12

Genetic Algorithms - Constraint Logic Programming. Hybrid Method for Job Shop Scheduling

K. MESGHOUNI[1], P. PESIN[2], S. HAMMADI[1], C. TAHON[2], P. BORNE[1]

[1] Ecole Centrale de Lille
LAIL - URA CNRS D 1440
BP 48
59651 Villeneuve d'Ascq Cedex - France

[2] Université de Valenciennes et du Hainaut Cambresis
LAMIH - URA CNRS n° 1775
Le Mont Houy
BP 311
59304 Valenciennes Cedex - France

e-mail : mesghouni@ec-lille.fr, pesin@univ-valenciennes.fr, hammadi@ec-lille.fr, tahon@univ-valenciennes.fr, p.borne@ec-lille.fr

Abstract

The job-shop scheduling problem is one of hardest problem (NP-complete problem). In lots of cases, the combination of goals and resources has exponentially increasing search space, the generation of consistently good scheduling is particularly difficult because we have very large combinatorial search space and precedence constraints between operations. So this paper shows the cooperation of two methods for the solving of job shop scheduling problems; Genetic Algorithms (GAs) and Constraint Logic Programming (CLP). CLP is a concept based on Operational Research (OR) and Artificial Intelligence (AI). It tends to rid itself of their drawbacks and to regroup their advantages. The GAs are searching algorithms based on the mechanics of natural selection, they employ a probabilistic search for locating the globally optimal solution. That starts with a population of randomly generated chromosomes, but the difficulty resides in the creation of initial population. This paper explains how to use the CLP to generate a first population and we apply the (GAs) to provide a job shop scheduling minimizing a makespan (Cmax) of the jobs.

L. M. Camarinha-Matos (ed.), *Re-engineering for Sustainable Industrial Production*

Keywords

Genetic algorithms, Initial population, Assignment and scheduling problems, Constraint Logic Programming

1 INTODUCTION

Production planning and job-shop scheduling involve the organization of production process, the coordination and control of the interaction of production factors and their transformation into products over time. Planning is mostly concerned by the description of the product routings for the real manufacturing process. Job-shop scheduling, which is one of the most important problem for companies in the domain of production management, focuses mostly on time-related aspects.

Moreover the scheduling problem is very hard to solve. It is then quite difficult to obtain an optimal solution satisfying the real time constraints using exact methods such as CLP because they require considerable computation time and/or complex mathematical formulation (Carlier, 1988). So GAs are used to solve job-shop scheduling problem and gives "good" solution near the optimal one. This paper presents the cooperation of two methods GAs and CLP for the solving of job shop scheduling problems.

The GAs employ a probabilistic search for locating the globally optimal solution. They have lots of advantages. They are robust in the sense that they provide a set of solutions near of the optimal one on a wide range of problems, they can easily be modified with respect to the objective function and constraints. Unfortunately, it is difficult to apply the traditionally GAs for the scheduling problem :

- There is no practical way to encode a job-shop scheduling as a binary string that does not have ordering dependencies. We fell that the chromosome representation used should be much closer to the scheduling problem. In a such cases a significant effort must be done in designing problem specific 'genetic' operators; however, this effort would pay off in increased speed and improved performance of the system. Therefore in this paper we have incorporated the schedule specific knowledge in operators and in chromosome representation. For this reason the parallel representation of the chromosome was created (Mesghouni 1996).
- How to find an efficient way to create an initial population satisfying all the constraints ?

CLP is a concept based on Operational Research (OR) and Artificial Intelligence (AI). It tends to rid itself of their drawbacks and to regroup their advantages. CLP provides solving facilities for Constraint Satisfaction Problems (CSPs) ; a CSP consists of a set of variables and a set of constraints on those variables . CLP allows to find all the combinations of values of each variable which satisfy all the constraints.
The features of CLP are :

- extension of PROLOG by adding solves on new domains such as finite domain restricted terms, boolean terms, linear rational terms,
- concept of constraint resolution which replace the concept of unification of PROLOG,

- active treatment on constraints thanks to the "test and generate" principle which induce a reduction of solving times,
- CLP don't give one (unique and optimal) solution but is able to give a set of solutions which satisfy a set of constraints.

This paper is organized as follows : The second section contains a detailed description and formulation of our flexible job-shop scheduling problem. The GAs and their operators are described in the third section. The fourth section contain a description of the CLP. The description of our hybrid process is presented in the fifth paragraph. Finally the last section contains an illustration of our method through some examples.

2 DESCRIPTION OF THE PROBLEM

The data and the constraints of this problem are (Hammadi, 1991) :

- There are n jobs, indexed by j, and these jobs are independent of each other,
- each job j has an operating sequence which is called g_j ,
- each operating sequence g_j is an ordered series of x_j operations $O_{i,j}$ for i = 1... x_j,
- the realization of each operation $O_{i,j}$ requires a resource or machine selected from a set of machines, $M_{i,j} = \{M_{i,j,k}$, k = 1,2, ...,M / M = the total number of machines existing in the shop} this implying the existing of the assignment problem.
- the processing time $P_{i,j,k}$ of an operation depends on the chosen machine,
- a start operation runs to completion (non preemption condition),
- each machine can perform operations one after the other (resource constraints).

3 GENETIC ALGORITHMS

3.1 Concepts

Genetic algorithm is a relatively new approach of optimum searching. GAs inherits its ideas from evolution of the nature. They are a parallel and global search technique because they simultaneously evaluates many points in the search space, they are more likely to converge towards a global solution. These algorithms are developed in a way to simulate a biological evolutionary process and genetic operators acting on the chromosome. They begin with a population of randomly generated strings and evolves towards a better solution by applying genetic operators; reproduction, crossover and mutation (Goldberg, 1989). They are so efficient that they can find a nearly optimum solution even for a large scale problem. For solving the traditional NP-completed problem some modification are indispensable (Michalewiez, 1992). In this paper genetic algorithms are mandatory by the following components :

- a specific genetic representation (or encoding) depending on the problem for determining the feasible solutions of the job-shop scheduling optimization problem (Syswerda, 1990),
- an efficient way to create an initial population of potential solutions,
- original genetic operators that alter the composition of children during reproduction.

3.2 Representation of the chromosome

Traditionally, chromosomes are simple binary vectors. This simple representation is not convenient for the complex job-shop scheduling problem when we must respect a certain order. A better solution is to change the chromosome syntax to fit the problem. In the case of our problem, the chromosome is represented by a set of parallel machines and each machine is a vector which contains the assignment operations for this machine. The operations are represented by three terms. The first one is the order number of the operation in its operating sequence, the second one is the number of the job, to which this operation belongs and the third one is the starting time of the operation if its assignment on this machine is definitive (Mesghouni, 1996). This starting time is calculated taking into account the resources constraints.

3.3 Design of crossover operator

In nature, crossover occurs when two parents exchange parts of their corresponding chromosomes. In a genetic algorithm, crossover recombines the genetic material in two parents chromosomes to make two children in order to generate a better solution.
The child one is given by the following algorithm (Mesghouni, 1997) :

```
step 1 :
        Parent 1, Parent 2 and the machine Mk are randomly selected.
step 2 :
        {O_i,j}_k of Child 1 = {O_i,j}_k of Parent 1
        i = 1
step 3 :
        /* M is the total number of the machines */
        While (i < M ) do
        Begin
                If ( i ≠ k ) then
                Begin
                        Copy the non existing operations of Mi of parent 2 into child 1
                        i = i + 1
                End
        End
step 4
        i = k
        If ( any operation of child 1 is missing ) then
                Scan Mk of parent 2 and copy the missing operation
        End if
```

To obtain child 2 go to step 2 and invert the role of parents 1 and 2.

3.4 Mutation

Mutation plays an important role in genetic algorithms, it introduces some extra variability into the population, it typically works with a single chromosome, it always creates another chromosome (Mesghouni, 1997).

4 Constraint Logic Programming

4.1 Concepts

CLP (Fron, 1994) is a concept based on Operational Research (OR) and on Artificial Intelligence (AI). It tends to rid itself of their drawbacks (lack of flexibility of the cost function, one unique and optimal solution if it exists, if it doesn't exist there is no response, impossibility to introduce qualitative criteria for the OR and unsuitability for numerical problems, high solving times for AI) and to regroup their advantages (ability to process numerical constraints for RO and power of data representation, independence data/data processing, solving of problems unsolved by classical algorithms for AI). We can sum up the different features of CLP :

- CLP is an extension of PROLOG by adding solvers on new domains such as finite domains restricted terms, boolean terms, linear rational terms, The concept of constraint resolution replace the concept of unification in PROLOG (Llia, 1991),
- The treatment on constraints is active thanks to the "test and generate" principle which induce a decrease of solving times (Esquirol, 1995),
- CLP doesn't give one (unique and optimal) solution but is able to provide a set of solutions which satisfy a set of constraints (Fron, 1994).

4.2 CLP and CSP

CLP provides solving facilities for CSP (Constraint Satisfaction Problems). A CSP (Boronad, 1993) consists of a set of variables, a set of definition domains for those variables and a set of constraints of those variables. As a consequence, you can solve a problem by this process :

- definition of variables and constraints,
- a priori propagation of the constraints,
- production of solutions according to definition domains and constraints.

So a scheduling problem can be regarded as a CSP, where variables are beginning time of each task, domains are definition domains of each variable and constraints are various constraints.

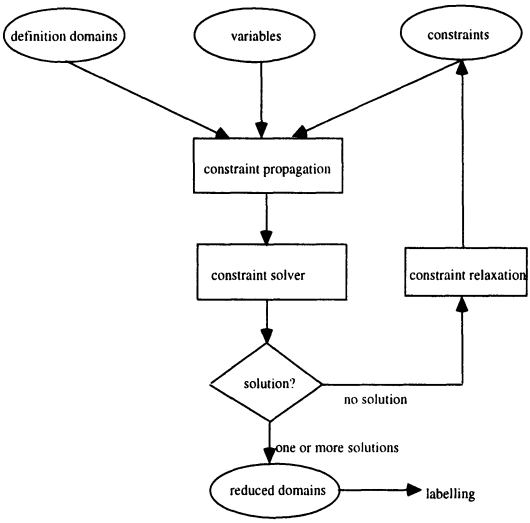

Figure 1 : Describe the structure of CLP using in scheduling

4.3 CLP and scheduling

In the field of scheduling, we can distinguish five main types of constraints (where $S_{i,j}$ = starting time of operation i of job j, $P_{i,j}$= processing time of operation i of job j after the choice of the resource, $I_{i1,j1\text{-}i2,j2}$=interval period between operation i1 of job j1 and operation i2 of job j2, $q_{i,jRt}$=amount of resource R selected for task i of job j at t, Q_R = the total amount of resource R available)

- relative location of an operation : to place an operation in relation with an other or with the others ($S_{i2,j2} - S_{i1,j1} >= I_{i1,j1\text{-}i2,j2}$ or $S_{i2,j2} - S_{i1,j1} >= P_{i1,j1}$ in the particular case),
- absolute location of a task : to place a task absolutely during the time ($S_{i,j} >= T$ where T is a given moment),
- disjunctive constraint : to force tasks not to be executed at the same time (if $S_{i1,j1} + P_{i1,j1} > S_{i2,j2}$ then $S_{i2,j2} + P_{12,j2} <= S_{i1,j1}$; if $S_{i2,j2} + P_{i2,j2} > S_{i1,j1}$ then $S_{i1,j1} + P_{i1,j1} <= S_{i2,j2}$),
- cumulative constraint : to force not to use the maximum amount of a resource available, for the set of all tasks using this resource($\sum q_{i,jRt} <= Q_R$),
- affectation constraint : to express the fact that one task must be processed by one resource and only one ($\sum A_{i,j,k} = 1$ during all the processing time of task i of job j, where $A_{i,j,k}$ is the affectation variable, k is the number of the resource, i,j means the task i of job j, $A_{i,j,k} = 0$ when task i of job j is not processed by k, $A_{i,j,k} = 1$ when task i of job j is processed by k).

CLP provides facilities to express all these constraints. All the constraints above can easily be expressed by predicates. We give the four main ones in the particular context of the constraint programming language CHIP (Constraint Handling in Prolog). The # means we are working on CHIP finite domains :

- succession constraints : succession ($S_{i2,j2}$,$S_{i1,j1}$, $P_{i1,j1}$):$S_{i2,j2}$ #>= $S_{i1,j1} + P_{i1,j1}$

- disjunctive constraints : disjunctive ($S_{i1,j1}$, $P_{i1,j1}$,$S_{i2,j2}$, $P_{i1,j1}$) : $S_{i1,j1}$ #>= $S_{i2,j2}$ + $P_{i2,j2}$
  disjunctive ($S_{i1,j1}$, $P_{i1,j1}$,$S_{i2,j2}$, $P_{i2,j2}$) : $S_{i2,j2}$ #>= $S_{i1,j1}$ + $P_{i1,j1}$.
- cumulative constraints (for a given resource) cumulative (Ld,Ldur, Lres, Lf,Lsur, Hau, Fin, Valinterm), with Ld:list of the starting times of the tasks processed by the given resource, Ldur:list of the duration of tasks, Lres: amount of resource given to each task, Lf:list of the ending times of the tasks processed by a given resource, other parameters are not very important in our context,
- affectation constraints (for a given task i of job j and for kn machines k1,k2,...,kn where i of job j may be processed) : affectation ($A_{i,j,k1}$,$A_{i,j,k2}$,...,$A_{i,j,kn}$) :-Ai,j,k1 + Ai,j,k2 + Ai,j,k3 +.....+ Ai,j,kn #= 1.

4.4 Hybrid process

After selecting an appropriate genetic representation of solutions to the problem, we should now create an appropriate initial population of solutions. This can be done in simple GAs in random way. However, some care should be taken in our problem because a set of constraints must be satisfied. Therefore a population of potential solutions found by CLP method is accepted as a starting point of our evolution GAs

In this paper, we present a cooperation of the CLP and the GAs to solve the flexible job-shop scheduling problem; we proceed in two steps.

Step 1 : We use CLP in order to provide a set of beginning solutions(set of scheduling) according to figure 1

Step 2 : we apply the (GAs) to solve a job shop scheduling minimizing a makespan (Cmax) of the jobs using a set of solutions found by the CLP like an initial population.

The first step is consists in two substeps :

- Finding a set of assignment (one for each task) with unlimited capacity which minimize the ending time of the scheduling. In this step succession constraints (for the tasks of each job) and assignment constraints (for each task of the scheduling problem) are applied. Ending time of scheduling is fixed to the minimum and all the possible sets of assignments are enumerate. An other solution consists in finding a set of assignment randomly for bigger size problem.
- Finding a set of scheduling with each assignment found above with limited capacity which minimize the ending time of the scheduling. In this step, cumulative constraints (for each resource) are applied. In this step succession constraints (for the task of each job) and assignment constraints (for each task of the scheduling problem) are applied

In the second step, we take this set of solutions(scheduling), we introduced it in GAs like an initial population, and we apply the genetic operators to improve the solution toward a optimal one.

5 NUMERICAL EXAMPLE

This section report on experiments made to show the performance of the proposed hybrid method. For the CLP we use the constraint programming language CHIP (Constraint Handling in Prolog) to generate a set of feasible schedule in a reasonable computing time. This set of

solutions represent a good initial population for the GAs which is implemented in a C-language program to provide quickly a near optimal schedule.

5.1 Little size problem

In this section we compare the results given by still only GAs, and the solution given by our hybrid method.
We study the following example, 5 machines and 3 jobs.
The operating sequences of these jobs are

Job 1 : O1,1, O2,1, O3,1.
Job 2 : O1,2, O2,2, O3,2.
Job 3 : O1,3, O2,3.

and the processing time of these operations are shown in table 2

	M1	M2	M3	M4	M5
O 1,1	1	9	3	7	5
O 2,1	3	5	2	6	4
O 3,1	6	7	1	4	3
O 1,2	1	4	5	3	8
O 2,2	2	8	4	9	3
O 3,2	9	5	1	2	4
O 1,3	1	5	9	3	2
O 2,3	5	9	2	4	3

Table 1 Processing time of the operations.

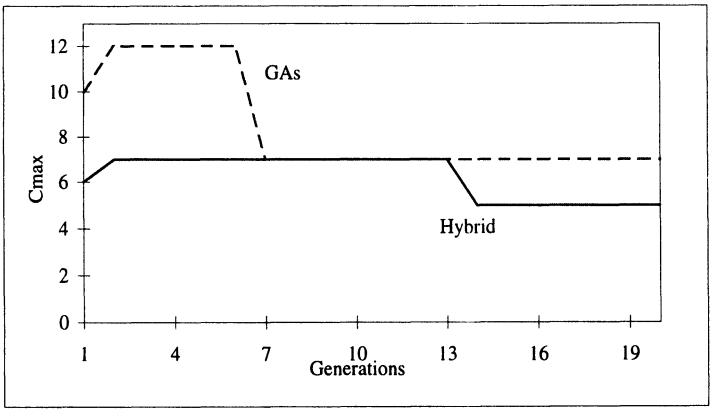

Figure 2 : Represent the Cmax for still only GAs and the Cmax for our hybrid method.

For the still only GAs, we proceed as following, we take one solution given by other methods (in this case by temporal decomposition method) and we apply the previously mutation operators to extend the population. We apply the GAs with the following operators :

- Size of population = 20
- Crossover probability (Pc) = 0.65
- Mutation probability (Pm) = 0.05
- Number of generation (NB gen) = 20.

For the CLP only, we use the procedure described in the section 4.3, in the figure 2 we represent the different cmax associated to set of scheduling

For the hybridization results, we take the set of solutions given by the CLP, we use it like the first population and we apply the GAs with the follow operators :

- Size of population = 12
- Crossover probability (Pc) = 0.65
- Mutation probability (Pm) = 0.05
- Number of generation (NB gen) = 20.

In this case the solution given by the hybrid method is better then of the solution gives by the GAs only and by the CLP only. But in order to prove the effectiveness of our hybrid method we should apply it for a big size problem.

5.2 Big size problem

We consider a big size problem with 10 jobs and 10 machines, each job has 3 operations and each operation can be performed by any machine(we have a total flexibility). In order to solve this problem by our hybrid method, we propose to proceed as in the following flow chart :

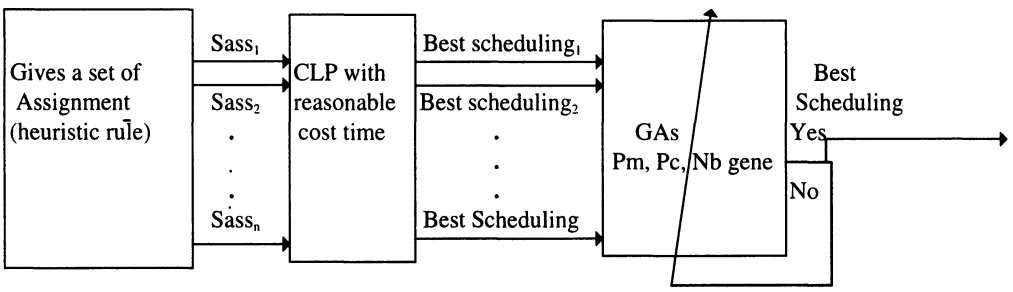

Figure 3 : Flow chart of hybrid method for big size problem.

With Sassi : a set of assignment of the operations to the machines.
i = 1 to n

This problem is very difficult to solve due to the very large combinatory process.
In this context, only applying CLP requires considerable computation time or complex mathematical formulation. Only applying GAs with a random initial population leads the 'genetic' operators to work in an inappropriate search space. So using an exact method such as branch-and-bound and dynamic programming or CLP method to create a good initial population is an important starting point for the evolution GAs.

6 CONCLUSION

The application of a new method to a flexible job-shop scheduling problem with real-word constraints has been defined. In this paper and for a little size problem the hybrid method is efficient and gives a good solution satisfying all of the constraints (precedence and resources constraints) and confirming the effectiveness of the proposed approach. This very good results gives by little size problem encourage for using this method like a figure 3 for a big size problem (10 jobs and 10 machines), for instance.

7 REFERENCES

Boronad-Thierry C. (1993) Planification et ordonnancement multi-site : une approche par satisfaction de contraintes, Thèse de doctorat, Université Paul SABATIER, Toulouse.

Carlier, J and Chretienne, P. (1988) Problèmes d'ordonnancement : Modélisation / complexité / algorithmes. Masson, Paris

Esquirol, P., Lopez, P. (1995) Programmation logique avec contraintes et ordonnancement : Automatique-Productique-Informatique Industrielle, Hermès, Vol. 29, n°4-5, pp379-407.

Fron A. (1994) Programmation par contraintes, Addison-Wesley.

Goldberg, D. E. (1989) Genetic algorithms in search, optimization, and machine learning. Addison-wesley.

Hammadi, S .Castelin, E. Borne, P.(1991) Efficient and feasible job-shop scheduling in flexible manufacturing systems : IMACS international symposium, Toulouse, France, March 13-15.

LLIA (1991) La programmation par contraintes, LLIA, n°73, July-august .

Mesghouni, K. Hammadi, S. Borne, P (1996). Production job-shop scheduling using Genetic Algorithms. proceedings of IEEE /SMC Conference, Beijing, China, Vol 2, October 14-17,. 1519-1524.

Mesghouni, K. Hammadi, S. Borne, P (1997) , Parallel Genetic Operators For flexible Job-Shop Scheduling, to appear in 1st International Conference on Engineering Design And Automation, Bangkok, Thailand, March 18-21.

Michalewicz, Z. (1992) Genetic Algorithms + Data Structures = Evolution programs: Springer Verlag.

Syswerda, G. (1990) Schedule optimization using genetic algorithm: in handbook of genetic algorithm (ed. Van Nostrand Reinhold), New-York.