Solving the Balanced Academic Curriculum Problem with an Hybridization of Genetic Algorithm and Constraint Propagation

T. Lambert[1,2], C. Castro[3], E. Monfroy[1,3,⋆], and F. Saubion[2]

[1] LINA, Université de Nantes, France
Firstname.Name@lina.univ-nantes.fr
[2] LERIA, Université d'Angers, France
Firstname.Name@univ-angers.fr
[3] Universidad Santa María, Valparaíso, Chile
Firstname.Name@inf.utfsm.cl

Abstract. In this paper, we are concerned with the design of a hybrid resolution framework including genetic algorithms and constraint propagation to solve the balanced academic curriculum problem. We develop a theoretical model in which hybrid resolution can be achieved as the computation of a fixpoint of elementary functions. These functions correspond to basic resolution techniques and their applications can easily be parameterized by different search strategies. This framework is used to solve a specific problem and we discuss the experimental results showing the interest of the of the model to design such hybridizations.

Keywords: CSP, genetic algorithms, constraint propagation, hybrid resolution.

1 Introduction

Constraint Satisfaction Problems (CSP) are usually defined by a set of variables associated to domains of possible values and by a set of constraints over these variables. They provide a modeling framework for many computer aided decision making practical applications (such as planning, scheduling, time tabling,...). CSP model is extended, for real world applications, in order to optimize a given objective function. The Balanced Academic Curriculum Problem (BACP) has been introduced in [6] and consists in planning the different courses of an academic curriculum on a given set of periods, satisfying some constraints to insure the most suitable organization for the students.

Solving such a problem consists in finding an assignment of values to the variables that satisfies the constraints and optimizes the given criterion. Many resolution algorithms have been proposed and can be classified in two main groups.

⋆ The author has been partially supported by the Chilean National Science Fund through the project FONDECYT N°1060373.

L. Rutkowski et al. (Eds.): ICAISC 2006, LNAI 4029, pp. 410–419, 2006.

Complete methods aim at exploring the whole search space in order to find all the solutions or to detect that the CSP is not consistent. Concerning complete resolution techniques, we are here mainly concerned with constraint propagation with split of domains of variables (e.g., enumeration), i.e., one of the most famous techniques of Constraint Programming (CP) [3].

Incomplete methods mainly rely on the use of metaheuristics providing a more efficient exploration of interesting areas of the search space in order to find some solutions. Most commonly used approaches are based on evolutionary [12,10] and local search algorithms [1]. In this paper, we focus on the first ones and more especially on genetic algorithms (GA).

In order to improve the efficiency of the solving algorithms, combinations of resolution paradigms and techniques have been studied (e.g. [8] presents an overview of possible uses of local search [1] in constraint programming [3]).

The benefit of the hybridization GA+CP is well-known (see [4] , [5]). Most of these works are rather algorithmic approaches which define a kind of master-slave combination, (e.g., LS to guide the search in CP, or CP to reduce interesting area of the search space explored in LS) or ad-hoc realizations of systems for specific classes of problems.

Our purpose is twofold: on the one hand, we aim at solving efficiently BACP by combining a genetic algorithm with constraint programming techniques, and on the other hand, we propose a general modelling framework to precisely design such hybrid resolution process and to highlight their characteristics and properties. This framework allows one to design and manage new and finer solving strategies and extensions.

To this end, we use our uniform generic hybridization framework [15] which is based on K.R. Apt's chaotic iterations [2], a mathematical framework for iterations of a finite set of functions over "abstract" domains with partial ordering. In this framework, basic hybrid resolution processes (such as domain variable reductions, offspring generation, and enumeration) are considered and managed at the same level by a single mechanism. Hence, we may adjust the application rate of the different resolution process in order to model various search strategies. We integrated GA functions and optimization aspects in our constraint system [7] (which is based on our hybrid framework) and solved instances of the BACP. The results show the benefits of our framework and of hybridization as well.

The paper is organized as follows: after having raised the problem in Section 2, we present in Section 3 an overview of GA and CP, confronted with experimentations in Section 4 before concluding in Section 5.

2 The Balanced Academic Curriculum Problem (BACP)

The problem consists in organizing courses in order to balance the work load of students for each period of their curriculum. Each course is given a number of credits representing the amount of work necessary to successfully follow the course. The load of a period is the sum of the credits of each course of the period.

Some more constraints are added: there is a maximum and minimum load per period, and some precedence relationships are established among some courses.

- Parameters:
  m, n: number of courses and number of periods
  α_i: number of credits of course i : $\forall i = 1, \ldots, m$
  β, γ: minimum and maximum academic loads allowed per period
  δ, ϵ: minimum and maximum numbers of courses per period
- Variables:
  x_i: period of course i , $\forall i = 1, \ldots, m$,$x_i \in [1..n]$
  c_i: academic load of course i , $\forall i = 1, \ldots, m$
- Objective function: $Min\ c = max(\sum_{k=1}^{m} c_k \mid x_k = j, \forall j = 1, \ldots, n)$
- Constraints:
  Courses b has course a as prerequisite : $x_a < x_b$
  Each period j has a greater or equal and a less or equal academic load allowed: $\beta \leq \sum_{k=1}^{m} c_k \mid x_k = j \leq \gamma$
  Each period j has a greater or equal and a less or equal number of courses allowed: $\delta \leq \sum_{k=1}^{m} 1 \mid x_k = j \leq \epsilon$

Thus, a solution is a fair assignment of courses to periods: we translate it as the minimization of the highest load period. For a more detailed description, the reader can refer to [6].

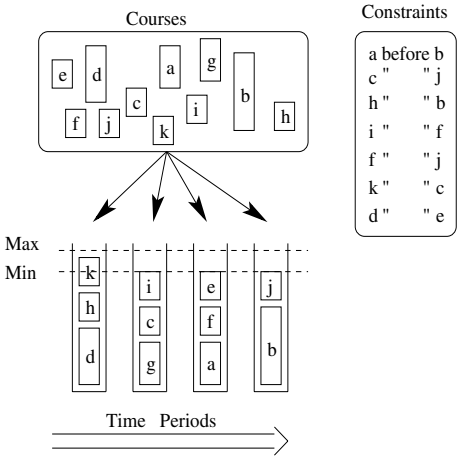

Fig. 1. Example of a courses distribution

As an example, for the 10 periods problem, 42 courses have to be assigned in the 10 periods. Every course has a cost affected from 1 to 5. The number of prerequisite constraints is 32 and for each period the number of courses has to be between 2 and 10, the charge (sum of load) between 10 and 24.

3 The Hybrid Algorithm

We first recall the resolution paradigms related to CSP, optimization, and GA and how they will be integrated in our hybrid algorithm.

3.1 Constraint Programming

A CSP is a tuple (X, D, C) where $X = \{x_1, \cdots, x_n\}$ is a set of variables taking their values in their respective domains $D = \{D_1, \cdots, D_n\}$. A constraint $c \in C$ is a relation $c \subseteq D_1 \times \cdots \times D_n$. In order to simplify notations, D will also denote the Cartesian product of D_i and C the union of its constraints. A tuple $d \in D$ is a solution of a CSP (X, D, C) if and only if $\forall c \in C, d \in c$.

Constraint propagation, one of the most famous techniques for solving CSP consists in iteratively reducing domains of variables by removing values that do not satisfy the constraints. However, reduction mechanisms use one or some of the constraints of the CSP. Thus, they enforce a local consistency property (such as arc-consistency) but not a global consistency of the CSP. These reductions must be interleaved with a splitting mechanism (such as enumeration) in order to obtain a complete solver, (i.e., solver which returns only solutions and does not loose any solution).

Constraint optimization problems, although similar to constraint solving, is comparatively harder because it only accepts solutions (i.e., values of variables) that minimize or maximize a given objective function while satisfying the constraints.

Prerequisite constraints are easily converted to binary constraint and abstracted to arc-consistency reduction functions on course domains. The constraints on periods: sum of loads and number of courses are represented as global constraints and used to prune the search tree by detecting inconsistencies. Thus, for the 8-period problem we have the following global constraints: $load(i, 10, 24), i = 1, .., 8$ standing for the allowed range (10,24) of the sum of load for the period i, and $courses(i, 2, 10), i = 1, .., 8$ for the allowed range (2,10) of the sum of courses for the period i.

During the search, in a given CSP, the global constraint $period(i, \delta, \epsilon)$ computes the number of domains within the value i. If less than δ occurrences of the period i are present in the different domains of courses, then the current CSP is locally inconsistent. The global constraints $load(i, \beta, \gamma)$ counts the charge range for a given period i in the current CSP.

3.2 Genetic Algorithms

Evolutionary algorithms are mainly based on the notion of adaptation of a population of individuals to a criterion using evolution operators like crossover [10]. Based on the principle of natural selection, *Genetic Algorithms* [12,13] have been quite successfully applied to optimization problems such as scheduling or transportation problems.

The key principle of this approach states that, species evolve through adaptations to a changing environment and that the gained knowledge is embedded in the structure of the population and its members, encoded in their chromosomes. If individuals are considered as potential solutions to a given problem, applying a genetic algorithm consists in generating better and better individuals w.r.t. the problem by selecting, crossing, and mutating them. This approach reveals very useful for problems with huge search spaces. We had to adapt some basic techniques and slightly modify some definitions to fit our context.

In the context of GA, for the resolution of a given CSP (X, D, C), the search space can be usually defined with the set of tuples $D = D_1 \times \cdots \times D_n$. We consider populations g of size i, $g \subseteq D$ such as $|g| = i$. An element $s \in g$ is an individual and represents a potential solution to the problem. Our genetic algorithm aims at generating from a population k, a new population k+1 of 60 individuals selected among 100 issued from k. Each time GA is called by the main algorithm, the following different cases can occur:

- The population $k+1$ has less than 100 individuals: an individual is selected randomly; then, either it is coupled with another parent to create 2 children in the population $k+1$, either it is submitted to mutation, or it is not change in the population $k+1$.
- Population $k+1$ has 100 individuals: a selection of the 60 best ones is made according to the evaluation function.

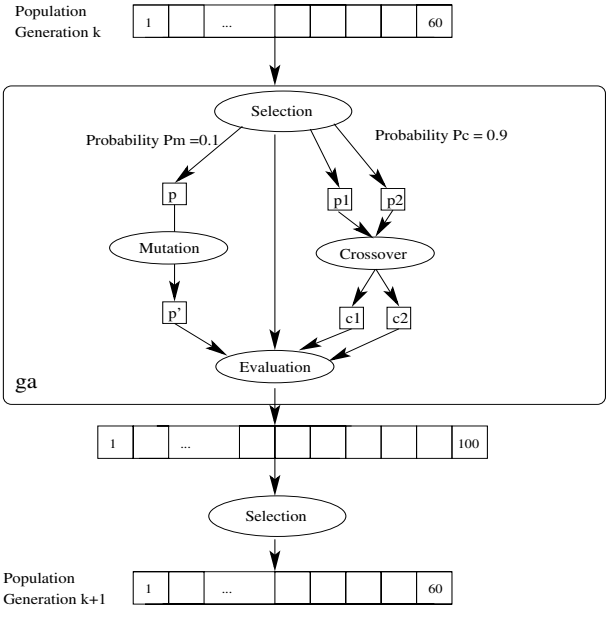

Fig. 2. Genetic Algorithm

3.3 Designing the Hybrid Algorithm

K.R. Apt proposed in [2] a general theoretical framework for modelling such reduction operators. In this context, domain reduction corresponds to the computation of a fixpoint of a set of functions over a partially ordered set. These functions, called *reduction functions*, abstract the notion of constraint. The model is extended with splitting operators and GA in order to model the different hybrid solving methods as the computation of a fixpoint of a set of functions. This model is also now extended to optimization problems such as BACP. The computation of the least common fixpoint of a set of functions F is achieved by the following algorithm:

GI: Generic Hybrid Algorithm

$d := \bot$;
$G := F$;
While $G \neq \emptyset$ do
    choose $g \in G$;
    $G := G - \{g\}$;
    $G := G \cup update(G, g, d)$;
    $d := g(d)$;
endwhile

where G is the current set of functions still to be applied ($G \subseteq F$), $d \in D$ a partially ordered set (the domains in case of CSP), and for all G, g, d the set of functions $update(G, g, d)$ from F is such that:

- A : $\{f \in F - G \mid f(d) = d \wedge f(g(d)) \neq g(d)\} \subseteq update(G, g, d)$.
- B : $g(d) = d$ implies that $update(G, g, d) = \emptyset$.
- C : $g(g(d)) \neq g(d)$ implies that $g \in update(G, g, d)$

This abstract framework corresponds to the hybridization of resolution techniques since we want to use a evolutionary optimization techniques combined with constraint propagation techniques.

Basically, we define three families of functions corresponding to domain reduction functions, split and the generation of a new population by a one step genetic algorithm. These functions are then alternatively chosen according to a given strategy and applied until a fixpoint is reached on the structure (this fixpoint characterizes either an optimal solution or a maximum number of allowed iterations). Therefore, hybridization and strategies are easily usable in this generic context as it will be shown in next section.

4 Experimental Results

In [14], we presented the constraint based solving system we built for hybridization of local search and constraints propagation. We have re-used our system and integrated the GA module (i.e., *ga* functions) in it. We have also added the notion of optimization to the notion of solution we had.

All tests are performed on a cluster with 22 processors used sequentially running at 2.2 GHz with 1 Go of RAM each.

4.1 Selected Problems

We consider the bacp8, bacp10 and bacp12 problems issued from the CSPlib [9] and latest data of these three curricula to form a new problem where some courses are shared by the different curricula. In order to give a point of reference, we present the results of [6] using the linear programming solver lp_solve for the 8-periods and 10-periods problems (Figure 3) and using our hybrid solver (Figure 4). If lp_solve is able to find the optimal solution for the first one, it is not the case for the second one.

Sol quality	bacp 8	Sol quality	bacp 10
24	137.08	33	9.11
23	218.23	32	25.38
21	218.43	30	25.65
20	712.84	29	1433.18
19	1441.98	27	1433.48
18	1453.73	26	1626.49
17 [1]	1459.73	24	1626.84

Fig. 3. Results in seconds using lp_solve

Sol quality	bacp 8	bacp 10	bacp 12
24	0.47	4.71	2.34
23	0.54	4.67	2.40
22	0.61	3.68	2.48
21	0.61	4.36	2.76
20	0.69	4.63	3.20
19	0.83	4.95	4.25
18	1.20	5.13	35.20
17	15.05 [1]	5.60	
16		6.39	
15		8.53	
14		34.84 [1]	

Fig. 4. Results using GA+CP

4.2 Strategies

We control the rates of each family of functions (reduction, split and genetic) by giving as strategy a tuple $(\%_{dr}, \%_{sp}, \%_{ga})$ of application rates. These values corresponds indeed to a probability of application of a function of each family but, in practice, we measure in Fig 5 the real rate of application (i.e., we only count the functions which are chosen according to the strategy and which have a real impact on the resolution).

4.3 Analysis

The most interesting in such an hybridization is the completeness of the association GA-CP, and the roles played by GA and CP in the search process (see Figures 5) : GA optimizes the solutions in a search space progressively being locally consistent (and thus smaller and smaller) using constraints propagation

[1] Optimum found.

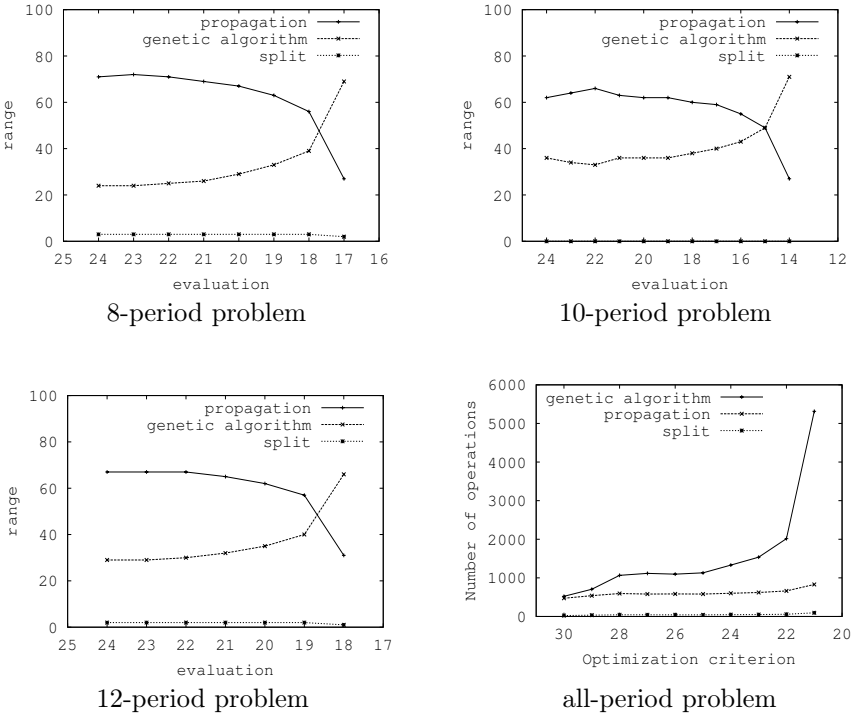

Fig. 5. Evolution of CP vs GA during the optimization process

and split. To evaluate the benefits of each of the components we have measured for CP: the number of effective reductions that are performed and the number of split and for GA: the fact that the next generation is globally better than the previous one.

Concerning the single problems (8, 10, 12), at the beginning, CP represents 70% of the effort: constraint propagation narrows the search space. On the contrary, GA represents about 30%. During this period, not enough local consistency is enforced by constraint propagation, and GA only finds solutions (satisfying all constraints) with a cost greater than 21. Then, at the beginning of the second half of the search process, in terms of costs, CP and GA converge: most of the sub CSP have reach the local consistency and tests over constraints do not improve domain reduction. At the end, GA performs 70% of the search effort to find the optimal solution.

Concerning strategies using GA **and** CP **alone**. In this implementation, CP is unable to find a feasible solution in 10 minutes cpu time. GA is able to find alone the optimal value but is 10 times slower w.r.t. the hybrid resolution $GA + CP$. Therefore, we have not included these results in the tables.

In the graph for the all-period problem, CP and GA start searching with the same efficiency but while CP seems to be stable, most of the operations are performed by the genetic process to obtain better solutions. This could be

explained by the fact that, in this problem, constraints are not strong enough w.r.t. the number of variables and the size of the generated search space. But, in our hybrid resolution system, GA appears as a powerful method even if most of the constraint operators have not reached their fixpoints.

5 Perspectives and Conclusion

Most of hybrid approaches are ad-hoc algorithms based on a master-slave combination: they favor the development of systems whose efficiency is strongly related to a given class of CSPs. In this paper, we have used a more suitable general framework to model hybrid optimization solving algorithms.

The results over the BACP show the benefits of our framework and of hybridization. They also allow us to identify the interaction between the different resolution mechanisms. Such studies could be used to tune general purpose hybrid solvers in the future.

A future extension will consist in providing "tools" to help designing finer strategies in the GI algorithm in our particularly suitable uniform framework. To this end, we plan to extend works of [11] where strategies are built using some composition operators in the GI algorithm. Moreover, this will also open possibilities of concurrent and parallel application of reduction functions inside the model.

References

1. E. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., 1997.
2. K. R. Apt. From chaotic iteration to constraint propagation. In *Proceedings of ICALP'97*, pages 36–55. Springer-Verlag, 1997.
3. K. R. Apt. *Principles of Constraint Programming*. Cambridge Univ. Press, 2003.
4. N. Barnier and P. Brisset. Combine and conquer: Genetic algorithm and cp for optimization. In *Proc. of CP'98*, page 463, LNCS 1520, Springer, 1998.
5. E. K. Burke, D. Elliman, and R. F. Weare. A hybrid genetic algorithm for highly constrained timetabling problems. In *Proc. of ICGA*, pages 605–610, Morgan Kaufmann, 1995.
6. C. Castro and S. Manzano. Variable and value ordering when solving balanced academic curriculum problems. In *Proceedings of 6th Workshop of the ERCIM WG on Constraints. CoRR cs.PL/0110007*, 2001.
7. E. Monfroy and F. Saubion and T. Lambert. Hybrid CSP Solving. Proceeding of 5th International Workshop Frontiers of Combining Systems (FroCoS). LNCS 3717, Springer, 2005
8. F. Focacci, F. Laburthe, and A. Lodi. Local search and constraint programming. In *Handbook of Metaheuristics*, Kluwer Academic, 2002.
9. I. Gent, T. Walsh, and B. Selman. http://www.4c.ucc.ie/ tw/csplib/, funded by the UK Network of Constraints.
10. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.

11. L. Granvilliers and E. Monfroy. Implementing constraint propagation by composition of reductions. In *Proc. of ICLP'2003*, *LNCS 2916*, pages 300–314. Springer, 2003.
12. J. H. Holland. *Adaptation in Natural and Artificial Systems.* 1975.
13. K. A. D. Jong. *An analysis of the behavior of a class of genetic adaptive systems.* Phd thesis, University of Michigan, 1975.
14. T. Lambert, E. Monfroy, and F. Saubion. Solving strategies using a hybridization model for local search and constraint propagation. In *Proceedings of ACM SAC'2005.* ACM, 2005.
15. E. Monfroy, F. Saubion, and T. Lambert. On hybridization of local search and constraint propagation. *Proc. of ICLP'04*, *LNCS 3132*, pp 299–313. Springer, 2004.