A Study of Genotype and Phenotype Distributions in Hybrid Evolutionary Algorithms to Solve the Flow Shop Scheduling Problem

Minetti G., Alfonso H.
Proyecto UNLPAM-09/F015[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, alfonsoh }@ing.unlpam.edu.ar
Phone/Fax: +54 02302 422780/422372, Ext. 6302

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

In this preliminary study the Flow Shop Scheduling Problem (FSSP) is solved by hybrid Evolutionary Algorithms. The algorithms are obtained as a combination of an evolutionary algorithm, which uses the Multi-Inver-Over operator, and two conventional heuristics (CDS and a modified NEH) which are applied either before the evolution begins or when it ends. Here we analyze the genotype and phenotype distribution over the final population of individuals trying to establish the algorithm behavior. Although the original Evolutionary Algorithm was created to provide solutions to the Traveling Salesman Problems (TSP), it can be used for this particular kind of scheduling problem because they share a common chromosome representation.

Keywords: flow shop sequencing problem, evolutionary algorithms, heuristics, CDS, NEH.

---

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

1. Introduction

The Flow-Shop Sequencing Problem is generally more precisely described as follow: There are *m* machines and *n* jobs, each job consists of *m* operations, and each operation requires a different machine. *n* jobs have to be processed in the same sequence on *m* machines. The processing time of job *i* on machine *j* is given by t_{ij} (i=1,.....,n; j=1,.....,m). In other words, jobs are to be processed on multiple stages sequentially. There is one machine at each stage. Machines are available continuously. A job is processed on one machine at a time without preemption, and a machine processes no more than one job at a time. The objective is to find a sequence of jobs minimizing the maximum flow time, which is called *makespan* [5].

Many different Evolutionary Algorithms (EAs), including parallel approaches, have been successfully applied to solve flow-shop sequencing problems [2, 3, 4, 6, 7, 16, 17, 19]. This kind of problem is essentially a permutation schedule problem, and the permutation of jobs can be naturally represented by a sequence of genes. In consequence, the algorithms, that were created to solve the TSP, can also be used for this kind of problems. This is the case of Multi-Inver-Over Evolutionary Algorithm [8, 9, 10].

In this work the conformation of final solutions, which were obtained by EAs and their combinations are studied. Although in a previous paper [14] better results were obtained combining the EA with local search techniques as Simulated Annealing and Tabu Search, an great computational effort was needed. For that reason, the hybridization with other techniques is investigated. These techniques are heuristics which specifically created for solving the FSSP. They are the CDS heuristic [1] and a modified version of the NEH heuristic [15].

2. Hybridized Algorithms

The Inver-Over Evolutionary Algorithm (*IO*) was developed first by Michalewicz [8]. This algorithm is currently considered as one of the best heuristics to solve TSP. In previous works we improved its performance by using a multirecombined method [10]. Further enhancement were achieved by HMEAs (*Hybrid Multi-inver-over Evolutionary Algorithms*), which consist in hybridizing multirecombined evolutionary algorithms (global search) with simulated annealing and tabu search (local search) [11, 12, 13, 14]. The Hybrid Multi-Inver-Over Evolutionary Algorithms presented here incorporate CDS heuristic and a modified NEH heuristic to the multirecombined *IO* versions (*IO*-n_1); where n_1 is the number of times that the Inver-Over operator is applied.

IO can be seen as a set of *m*-parallel hill climbing procedures. In each procedure, the number of inversions and the segment to be inverted depend on the current population. The algorithm is an evolutionary one, with an adaptive operator, which is a combination of inversion and crossover (e.g. the city to go is selected on the basis of another individual from the population).

The Nawaz, Enscore, and Ham (NEH) heuristic is based on the assumption that a job with a higher total processing time on all machines should be given higher priority than a job with a lower total processing time [15]. The NEH algorithm builds the final sequence in a constructive way, adding a new job at each step and finding the best partial solution. The elimination of the first step in the NEH heuristics is the modification that was made in this algorithm. This change was carried out to find different schedules from distinct sequences of a same instance.

The Campbell, Dudek and Smith (CDS) heuristic is basically an extension of Johnson´s algorithm [1]. Its efficiency relies on two properties:

- ✓ Uses Johnson´s rule in a heuristic fashion.
- ✓ Then, by solving *m*-1 two machine problems, creates several schedules from which a best schedule can be chosen.

This HMEA includes the above-described heuristics to the *IO*-n_1 versions separately. Each heuristic is applied to a percentage of individuals of the initial population and the evolutionary algorithm begins from this improved population. Individuals can be selected randomly (R), or they are the best (B) or the worst (W) in the population. In the case of NEH, the heuristic is applied in the final population, too. As a result the following algorithms were designed (*w* is a wild card standing for R, B, or W indistinctly):

- IO-n_1+CDS-*w*. Applies CDS in its original version. As CDS produces a unique best schedule from different sequences of the same instance, a number of identical individuals are inserted in the initial population.
- IO-n_1+CDS1. This modified heuristic procedure inserts all the schedules created during the different stages of CDS, in the initial population.
- IO-n_1+NEH1-*w*. Applies the modified NEH version in the initial population.
- IO-n_1+NEH2-*w*. Applies the modified NEH version in the final population.

CDS and CDS1 heuristics are only applied to the initial population because they always produce the same set of schedules. The modified version of NEH can be applied also in the final population because the schedule created depends on the initial sequence provided by the individual to be hybridized.

3. Experiment Description

According to the above described hybrid algorithms, a set of experiments were performed. All of them used the multirecombined inver-over operator. Five different approaches, *IO-1* to *IO-5* were conducted applying from 1 to 5 inver-over operations, respectively.
All approaches were tested for fourteen instances, extracted from [18]. They are:

Instances	ID	Size n x m	Upper Bound
Tail001, Tail003, Tail005, Tail007 & Tail009	T001, T003, T005, T007 & T009	20 x 5	1278, 1081, 1235, 1234, & 1230, respectively.
Tail012, Tail014, Tail016, Tail018 & Tail020	T012, T014, T016, T018 & T020	20 x 10	1659, 1377, 1397, 1538, & 1591, respectively.
Tail033 & Tail037	T033 & T037	50 x 5	2621 & 2725 respectively
Tail044 & Tail049	T044 & T049	50 x 10	3063 & 3897 respectively

For each instance a series of fifty runs was performed. All the Multirecombined Evolutionary Algorithms used the following parameter settings. Population size of 100 individuals, probability *p* set to 0.02, elitism to retain the best valued individual found so far, maximum number of generations fixed at 4000 and a stop criterion is established as follow: once the first 500 generations are accomplished the algorithm stops if the best individual does not change after 100 consecutive generations. The heuristics are applied over a 5% of population.

4. Results

To evaluate the algorithms the following relevant performance variables were chosen:

- **Ebest:** It is the percentile error of the best-found individual when compared with the known, or estimated, optimum value *opt_val*.
- **Epop:** It is the percentile error of the population mean fitness when compared with *opt_val*.
- **Gbest:** Identifies the generation where the best individual (retained by elitism) was found.

- **SD:** Standard Deviation.

Algorithm	T001	T003	T005	T007	T009	T012	T014	T016	T018	T020	T033	T037	T044	T049
***IO*-1**	1286	1098	1278	1269	1245	1696	1394	1423	1559	1610	2655	2776	3220	3087
***IO*-2**	1281	1096	1275	1269	1236	1684	1398	1412	1563	1613	2647	2758	3197	3039
***IO*-3**	**1278**	1089	1271	1269	1237	1681	1392	1415	1554	1612	2641	2745	3170	3031
***IO*-4**	**1278**	1098	1268	1269	1247	1678	1393	1410	1555	1610	2640	2741	3165	3025
***IO*-5**	**1278**	1088	1271	1269	1240	1678	1389	1406	1549	1603	2630	2736	3158	3012
***IO*-1+CDS1**	1284	1098	1244	1239	1247	1692	1410	1423	1566	1614	2631	2750	3144	3032
***IO*-2+CDS1**	**1278**	1087	1243	1239	1240	1685	1393	1416	1550	1619	2626	2735	3134	3000
***IO*-3+CDS1**	**1278**	1095	1243	1239	**1230**	1678	1395	1407	1558	1605	2624	2746	3125	2995
***IO*-4+CDS1**	**1278**	1096	1243	1239	**1230**	1673	1385	1412	1550	1604	2625	2737	3126	2976
***IO*-5+CDS1**	**1278**	1088	1236	1239	1233	1677	1388	1403	1555	1610	2624	2736	3116	2979
***IO*-1+CDS-R**	1279	1097	1244	1239	1243	1703	1404	1424	1558	1612	2631	2763	3171	3030
***IO*-2+CDS-R**	**1278**	1089	1244	1239	1240	1688	1398	1410	1559	1610	2624	2754	3162	2991
***IO*-3+CDS-R**	**1278**	1088	1236	1239	**1230**	1679	1392	1408	1548	1608	2622	2746	3135	2991
***IO*-4+CDS-R**	**1278**	1088	1241	1239	1233	1676	1393	1405	1549	**1591**	2624	2746	3120	2980
***IO*-5+CDS-R**	**1279**	**1081**	1236	1239	**1230**	1672	1394	**1397**	1551	1599	2624	2746	3106	2973
***IO*-1+CDS-W**	1279	1089	1236	1239	1251	1692	1404	1408	1564	1624	2631	2765	3179	3002
***IO*-2+CDS-W**	**1278**	1088	1244	1239	1237	1678	1392	1418	1556	1613	2624	2755	3147	2989
***IO*-3+CDS-W**	**1278**	1087	1236	1239	**1230**	1690	1388	1406	1544	1609	2624	2756	3149	3002
***IO*-4+CDS-W**	**1282**	**1081**	1236	1239	**1230**	1660	1389	1410	1546	1608	2624	2746	3129	2975
***IO*-5+CDS-W**	**1278**	**1081**	1235	1239	1230	1682	1387	1401	1553	1610	2622	2736	3126	2972
***IO*-1+CDS-B**	**1278**	1090	1236	1239	1252	1697	1395	1416	1559	1613	2631	2763	3176	3024
***IO*-2+CDS-B**	**1278**	1088	1243	1239	1240	1673	1402	1414	1560	1620	2622	2758	3141	2999
***IO*-3+CDS-B**	**1278**	1089	1236	1239	1239	1678	1393	1408	1546	1614	2622	2750	3139	2983
***IO*-4+CDS-B**	**1278**	1088	1243	1239	**1230**	1685	1393	1401	1554	1610	2622	2746	3132	2988
***IO*-5+CDS-B**	**1278**	1086	**1235**	1239	**1230**	1672	1388	1402	1554	1611	**2621**	2736	3126	2974
***IO*-1+NEH1-R**	**1278**	1091	1236	1239	1238	1686	1389	1419	1561	1605	2627	2732	3105	2987
***IO*-2+NEH1-R**	**1278**	1088	**1235**	1239	1241	1674	1391	1417	1556	1608	**2621**	2732	3113	2970
***IO*-3+NEH1-R**	**1278**	**1081**	**1235**	1239	**1230**	1667	1387	1408	1554	1596	2622	2732	3106	2976
***IO*-4+NEH1-R**	**1278**	1088	1237	1239	**1230**	1664	1383	**1397**	1544	1606	**2621**	**2725**	3118	2987
***IO*-5+NEH1-R**	**1278**	**1081**	**1235**	1239	**1230**	1664	1391	**1397**	1553	1602	2622	2732	3110	2982
***IO*-1+NEH1-W**	**1278**	1088	1236	1239	1240	1675	1392	1417	1560	1611	**2621**	2732	3115	2985
***IO*-2+NEH1-W**	**1278**	1088	**1235**	1239	1235	1675	1390	1400	1548	1612	2626	2732	3099	2971
***IO*-3+NEH1-W**	**1278**	1088	**1235**	1239	**1230**	1677	1386	1411	1555	1598	2623	2732	3114	2965
***IO*-4+NEH1-W**	**1278**	1088	**1235**	1239	**1230**	1679	1392	1407	1554	**1591**	**2621**	**2725**	3111	2959
***IO*-5+NEH1-W**	**1278**	1088	**1235**	1239	**1230**	1664	**1377**	1406	1555	1604	2622	2732	3105	2974
***IO*-1+NEH1-B**	**1278**	1089	1243	1239	**1230**	1679	1392	1417	1559	1613	2624	**2725**	3121	2986
***IO*-2+NEH1-B**	**1278**	1087	1239	1239	**1230**	1671	1392	1400	1554	1609	**2621**	2732	3110	2980
***IO*-3+NEH1-B**	**1278**	1088	1236	1239	1233	1668	1383	1411	1553	1608	**2621**	2732	3110	2981
***IO*-4+NEH1-B**	**1278**	**1081**	1236	1239	**1230**	1671	1387	1407	1548	1604	2623	2736	3104	2978
***IO*-5+NEH1-B**	**1278**	**1081**	**1235**	1239	**1230**	1672	1383	1406	1556	1608	2622	2726	3110	2975

Algorithm	T001	T003	T005	T007	T009	T012	T014	T016	T018	T020	T033	T037	T044	T049
***IO*-1+NEH2-R**	1281	1089	1243	1239	1253	1676	1400	1421	1555	1615	2627	2735	3116	3000
***IO*-2+NEH2-R**	1283	1094	1244	1239	1250	1685	1379	1418	1560	1615	2639	2746	3114	3009
***IO*-3+NEH2-R**	**1278**	1088	1239	1239	1238	1677	1394	1415	1554	1614	2639	2743	3110	3014
***IO*-4+NEH2-R**	**1278**	1088	**1235**	1239	1237	1670	1393	1398	1551	1607	2634	2739	3110	2999
***IO*-5+NEH2-R**	**1278**	1089	**1235**	1239	**1230**	1679	1379	1406	1550	1610	2627	2738	3113	2996
***IO*-1+NEH2-W**	**1278**	1098	1246	1239	1244	1689	1394	1424	1569	1618	2635	2736	3123	2995
***IO*-2+NEH2-W**	**1278**	1098	1236	1239	1240	1684	1390	1414	1562	1613	2638	2736	3118	2988
***IO*-3+NEH2-W**	**1278**	1090	1243	1239	1244	1676	1393	1406	1551	1608	2629	2736	3108	2992
***IO*-4+NEH2-W**	1279	1085	**1235**	1239	1233	1684	1379	1415	1553	1610	2637	2736	3125	2996
***IO*-5+NEH2-W**	1281	1087	1239	1239	**1230**	**1659**	1387	1406	1552	1610	2624	2736	3115	3007
***IO*-1+NEH2-B**	1288	1090	1250	1251	1255	1692	1401	1424	1565	1618	2635	2751	3118	3017
***IO*-2+NEH2-B**	1297	1091	1244	1247	1253	1688	1403	1442	1573	1636	2640	2758	3119	3017
***IO*-3+NEH2-B**	1297	1100	1248	1249	1253	1695	1397	1424	1572	1641	2632	2738	3121	3022
***IO*-4+NEH2-B**	1286	1088	1250	1250	1260	1686	1393	1433	1553	1636	2637	2746	3138	3025
***IO*-5+NEH2-B**	1286	1097	1250	1249	1253	1694	1392	1431	1564	1641	2636	2753	3126	3025

Table 1: The best makespan values found by each algorithmic option

Table 1 shows the lower makespan values obtained by each algorithm in every instance. Boldfaced values indicate that an algorithmic option has reached the upper bound for a particular instance.

Any algorithmic option is quite good for smaller instances (20 x 5 size), but most hybridized options work better for bigger instances.

Figures 1, 2, 3 and 4 show minimum and mean Ebest and Epop values for representative instances of their respective sizes.

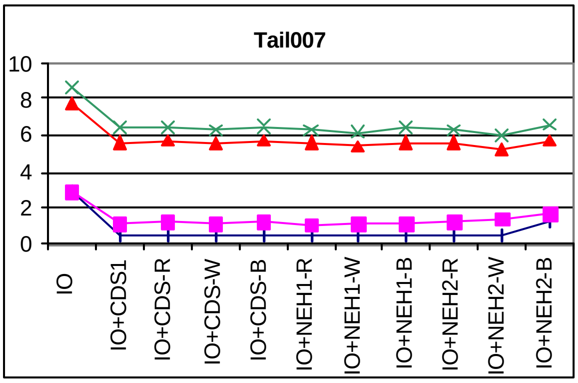


Figure 1: Ebest and Epop values for Tail007 Instance

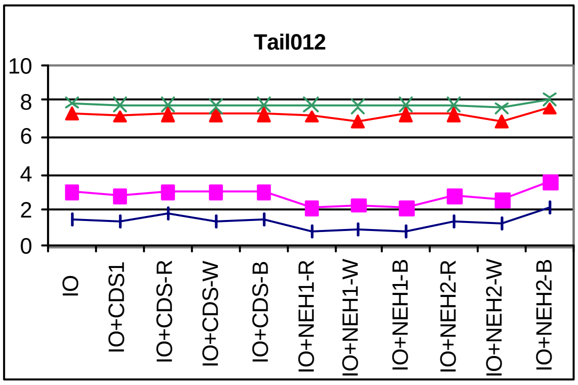
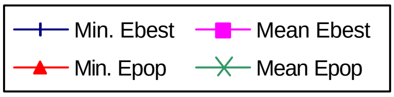


Figure 2: Ebest and Epop values for Tail012 Instance



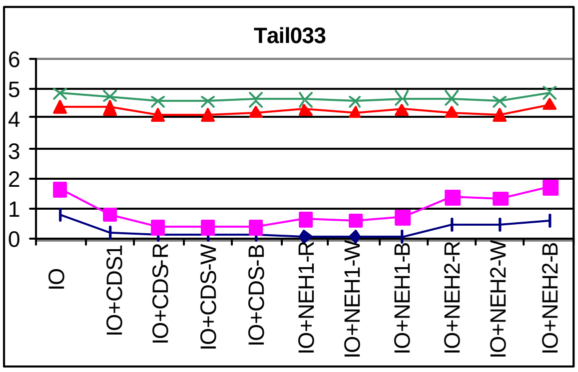


Figure 3: Ebest and Epop values for Tail033 Instance

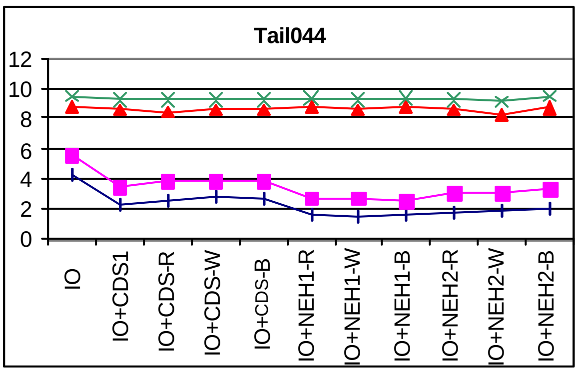


Figure 4: Ebest and Epop values for Tail044 Instance

A general overview on Ebest and Epop values, suggests that best individuals in the final population are surrounded by many other individuals. It can be inferred for two reasons:

1. The difference between both performance measures is small.
2. The behavior pattern is very similar for both error values.

The Epop values are lesser than 10%; that means the final population is close to the upper bound values. Besides in most cases Ebest values are equal or next to zero which means that best individuals attains or are very close to the optimum or estimated optimum.

In a more detailed analysis, better Ebest and Epop values are observed when the heuristic is applied to initial populations over random individuals or worst individuals (diversification helps).

In figure 5 the average of Minimum and Mean Gbest values for all instances is shown. Here we can see that the algorithms, which apply the heuristic in the initial population, obtain their best individual in earlier generations and consequently, lesser computational effort is required. Also, these curves follow the pattern showed by the Ebest and Epop values. This indicates that a better performance is reached when the heuristic is applied to initial populations.

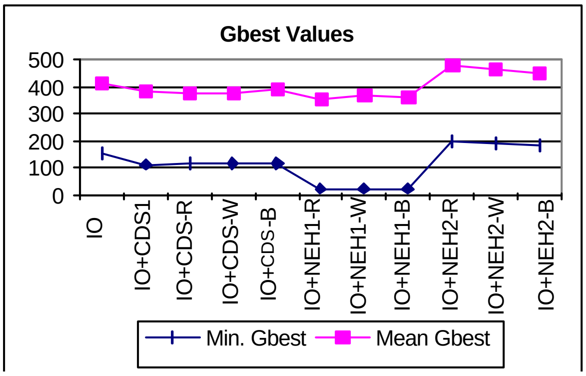


Figure 5: Average of Minimum and Mean Gbest Values for all instances.

4.1. Genotypic Distribution in the final Population

In this section we discuss on genotype distribution over the final population. The idea is to find a possible pattern of jobs allocation on chromosomes. We show here results for the same four representative instances. The following figures illustrate he number of occurrences where a job appears in a determined position. In the Y-axis, jobs (C1, C2…) are sequenced in non-decreasing order of processing time. The X-axis represents the positions on chromosomes, and the Z-axis shows the number of occurrences of a job in a determined position.

In figure 6 an almost uniform distribution is observed for Tail007 instance. Some deviations are present in the 20th position, were a short and a medium job predominate. Besides, in this position the absence of some long jobs is almost total. Hardly ever some medium or long job appear in the first locations. These characteristics are repeated for every algorithm with this instance.

In the case of the Tail012 instance, the ten shortest jobs begin by grouping from intermediate positions and then towards one of the borders (endmost position in the chromosome), while in the opposite border they are almost absent. The remaining jobs are distributed in a relative uniform way. These characteristics are given under any algorithmic option.

In Tail033 instance (fig. 8), the job allocation is also quite similar under any algorithmic option. Here, shortest jobs are almost uniformly distributed on the chromosome, while the remaining jobs, are concentrated in one of the borders.

In the last instance (Tail044, fig. 9) a similar job distribution is observed for each algorithm. But there is not a pattern, which does not allow us to describe the job allocation in relation with their length.

Summarising, for each particular instance jobs are distributed following a quite similar pattern under any of the hybrid algorithms used to solve the problem. Consequently, we can infer that the evolutionary algorithm orients the job distribution independently of the heuristic. This happens because heuristics are only applied on some individuals of the initial or final populations. A uniform job distribution means that different schedules are produced; then a high genotypic diversity is obtained.

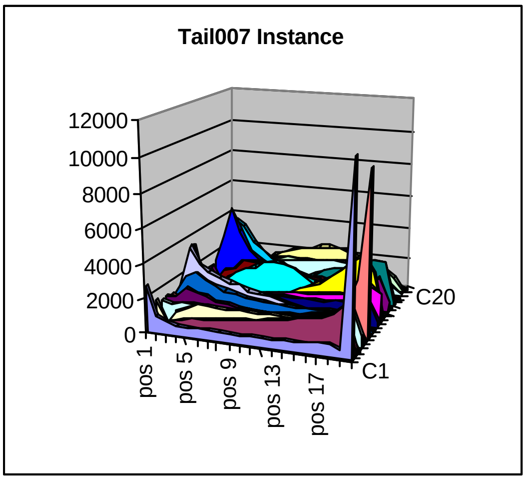


Figure 6: Jobs Distribution for Tail007 instance

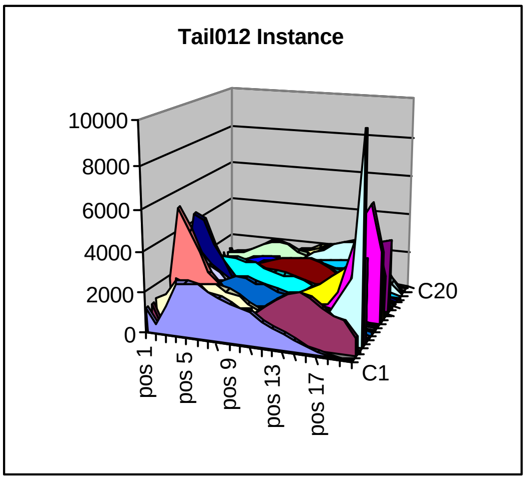


Figure 7: Jobs Distribution for Tail012 instance

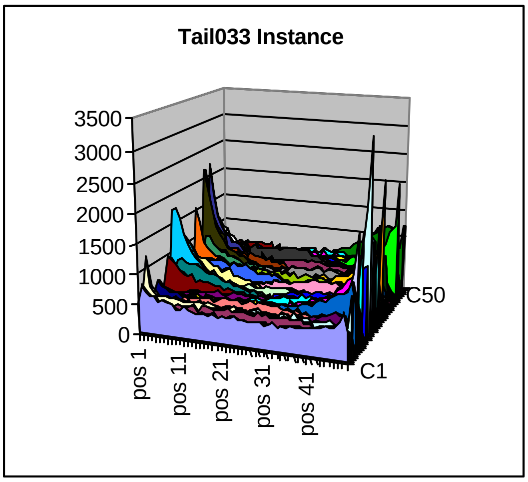


Figure 8: Jobs Distribution for Tail033 instance

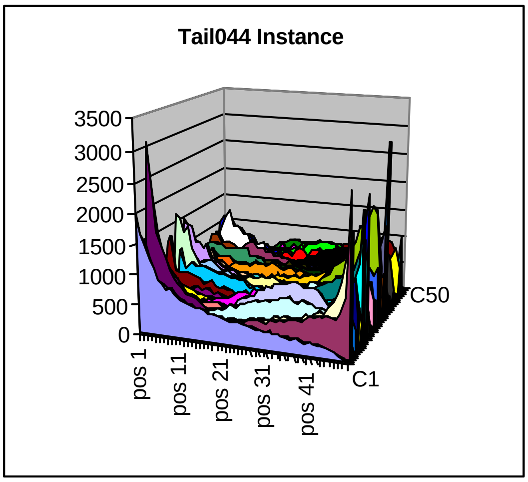
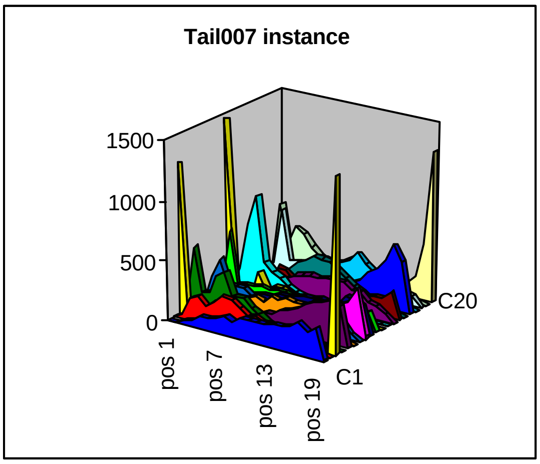


Figure 9: Jobs Distribution for Tail044 instance

Figures 10 and 11 show the job allocation at best individual chromosomes for two different problem sizes. In general the shortest and longest jobs have a tendency to group in the endmost positions of the chromosomes, while the rest of them are located in the internal positions.



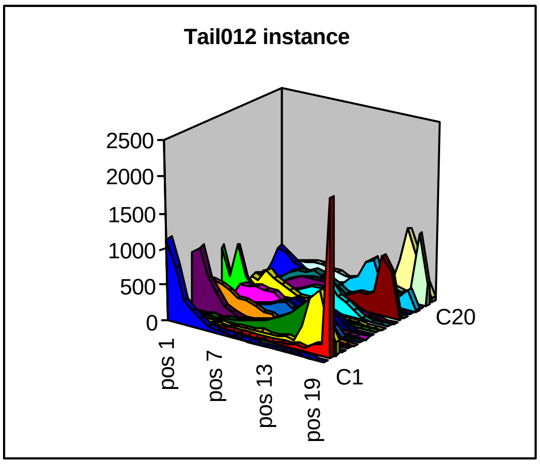


Figure 10: Jobs Distribution for the best indiv. in T007 Figure 11: Jobs Distribution for the best indiv. in T012

4.2. Phenotypic distribution in the final population

As the main characteristics of the phenotypic behaviour are quite similar for all different kinds of instances selected in this work, we only illustrate here the behaviour achieved under distinct multiplicity levels (1,2,…,5) of the *IO-n1* plus NEH1 algorithms, for the Tail044 instance. Figure 12. shows the mean makespan value for each of the 50 final populations obtained from the 50 runs of the algorithm under each multiplicity level.

In more detailed analysis it was determined that the standard deviation of individuals in the final population is less than one, for all algorithmic options. Also, the makespan values were decreased (improved) significantly when the multiplicity was augmented. This means that the combination of multiplicity and hybridization features provides better solutions without altering phenotypic diversity. Furthermore, these low standard deviations indicate a not too high variety of makespans values.

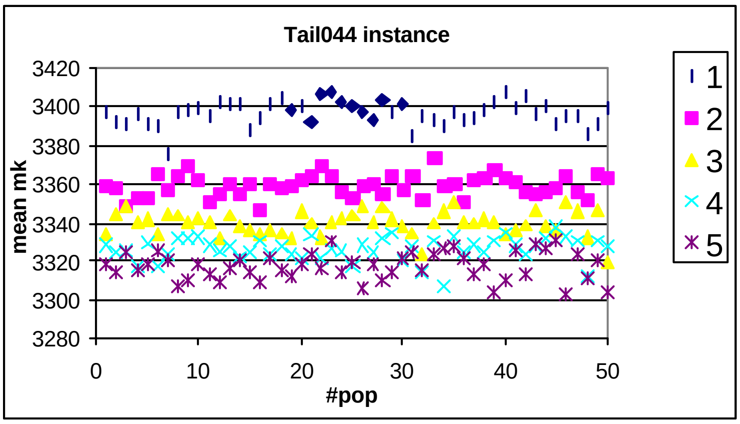


Figure 12: Mean Makespan for Tail044 instance under *IO-n1*+NEH1

The uniform job distribution, the low standard deviation of makespan values and the small Epop values (less than 10%), indicate a big number of schedules, which are built of many different ways and give high quality solutions. These solutions are close to the known optimum value and in many cases this value is reached.

5. Conclusions

We have studied and analyzed the combination of conventional heuristics and evolutionary algorithms, for solving scheduling problems, in particular the Flow Shop Scheduling Problem. The Evolutionary Algorithm searches and drives the search toward lower makespan values for each instance, while the conventional heuristics introduce individuals with problem specific knowledge. Consequently, the AE does a quicker and more efficient search. The hybridization used here provide good solutions without the computational effort required when tabu search or simulated annealing is applied to some individuals of the evolving population.

In our genotypic study we observed that similar patterns are obtained independently of the conventional heuristic used. Therefore, we can conclude that due to the level of hybridization, the heuristics (CDS and NEH) and the type of EA (Multi-inver-over) used, the latter is the main responsible of building the final solutions.

In our phenotypic study we observed low standard deviation of makespan values and low Ebest and Epop values. This ensures to provide a significant number of schedules, which are constructed by different permutations and are close to, or reach, the known optimum value. To have at hand a set of quasi optimal schedules is of utmost importance when the availability of ready jobs can change in the system.

Future work will be devoted to similar studies related with the behaviour of evolutionary approaches by analysing genotypic and phenotypic characteristics of the individuals in the final population.

6. References

[1]Campbell, H., Dudek, R., Smith, M., A heuristic algorithm for the *n*-job *m*-machine sequencing problem, Management Science, vol 16B, pp. 630-637, 1970.

[2]Chen C., Vempati V., Aljaber N., An application of genetic algorithms for flow shop problems, European Journal of Operational Research, vol.80, pp. 389-396.

[3]Esquivel S., Zuppa F., Gallard R., Multirecombinated Evolutionary Algorithms for the Flow Shop Scheduling Problem

[4]Esquivel S., Printista M., Zuppa F., Gallard R., Parallel and sequential evolutionary algorithms for the flow shop scheduling problem

[5]Gen M., Cheng R., Genetic Algorithms & Engineering Design, Wiley Interscience, 1997.

[6]Ishibuchi H., Yamamoto N., Murata T., Tanaka H., Genetic Algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems, Fuzzy Sets and Systems.

[7]Kubota A., Study on Optimal Scheduling for Manufacturing System by Genetic Algorithms, Master's thesis, Ashikaga Institute of Technology, Ashikaga, Japan, 1995.

[8]Michalewicz Z., Esquivel S., Gallard R., Michalewicz M., Tao G, Trojanowski, K., The Spirit of Evolutionary Algorithms, special issue on Evolutionary Computing of the Journal of Computing and Information Technology, University Computing Centre, Zagreb, Croatia, pp.1-18, 1999.

[9]Michalewicz Z, Fogel D.B., How to solve it: modern heuristics, Springer-Verlag Heidelberg, Germany 2000.

[10]Minetti G., Gallard R., Alfonso H., Inver-Over variants for de Euclidean Travelling Salesman Problem, Proceedings of the Second International ICSC Symposium on Engineering Of Intelligent Systems, EIS'2000, Scotland, pp. 458-463, 2000.

[11]Minetti G., Gallard R., Alfonso H., Hybrid Multi-Inver-Over Evolutionary Algorithms for the Euclidean Travelling Salesman Problem, International ICSC Symposia On Soft Computing (SOCO 2001) and Intelligent Systems For Industry (ISFI 2001), pp. #1824-094 of CD-Rom Proceedings ISBN 3-906454-27-4, Universidad de Paisley – Scotland, June 2001.

[12]Minetti G., Gallard R., Alfonso H., Influence Of Temperatures Settings For Simulated Annealing In A Hybrid Evolutionary Algorithm, 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001) and the 7th International Conference on Information Systems Analysis and Synthesis (ISAS 2001), pp. 397-400 of Proceedings ISBN 980-07-7543-9, Orlando – USA July 2001.

[13]Minetti G., Gallard R., Alfonso H., Bermudez C., Hybridizing Multi-Inver-Over Evolutionary Algorithms With Tabu Search For The Symmetric TSP, VII Congreso Argentino de Ciencias de la Computación – CACIC'2001, pp. 1141 a 1150 - ISBN 987-96-288-6-1, Calafate – Argentina, October 2001.

[14]Minetti G., Gallard R., Alfonso H., Solving Flow-Shop Sequencing Problem with Evolutionary Algorithms for TSP, International Conference on Computer Science, Software Engineering, Information Technology, e-Bussiness, and Applications – CSITeA'02, pp. 121-125 – ISBN: 0-9700776-3-7, Foz do Iguazu, Brazil, June 2002.

[15]Nawaz M., Enscore E., Ham I., A heurisitic algorithm for the *m*-machine *n*-job flow shop sequencing problem, Omega, vol. 11, pp. 11-95, 1983.

[16]Reeves C., A genetic algorithm for flow shop sequencing, Computers and Operations Research, vol. 22, pp. 5-13, 1995.

[17]Sannomiya N., Iima H., Application of genetic algorithms to scheduling problems in manufacturing process, in Proceedings of the Third IEEE Conference on Evolutionary Computation, IEEE Press, Nagoya, Japan, pp. 523-528, 1996.

[18]Taillard E., Benchamarks for basic scheduling problems, European Journal of Operational Research, vol. 64, pp. 278-285, 1993.

[19]Tsujimura Y., Gen M., Kubota E., Flow-shop scheduling with fuzzy processing time using genentic algorithms, The 11th Fuzzy Systems Symposium, pp. 248-252, Okinawa, 1995.