MULTIPLICITY OF PARENTS AND CROSSOVERS, THE STUD AND THE NEH HEURISTIC FOR SEARCHING THE OPTIMAL MAKESPAN IN FSSP

ESQUIVEL S.C., ZUPPA F., GALLARD R.

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)[1]
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 - Local 106
5700 - San Luis
E-mail: {esquivel,fede,rgallard}@unsl.edu.ar

Phone: +54 2652 420823
Fax : +54 2652 430224

ABSTRACT

Different evolutionary approaches using genetic algorithms were proposed to solve the Flow Shop Scheduling Problem (FSSP). Variants point to the selection mechanism, genetic operators and the decision to include or not in the initial population an individual generated by some conventional heuristic (Reeves). New trends to enhance evolutionary algorithms for solving the FSSP introduced *multiple-crossovers-per couple* (MCPC) and *multiple-crossovers-on-multiple-parents* (MCMP).

MCMP-S, a multiple-crossovers-on-multiple-parents variant, selects the stud (breeding individual) among the multiple intervening parents and mates it, more than once, with every other parent in a multiple crossover operation. In previous works, two versions of MCMP-S were faced. In the first one (MCMP-SOP), the stud and every other parent were selected from the old population. In the second one (MCMP-SRI), the stud was selected from the old population, and the other parents (random immigrants) were generated randomly.

This paper introduces MCMP-NEH. The idea is to use the NEH heuristic, where the stud mates individuals in the mating pool coming from two sources: random immigrants and NEH-based individuals. These NEH-individuals are produced from randomly chosen individuals of the population and used as the starting points of the NEH heuristic. Experiments were conducted to contrast this novel proposal with a conventional evolutionary algorithm, with the only objective of establishing the improvement degree despite computational effort. Implementation details and a comparison of results for a set of flow shop scheduling instances of distinct complexity, using every evolutionary approach, are shown.

1. INTRODUCTION

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 [12]. In our FSSP model we assumed, that each job is processed on all machines in the same order, each

[1] The Research Group is supported by the Universidad Nacional de San Luis and the ANPCYT (National Agency to Promote Science and Technology). http://www-pr.unsl.edu.ar/proyecto338403//home_page.html.

machine processes a job at a time, and each job is processed in a single machine at a time. The operations are not preemptable and set-up times are included in the processing times. As the flow shop problem is essentially a permutation schedule it is natural for chromosomes to be encoded as permutations. Consequently, adequate genetic operators such as ordered (OX)[1], cyclic (CX)[11], partial mapped (PMX)[10], and one-cut-point (OCPX)[1] crossovers should be used. In opposition to the conventional single crossover per couple approach (SCPC), recent improvements in evolutionary algorithms include a multiplicity feature, which allows multiple recombination on multiple parents (MCMP)[3][4][5][6]. Exploring the abilities of multirecombination this proposal introduces an implementation of the MCMP approach which selects the stud (breeding individual) from the multiple parents pool and mates it in a multiple crossover operation with two kinds of chromosomes. The first ones are randomly created chromosomes, which represents foreign individual and prevent losing genetic diversity. The second ones are individuals chosen randomly from the population and used as the starting points of the NEH heuristic. In this way, we are providing a better genetic diversity and a hybridization or local search (which is represented by the NEH heuristic applied to the evolved chromosomes) through NEH. This presentation discusses details of implementation and compares MCMP-NEH with SCPC when dealing with the flow shop scheduling problem in the search of optimal or near-optimal makespan.

2. THE EVOLUTIONARY MULTICROSSOVER ON MULTIPARENT-NEH APPROACH

Latest trends in Evolutionary Computation point to multirecombination. In Eiben's *multiparent* (MP) approach[2], offspring creation is based on a larger sample from the search space and, thus, a larger diversity is supplied. This can help to avoid premature convergence. Eiben initially used three *scanning crossover* (SX) methods, which essentially take genes from parents to contribute to build the offspring. As defined, SX is not directly applicable to permutations because invalid offspring are created. In our proposal, the SX scanning method is replaced by PMX. Preliminary steps towards that MCMP approach included MCMP-SOP and MCMP-SRI. In the first one, every member of the mating pool was selected from the existing population. Within this pool, the best individual was selected as the stud and it was coupled more than once with every other member of the pool. In MCMP-SRI only the stud is selected from the population and the rest of the individuals are created randomly. From both MCMP-SOP behaved better. MCMP-NEH is very similar to MCMP-SOP and MCMP-SRI, but here, only the stud is selected from the population and the rest of the individuals in the mating pool are created in two ways. Part of them are random immigrants and part of them are NEH-based individuals created from randomly chosen population individuals. The NEH algorithm can be sketched in the following stages:

1. *Rank the jobs by decreasing order of the sums of processing times on all machines.*
2. *Take the first two jobs and schedule them minimizing the makespan.*
3. *For k = 3 to n do: insert the k^{th} job at a place, which minimizes the partial makespan.*

Contrasted with evolutionary and conventional heuristics, NEH has proved to be a good conventional heuristic to solve the FSSP providing a single solution [7]. To provide more than a single solution, instead of ranking the jobs as in the first step, a chromosome chosen from the population is used as it is. The whole process could be seen as applying a local search to that chromosome. In this implementation, we combined both types of chromosomes, along with the stud selected from the population. In order to decide which method is to be used to generate the individuals for mating, a number is randomly generated. After generating n_1 mating individuals (including the stud), multiple crossovers are performed. The crossover points are determined in a random way and the stud is combined with the rest of the selected parents. From that multirecombination, $2*(n_1 -1)$ offspring

are obtained, but only the best one, which is stored in a temporary structure, survives. After completing all the n_2 crossovers, the best individual in the temporary structure is selected for insertion in the next population.

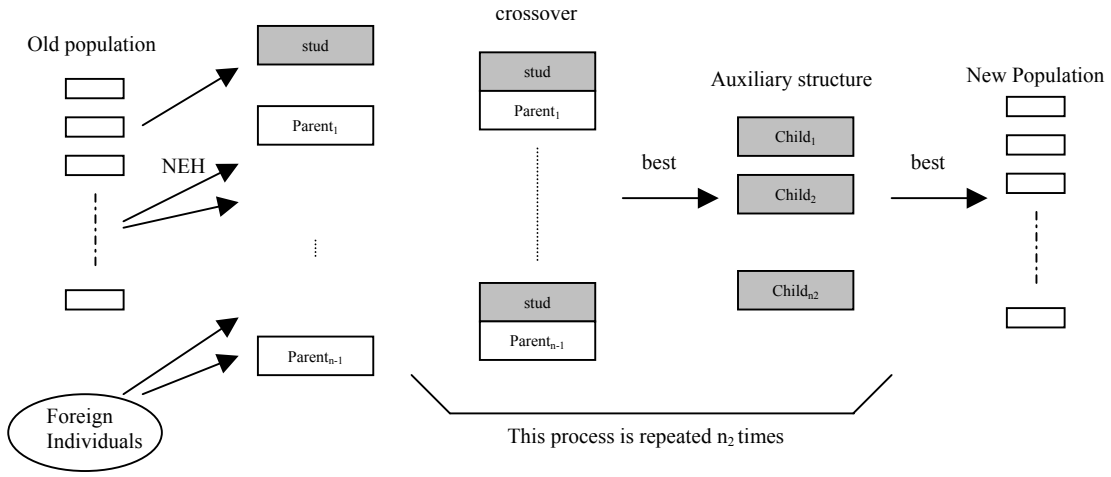

Fig. 1. The multirecombination process in MCMP-NEH.

procedure: MCMP-NEH
// *n* represents the size of the population
// n_1 represents the number of parents to select *aux_child1* array[2*(n_1-1)]
//contain children generated by the crossover operator

while $j \leq n$ **do**
begin
If (random[0,1] < crossover probability) **then**
Begin
Select n_1 parents from *P*(*t*) and build the *mating_pool*
If (random[0,1] < method probability) **then**
Generate chromosome randomly
else
Select the next individual from the mating pool and apply NEH
end if
for *cont=1* **to** n_2
for *i* =2 **to** n_1 **do**
CROSS(mating_pool[1], mating_pool[i]) to form aux_child
Evaluate aux_child1[i]
end for
Sort *aux_child1* // *the best child is in the first position*
Aux_child2[cont] = aux_child1[1];
end for
else
aux_child2[1] = Select one parent randomly from the mating pool
If (random[0,1] < mutation probability) **then**
begin
mutate *aux_child2*[1]
insert *aux_child2*[1] in *New_pop*
end if
end if
end while
end procedure

3. EXPERIMENTAL TESTS AND RESULTS

To contrast the behavior of MCMP-NEH, with the conventional SCPC, we devised five different EAs. All of them have a population size of 100 individual, use proportional selection, a maximum

number of generations fixed at 2000 and probabilities of crossover set to 0.65. Mutation probabilities were set to 0.1 for the SCPC and MCMP-NEH versions. For crossover, PMX was used and for mutation, random exchange mutation (RXM)[10] was used in every version. In the multicrossover-multiparent versions, 6 crossovers were performed on five parents selected to conform the mating pool. The probability used to select the method was 0.5, in these preliminary experiments. Further details are described below:

- SCPC: A conventional evolutionary algorithm using PMX and RMX.
- MCMP-NEH: A multiple-parent, multiple crossovers with the inclusion of the NEH heuristics in every moment of the evolution.

The conventional and evolutionary algorithms were ran on the complete set of the following Taillard's benchmarks[13] : 20x5, 20x10 and 20x20. A series of ten runs was performed for each instance under each algorithm. To compare the algorithms, the following performance variable was 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 on how far the best individual is from that *opt_val*.

When the 10-run series were accomplished, mean and minimum *Ebest* values for each instance were determined and finally average mean and average minimum *Ebest* values were determined over all instances. In the following tables, the first column identifies the corresponding instance, the following columns show the results under each algorithm and the last row gives the average result values.

20 x 5 instances

Instance	SCPC	MCMP NEH
a	1.29	**0**
b	0.34	**0**
c	1.23	0,157
d	1	0,418
e	0.93	0,251
f	1.1	0,929
g	0.97	**0**
h	0.35	**0**
i	1.48	**0**
j	1.29	**0**
Avrg.	0.998	0.1755

Table 1. Mean *Ebest* values for 20x5 instances

Instance	SCPC	MCMP NEH
a	**0**	**0**
b	**0**	**0**
c	**0**	**0**
d	0.3	**0**
e	0.56	-0.081
f	0.5	**0**
g	0.96	**0**
h	0	**0**
i	0.24	**0**
j	**0**	**0**
Avrg.	0.25	**-0.0081**

Table 2. Minimum *Ebest* values for 20x5 instances

Tables 1 and 2 show detailed results for the 20x5 problem size. Table 1 refers to mean *Ebest* values obtained from the ten runs for each instance while table 2 refers to minimum *Ebest* values. These tables show a significant improvement in both mean and minimum *Ebest* values when MCMP-NEH is compared with SCPC. From previous [8], not included in this report, performance enhancements are also observed in the mean *Ebest* values when compared with MCMP-SOP. The corresponding

values are 0.1755% under MCMP-NEH versus 0.807% under MCMP-SOP. It worth to remark that the negative value for minimum *Ebest* value of the *e* instance means that a schedule with a lower makespan for this instance was found, and consequently a new upper bound is available. All other zero values mean that the corresponding known upper bound was reached.

20 x 10 instances

Instance	SCPC	MCMP NEH
a	2.29	0.278
b	2.56	0.82
c	2.27	0.775
d	2.06	0.675
e	2.42	0.366
f	1.85	1.353
g	1.46	0.162
h	2.65	0.728
i	2.09	0.264
j	2.09	0.999
Avrg.	2.174	0.642

Table 3. Mean *Ebest* values for 20x10 instances

Instance	SCPC	MCMP NEH
a	1.7	0.063
b	1.2	0.06
c	0.8	0.334
d	1.16	0.073
e	0.7	**0**
f	0.28	0.215
g	0.6	**0**
h	1.04	0.325
i	1.12	0.063
j	1.38	0.44
Avrg.	0.998	0.1573

Table 4. Minimum *Ebest* values for 20x10 instances

Table 3 and 4 show the averages of the 20x10 instances compared with the SCPC approach. In the Mean *Ebest* values table it can be seen how this approach outperformed the SCPC approach in every instance, averaging 0.642% against 2.174% of the SCPC algorithm. In the Minimum *Ebest* Table, it can be seen that MCMP reach 2 optimum values and 4 times nearly arrived to the optimum (0.0x of *Ebest* means in this size of instance that the makespan reached is 1 unit more than the optimum). The final average of the MCMP-NEH is 0.1573% against nearly 1% of the SCPC algorithm.

20 x 20 instances

Instance	SCPC	MCMP NEH
a	2.12	0.501
b	1.55	0.048
c	1.91	0.804
d	1.69	1.015
e	1.61	0.730
f	1.55	0.315
g	1.83	0.637
h	1.83	0.464
i	1.89	0.174
j	2.94	0.698
Avrg.	1.892	0.538

Table 5. Mean *Ebest* values for 20x20 instances

Instance	SCPC	MCMP NEH
a	1.21	0.261
b	0.9	0.048
c	1.33	0.270
d	0.31	0.731
e	0.78	0.352
f	0.71	**0**
g	1.14	0.218
h	0.63	0.091
i	0.8	**0**
j	1.56	0.092
Avrg.	0.937	0.206

Table 6. Minimum *Ebest* values for 20x20 instances

Table 5 and 6 show the results obtained in the 20 x 20 instances. The MCMP-NEH algorithm again performed very well. In the Mean *Ebest* table it can be seen how while the SCPC approach obtained an Average Mean *Ebest* of 1.892%, the multirecombinative approach reach 0.53%. The improvement is 72%. In the Minimum Ebest table the SCPC approach has an average of 0.937% while the multirecombinative approach has an average of 0.206%. The improvement is 78%.

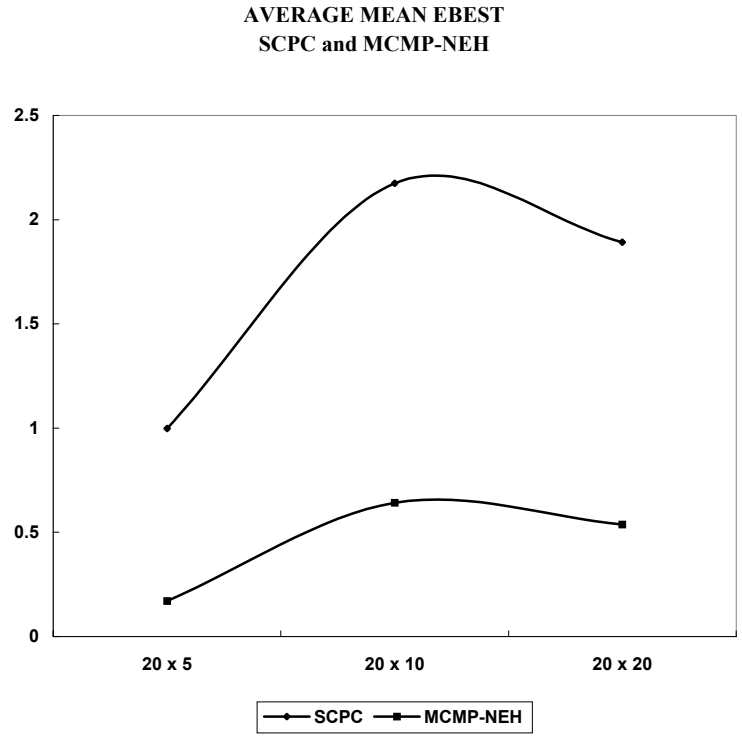

Fig. 2. Average of Mean *Ebest* values over all instances for each problem size under conventional and multirecombined evolutionary approaches.

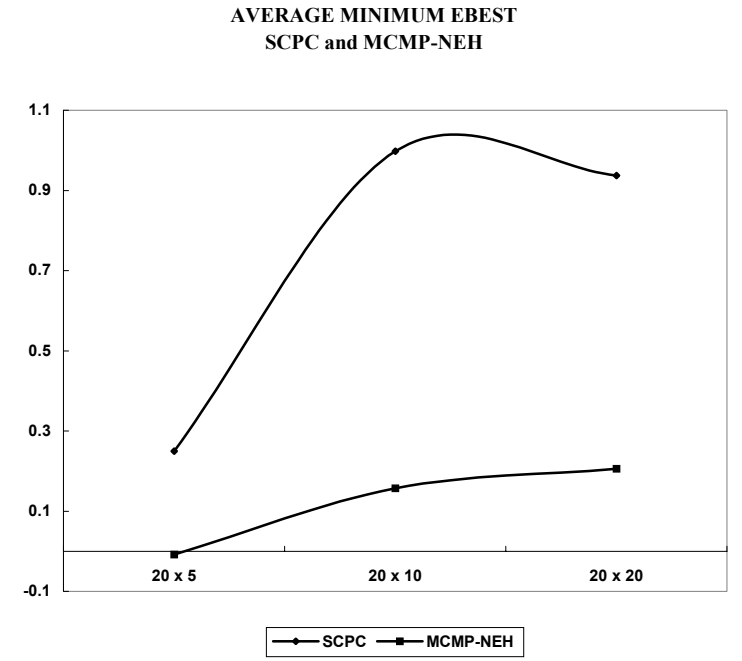

Fig. 3. Average of Minimum *Ebest* values over all instances for each problem size under conventional and multirecombined evolutionary approaches.

Figures 1 and 2 show the performance of the contrasted methods over all problem sizes. In general the multiparent-multicrossover-neh method outperformed the conventional method in both average mean and average minimum Ebest. In the SCPC approach the errors range from 0.998% in the 20x5 instance to 2.174% in the 20x10 instances. In the MCMP-NEH the range was from –0.008% in the 20x5 instances to 0.206% in the 20x20 instances.

4. CONCLUSIONS

Following current trends to solve difficult optimization problems, the Flow Shop Scheduling Problem was faced by means of distinct evolutionary computation approaches. The present contribution contrasted the behaviour of new multiparent-multiplecrossover-neh based approaches with conventional evolutionary algorithms. Quality of results was the only performance metric used in this study. Results obtained on a selected test suite deserve the following observations.
This multirecombinative method outperformed the SCPC approach significantly in every problem size that we tested. Moreover, according to the results, it can be seen that it has a remarkable balance between exploration and explotation. The only problem that was detected is the computational effort required, which slow too much the process. After making a few calculations we realize that the NEH heuristic, which is the slowest heuristic, was performed twice by each individual that was to be inserted in the next population. This adds to 200 each generation and 400000 at the end of the run. Further considerations have to be done, in order to apply NEH fewer times and therefore reduce the time of execution. At the light of these results, research will be oriented to continue searching for further enhanced evolutionary algorithms.

5. ACKNOWLEDGEMENTS

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

BIBLIOGRAPHY

[1] Davis, L., Applying Adaptive Algorithms to Epistatic Domains, Proceedings of the International Joint Conference on Artificial Intelligence, pp. 162-164, 1985. (4)

[2] Eiben A.E., Raué P-E., and Ruttkay Zs., *Genetic algorithms with multi-parent recombination*. In Davidor, H.-P. Schwefel, and R. Männer, editors, Proceedings of the 3rd Conference on Parallel Problem Solving from Nature, number 866 in LNCS, pages 78-87. Springer-Verlag, 1994

[3] Esquivel S., Gallard R. and Michalewicz Z., *MCPC: Another Approach to Crossover in Genetic Algorithms*, Proceeding of Primer Congreso Argentino de Ciencias de la Computación, pp 141 - 150, 1995.

[4] Esquivel S., Leiva A., Gallard R., - *Multiple Crossover per Couple in Genetic Algorithms*, Proceedings of the Fourth IEEE Conference on Evolutionary Computation (ICEC'97), pp 103-106, ISBN 0-7803-3949-5, Indianapolis, USA, April 1997.

[5] Esquivel S., Leiva H., 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.

[6] Esquivel S., Zuppa F., Gallard R., Multirecombinated Evolutionary Algorithms for the Flow Shop Scheduling Problem. PPSN IV. P. 263, 272.

[7] Esquivel S., Zuppa F., Gallard R., Contrasting Conventional and Evolutionary Approaches for the Flow Shop Scheduling Problem. EIS II.

[8] Esquivel S.C., Zuppa F., Gallard R. Multiple Crossovers, Multiple Parents and the Stud for Optimization in the Flow Shop Scheduling Problem, LIDIC, SCI 2001.

[9] Goldberg, D.E. and Lingle, R., Alleles, Loci, and the TSP, pp. 154-159.

[10] Mitsuo Gen, Runwei Cheng, Genetic Algorithms and Engineering Design. A Wiley and-Interscience Publication.

[11] Oliver, I.M., Smith, D.J., abd Holland, J.R.C., A study of Permutations Crossover Operators on the Traveling Salesman Problem.

[12] Pinedo M., Scheduling Theory, Algorithms and Systems. Prentice Halls international series in industrial and systems engineering. (16)

[13] Taillard E., *Benchmarks for basic scheduling problems*, European Journal of Operational Research, vol 64, pp 278-285, 1993. (http://www.idsia.ch/~eric/problemes.dir/ordonnancement.dir/ordonnancement.html)(22)