Constraint Satisfaction Problems In Constraint Programming

Amey Kelkar
Mulund College of Commerce

Abstract

Constraint satisfaction is arguably the most relevant and practical facet of programming in operational research. As a branch of constraint programming, it has received unprecedented application in airline, shipping and freight industries particularly in scheduling and rostering. The formulation and research into algorithms for constraint satisfaction problems solution has received a lot of interest among researchers owing to the robustness of solutions that can be harnessed. Beginning with an introduction on constraint programming, the paper defines and isolates a typical constraint satisfaction problem. It then delves into the topic of constraint satisfaction as a branch of constraint programming, the modes of algorithm formulation to solve constraint satisfaction problems and its applicability in real life scenarios.

1. Introduction

Constraint programming is a programming paradigm in which constraints are used to state or define the relationships between variables. Constraint programming uses various types of variables such as variables emergent from the solution of simplex algorithms, variables that satisfy a constraint among others (ETAPS, 2003). By definition, a constraint refers to a relation in logic, of several variables (which are unknown) for which, each assumes a value in a specific domain. It restricts the values that can be assigned to a specific variable, thereby providing partial details on the given variable. The programming process in constraint programming entails building or coming up with requirements, also referred to as constraints, and then by use of constraint solvers, coming up with a solution to the previously specified requirements. Constraint programming is divided into two major branches, Constraint Satisfaction and Constraint Solving [1].

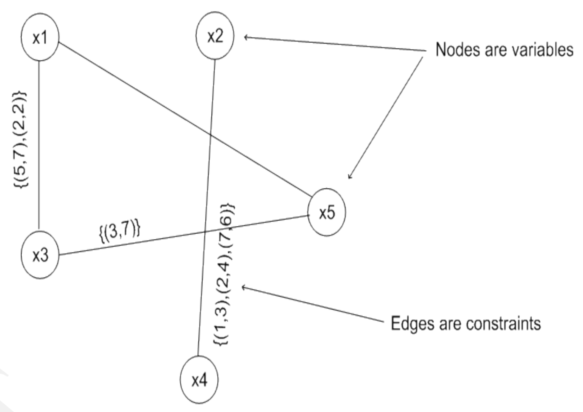

Figure 1: Relationships between constraints and variables [8]

2. Constraint Satisfaction Problems

This paradigm of programming has received widespread applicability especially in the areas of scheduling and planning. In a constraint programming perspective, this can be viewed as constraint satisfaction problems, which is squarely within the constraint satisfaction branch of constraint programming. A typical definition of a constraint satisfaction problem can be expressed as follows; a situation that provides a definite set of variables and values that can be assigned to the variables. In addition, there is a set of constraints, the solution is to find values from the given set to be assigned to the variables and satisfy the constraints [2].

A constraint satisfaction problem consists of

- a set of variables $X = \{x_1,\ldots,x_n\}$
- a domain for each variable $D = \{D_1,\ldots D_n\}$
- a set of constraints $C = \{C_1,\ldots,C_n\}$

Thus, a constraint satisfaction problem can be defined as triple P where P = (X, D, C)

A finite Constraint Satisfaction Problem consists of finite set of variables and a finite domain for each variable.

3. Examples

Cryptarithmeticpuzzles can be expressed as a Constraint Satisfaction Problem. Consider the following.

```
CRASH
+ R E B O O T
HACKER
```

In the above example the set of variables are the letters X = {C, R, A, S, H, E, B, O, T, K}. Their domains are the set of digits and can be represented by D = {0, 1, ..., 9}, and constraint is that each of the ten variables should be assigned to a different value in the domain.

Map colouring can also be expressed as a Constraint Satisfaction Problem. Consider the following

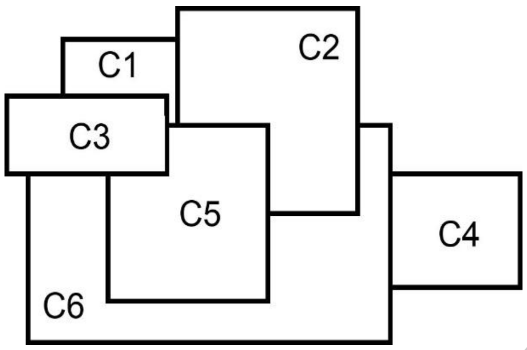

Figure 2: Map of six countries

In the above example the set of variables are the six countries X = {C1, C2, C3, C4, C5, C6} and domain is the set of colours D = {Red, Green, Blue}. The constraint is that no adjacent countries should have the same colour in the map. Therefore, constrains will be Colour of C1 ≠ Colour of C2, Colour of C1 ≠ Colour of C3, Colour of C1 ≠ Colour of C5 and so on.

4. Techniques

Problems in constraint satisfaction are combinatorial in nature, and thus existence of an effective algorithm is unlikely. To find a solution to such problems, the algorithms are enumerative, and their time requirement is exponential. The manner by which these algorithms come up with a solution is either, Consistency driven, Backtracking or Generate and Test [4]. Based on the information available in the constraints, algorithms that are consistency based minimize the solution-search space at the very initial levels. An example is the Arc consistency algorithm which when described in the simplest form means, if all constraints in a problem affect only 2 variables (binary), a constraint graph can be used to represent the constraints and variables. On the graph, the variables are represented by nodes, and on the condition that there is a constraint between two variables, there is an edge joining the two nodes that represent the variables [2].

5. The Backtracking Algorithm

In a backtracking algorithm, the variables in the problem are ordered starting with the ones that have a smaller range or are highly constrained. This ordering fashion greatly increases the efficiency of the algorithm. By checking that constraints are satisfied at the earliest possible stage, it assigns values to variables for constraints that involve bound variables at these early stages. This mode employs chronological backtracking (involves unbinding variables inversely to the order they were bound) and dependency directed backtracking (similar to chronological backtracking but identifies and corrects failures resulting from chronological backtracking).

The algorithm is as following

```
BT (Level)
  If all variables are assigned
        PRINT value of each variable
        RETURN (for multiple solutions) or
        EXIT (for single solution)
V = PickUnassignedVariable()
Variable [Level] = V
Assigned [V] = TRUE
FOR d EACH member of Domain (V)
        Value[V] = d
        OK = TRUE
FOR EACH constraint C such that V is a variable of
Cand all other variables of C are assigned.
IF C is not satisfied by the current set of assignments
        OK = FALSE
IF(OK)
BT (Level+1)
RETURN
```

Although the Backtracking algorithm is better than Generate and Test algorithm, its efficiency in some nontrivial problems is very low [5]. The three limitations of backtracking are

1. Thrashing – occurs due to repeated failure for the same reason. The popular technique to avoid thrashing is using Backjumping technique.
2. Redundancy – Even if conflict is avoided by Backjumping, they aren't stored for detection of conflict of the same kind which may occur in future.

3. Deferred detection of conflicts – Backtracking algorithm cannot detect conflicts before they really occur.

6. Generate and Test Algorithm

Generate and test algorithms are the simplest but are intolerably slow. They entail generating of all possible combinations of variable assignments and testing them to see if they fulfil the constraints. These algorithms generally make use of nested loops where testing of the constraints condition is done in the innermost loop [4].
The main limitation of the Generate and Test Algorithm is that it is capable of generating several wrong values which are assigned to variables. The values excluded in the testing phase [6].

7. Conclusion

Constraint Satisfaction problem can definitely be solved using the traditional algorithms, but at a certain cost. Instead, scheduling, timetabling, and rostering are practical examples of constraint satisfaction problem solving.

In scheduling jobs are to be done by machines given that a machine can only handle a job at a time, and different jobs have different priorities. Employee shift scheduling can be a perfect example of scheduling, where each employee has to complete a certain amount of work in a certain time period, based on its priority. Scheduling helps increasing productivity as well as throughput and reducing the turnaround time.

In timetabling, an example involves creating an exam timetable where different exams have to be done in different periods, by different students, in different rooms of different room sizes and many other constraints around the problem.
Crew rostering is an example application in rail and air transport industries [2].

7. References

[1]Bartak, R., “Constraint Programming. On-Line Guide to Constraint Programming”, 1998
[2] Sally C. Brailsford and Chris N. Potts and Barbara M. Smith, “Constraint Satisfaction Problems: Algorithms and Applications”, University of Southampton, Leeds, UK, 1998
[3] ETAPS, “Foundations of Constraint Programming”, Warshaw, Poland, 2003
[4] Constraint Satisfaction Problems. n.d. Retrieved from http://www.cis.temple.edu/~giorgio/cis587/readings/constraints.html
[5] Vipin Kumar, “Algorithms for Constraint-Satisfaction Problems: A Survey”, AI Magazine Volume 13 Number 1, 1992
[6] Roman Barták, “ConstraintPropagation And Backtracking-Based Search”, Faculty of Mathematics and Physics, Charles University
[7] Krzysztof Apt, “Principles of Constraint Programming”, 2009
[8] http://www.doc.ic.ac.uk/~sgc/teaching/pre2012/v231/lecture15.html