Constraint Logic Programming Applied to the Resolution of a Problem of picking of Warehouse

SIGMUNDO PREISSLER JUNIOR[1]
CLAUDIO CESAR DE SÁ[2]
FERNANDO DEEKE SASSE[3]

UDESC - Universidade do Estado de Santa Catarina
Departamento de Ciência da Computação (DCC)
Campus Universitário Prof. Avelino Marcante, S/N
Centro de Ciências Tecnológicas (CCT)
Joinville - SC - Brasil
[1]sigmundo@unerj.br
[2]claudio@joinville.udesc.br [3]fsasse@alumni.uwaterloo.ca

Abstract. In this paper we apply logical programming with constraints to the problem of minimizing operational costs of a mobile agent that searches and moves objects in shelves, obeying a known physical configuration.

Keywords: Logistic, Constraint Logic Programming, CSP.

(Received June 25, 2007 / Accepted December 07, 2007)

1 Introduction

The constraint programming approach consists of searching for a state of the world in which a large number of constraints are satisfied simultaneously. In this context a problem is typically stated as a state of the world containing a number of unknown variables. The constraint program searches for the values of these variables in certain domains.

Constraint programming (CP) is a programming paradigm where relations between variables can be stated in the form of constraints and consists of an embedding of constraints in a language. Inside the scope of CP, where the problems are solved by propagating their constraints, there are the constraint satisfaction problems (CSP). A CSP is defined by a set of variables, $X_1, X_2, \ldots, X_n$, and a set of constraints, $C_1, C_2, \ldots, C_m$. Each variable X_i has a nonempty domain D_i of possible values. Each constraint C_i involves some subset of the variables and specifies the allowable combinations of values for that subset. A state of the problem is defined by an assignment of values to some or all of the variables, $\{X_i = v_i, X_j = v_j, \ldots\}$. An assignment that does not violate any constraints is called a *consistent* or*legal* assignment. A complete assignment is one in which every variable is mentioned, and a solution to a CSP is a complete assignment that satisfies all the constraints. Some CSPs also require a solution that maximizes an objective function [12].

Another approach to deal with these problems consists in using programming in logic, which is known as CLP [8]. Usually CLP is considered as a form of CP, in which logic programming is extended to include concepts from constraint satisfaction (the process of finding a solution to a set of constraints). The CLP exhibits many important features, like logical variables, backtracking, declarativeness, where the programs are easily developed and maintained. The logic programming languages are mostly Prolog-like and its derivatives [5]. The CLP can also be viewed as a strategy for representing descriptions and finding solutions of combinatorial problems, in many areas such as planning, scheduling, DNA sequencing, nesting problems, etc [2, 3].

This paper addresses a problem of logistics in the process of searching for products in a storage area of a warehouse. The solution provided by CLP is applied to solve the problem of finding a solution that minimizes operational costs associated with the length of the path used to accomplish the task.

This paper is organized as follows: In Section 2 we present the relationship between logistics and problems of origin implementations using CLP. In section 3 is presented CLP of the technique and its applications in areas related to the problem of collection picking in the warehouse. A modeling to the problem, proposed [11] is presented in the session 4. In section 5 is the experiment using CSP. In section 6 are shown the results and comparative analysis between the models presented.

2 Issues section of Logistics

Problems from operational research (OR), in combinatorial optimization (CO) and the logistics of distribution (LD) can typically be applied to searches that relate the establishment of routes [11] in search of better path. The travel salesman problem [12] is a typical example in the area of logistics and the search for the best path. If a traveler wish to visit *(*n) cities, distant among themselves and between them there are ways (edges or arcs) by which the traveler can move around, how to find the route of least cost? No daily lives of industries is no different and various problems similar to this are found. Some basic features such as entry and exit of goods in storage areas can generate delays when these processes are not properly optimized. The aprooaches for solution in these types of situations are diverse as: the use of the theory of graphs [11], search algorithms, in-depth [7], simulation of a colony of ants [12] and constraint logic programming [6]. An example of modeling and application using CSP is presented in the next session.

3 Constraint Logic Programming

In several moments of our day-to-day constraints are used in the search for resolution of problems of common sense. Intuitively, can be considered as a constraint in a space of possibilities [13]. The restrictions are limitations to the possible values that variables can take a problem within a certain domain [1]. We can also say that this deal with a rule or set of rules imposed on variables in a problem, possibly included in these fields. The CLP [13] uses a computer model that combines the declaratory nature of the logic programming in the efficiency of methods of solving problems using the restrictions. That is, it is the union of two distinct natures, on the one hand the programming logic and the other theorems restrictive [3]. The constraint satisfaction problems deals with two classes of problems: fields and finite or infinite fields complex The problems of satisfaction of restrictions can be specified [1] by (V, D, R), where:

- V:set of variables is used in the modeling of the problem.
- D:is the area(s) in which the variables of V may have values.
- R:is the set of restrictions that affect the variables V.

Each restriction is applied to a subset of variables, conditional their values. An assignment is not known if this consistently violate any constraints. The solution is found when all variables have a value consistent [12]. Some classic examples of applications that use as a method of resolving the programming by restrictions are: coloring on maps, management of network (traffic data), digital circuits, production management, scheduling of tasks, time tabling [12]. Another widespread problem in the area of AI is the Travel Salesman Problem (TSP) [10]. A traveler needs to visit n differents cities, starting and ending their journey in the first city. No matter the order in which cities are visited, the traveler must go directly from one city to its neighbor visiting all of them. The problem is to find the shortest path for the traveler visit all cities.

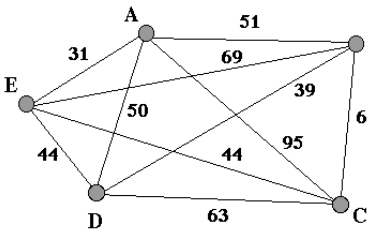

Figure 1: Travel Salesman Problem

- **Variables:** Order in a citie that is visited;
- **Objective:** Determine the best (shorter or lower cost) route to be traveled by the traveler;
- **Constraints:** Do not repeat visits to cities and ensure the global cycle;
- **Domain:** Finite Domains (number of cities)

4 The Problem

The problem presented here [11] deals with the search for the best way and lower operational cost to a network of products distributed in a Warehouse. In industry, such problems are largely found.

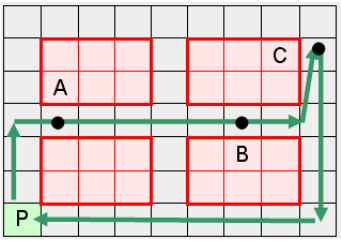

Figure 2: Directions to be traveled

The figure 3 presents the prototype of organizing useful area of the warehouse. The definition of points of location (nodes), allows adequate handling equipment. For comparison of results is efficient, we are using the modeling proposed by [11], where the edges that connect the nodes are used for the definition of 'road' to the locomotion of products in the store. The sequence number of nodes was increased up to one and following from left to right and from the bottom up [11].

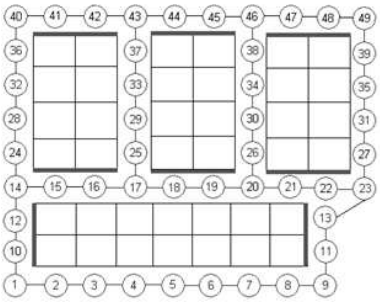

Figure 3: Prototype for definition of variables

The areas highlighted on the shelves symbolize the points blind to each of them. This is the place where the product can not be accessed. Thus, it would be impossible to collect any product by node number 12, for example. However, if the employee was in the node 26, could collect both two products. The characteristics are:

- One shelf contains a finite number of products.
- Each node has one or more related products.
- The node have connections between them that determine the possible paths.
- The distance between a node and another has the same fixed amount.

For better understanding of the configuration and connection between products and nodes opted by the model explained in Figure 4. The graph, Figure 4, is the end result, after the definition of products related to nodes. The products are coded 1 to 38. That is, thirty-eight different products associated with their respective shelves. Each of the 49 we have an effective configuration of that product(s) may or may not sign. The road is the link between nodes (edges). For the solution of the problem

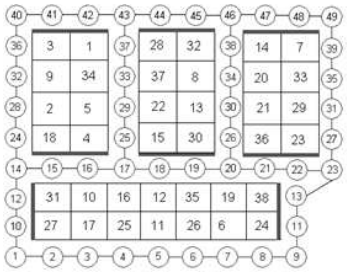

Figure 4: Relations between products and nodes

is proposed sought to formulate an algorithm using the language PROLOG and making use of the characteristics of Constraint Logic Programming. Then the results are evaluated after the implementation in comparison to the application proposed by [11] on their work [11]. The definition of variables, fields and restrictions to the problem and the results are discussed in the session below.

5 The Experiment

In his work, [11] presents a solution to the problem proposed using the Hamiltonian method. The application makes use of the algorithm, *(*Diskjtra) for the search of the nodes in a graph. Problems pleased restrictions can be applied under use of boolean fields, finite, real or restrictions linear intervals. The basis of this system of constraints is the use of finite fields CLP(FD), commonly applied to the problems of constraints. It is the use of finite subset of values to the variables of field. For the development of this application, has chosen is to use the GNU-PROLOG and its library Constraint Satisfaction Problems for Finite Domains (FD) [1]. The steps used for the construction of the algorithm were: definition of nodes, definition of which products can be accessed by node, the definition of link between nodes, or the edges. As the distance between nodes is a fixed amount, set up in this algorithm. Later the constraints were implemented between

variables through requisition arising from the user. An example: for a requisition of 3 products is defined in the code the need for at least one occurrence of each product required in the solution as well as the constraint of the result by the start node one and finalize this same node. The CLP(DF) provides the interpreter various types of predicates for use restrictive. These are divided into two classes: arithmetic and symbolic. The restrictions arithmetic deal with problems in use of equations of difference $(v_1 \neq v_2)$, equal$(v_1 = v_2)$ or inequations $(v_1 < v_2$,$v_1 > v_2$,$v_1 \leq v_2$ or $v_1 \geq v_2)$. They can also express whith conectives logical: conjunction, disjunction, involvement, equivalence or denial. Conversely, the constraint symbolic use predicates that can express no arithmetical relationship between the variables of the area. These are developed for solving problems of staggering, combinations, grouping, unique among others. But can syntactic change in the use of different interpreters. The simplest form of the use of consistency is known for consistency per node. Obtained removing themselves individually the values of each variable field. This, therefore, it is a problem of binary type CSP restrictions exist for not involving more than two variables at a time. As a first attempt in search of the shortest path using CSP, chosen by the method proposed by [4] believes that the consistency of arc as a means to obtain the solution. The technique uses restrictions between variables in order to determine that the parent node is less than or equal to the child node, and so on. In the first experiment, the variables have been declared and assigned in order to represent only part of the circuit total proposed in the model. The time resulting from the search of a product was around seven seconds. Later, with the increase in the area of search and the increase in the number of requests, represented by a sequence of products to be collected, the time has risen exponentially. This is a first experiment that did not achieve satisfactory results opted for the change in restrictions. The conditional representing the sequence of the variables to be visited that meet the criteria of less than or equal to the father son were withdrawn. To find the shortest path was used the method where the algorithm seeks the solution by dividing the list of variables in parts. The model proposed here considers the variables we likely to visit by the inspector. The move is the node to each node five and return to a node agent that travels a range of nine variables. Therefore, the greater the number of variables visited, the greater the path found by the search algorithm of the solution. So this is a model where an official part of a starting point, it collects the products requested and returns to the point of origin, the algorithm has restrictions on the value of departure and arrival of the agent.

6 Results

In the proposed experiment variables were considered as the nodes to be covered by the agent during the collection. Based on this information and know that the area of search is understood by 49 we held was a reading time for the processing of the application when in situations where the number of nodes is increased. It began with 50 variables, later, 100, 200 to 700. The processing times are expressed in the graph below which shows a linear behavior. The elevations are perceived in relation to the time that ranged from 100 to 810 milliseconds. A considerably good time, for it was processing that did not register one according to 800 variables. This measurement of time in relation to the number of variables was important to record that the increase in processing time for a graph of greater size is accepted. That is, if there is a need to search in a larger space, resulting from growth in the number of products this algorithm or we will not have a response time consuming. To make a comparison of performance using the two

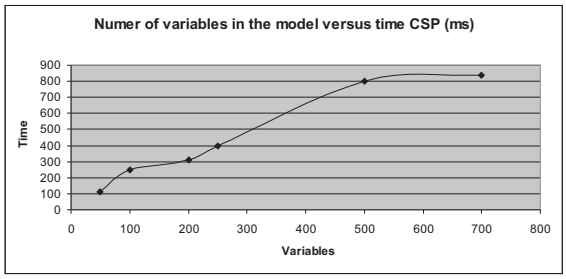

Figure 5: Number of variables in the model versus time CSP(ms)

technologies for development: the Hamiltonian cycle and use the CLP, we primarily by increasing the number of requests to the system and achieve far as processing time taking into account a similar configuration of hardware for the two experiments. The requests are accepted these models as: a request by the user, which want to collect products. The products have already registered relationship with us in the model described above CLP. Each node can access one or more products. The processing time for the algorithm proposed by [11]. Are prepared in the chart below. This assessment is obtained by measuring the processing time obtained from the beginning of the request until its final response. The figure below shows the relationship between requests versus processing time, obtained by the use of the algorithm Hamiltonian cycle [11]. The time had is between one and fifty seconds.

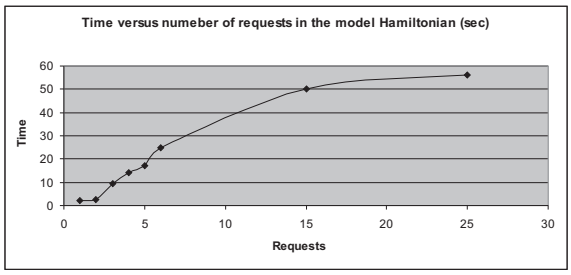

Figure 6: Time versus number of requests in the model Hamiltonian (sec)

Compared with the same number of requests (requests) to the system, the algorithm that uses CSP showed variation between 80 to 180 milliseconds. The results did not reach the mark of the second.

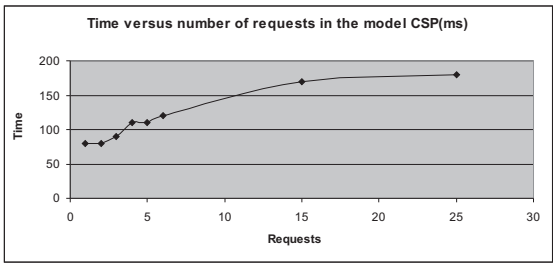

Figure 7: Time versus number of requests in the model CSP(ms)

We can see that in both cases the increase in the number of requests is also associated with the increase in processing time. In a comparative analysis between the two models: Hamiltonian cycle and CLP, we can see that there is an increase in processing time when the number of requests the system also increases. As they increased the number of requests, the Hamiltonian method presents a higher growth in relation to the CLP that remains constant suffering slight changes of growth. These findings are expressed in the above charts that show increases in the brand of seconds using Hamiltonian cycle in milliseconds using CLP.

7 Conclusions

The main motivations for carrying out this experiment were: the pursuit of knowledge necessary for the resolution of problems through the CLP, their effective use in practical applications of day-to-day industry and proportional dissemination of knowledge of this area that extends logic programming and constraint satisfaction. The constraint logic programming represents a classic approach of programming that requires the individual with a total dedication to the modeling of the problem, making the hidden effective process for resolution [13]. The main contribution of CLP is the reduction of lines of code of the applications. This technique aims to reduce the time to search for the use of variables restrictive. The more restrictive is a variable smaller is the time spent by the algorithm to find a solution. An important contribution of CLP is the possibility of a problem can be modeled using variables. These variables can be related to each other, are contained in fields. Such characteristics can be introduced into an algorithm that uses conventional programming into logic in order to optimize the response time, lines of code and simplify conditional checks. The programming logic by restrictions demonstrated to be effective in the construction of elegant and efficient programs. But the field of theoretical foundation and the techniques of programming needed to use the full potential of systems based on spread of restrictions is still a no trivial task. Normally, know the resources available in a system for scheduling restrictions is sufficient to model problems properly, but not for the achievement of effective results [9]. The experience of creating a software that uses logic programming with restrictions was necessary to the understanding of the technique, but mainly showed the importance of using a modeling consistent. The values obtained in the capture of processing time of the first experiment showed that as important as the use of restrictions in programming logic is in its modeling. The results obtained by applying the CLP on the problem of picking the warehouse have proved satisfactory in relation to the use of the Hamiltonian cycle [11]. The two methods for solving the same problem, proved effective. The comparative present at the session 5 showed a better performance in relation to the CLP cycle Hamiltonian when measured response to the times in the processing of both. The CLP can be used in many cases of application. Problems such as time tabling, staggering, tasks, processes and sequencing of several other problems, mainly from the operational research. But not all the real problems can be solved using this technique. Therefore, the main contribution of this work is to demonstrate the efficiency in the use of algorithms using the CLP in problems of routes. It brought a comparison of the use of Hamiltonian cycle [11] and the CLP for the same problem. The data presented in the previous chapter showed a significant reduction in processing time by CLP, and the difference in growth versus time processing between the two approaches when the increase in the area of search. Some proposals for future work are: the use of techniques such as hybrid genetic algorithms (GA), the use of the metaphor of ants and other that addressing the improvement in the

comparison of the results. We can use the same model or other techniques to search for consistency.

References

[1] Barahona, P. *Programação por Restrições*. Disponível em http://ssdi.di.fct.unl.pt/ pb/cadeiras/pr., Acessado em 25/03/2007, 2007.

[2] Barták, R. *Constraint Programming: In Pursuit of the Holy Grail in Proceedings of the Week of Doctoral Students (WDS99*. Part IV, MatFyzPress, Prague, June 1999, pp. 555-564, 1999.

[3] Barták, R. *Theory and Practice of Constraint Propagation in Proceedings of the 3rd Workshop on Constraint Programming for Decision and Control (CPDC2001)*. Wydavnictvo Pracovni Komputerowej, Gliwice, Poland, June 2001, pp. 7-14., 2001.

[4] Bittencourt, G. *Inteligência Artificial: Ferramentas e Teorias*. 10ª Escola de Computação. Editora da Unicamp, São Paulo, 1996.

[5] Fruewirth, T. and Abdennadher, S. *Essentials of Constraint Programming*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[6] Hubner, J. *Constraint Satisfaction Problems*. Material da disciplina de IA. Disponível em http://www.furb.br/ jomi., 2007.

[7] Luger, G. *Inteligência Artificial: estruturas e estratégias para a solução de problemas complexos.* 4 ed. Porto Alegre: Bookmann, 2004.

[8] Marriott, K. and Stuckey, P. J. *PProgramming with Constraints: an Introduction*. University of Melbourne and Monash University. MIT Press. 476 pp, 1998.

[9] Martins, A. *Introdução aos sistemas de restrições.* Faculdade de engenharia do Porto: FEP, 2005.

[10] Pedroso, J. P. *Knapseck Problem.* Disponível em http://www.ncc.up.pt/ jpp/cia/node47.html. Acessado em 23/03/2007, 2007.

[11] Pizani, A. S. *Aplicação do Ciclo Hamiltoniano em picking de almoxarifado*. Trabalho de Conclusão de Curso: UDESC, 2007.

[12] Stuart Russel, P. N. *Artificial Intelligence - A Modern Approach, 2ed.* Prentice Hall, 1995.

[13] Sucupira, I. R. *Programação por Propagação de Restrições: Teoria e Aplicações.* IME: USP, 2003.