New Codification Schemas for Scheduling with Genetic Algorithms*

Ramiro Varela, David Serrano, and María Sierra

Dep. of Computer Science, University of Oviedo,
Artificial Intelligence Center,
Campus de Viesques, 33271 Gijón, Spain
{ramiro, mariasierra}@aic.uniovi.es
http://www.aic.uniovi.es/Tc

Abstract. Codification is a very important issue when a Genetic Algorithm is designed to dealing with a combinatorial problem. In this paper we introduce new codification schemas for the Job Shop Scheduling problem which are extensions of two schemas of common use, and are worked out from the concept of underlying probabilistic model. Someway the underlying probabilistic model of a codification schema accounts for a tendency of the schema to represent solutions in some region of the search space. We report results from an experimental study showing that in many cases any of the new schemas results to be much more efficient than conventional ones due to the new schema tends to represent more promising solutions than the others. Unfortunately the selection in advance of the best schema for a given problem instance is not an easy problem and remains still open.

1 Introduction

Genetic Algorithms (GAs) are a flexible search paradigm for dealing with complex optimization and combinatorial problems. Even though a conventional GA often produce moderate results, it is well-known that its efficiency can be improved by incorporating heuristic knowledge from the problem domain in any of the genetic operators, or by combining the GA with a local search procedure as done, for example, by D. Mattfeld in [5] and by T. Yamada and R. Nakano in [8] for the Job Shop Scheduling (JSS) problem.

In this paper we consider the issue of codification and propose new codifications schemas. These new schemas are worked out from the study of two schemas commonly used in scheduling problems: conventional permutations (CP) and permutations with repetition (PR). Through the concept of underlying probabilistic model proposed in [7] we developed two new schemas termed as partial PR (PPR) and extended PR (EPR) respectively, as extensions of PR and CP.

* This work has been supported by project FEDER-MCYT TIC2003-04153 and by FICYT under grant BP04-021.

J. Mira and J.R. Álvarez (Eds.): IWINAC 2005, LNCS 3562, pp. 11–20, 2005.

These new codifications improve the capacity of CP and PR to represent good schedules and consequently in many cases are able to improve the GA performance as well.

The rest of the paper is organized as follows. In section 2 we formulate the JSS problem and describe the search space of active schedules together with the G&T algorithm that allows for greedy search over this space. In section 3 we review two common codifications for scheduling problems: CP and PR, and propose the new models: PPR and EPR. Then in section 5 we report results from an experimental study over a subset of problem instances taken from a standard repository. Finally in section 6 we summarize the main conclusions and propose a number of ideas for future work.

2 Problem Formulation and Search Space for the JSS Problem

The JSS problem requires scheduling a set of N jobs $\{J_0, \ldots, J_{N-1}\}$ on a set of M physical resources or machines $\{R_0, \ldots, R_{M-1}\}$. Each job J_i consists of a set of task or operations$\{\theta_{i0}, \ldots, \theta_{i(M-1)}\}$ to be sequentially scheduled. Each task θ_{il} having a single resource requirement, a duration $du\theta_{il}$ and a start time $st\theta_{il}$ whose value should be determined. The JSS has two binary constraints: *precedence constraints* and *capacity constrains*. Precedence constraints defined by the sequential routings of the tasks within a job translate into linear inequalities of the type: $st\theta_{il} + du\theta_{il} \leq st\theta_{i(l+1)}$ (i.e. θ_{il} before $\theta_{i(l+1)}$). Capacity constraints that restrict the use of each resource to only one task at a time translate into disjunctive constraints of the form: $st\theta_{il} + du\theta_{il} \leq st\theta_{jk} \vee st\theta_{jk} + du\theta_{jk} \leq st\theta_{il}$. The objective is to come up with a feasible schedule such that the completion time, i.e. the *makespan*, is minimized.

The JSS problem has interested to many researches over the last three decades. In [4] Jain and Meeran review the most interesting approaches to this problem. One of the first efficient approaches is the well-known algorithm proposed by Giffler and Thomson in [3]. Here we consider a variant termed as hybrid G&T (see Algorithm 1). The hybrid G&T algorithm is an active schedule builder. A schedule is active if to starting earlier any operation, at least another one must be delayed. Active schedules are good in average and at the same time this space contains at least one optimal schedule. For these reasons it is worth to restrict the search to this space. Moreover the search space can be reduced by means of the parameter $\delta \in [0, 1]$ (see Algorithm 1, step 7). When $\delta < 1$ the search space gets narrowed so that it may contain none of the optimal schedules. At the extreme $\delta = 0$ the search is constrained to non-delay schedules: in such a schedule a resource is never idle if an operation that requires the resource is available. The experience demonstrates that as long as parameter δ decreases, in general, the mean value of solutions within the search space improves.

Algorithm 1 Hybrid G&T

1. Let $A = \{\theta_{j0}, 0 \leq j < N\}$;
while $A \neq \emptyset$ **do**
2. $\forall \theta_i \in A$ let $st\theta_i$ be the lowest start time of i if scheduled at this stage;
3. Let $\theta_1 \in A$ such that $st\theta_1 + du\theta_1 \leq st\theta + du\theta, \forall \theta \in B$;
4. Let $R = MR(\theta_1)$; $\{MR(\theta)$ *is the machine required by operation* $\theta\}$
5. Let $B = \{\theta \in A; MR(\theta) = R, st\theta < st\theta_1 + du\theta_1\}$
6. Let $\theta_2 \in B$ such that $st\theta_2 \leq st\theta, \forall \theta \in B$;
{*the earliest starting time of every operation in B,if it is selected next,is a value of the interval* $[st\theta_2, st\theta_1 + du\theta_1[\}$
7. Reduce the set B such that
$B = \{\theta \in B : st\theta \leq st\theta_2 + \delta((st\theta_1 + du\theta_1) - st\theta_2), \delta \in [0, 1]\}$;
{*now the interval is reduced to* $[st\theta_2, st\theta_2 + \delta((st\theta_1 + du\theta_1) - st\theta_2]\}$
8. Select $\theta^* \in B$ at random and schedule it at time $st\theta^*$;
9. Let $A = A \backslash \{\theta^*\} \cup \{SUC(\theta^*)\}$;
{$SUC(\theta)$ *is the next operation to* θ *in its job if any exists*}
end while

3 Codification Schemas for JSS with GAs

In this work we consider a standard GA such as the one showed in Algorithm 2, and for chromosome codification in principle we consider CP and PR schemas. In both cases a chromosome expresses a total ordering among all operations of the problem. For example, if we have a problem with $N = 3$ jobs and $M = 4$ machines, one possible ordering is given by the permutation *(*θ_{10} θ_{00} θ_{01} θ_{20} θ_{21} θ_{11} θ_{21} θ_{02} θ_{12} θ_{13} θ_{03} θ_{22}*)*, where θ_{ij} represents the operation $j, 0 \leq j < M$, of job $i, 0 \leq i < N$. In the CP schema, the operations are codified by the numbers $0, \ldots, N \times M - 1$, starting from the first job, so that the previous ordering would be codified by the chromosome *(4 0 1 8 9 5 10 2 6 7 3 11)*. Whereas in the PR schema an operation is codified by just its job number, so that the previous order would be given by *(1 0 0 2 2 1 2 0 1 1 0 2)*. PR schema was proposed by C. Bierwirth in [1]; and CP were also used by C. Bierwirth and D. Mattfeld

Algorithm 2 Genetic Algorithm

1. Generate the Initial Population;
2. Evaluate the Population
while No termination criterion is satisfied **do**
3. Select chromosomes from the current population;
4. Apply the Crossover and Mutation operators to the chromosomes selected at step 1. to generate new ones;
5. Evaluate the chromosomes generated at step 4.;
6. Apply de Acceptation criterion to the set of chromosomes selected at step 3. together with the chromosomes generated at step 4.;
end while
7. Return the best chromosome evaluated so far;

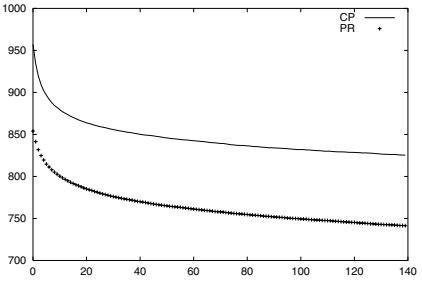


Fig. 1. Mean makespan evolution, over 140 generations, with CP and PR codifications for the problem instance ABZ8. In either case the GA was run 30 times starting from random initial populations

in [2]. In any case we chose the G&T algorithm as decoder. This only requires modify the selection criteria a step 8 (see Algorithm 1) by a deterministic one; in this case

8. Select $\theta^* \in B$ such that θ^* is the leftmost operation of B in the chromosome and schedule it at time $st\theta^*$

As genetic operators we consider generalized crossover (GOX) and mutation by swapping two consecutive operations, as described in [2]. Regarding selection and acceptation criteria we consider 2-2 tournament acceptation after organizing all chromosomes of the previous generation in pairs and apply crossover and mutation to every pair accordingly to crossover and mutation probabilities.

In [7] we have demonstrated that PR codification is better than CP. By simple experimentation it is easy to demonstrate that an initial population composed by random PRs is, in most of the cases, much better than a random population of CPs. Moreover a typical GA converges towards better solutions when PRs are used. Figure 1 shows the mean makespan convergence of the GA to solve the problem instance ABZ8 using PR and CP codifications. Furthermore we have also provided some explanation for such a behavior: a PR tends to represent *natural orders*. We explain this by means of an example. Let us consider that operations θ_{12} and θ_{20} require the same resource. As θ_{12} is the third operation of job 0 and θ_{20} is the first operation of job 2, the most natural or probable order among these two operations within an optimal (or at least a good) schedule can be considered in principle as $(\theta_{20}\ \theta_{12})$. The intuition behind this assumption is that the operation θ_{12} has to wait for at least two operations, θ_{10} and θ_{11}, to be processed, while the operation θ_{20} could be processed with no waiting at all. Now if we consider the probability that the operation θ_{20} appears before the operation θ_{12}, in a random PR this value is 0.95 whereas it is 0.5 in a random CP. In general, the probability that operation θ_{li} falls in a previous position to operation θ_{mj} in a random PR depends on the value of M and the positions i and j and is calculated by

$$P_{PR}(i,j) = (j+1) * \binom{M}{j+1} * \sum_{k=i+1}^{M} \frac{\binom{M}{k}}{\binom{2M}{k+j+1} * (k+j+1)} \quad (1)$$

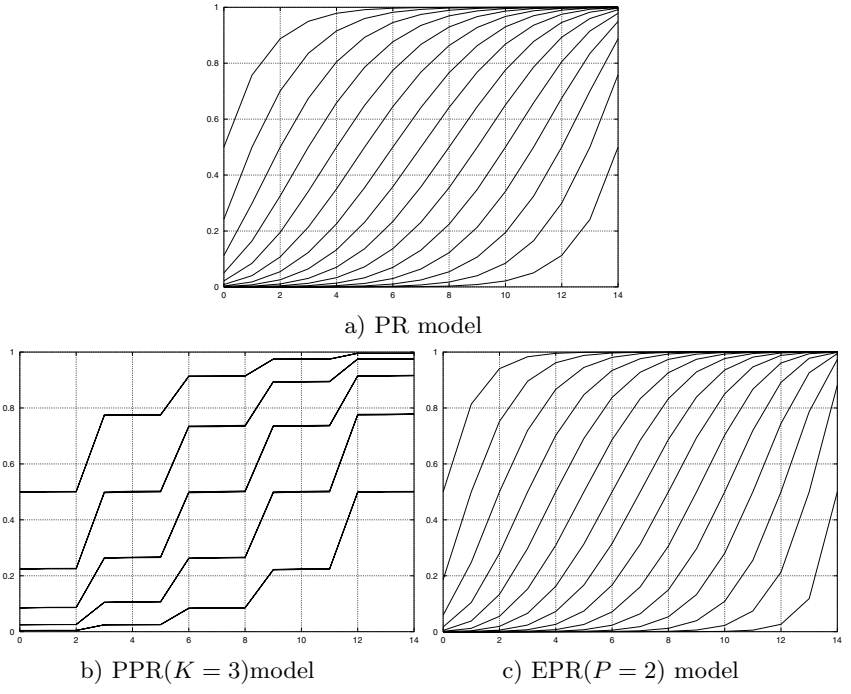
whereas for random CPs we have $P_{CP}(i,j) = 0.5$. We call these probability distributions *underlying probabilistic models* of the corresponding schemas PR and CP respectively. Figure 2a shows the PR underlying probabilistic model for a problem with 15 machines.

From PR and CP codifications, in particular from their probabilistic models, the following question raises: would be it possible to improve the PR schema?, If so, how should the underlying probabilistic model be? We conjecture that the essential of a given probabilistic model is the slope of the probability curves, in particular the way the slope varies as long as j varies from 0 to $M-1$ for a given i. This way we look for schemas with probabilistic models having slopes raising lower than PR, or to the contrary slopes raising more quickly than PR. In the first case we would have an intermediate schema between CP and PR, and in the second we would have an extension of PR in the opposite direction to CP. From this rationale we propose two extensions of the CP and PR schemas. The first one is termed Partial PR (PPR) and the second is Extended PR (EPR).

PPR consists on using a number of K different numbers to codify the set of operations of each job, M being the number of machines and K being an integer number that divides to M. In PPR for a given K the operations of job 0 are represented by numbers $0, 1, \ldots, K-1$; the operations of job 1 by $K, \ldots, 2(K-1)$, and so on. As every job has M operations each number should be repeated M/K times. For example operations of job number 0 are numbered by $0, 1, \ldots, K-1, 0, 1, \ldots, K-1, \ldots$ This way the PR chromosome *(1 0 0 2 2 1 2 0 1 1 0 2)* is equivalent to the $PPR(K=2)$ chromosome *(2 0 1 4 5 3 4 0 2 3 1 5)*. As we can observe CP and PR are limit cases of PPR with $K = M$ and $K = 1$ respectively.

On the other hand EPR is an extension of PR that consists on representing each operation by P numbers, instead of just 1 as in PR, taking into account that the last of the P numbers is the one that actually represents the operation when the decoding algorithm is applied, whereas the previous $P-1$ are only for the purpose of modify the probabilistic model in the way mentioned above. For example, the PR chromosome *(1 0 0 2 2 1 2 0 1 1 0 2)* is equivalent to the EPR($P=2$) chromosome *(0 1* ***1 0*** *0 2* ***0 2*** *2* ***2*** *2 1* ***1 2*** *1 0* ***0 1*** *2 0 1* ***1 0*** ***2)***.

Figures 2b and 2c show probability distributions corresponding to PPR and EPR schemas, with $K = 3$ and $P = 2$ respectively, for a problem with 15 machines. As we can observe PPR curves raise lower than PR curves, whereas EPR curves raise more quickly. In these cases the curves are calculated experimentally from a random population of 5000 chromosomes. However equations analogous to expression (2) for PR can be derived from PPR and EPR schemas as well. PPR curves are composed by steps of size K due to for every group of K consecutive operations with different numbers in a job all the relative orders among them in a random chromosome are equally probable. Moreover, for the same reason, every group of curves corresponding to K consecutive values of i, starting from $i = 0$, degenerates to the same curve.



a) PR model

b) PPR($K = 3$)model

c) EPR($P = 2$) model

Fig. 2. Probability profiles for a problem with $M = 15$ for RP , PPR and EPR models. In any case each of the curves represents the values of $P(i, j)$ for a fixed i whereas j ranges from 0 to $M-1$. The curves from up to bottom correspond to values of i ranging from 0 to $M - 1$. PR model is obtained by evaluation of expression (2), whereas PRP and EPR are obtained empirically from random populations of 5000 chromosomes

In any case it can be observed that for PPR, the larger the value of parameter K, the lower the rise of the curves. Whereas for EPR, the larger the value of P, the larger the rise. Here it is important to remark that for EPR the chromosome gets larger with increasing values of P so that some genetic operators get more time consuming.

4 Experimental Study

In this study we have experimented with a set of selected problems taken from a well-known repository: the OR library. These selected problems are recognized as hard to solve for a number of researches such as D. Mattfeld [5]. Table 1 reports results about the mean values of makespan of random initial populations regarding various codifications. As we can observe, with the only exception of FT20 instance, PR populations are much better than CP populations. Moreover PPR populations gets better and better as long as the value of parameter K augments, but in neither case PPR is better than PR. Regarding EPR, popula-

Table 1. Summary of mean values from random populations of 100 chromosomes with different codifications. In PPR a "-" means that the corresponding value of K is not applicable (it does not divide to M). Chromosomes are evaluated by means of the G&T algorithm with $\delta = 0.5$

Problem Instance		CP	PPR			PR	EPR		
Name(size)	BS		K=5	K=3	K=2		P=2	P=3	P=4
$abz7(20 \times 15)$	665	940	876	854	-	818	801	795	**792**
$abz8(20 \times 15)$	670	957	905	882	-	854	828	819	**815**
$abz9(20 \times 15)$	686	950	915	903	-	895	883	882	**881**
$ft10(10 \times 10)$	930	1265	1242	-	1218	1200	1185	**1181**	**1181**
$ft20(20 \times 5)$	1165	**1510**	-	-	-	1531	1571	1590	1603
$la21(15 \times 10)$	1046	1432	1380	-	1364	1324	1315	1315	**1314**
$la24(15 \times 10)$	935	1321	1260	-	1216	1192	1164	1150	**1141**
$la25(15 \times 10)$	977	1400	1357	-	1277	1238	1204	1193	**1185**
$la27(20 \times 10)$	1235	1691	1624	-	1572	1555	**1541**	1542	1545
$la29(20 \times 10)$	1153	1685	1595	-	1521	1481	1444	1431	**1422**
$la38(15 \times 15)$	1196	1660	1591	1576	-	1530	1519	1512	**1509**
$la40(15 \times 15)$	1222	1675	1599	1581	-	1520	1491	1479	**1473**

Table 2. Summary of results from the GA with various codification schemas starting from random initial populations. The GA was ran 30 times for each problem instance with crossover probability 0.7, mutation probability 0.2, population size of 100 chromosomes, 200 generations and parameter $\delta = 0.5$ in decoding algorithm G&T. For each one of the codifications the mean error in percent of the best solutions reached in the 30 trials is represented

Problem Instance		CP	PPR			PR	EPR		
Name(size)	BS		K=5	K=3	K=2		P=2	P=3	P=4
$abz7(20 \times 15)$	665	15.6	7.7	5.6	-	2.4	**1.7**	1.8	1.8
$abz8(20 \times 15)$	670	17.0	8.3	11.1	-	8.3	**7.8**	**7.8**	8.0
$abz9(20 \times 15)$	686	18.7	15.9	14.1	-	**11.5**	12.1	12.7	13.2
$ft10(10 \times 10)$	930	6.5	5.1	-	**3.7**	**3.7**	4.3	4.7	5.6
$ft20(20 \times 5)$	1165	**1.4**	-	-	-	3.2	5.7	6.8	7.4
$la21(15 \times 10)$	1046	10.0	8.5	-	5.8	**4.4**	4.6	4.7	4.9
$la24(15 \times 10)$	935	9.8	8.3	-	5.6	4.9	**4.3**	4.5	4.8
$la25(15 \times 10)$	977	8.4	6.2	-	3.7	**3.1**	3.3	4.1	4.3
$la27(20 \times 10)$	1235	12.9	10.7	-	7.0	4.9	4.8	**4.7**	4.9
$la29(20 \times 10)$	1153	15.2	12.0	-	7.3	**6.7**	7.1	7.8	8.2
$la38(15 \times 15)$	1196	10.1	7.9	7.7	-	**7.3**	9.1	10.3	10.9
$la40(15 \times 15)$	1222	9.1	7.0	5.6	-	**4.5**	**4.5**	5.2	5.6
	Average	11.2	8.9	8.8	5.5	**5.4**	5.8	6.3	6.6

tions slightly improve as long as parameter P augments, but we have to take into account that in this case the chromosome size is proportional to the value of P.

In the next experiment we have run the GA with the codifications considered in previous experiment; in any case starting from random initial populations.

Table 2 reports results about the mean makespan in percent of the solutions reached by the GA calculated as

$$((Mean - Best)/Best) * 100, \tag{2}$$

Mean being the mean of the best solutions reached in the 30 trials of each version of the GA, and *Best* being the makespan of the optimal solution to the problem instance.

As we can observe the original PR schema produces the best results in average, even though PR initial populations are not the best ones, as shown in Table 1. However EPR produces the best results for 5 of the 12 problem instances. Again problem instance FT20 is an exception, in this case CP is the best schema, not only for initial population but also regarding the GA evolution.

From comparison of Tables 1 and 2 we can observe that in principle the final values of the mean error reached by the successive GA versions improves as long as the mean makespan of initial populations gets lower. However the mean error reaches a minimum around PR schema, and beyond this point the mean error values augment in spite of the fact that the initial populations have a lower mean makespan. This behavior can be explained by taking into account the degree of diversity of the population. As showed in Table 3, as the initial populations get lower values of mean makespan, these populations get also a lower diversity that translates into a premature convergence of the corresponding GA version. Therefore we could conclude that, regarding a conventional GA such as the one proposed in section 3, PR schema presents the best tradeoff between quality and diversity, and therefore in general the GA reaches better solutions when starting from random initial populations. However in a remarkable number of cases EPR performs better than PR. This fact suggest us that EPR is a suitable codification that can be considered as comparable to PR. Regarding the remaining codifications, mainly EPR($P > 2$), a control mechanism should

Table 3. Summary of standard deviation values of makespan from random populations of 100 chromosomes with different codifications

Problem Instance		CP	PPR			PR	EPR		
Name(size)	BS		K=5	K=3	K=2		P=2	P=3	P=4
$abz7(20 \times 15)$	665	54.3	39.8	33.1	-	26.7	21.8	20.3	19.6
$abz8(20 \times 15)$	670	54.3	40.2	36.8	-	30.1	24.8	22.8	21.0
$abz9(20 \times 15)$	686	46.1	35.1	31.9	-	32.7	29.1	27.1	25.5
$ft10(10 \times 10)$	930	76.5	70.6	-	64.8	59.7	51.6	46.0	44.1
$ft20(20 \times 5)$	1165	72.6	-	-	-	62.7	54.6	48.7	44.6
$la21(15 \times 10)$	1046	87.9	70.5	-	67.4	57.6	49.9	45.5	43.2
$la24(15 \times 10)$	935	98.5	75.3	-	67.3	60.0	51.1	44.7	43.2
$la25(15 \times 10)$	977	95.7	94.8	-	74.3	65.1	51.5	45.9	43.7
$la27(20 \times 10)$	1235	92.0	70.5	-	63.2	59.9	58.9	59.5	58.2
$la29(20 \times 10)$	1153	91.1	80.8	-	66.3	57.7	50.7	44.9	44.0
$la38(15 \times 15)$	1196	104.4	79.7	71.5	-	59.3	49.7	45.3	44.4
$la40(15 \times 15)$	1222	108.5	88.8	85.2	-	67.2	60.7	53.8	50.8

be introduced into the GA to maintain an acceptable degree of diversity; not only in the initial population, but also along the GA evolution. Maybe in this case EPR($P > 2$) could be competitive. At the same time PPR and CP in principle seem to be not competitive with the remaining ones due to the GA converges to much worse values in spite of the high diversity degree that these codifications produce into the initial populations. We have also conducted experiments over a number of 1000 generations with similar conclusions.

Regarding time consumption, CP, PPR and PR schemas needs approximately the same values. For example an execution in the conditions reported in Table 2, that is over 200 generations, takes about 1.9 secs. for a problem of size 10×10 and about 7.5 secs. for a problem of size 20×15. However for EPR schema the time consumption augments with the value of parameter P so that the times required for the 10×10 instance are about 2, 2.3 and 2.5 secs. for values of P of 2, 3 and 4 respectively. Whereas these times are about 8.5, 9 and 10 secs. for the 20×15 instance. The experiments were conducted on a Pentium IV processor at 1.7 Ghz. and 512 Mbytes of RAM, under Windows XP operating system.

5 Concluding Remarks

In this paper we have proposed two new codification schemas to solve scheduling problems by means of genetic algorithms: PPR and EPR. These new schemas are in fact extensions of two codifications of common use in scheduling and related problems: CP and PR. By simple experimentation we have demosntrated that PR is in general much better than CP. We have observed that populations of randomly generated PRs has a mean value of makespan much lower than random populations of CPs. Moreover the convergence of a conventional GA is much better when PR is used. In [7] we have provided an explanation to these facts: PRs tends to represent natural orders among the operations requiring the same resource. In other words, operations that are more likely to appear at the beginning of a good schedule have a larger probability of appearing at the first positions of the chromosome as well. Moreover we have formalized the explanation by means of the underlying probabilistic model of a codification. This is a probability distribution that accounts for the probability that operation at position i in its job sequence appears before than operation at position j of another job. By observation of the probabilistic models of CP and PR codifications we have worked out two extensions: EPR and PPR. EPR generalized both CP and PR and is parameterized by a value K ranging from M to 1. In principle only values of K that divides to M are considered in order to simplify the genetic operators. PPR($K = M$) is the same as CP and PPR($K = 1$) is the same as PR. On the other hand, EPR is an extension of PR but in the opposite direction to CP. EPR is parameterized as well by a value $P \geq 1$ that indicates the number of digits we use to represent an operation. EPR($P = 1$) is the same as PR and for larger values of P has the inconvenient of requiring a chromosome length proportional to P, what in practice restricts this value to small values as 2, 3 or 4.

The reported experimental results shown that the performance of the GA depends on the codification chosen. Moreover in average PR is the best schema, however for a significative number of problem instances other schemas, generally close to PR such as PPR with a small value of K or EPR with a small value of P, are better. Moreover schemas far from PR such as CP and EPR with larger values of P, are in general the worse ones. This fact suggest that it is worth to consider schemas other than PR. However as it does not seem easy to envisage a method to select in advance the best schema for a given problem instance, in principle the only way is trying various schemas at the same time and take the value provided for the best one. It would also be possible to allow the GA consider in the initial population chromosomes codified with different schemas and let the evolution process selecting the best one by itself.

References

1. Bierwirth, C.: A Generalized Permutation Approach to Jobshop Scheduling with Genetic Algorithms. OR Spectrum **17** (1995) 87-92.
2. Bierwirth, C, Mattfeld, D.: Production Scheduling and Rescheduling with Genetic Algorithms. Evolutionary Computation **7(1)** (1999) 1-17.
3. Giffler, B. Thomson, G. L.: Algorithms for Solving Production Scheduling Problems. Operations Reseach **8** (1960) 487-503.
4. Jain, A. S. and Meeran, S.: Deterministic job-shop scheduling: Past, present and future. European Journal of Operational Research **113** (1999) 390-434.
5. Mattfeld, D. C.: Evolutionary Search and the Job Shop. Investigations on Genetic Algorithms for Production Scheduling. Springer-Verlag, November 1995.
6. Varela, R., Vela, C. R., Puente, J., Gmez A.: A knowledge-based evolutionary strategy for scheduling problems with bottlenecks. European Journal of Operational Research **145** (2003) 57-71.
7. Varela, R., Puente, J. and Vela, C. R.: Some Issues in Chromosome Codification for Scheduling Problems with Genetic Algorithms. ECAI 2004, Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (2004) 7-16.
8. Yamada, T. and R. Nakano.: Scheduling by Genetic Local Search with multi-step crossover. Fourth Int. Conf. On Parallel Problem Solving from Nature (PPSN IV), Berlin, Germany, (1996) 960-969.