Proceedings of the 6th WSEAS Int. Conf. on EVOLUTIONARY COMPUTING, Lisbon, Portugal, June 16-18, 2005 (pp214-219)

Constraint Programming and Genetic Algorithms to Solve Layout Design Problem

JOSÉ TAVARES
GECAD – Knowledge Engineering and Decision Support Group
Institute of Engineering – Polytechnic of Porto
Rua Dr. António Bernardino de Almeida, 4200-072 Porto – PORTUGAL
Phone: +351 2 8340500, Fax: +351 2 8321159,
http://www.dei.isep.ipp.pt/~jtavares

Abstract: - Real world optimization problems are typically complex and difficult to solve. In this work it is intended to address these problems through a combination of two techniques: Constraint Logic Programming (CLP) and Genetic Algorithms (GA). This approach aims to benefit, on the one hand, from the easiness and naturalness of the CLP to express problems whose formulation is based on constraints, and on the other hand, from the ability that GA have in attaining good solutions to a particular problem, manly when specific and efficient methods to solve the problem suitable way do not exist. As a case study these ideas were tested to solve the Facility Layout Problem which is one of the most difficult problems that face researchers experimenting with complex systems to real world applications. It relies with the design and location of production lines, machinery and equipment, inventory storage and shipping facilities.

Key-words: Facilities Layout Design, Constraint Satisfaction, Constraint Logic Programming, Genetic Algorithms.

1 Introduction

This paper describes the work done by our research team that originates the new starting RECEO project. The project main goal is to combine the Constraint Logic Programming (CLP) [1] with Genetic Algorithms (GA) [2,3,4] to solve certain kind of optimization problems. These problems are essentially characterized by holding a wide variety of constraints and by presenting, most of times, a too much large solutions space to be computationally practicable the search for the optimal solution. In the end we expect have a combination of CLP with GA that represents a new approach that combines the advantages of the techniques solve problems. CLP will offer an ease and natural way to express problems whose formulation is based on constraints and that GA have the ability in attaining good solutions to a particular problem, manly when specific and efficient methods to solve them in a suitable way do not exist.

1.1 The Technology

In the last decade CLP emerged as new technology to deal with complex combinatorial problems. This technology matches the declarative aspects of the Logic Programming (LP) paradigm with the techniques for constraint satisfaction [5] in a proper way for problem solving. This hybrid technique improves the search strategies used in logic programming, once it adds constraints and consistence verification techniques. With this scheme, the solution space can be largely reduced.

The constraints and consistency verification techniques were initially developed to solve the Constraint Satisfaction Problems (CSP), which for a long time had been an Artificial Intelligence (AI) research field. Many combinatorial problems, characterized by a large number of constraints, are well suited for CLP, namely scheduling problems, timetabling, planning, placement, configuration, and routing. Others areas of application goes from the natural language processing, to the circuit analysis and games theory. CSP seeks assignments to a set of variables $X = \{x_1, x_2, ..., x_m\}$ from a set of corresponding domains $D=\{d_1, d_2, ..., d_m\}$, one per variable, satisfying a set of constraints $C=\{c_1, c_2,..., c_n\}$ over subsets of the cartesian space spanned by D. CSP is a binary problem, in which a set of assignments to the variables X satisfies or not all the constraints [6]. A solution for a CSP is a domain value assignment for each variable, in a way that all the

constraints are satisfied. It has been verified that CLP offers a more natural way to express real world problems in a computer program, the development time is shorter, the maintenance processes are simpler and the efficiency is equivalent to that of the programs developed in procedural languages according to the paradigm of constraint satisfaction [6].

Since the end of the eighties the CLP technology, and in particular the Constraint Logic Programming with Finite Domains (CLP(*FD*)) [1,6], has been applied successfully solve problems in several areas where other technologies had lapsed. It provides declarative, abstract and an elegant way to specify problems. The systems based on constraints present a strong theoretical component, being equally appealing to the industry. Many of these problems present common features to the combinatorial problems and, therefore, they are difficult to solve.

1.2 Test Case Problem

The Facilities Layout Problem (FLP) was the first problem selected to experiment our ideas for combining CLP with GA. We remember that is our intention to extend this approach to a general framework. FLP is one of the most complex problems in the industry. It is defined as the planning of the proper location of machines, employees, workstations, warehouses and client service areas. It also involves the design of the material and people flow pattern around, the movement inside, at the input and at the output of the productive plants. In a factory, the layout is a fundamental issue. From it, the equipment and human resources have a great influence on the real output, whatever is the manufacturing plant's theoretical installed capacity. It is necessary to plan the operations scheduling among the available equipment for each operation type and the flow of the materials and people among them. The warehouses location, how they are supplied from outside, the areas and how the distribution transportation is loaded are also tasks of the planning process. Issues related with layout, like work conditions (noise levels, temperature and air quality), have to be considered. The correct design and the dynamic management of the manufacturing plant is a manager's fundamental task in order to have an efficient manufacturing process using the available material and human resources.

The FLP was originally defined by [7], [8] and [9]. Given the complexity of the FLP, a strong effort was given in the research and development of techniques to assist layout design specialists [10,11,12]. These techniques use procedures classified as optimal and sub optimal algorithms. For the first ones, the attainment of the optimal solution for problems with some dimension has shown problematic and, therefore, other ways were explored that give good solutions in useful time. These algorithms are in the group of the sub optimal algorithms.

In the modern manufacturing systems, the traditional FLP assumptions are more and more difficult to support. In first place, there is a tendency to consider a third dimension given, for example, lighter machines, higher prices of the available areas, among others. In second, it is evidenced that in the current industrial environment, there is a strong trend for an increasing level of volatility and uncertainty, where more and more companies are present in a global market. It is also evidenced, an increasing technological innovation and changes in the specifications of the products, these demanded by the consumers. All these factors contribute to reduce the life cycle of a manufacturing layout.

1.3 Solving the Problem

Solving the FLP with the CLP(*FD*) technology requires, however, the development of new models or, at least, the adaptation of some models that are already been used. In our particular case we had to take in account the aspects related with the use of the CLP(*FD*) technology. The model was inspired in models of space assignment problems [10,11,12]. The identification of the problem variables as well the definition of its domains and the specification of the constraints, that obviously have a geometric nature, were important concerns in the model development.

Very early experiments to solve FLP using CLP(FD) with and uses a Branch&Bound (*B&B*) algorithm for the optimization task showed this requires a huge computational power and is not practicable to explore the entire search space for real world problems. This scenario suggests that other optimisation techniques should be used in order to deal with such huge search space. The chosen technique was GAs.

The experiments done by combining CLP with GA in order to solve the FLP resulted in a system module named as LayGeRL. This approach aims to benefit, on one hand, from the easiness and naturalness of the CLP to express problems whose

formulation is based on constraints, and on the other hand, from the ability that GA have in attaining good solutions to a particular problem, manly when specific and efficient methods to solve the problem in a suitable way do not exist. In fact, in the method developed the CLP has the task of constraint reasoning and GA the optimisation task. In fact, this combination of techniques showed not only a great potential to solve FLP but others kinds of complex problems.

2 CLP Modeling for the FLP

This section describes the model develop in order to solve the FDP. In general, the model used follows the multi-row model [10], where the production units can be located anywhere in the facility plant and deals only with single-floor facilities. The manufacturing plant is modelled considering a rectangle with width *W* and length *L* which surrounds its shape as shown in Fig. 1. All production resources are located inside of that rectangle. Locations inside the rectangle that do not belong to the facilities plant are not considered by imposing some position constraints that are going to described below.

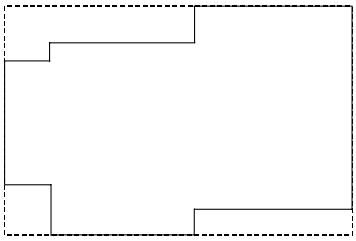

Fig. 1: Rectangle surrounding the Manufacturing Plant.

Production units (PU) are the production resources to be located in the plant. These could be a simple workstation with a machine and, optionally, with a small area for temporary storage of materials, or a collection of workstations where the facility itself is a layout sub problem. In our model it is also acknowledge that there are some alternative PUs to perform the same operation. To the set of PUs that are able to carry out the same process operation we call a PU class.

As in the plant, a surrounding rectangular shape is used to represent the actual PU shape. Fig. 2 shows the rectangular model for two PUs. This rectangular shape is as a function of three parameters: the *width*, the *length* and the optional *gap* value that represents the minimal distance that has to be respected in relation to the others PUs.

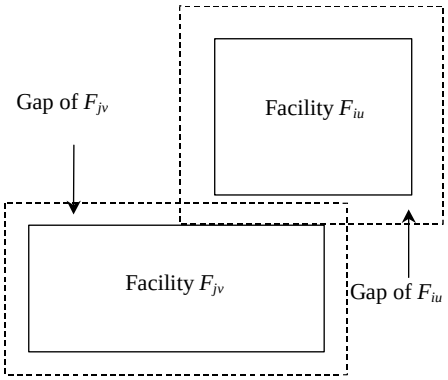

Fig. 2: Two facilities with a rectangular shape approximation.

PUs require always a constant minimum area, but their width and length may change inside predefined limits. Considering the areas as fixed, it is obvious that, if the width increases then the length has to decrease. This approach is used to deal with two situations. One situation occurs when a PU is a department or a cell containing several machines requiring a given area, but with flexible shape. The other occurs when the PU has fixed area and shape, and it must be decided what will be the best orientation in the plant (north-south or east-west).

To evaluate the quality of the layout solutions it is possible to consider a large number of cost functions types. Those cost function types may use qualitative and/or quantitative factors. The selected cost function focuses on quantitative factors. It is based in two parameters: the product demand (which defines the flow volume) and the distance between each PU pair having material flowing between them. This leads to the solution cost computation made by the expression (1), where f_{iujv} is the flow value between PU *u* of class *i* and PU *v* of class *j* and d_{iujv} is their relative distance being usually given by an euclidean or a rectilinear metric. This distance is dependent, obviously, from the position where the facilities are placed.

$$Custo = \sum_{j=1}^{n-1} \sum_{i=j+1}^{n} \sum_{u=1}^{NI_i} \sum_{v=1}^{NI_j} f_{iujv} \times d_{iujv} \qquad (1)$$

It is obvious that the PUs in the plant placement have to obey to some constraints. Some of them are implicit to the problem in order to get a correct layout; others are related to the problem specific requirements or user requirements, like technological ones,

environmental ones, strategic ones and others. It was identified some geometric constraints that are supposed to cover a wide range of FLP cases.

No Overlap is the constraint that should always be present and which imposes that any PU must be placed in the plant in such way that is not going to overlap with the others;

Neighbourhood is used to deal with situations where it is desirable to locate two PUs close to each other as, for example, when there is a large volume of material flowing between them;

Distance is a constraint used to impose a given relation of distance between two PUs or between a known point and a PU. One possible situation occurs when some PUs have to operate in a temperature-controlled environment not compatible with others, located in the neighbourhood;

Absolute Position constraints are used to force PUs to be located, either inside or outside of a given area of the manufacturing plant. With these constraints, it is possible to reserve space areas for different purposes like offices or warehouses. These constraints are also used to prevent the location of the PUs in areas that are not inside in the real plant but are inside of the plant's surrounding rectangular shape;

Relative Position constraints are the ones that make possible handling situations like, for example, "PU A is at right of PU B". There are four possible relative position constrains: 'at right of'; 'at left of'; at front of' and 'at back of';

Orientation constraints deals with situations like the ones that it is necessary to constraint the orientation of a PU or define that several PUs have some kind of relation in terms of its orientation.

3 The Optimization Algorithm

Solving a problem following the CLP(*FD*) paradigm usually involves three steps: i) the definition of the problem variables and their domain; ii) the statement of constraints and, finally, iii) the solution enumeration step, which instantiates the variables, one by one, with a value from their domain. Frequently, the last step is an optimization task using a B&B algorithm. The next subsection describes the developed approach to combine CLP paradigm with GA for the optimization task.

3.1 The combination approach

The approach followed is similar to the work done in order hybridize a B&B algorithm and GA [13]. Their work follows three main principles: i) use current problem encoding; ii) hybridize if and where possible; iii) and adapt the genetic operators [3]. In this work the GA operators are implemented in CLP paradigm as illustrated in Fig. 3. The main process is on the GA side. This process can be viewed as the client and the CLP engine can be viewed as the server. The main process needs to start the CLP engine, to be able to use its services (see Fig. 4). When the CLP starts, it begins by creating the problem variables with their finite domain, and then places the constraints according to the problem specifications. The CLP engine resumes to its start up by returning its current state to the main process.

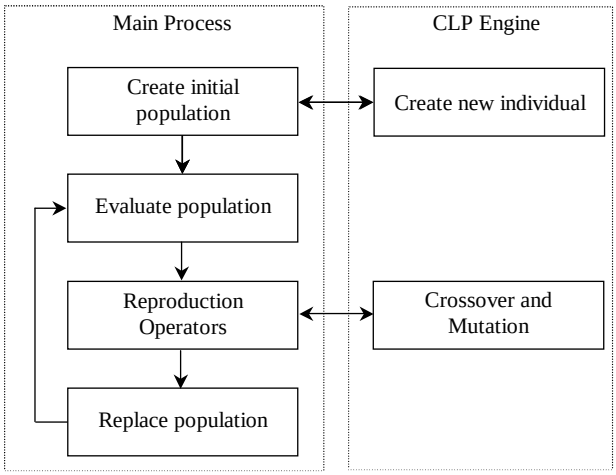

Fig. 3: Outline of the CLP and GA combination.

The main process uses the state returned by the CLP engine to be able to create its initial population, perform the crossover and mutation operations, and finally, evaluate the individuals (see Fig. 5). The individuals produced by the CLP engine (creating a new individual or executing a crossover and a mutation operation), are consistent with the problem constraints placed during the CLP engine start up.

```
procedure MainApp
begin
        <initialisation stuff>
        Clp_State ← clp_startup
        Min ← GA_Optimise(Clp_State)
        <exit stuff>
end
```

Fig. 4: The main process source code skeleton.

Once the CLP engine is started, the optimisation task begins. This task is a GA like the source code skeleton showed in the Fig. 5. The operations in italic with the name starting with *clp_* are implemented in the CLP(*FD*) paradigm.

```
procedure GA_Optimise(Clp_State)
begin
    t ← 0
    P0 ← clp_create_initial_population
    clp_evaluate (Clp_State, P0)
    while not Final Condition do
        Pt' ← select_from Pt
        Pt'' ← clp_crossover (Clp_State, Pt')
        Pt''' ← clp_mutate (Clp_State, Pt'')
        clp_evaluate (Clp_State, Pt''')
        Pt+1 ← replace (Pt, Pt''')
        t ← t + 1
    end
end
```

Fig. 5: The GA skeleton with operators implemented using the CLP paradigm.

3.2 The Genotype Representation

Since CLP engine executes all the GA operators, the representation of the solutions is done directly using the Logic Programming (LP) like data structure syntax. Each individual in the GA population is only a reference to its respective LP representation. The genotype of each individual is a list of genes, where each one contains information about the respective PU.

3.3 Recombination

The recombination is done by the CLP engine and in a certain way the developed recombination operator for FLP performs a slightly form of mutation to ensure that the result of this operator will be consistent with the problem constraints.

The recombination starts by breaking the parent genotype in two random halves. The length of the two halves and the genes in each half are also random. After breaking the parents in two halves, the recombination operation is carried out and the generated offsprings should be consistent with the problem constraints. However, it may happen that the recombination operation fails to generate an offspring. This happens when the operator cannot locate the facilities (genes) of the second half in the available space of the manufacturing plant, given the location of the facility of the first half and the problem constraints. A null fitness value is assigned to those failed offsprings and they die before the next generation.

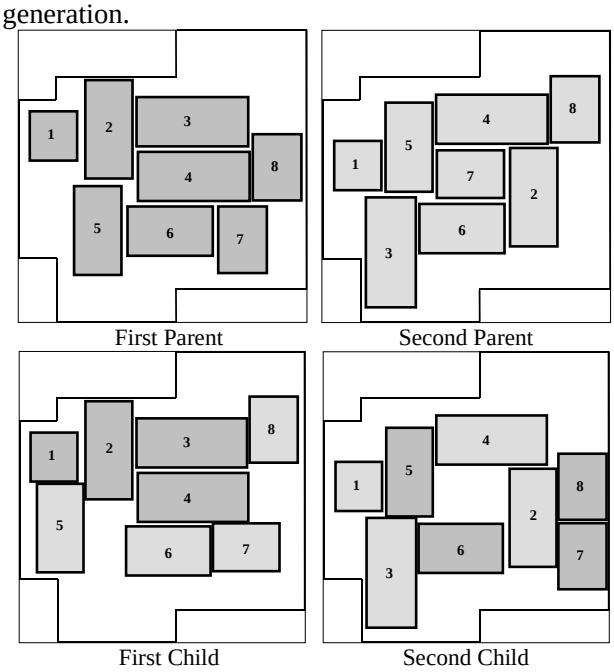

Fig. 6: The crossover operation.

3.4 Mutation

As it was referred in the previous section the recombination operator has a side effect, which consists in one kind of mutation. This kind of mutation modifies slightly the position of some facilities. However, it is desirable that, from time to time, the orientation or the shape of some facilities gets also modified. Among different possible mutation operators, we selected the one that operates like the one illustrated Figure 6. It shows an example of two mutated random selected genes. As result e shape of the respective facilities was modified.

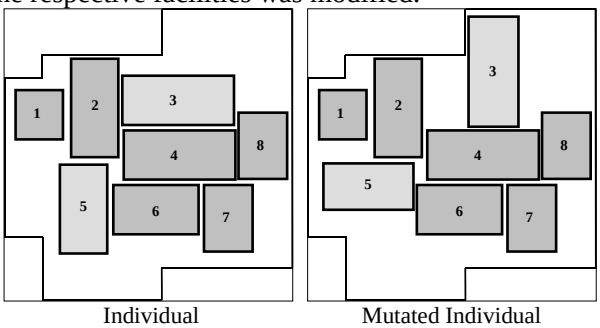

Figure 1: The result of a mutation operation when applied to an individual.

4 Conclusions

As referred in the beginning, this paper describes the work done by our research team that originates the new starting RECEO project. The project goal is to combine the CLP with GA to address difficult optimisation problems. These problems are essentially characterized by holding a wide variety of constraints and by presenting, most of times, a too much large solutions space to be computationally practicable the search of the best solution. It is intended that the referred approach will result in an original tool that finds a large applicability.

In the recent work of the research team this approach has been explored as a specific method to solve FLP. This approach aims to benefit, on the one hand, from the easiness and naturalness of the CLP to express problems whose formulation is based on constraints, and on the other hand, from the ability that GA have in attaining good solutions, manly when specific and efficient methods to solve the problem ina suitable way do not exist.

As it was referred, the work already done consists in a specific approach for the case of FLP. However, on the basis of the results and on the experience acquired, the combination of these two techniques presents interesting potentialities to evolve to a general-purpose tool to solve optimisation problems. In fact, the consolidation of this approach as a generic tool is one of the main and innovative project goals.

One limitation of the work already developed until the moment is related with the way of dealing with constraints. These, for the problem in question, will have to be always satisfied (hard-constraints). This fact is not sustainable in all cases, given that many times there are situations where the constraints satisfaction requirements have different levels in a way that they could be hold or not (soft-constraints). This is an aspect to be considered in the tool that will result from RECEO project with the development of a mechanism that allows soft-constraints reasoning.

The biggest difficulty in the combination of these techniques is to find a suitable problem solution representation and to choose the operators that, in principle, can be applied to the GA evolution. In this situation it is intended, beyond the development of operators with generic characteristics, to provide the ability to develop new operators, possibly specific ones to solve the problem in question and which incorporate characteristics of the techniques originally used to solve them.

References:

[1] Frühwirth, T., Herold, A., Küchenhoff, V., Provost, T., Lim, P., Monfroy E., and Wallace, M. (1993). Constraint Logic Programming - An Informal Introduction, European Computer-Industry Research Centre.

[2] Holland, J. H. (1975) Adaptation in Natural and Artificial Systems. Univ. of Michigan Press, Ann Arbor, MI.

[3] Davis, L. (1991). Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, NY.

[4] Michalewicz, Z. (1992). Genetic algorithms + Data Structures = Evolution Programs, Springer-Verlag, New York, NY.

[5] Kumar, V. (1992). Algorithms for Constraint Satisfaction Problems: A Survey, AI Magazine 13(1):32-44, 1992.

[6] Jaffar, J., and Lassez, J. (1987). Constraint logic programming. In Proceedings of the 14th ACM Symposium on Principles of Programming Languages, Munich, Germany, pages 111-119.

[7] Koopmans, T. C., and Beckman, M. 1957. “Assignment Problems and the Location of Economic Activities”. Econometrica, 25, pp 53-76.

[8] Armour, G.C., and Buffa, E. S. (1963). A heuristic algorithm and simulation approach to relative location of facilities. Management Science, 9:294-309.

[9] Vollman, T. E. and Buffa, E. S. (1966). Facilities layout problem in perspective. Management Science, 12:450-468.

[10] Heragu, S. (1997). Facilities Design, PWS Publishing Company, ISBN 0-534-95183-X.

[11] Heragu, S. S. and Kusiak, A., 1987. “The Facility Layout Problem”, European Journal of Operational Research, 53, pp 1-13.

[12] Montreuil, B. H., Venkatadri, U., e Ratliff, H. D. 1993. “Generating a layout from a design skeleton”. IIE Transactions, 25(1), pp 3-15.

[13] Cotta, C., Aldana, J.F., Nebro, A.J., and Troya, J.M. (1995). Hybridizing Genetic Algorithms with Branch and Bound Techniques for the Resolution of the TSP. Artificial Neural Nets and Genetic Algorithms, D.W. Pearson, N.C. Steele, R.F. Albrecht (eds.), Springer Verlag Wien - New York, 277-280, ISBN 3-211-82692-0.