International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.3., Issue.6., 2015
(Nov.-Dec.,)

REVIEW ARTICLE

ISSN: 2321-7758

CONSTRAINT SATISFACTION PROBLEMS : SURVEY OF ALGORITHM

SAMEER DESWAL[1], SAVITA BISHNOI[2]
[1]Computer Science & Engineering, Rohtak Institute of Technology & Management, Rohtak, Haryana, India
[2] Assistant Professor, CSE Department,Rohtak Institute of Technology & Management, Rohtak, Haryana, India

International Journal of Engineering Research-online (IJOER)
ISSN:2321-7758
www.ijoer.in

ABSTRACT
Many problems in AI can be modeled as constraint satisfaction problems CSPs , Some of them are related to real life problems like timetable planning , vehicle routing problem etc. Hence there are various heuristics, meta-heuristics are present which can be used to solve these problems . In this paper numerous algorithm have discussed and their comparison is presented.

I. INTRODUCTION

Constraint satisfaction problems (CSPs) are mathematical problems defined as a set of objects whose state must satisfy a number of constraints or limitations. CSPs represent the entities in a problem as a homogeneous collection of finite constraints over variables, which is solved by constraint satisfaction methods.Examples of simple problems that can be modeled as a constraint satisfaction problem are like Eight queens puzzle , Map coloring problem ,Sudoku etc. Timetable creation comes in automated planning[1] section which are considered as real life examples .

II. Related Work

Constraint satisfaction problems on finite domains are typically solved using a form of search. The most used techniques are variants of backtracking, constraint propagation, and local search[2].

Backtracking is a recursive algorithm. It maintains a partial assignment of the variables. Initially, all variables are unassigned. At each step, a variable is chosen, and all possible values are assigned to it in turn. For each value, the consistency of the partial assignment with the constraints is checked; in case of consistency, a recursive call is performed. When all values have been tried, the algorithm backtracks. In this basic backtracking algorithm, consistency is defined as the satisfaction of all constraints whose variables are all assigned. Several variants of backtracking exists. Backmarking improves the efficiency of checking consistency. Backjumping allows saving part of the search by backtracking "more than one variable" in some cases. Constraint learning infers and saves new constraints that can be later used to avoid part of the search. Look-ahead is also often used in backtracking to attempt to foresee the effects of choosing a variable or a value, thus sometimes determining in advance when a subproblem is satisfiable or unsatisfiable.

Constraint propagation techniques are methods used to modify a constraint satisfaction problem. More precisely, they are methods that enforce a form of local consistency, which are conditions related to the consistency of a group of variables and/or constraints. Constraint propagation has various uses. First, it turns a problem into one that is equivalent but is usually simpler to solve. Second, it may prove satisfiability or unsatisfiability of problems. This is not guaranteed to happen in general; however, it always happens for some forms of constraint propagation and/or for some certain kinds of problems. The most known and used form of local consistency are arc consistency, hyper-arc consistency, and path consistency. The most popular constraint propagation method is the AC-3 algorithm, which enforces arc consistency.

Local search methods are incomplete satisfiability algorithms. They may find a solution of a problem, but they may fail even if the problem is satisfiable. They work by iteratively improving a complete assignment over the variables. At each step, a small number of variables are changed value, with the overall aim of increasing the number of constraints satisfied by this assignment. The min-conflicts algorithm is a local search algorithm specific for CSPs and based in that principle. In practice, local search appears to work well when these changes are also affected by random choices. Integration of search with local search have been developed, leading to hybrid algorithms.

Decision problems

CSPs are also studied in computational complexity theory and finite model theory. An important question is whether for each set of relations, the set of all CSPs that can be represented using only relations chosen from that set is either in P or NP-complete. If such a dichotomy theorem is true, then CSPs provide one of the largest known subsets of NP which avoids NP-intermediate problems, whose existence was demonstrated by Ladner's theorem under the assumption that P ≠ NP. Schaefer's dichotomy theorem handles the case when all the available relations are boolean operators, that is, for domain size 2. Schaefer's dichotomy theorem was recently generalized to a larger class of relations[3].

Most classes of CSPs that are known to be tractable are those where the hypergraph of constraints has bounded treewidth (and there are no restrictions on the set of constraint relations), or where the constraints have arbitrary form but there exist essentially non-unary polymorphisms[clarification needed] of the set of constraint relations.

Every CSP can also be considered as a conjunctive query containment problem[4].

Function problems

A similar situation exists between the functional classes FP and #P. By a generalization of Ladner's theorem, there are also problems in neither FP nor #P-complete as long as FP ≠ #P. As in the decision case, a problem in the #CSP is defined by a set of relations. Each problem takes as input a Boolean formula as input and the task is to compute the number of satisfying assignments. This can be further generalized by using larger domain sizes and attaching a weight to each satisfying assignment and computing the sum of these weights. It is known that any complex weighted #CSP problem is either in FP or #P-hard[5].

There are currently many different solution generation algorithms in existence. Some are so well known that they are the subject of standard textbook material (see, eg. , Scharef, 1996; Osman and Kelly, 1996). This section presents major algorithms that have appeared in the literature on generating timetabling solutions

Tabu Search

A brief description of these fundamental techniques is now presented. Each of these descriptions are structured to begin with a general formulation of the technique followed by a mathematical interpretation of each technique within the context of the generalised timetable problem.

Linear Programming/Integer Programming

The Linear and Integer Programming techniques, the first applied to timetabling, were developed from the broader area of mathematical programming. Mathematical programming is applicable to the class of problems characterised by a large number of variables that intersect within boundaries imposed by a set of restraining conditions . The word "programming" means planning in this context and is related to the type of application . This scheme of programming was developed during World War II in connection with finding optimal strategies for conducting the war effort and used afterwards in the fields of industry, commerce and government services .

Evolutionary and Genetic Algorithms

Evolutionary Algorithms (EAs) are a class of direct, probabilistic search and optimisation algorithms gleaned from the model of organic evolution. A Genetic Algorithm (GA) is a type of EA and is regarded as being the most widely known EA in recent times.

Simulated Annealing

Simulated Annealing (SA) is a randomised local search optimisation technique for finding solutions to optimisation problems. The name is derived from the analogue to the chemical physics simulation of the cooling of a collection of a Boltzmann distribution of atoms. SA is highly resource intensive and one of its setbacks is its requirement of utilising a large amount of computational time for obtaining a near- optimal solution. As such some attempts at speeding up annealing algorithms have been based on shared memory multiprocessor systems, and parallelization for certain problems on distributed memory multiprocessor systems.

Timetable construction problems at schools

In this thesis, we focus on timetable construction problems at schools. For these timetabling problems, the events are lessons in a subject, taught by a teacher to a group of students, sometimes referred to as a class, in a room. We show the concepts of timetabling in general (a) and timetable construction at schools in particular (b). In both cases there is a central concept to which all other concepts, most of them representing the necessary resources, are related. Decisions. Timetable construction problems are mainly about allocating resources, i.e., teachers, students, rooms, and time slots, to lessons.

Table 1 Algorithm Comparison

Algorithm	Scalable?	Optimal?	Easy to use?	Tweakable?	Requires CH?
Exhaustive Search (ES)					
Brute Force	0/5	5/5	5/5	0/5	No
Branch And Bound	0/5	5/5	4/5	2/5	No
Construction heuristics (CH)					
First Fit	5/5	1/5	5/5	1/5	No
First Fit Decreasing	5/5	2/5	4/5	2/5	No
Weakest Fit	5/5	2/5	4/5	2/5	No
Weakest Fit Decreasing	5/5	2/5	4/5	2/5	No
Strongest Fit	5/5	2/5	4/5	2/5	No
Strongest Fit Decreasing	5/5	2/5	4/5	2/5	No
Cheapest Insertion	3/5	2/5	5/5	2/5	No

Algorithm	Scalable?	Optimal?	Easy to use?	Tweakable?	Requires CH?
Regret Insertion	3/5	2/5	5/5	2/5	No
Metaheuristic (MH)					
Local Search					
Hill Climbing	5/5	2/5	4/5	3/5	Yes
Tabu Search	5/5	4/5	3/5	5/5	Yes
Simulated Annealing	5/5	4/5	2/5	5/5	Yes
Late Acceptance	5/5	4/5	3/5	5/5	Yes
Step Counting Hill Climbing	5/5	4/5	3/5	5/5	Yes
Evolutionary Algorithms					
Evolutionary Strategies	4/5	3/5	2/5	5/5	Yes
Genetic Algorithms	4/5	3/5	2/5	5/5	Yes

III. Conclusion

This paper gives understanding the domain of constraint satisfaction problem and different ways to solve these problem . I have focused the methods which are used to solve timetable problems . These methods included algorithms which provide satisfiable solutions . I have provided the comparison between algorithm in terms of scalability , optimization , easy to use etc.

Backtracking is used among algorithms because it provide a way to search the solution by moving back and from in solution space . It ease the process fo finding optimal solution.

Heuristic Algorithm is used to generate the expected solutions .

This paper gives a view of various methods to solve the constraint satisfaction problem . In future further detailed study on algorithm could be done revealing other important properties .

IV. References

[1]. Dynamic Flexible Constraint Satisfaction and Its Application to AI Planning, Ian Miguel - slides. Stuart Jonathan Russell, Peter Norvig (2010). Artificial Intelligence: A Modern Approach. Prentice Hall. p. Chapter 6. ISBN 9780136042594.

[2]. Bodirsky, Manuel; Pinsker, Michael (2011). "Schaefer's theorem for graphs". Proceedings of the 43rd Annual Symposium on Theory of Computing (STOC '11). Association for Computing Machinery. pp. 655–664.

[3]. Kolaitis, Phokion G.; Vardi, Moshe Y. (2000). "Conjunctive-Query Containment and Constraint Satisfaction". Journal of

Computer and System Sciences 61 (2): 302–332. doi:10.1006/jcss.2000.1713.

[4]. Cai, Jin-Yi; Chen, Xi (2012). Complexity of counting CSP with complex weights. pp. 909–920.

[5]. Miguel, Ian (July 2001). Dynamic Flexible Constraint Satisfaction and its Application to AI Planning (Ph.D. thesis). University of Edinburgh School of Informatics. hdl:1842/326;

[6]. Dechter, R. and Dechter, A., Belief Maintenance in Dynamic Constraint Networks In Proc. of AAAI-88, 37-42.

[7]. Solution reuse in dynamic constraint satisfaction problems, Thomas Schiex

[8]. Duffy, K.R.; Leith, D.J. (August 2013), "Decentralized Constraint Satisfaction"