso, H., and Gallard, R., *Multiplicity and incest prevention in evolutionary algorithms to deal with the job shop problem*, Proceedings of the Second ICSC Symposium on Engineering of Intelligent Systems, University of Paisley, Scotland, pp. 451-457, 2000.

[19]. Storer, R., W, S., and Vaccari, R., *New search spaces for sequencing problems with application to job shop scheduling*, Management Science, vol. 38, no. 10, pp. 1495-1510, 1992.

[20]. Syswerda G. *Schedule optimization using genetic algorithms*. In L. Davis, editor, Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, pp. 332-349, 1991.

[21]. Yamada T. and Nakano R. *A genetic algorithm applicable to large-scale job-shop problems*. In R. Maenner and B. Manderick, editors, Parallel Problem Solving from Nature 2 , pp. 281-290. Elsevier, Amsterdam, 1992.