Constraint Networks: A Survey

Christopher C. Yang[†] and Ming-Hsuan Yang[*]
†: Department of Computer Science
The University of Hong Kong
Email: yang@cs.hku.hk
*: AI Group, Beckman Institute
The University of Illinois at Urbana-Champaign
Email: myang1@uiuc.edu

Abstract

A constraint satisfaction problem (CSP) involves a set of variables, a domain of potential values for each variable, and a set of constraints, which specifies the acceptable combinations of values. One popular approach is to represent the original problem as a constraint network where nodes represent variables and arcs represent constraints between variables. Node consistency and arc consistency techniques are first applied to prune the domains of variables. Constraint propagation techniques are then applied to solve the problem. Many AI and engineering problems can be formulated as CSPs and solved by various CSP algorithms such as constraint propagation, backtracking, forward checking, and hybrids. This paper gives an overview of these algorithms.

In particular, we present a review of the interval constraint satisfaction problems (ICSP). Real intervals or sets of discrete values label the variables. The constraints can be binary relationships or n-ary mathematical operations. The techniques for solving the interval constraint satisfaction problem such as Waltz filtering and tolerance propagation are presented.

1. Introduction

Many tasks in artificial intelligence can be seen as constraint satisfaction problems. The task specification can be formulated to consist of a set of variables, each of which must be instantiated in a particular domain and a set of constraints (predicates) that the values of the variables must simultaneously satisfy. The solution of CSP is a set of value assignments such that all the constraints are satisfied.

Many approaches, as summarized in [14], have been developed to solve constraint problems such as Predicate Calculus [28], Propositional Logic [22], Truth Maintenance [21], Integer Programming [33], Automata Theory [39], Graph Theory [20], Hill Climbing [31], Neural Networks [5], Genetic Algorithms [37], Relational Algebra [1], Constraint Synthesis [11], Disjunctive Decomposition [13], Conjunctive Decomposition [8], Constraint Logic Programming [18], and GSAT [35]. We can classify these techniques into (1) problem reduction, (2) solution synthesis, and (3) searching. In problem reduction, we first identify redundant information and then remove them. On the contrary to problem reduction, we generate legal compound labels in solution synthesis instead of removing redundant labels. In searching, we search from the initial node in forward or backward direction based on the constraints until all the constraints are satisfied.

In this paper, we have also investigated the interval constraint satisfaction problems (ICSPs). The variables can be labeled as real intervals or sets of discrete values. The constraints can be binary or n-ary mathematical operations.

2. Constraint Satisfaction Problems

Constraint Satisfaction Problem (CSP) is a problem composed of a set of variables and a set of constraints. Values are assigned to the variables such that all the constraints are satisfied. A compound label is the simultaneous assignment of values to a set of variables. The assignment, $\{ <x_1, v_1>, <x_2, v_2>, \ldots, <x_n, v_n> \}$, denotes the compound label of assigning $v_1, v_2, \ldots, v_n$ to $x_1, x_2, \ldots, x_n$, respectively. For example, given the map of the United States, we want to color every state and no adjacent states have the same color. We represent each state as a node in the constraint network and the arcs of the network correspond to the adjacency of the states. The values of the variables are therefore the possible choices of the colors such as red, blue, green, and orange. Montanari presented the fundamental properties of constraint networks in [32] and Mackworth discussed the logic of CSPs in [28].

Node, arc, and path consistency are common consistency concepts for binary constraint problems, and k-consistency is concept for general CSPs. It defines the satisfaction of the constraint problems. A node is consistent if and only if all values in its domain satisfy the constraints on the corresponding variable. An arc (v_i, v_j) is consistent if and only if for every value x in the domain of v_i, there is some value y in the domain of v_j such that $v_i = x$ and $v_j = y$ are permitted by the constraint between v_i and v_j. A path is consistent if and only if for all the values in the initial node and final node, there are some values in every node along the path such that the constraints between all the nodes on the path are satisfied. In order to obtain the solutions of CSP, researchers have explored three major techniques, problem reduction, solution synthesis, and searching.

2.1 Problem Reduction

The goal of problem reduction is to remove redundant values and redundant compound labels. The key of this technique is to identify such values and compound labels. If the presence of a value in a domain or a compound label in a constraint falsifies the constraints, then it can be deduced to be redundant.

Since the size of search space is the grand product of all the domain sizes in the problem and it is time consuming to determine solution in such a large search space, reducing the search space is helpful in solving the problem. When we remove redundant values and redundant compound labels, we avoid searching those futile subtrees repeatedly.

Mackworth [27] developed algorithms to achieve node, arc, and path consistency. In node-consistency achievement algorithms, all values that fail to satisfy the unary-constraints are deleted from the domains. In arc-consistency achievement algorithms, we remove any values that do not satisfy the constraint represented by the arc. If any values are removed, the arc will be examined again. The path-consistency achievement algorithms are developed in a similar manner.

2.2 Solution Synthesis

Solution synthesis [11] incrementally builds a lattice, called minimal problem graph that represents the minimal problem. Upward propagation and downward propagation are used to eliminate compound labels in nodes of a higher order and lower order, respectively. All illegal compound labels are removed to ensure soundness and no legal compounds are removed to ensure completeness. It constructively generates legal compound labels instead of eliminating redundant values or redundant compound labels.

2.3 Searching

Researchers have developed several searching strategies for CSPs. The searching algorithms start from an initial node, pick a value out from its domain and save it into the working list. It continues from node to node until all variables are assigned a value. If a feasible solution cannot be obtained during the searching process, different searching strategies have different methods to search for alternative solutions.

2.3.1 Backtracking Algorithm

Backtracking algorithm [2,3,4,12,29] is an exhaustive search that explores the whole search space. It has no attempt to reduce the search space by pruning off the search space. When no feasible value can be assigned to the current node (a dead-end situation occurs), it goes back to the previous node, picks another value and continues. Backjumping, backchecking, and backmarking algorithms modify backtracking algorithm such that a smaller search space is explored.

Backjumping algorithm [16] analyzes the situation to identify the culprit decisions that cause the failure when it backtracks. It then backtracks to the most recent culprit decision instead of the immediate past node as in the backtracking algorithm. Backchecking algorithm reduces compatibility checks that are computationally expensive. Backmarking [15] is an improved backchecking algorithm. It keeps records of the incompatible labels for every label and avoids repeating compatibility checks that have been performed and hence reduce compatibility checks.

2.3.2 Forward Checking (Lookahead) Algorithm

On the contrary to the backtracking algorithm, forward checking algorithm [26] looks ahead to remove the incompatible values in the unlabelled variables before it hits the problems. Every time when a value is assigned to a node, it determines the values for all the other unlabelled variables that are incompatible with the current assignment. When it moves on the next node, it only picks value from the compatible values in the domain.

AC-Lookahead algorithm looks further ahead than forward checking algorithm. When it assigns a value to the current node, it does not only determine the incompatible values in the unlabelled variables, but it also determines whether the current assignment will cause any unlabelled variable infeasible to obtain any values.

2.3.3 Truth Maintenance System

To avoid chronological backtracking in search, Sussman and Stallman developed Dependency-directed Backtracking [36] in that decisions of backtracking are made based on logical dependencies rather than the chronological order. Truth Maintenance System (TMS) [9], developed by Doyle, is a method of providing ability to do dependency-directed backtracking and to support nonmonotonic reasoning.

A TMS-based problem solver consists of two components: an inference engine and a TMS. The inference engine is used to derive new facts from old ones, while the TMS records the justifications of the derivations (i.e. how a fact has been derived) which in turn provides with both assertions and dependencies among assertions. A TMS allows assertions to be connected via a network of dependencies and its tasks is to ensure a *consistent, well-founded labeling* (i.e. the proper grounding of a chain of justifications on a set of nodes that do not themselves depend on the nodes they support) and to resolve *contradictions* (which, in TMS, does not represent logical contradictions but rather states of the database explicitly declared to be undesirable). Based on this, McAllester proposed a Logic-Based Truth Maintenance System (LTMS) [30] and de Kleer developed an Assumption-Based Truth Maintenance System (ATMS) [10,22,23]. Both TMS and LTMS pursue a single line of reasoning (i.e. a *single* context) at a time, and dependency-directed backtracking occurs when it is necessary to change the system's assumptions. In ATMS, alternating paths (i.e. *multiple* contexts) are maintained in parallel. The assumptions differ from the justifications in the fact that they are believed unless there is evidence to the contrary and can thus prove false. Backtracking is avoided at the expense of maintaining multiple contexts, each of which corresponds to a set of consistent assumptions. The universe of consistent contexts is pruned as reasoning proceeds in an ATMS-based problem solver. The remaining consistent contexts are used to label assumptions, thus indicating the contexts in which each assumption has a valid justification. The problem solver prunes any assumption that does not have a valid justification. Essentially, an ATMS does breadth-first search (i.e. considers all possible contexts at once) while both LTMS and TMS operate depth-first.

The tasks of CSP and ATMS seem, on the surface, very different. However, both require combining distinct modes of reasoning to minimize the overall computational cost to complete their tasks. In [22], de Kleer made a comparison of ATMS and CSP techniques and showed how CSPs can be mapped to ATMS problems and vice versa.

2.3.4 Constraint Propagation

Constraint propagation is a technique to maintain consistency among nodes of a constraint network whenever an instantiation occurs. Whenever a value has been assigned to a node, this assignment is passed on to neighboring nodes in order to ensure the consistency (usually arc-consistency) among them. This process can be further rippled down the constraint network to prune insistent values based on the current instantiation. However, experiments of a variety of problems have indicated that it is better to apply constraint propagation only in a limited form [17].

Usually, constraint propagation is embedded in a backtracking algorithm to solve a CSP. In this approach, a root node is created to solve the original CSP. Whenever a node is visited, constrain-propagation algorithm (such as node consistency, arc consistency) is used to attain a desired level of consistency. During the process, if the cardinality of each variable becomes 1 and corresponding CSP is arc consistent, then the current instantiation of each node represents a solution. If in the process of performing constraint propagation at the node, the domain of any variable becomes null, then the node is pruned. Otherwise one of the variables (whose current cardinality is grater than 1) is selected, and a new CSP is created for each possible assignment of this variable. Each such new CSP is depicted as a successor node of the node that representing parent CSP. A backtracking algorithm visited these nodes in depth-first search until a solution is found.

3. Interval Constraint Satisfaction Problems

Interval constraint satisfaction problem (ICSP) is a constraint satisfaction problem with the domain of variable as interval. Each variable is assigned an interval instead of a single value. Some researchers have investigated the domain of real intervals [6,19,25,40,41,42], and some have investigated the domain of set of discrete values [7,27]. Some have examined the binary constraints [7,25] and some have examined the n-ary mathematical constraints [6,19,27,40,41,42].

Decheter and Pearl [7] developed a method of generating heuristic advice to guide the order of value assignments based on sparseness in the constraint network and the simplicity of tree-structured CSPs. The domains of variables are sets of discrete values and the constraints. The approach of automatic advice generation orders the candidates of an unlabeled node according to the confidence that they can be extended further to a full solution in backtracking. Confidence are obtained by making assumptions about the continuing portion of the search graph and estimating the likelihood that it will contain a solution even when the simplifying assumptions are removed.

Mackworth and Freduer [27] analyzed the time complexity of several node, arc and path consistency algorithms in CSPs. The domains of variables are sets of discrete values. It shows that arc achievable in time linear in the number of binary constraints.

Ladkin and Reinefeld [25] developed a technique to solve qualitative interval constraint problems. The domains of variables are real intervals and the constraints are binary. Ladkin and Reinfeld employs the relational matrix composition algorithm to check the initial path-consistency and the uses path-consistency further as a pruning technique to reduce the size of the solution space.

Davis [6] adapted the Waltz filtering algorithm for screening impossible values from the variable domain to solve the ICSPs. The domains of variables are real intervals and the constraints are n-ary mathematical operations. However, the Waltz filtering algorithm cannot determine the global solutions in general but only the local solutions.

Hyvonen [19] used the tolerance propagation approach, which combines the consistency techniques based on the topology of the constraint net with techniques of interval arithmetic, to solve the ICSPs. The domains of variables and constraints are the same as what Davis have studied. The tolerance propagation techniques determine both global and local solutions.

The ICSPs that studied by Davis and Hyvonen are defined based on variable consistency. A variable, v_i, is consistent if and only if for all values in v_i, there are some values for all other variables such that all constraints in the network are satisfied. Yang et al. [40] define a new constraint consistency, and develop a new constraint propagation techniques for the applications on tolerance design, robotics, and others. A constraint is consistent if and only if for all values in every input variables of the constraint, there are some values for the output variables such that the constraint is satisfied. Forward and backward propagation techniques for single constraint and multiple constraint in a hierarchical interval constraint network are developed.

4. Conclusion

In this paper, we present a general view of the constraint satisfaction problems. Many techniques have been developed to solve the constraint satisfaction problems, such as predicate calculus, integer programming, neural network, truth maintenance, etc. We categorize the methodologies into four categories, (i) problem reduction, (ii) solution synthesis, and (iii) searching. In particular, we have also presented a brief discussion of the interval constraint satisfaction problems.

References

[1] Bibel, W., "Constraint Satisfaction from a Deductive Viewpoint," *Artificial Intelligence*, vol. 35, pp.401-416, 1988.

[2] Binter, J. R., and Reingold, E. M., "Backtrack Programming Techniques," *Communications of the ACM*, vol.18, pp.651-656, 1975.

[3] Bruynooghe, M., and Pereira, L. M., "Deduction Revision by Intelligent Backtracking," *Implementation of Prolog*, Ellis Horwood, Chichester, U. K., pp.194-215, 1984.

[4] Cox, P. T., "Finding Backtrack Points for Intelligent Backtracking," *Implementation of Prolog*, Ellis Horwood, Chichester, U. K., pp.216-223, 1984.

[5] Daveport, A., Tsang, E., Wang, C., and Zhu, K., "GENET: A Connectionist Architecture for Solving Constraint Satisfaction Problems by Iterative Improvement," In *Proceedings of Twelfth National Conference on Artificial Intelligence*, pp.325-336, 1994.

[6] Davis, E., "Constraint Propagation with Interval Labels," *Artificial Intelligence*, vol.32, pp.281-331, 1987.

[7] Dechter, R., and Pearl, J., "Network-Based Heuristics for Constraint-Satisfaction Problems," *Artificial Intelligence*, vol.38, pp.351-366, 1989.

[8] Dechter, R., and Pearl, J., "Tree Clustering for Constraint Networks," *Artificial Intelligence*, vol. 38, pp.353-360, 1989.

[9] Doyle, J., "A Truth Maintenance System," *Artificial Intelligence*, vol.12, pp.231-272, 1979.

[10] Forbus, K. D., and Kleer, J. de, *Building Problem Solvers*, Cambridge, MA, MIT Press, 1993.

[11] Freuder, E. C., "Synthesizing Constraint Expressions." *Communications of the ACM*, vol.21, no.11, pp.958-966, 1978.

[12] Freuder, E. C., and Quinn, M., "Taking Advantage of Stable Sets of Variables in Constraint-Satisfaction Problems," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Menlo Park, California, 1985.

[13] Freuder, E. C., and Hubbe, P. D., "A Disjunctive Decomposition Control Schema for Constraint Satisfaction," *Constraint Programming*, Saraswat and Van Hentenryck, editors, MIT Press, 1994.

[14] Freuder, E. C., "The Many Paths to Satisfaction,'' In M. Meyer, Editor, *Constraint Processing*, Lecture Notes in Computer Science vol.923, pp. 103-119, Springer-Verlag, 1995.

[15] Gasching, J., "A General Backtracking Algorithm that Eliminates Most Redundant Tests," *Proceedings of International Joint Conference on Artificial Intelligence*, Cambridge, MA, 1977.

[16] Gasching, J., "Experimental Case Studies of Backtrack vs. Waltz-Type vs. New Algorithms for Satisfying Assignment Problems," *Proceedings Second National Conference of the Canadian Society for Computational Studies of Intelligence*, Toronto, Ontario, pp.268-277, 1978.

[17] Haralick, R., and Elliot, G., "Increasing Tree Search Efficiency for Constraint Satisfaction Problem", *Artificial Intelligence*, vol. 14, no.3, pp. 263-313, 1980.

[18] Hentenryck, V., Simonis, H., and Dinebas, M., "Constraint Satisfaction Using Constraint Logic Programming," *Artificial Intelligence*, vol. 58, pp. 113-160, 1992.

[19] Hyvonen, E., "Constraint Reasoning Based on Interval Arithmetic: the Tolerance Propagation Approach," *Artificial Intelligence*, vol.58, pp.71-112, 1992.

[20] Jegou, P., "Decomposition of Domains Based on the Micro-Structure of Finite Constraint Satisfaction Problems," *Proceedings of Eleventh National Conference on Artificial Intelligence*, 24, pp.731-736, 1993.

[21] Kleer, J. de, "An Assumption-based TMS," *Artificial Intelligence*, vol.28, 1986.

[22] Kleer, J., de, "A Comparison of ATMS and CSP Techniques,'' *Proceedings of the Eleventh International Joint Conference on*

Artificial Intelligence, Menlo Park, CA., pp.290-296, 1989.

[23] Kleer, J. de, "A Perspective on Assumption-based Truth Maintenance," *Artificial Intelligence*, vol.59, 1993.

[24] Kumar, V., "Algorithms for Constraint-Satisfaction Problems: A Survey," *AI Magazine*, vol.13, no.1, 1992.

[25] Ladkin, P. B., and Reinefeld, A., "Effective Solution of Qualitative Interval Constraint Problems," *Artificial Intelligence*, vol.57, pp.105-124, 1992.

[26] Mackworth, A. K., "Consistency in Network of Relations," *Artificial Intelligence*, vol.8, no.1, 1977.

[27] Mackworth, A. K., and Freuder, E. C., "The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problem," *Artificial Intelligence*, vol.25, pp.65-74, 1985.

[28] Mackworth, A. K., "The Logic of Constraint Satisfaction,'' *Artificial Intelligence* vol. 58, pp. 3-30, 1992.

[29] Matwin, S., and Pietrzykowski, T., "Intelligent Backtracking in Plan-based Deduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.7, no.6, pp.682-692, 1985.

[30] McAllester, D. A., "An Outlook on Truth Maintenance," *Tech. Rep. AI Memo 551*, MIT Artificial Intelligence Lab, Cambridge, MA., 1980.

[31] Minton, S., Johnston, M., Philips, A., and Laird, P., "Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems," *Artificial Intelligence*, vol. 58, pp.161-206, 1992.

[32] Montanari, U., "Networks of Constraints: Fundamental Properties and Application to Picture Processing," *Information Science*, vol. 7, no. 2, pp. 95-132, 1974.

[33] Rivin, I., and Zabig, R., "An Algebraic Approach to Constraint Satisfaction Problems," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp.284-289 1989.

[34] Rossi, F., Petrie, C., and Dhar, V., "On the Equivalence of Constraint Satisfaction Problems," Technical Report ACT-AI-222-89, MCC Corp., Austin, Texas, 1989.

[35] Selman, B., Levesque, H., and Mitchell, D., "A New Method for Solving Hard Satisfiability Problems," *Proceedings of National Conference on Artificial Intelligence*, pp.440-446, 1992.

[36] Stallman, R. M., and Sussman, G. J., "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis," *Artificial Intelligence*, vol.9, no. 2, 1977.

[37] Tsang, E., and Warwick, T., "Applying Genetic Algorithms to Constraint Satisfaction Problems," *Proceedings of European Conference on Artificial Intelligence*, pp.649-654, 1990.

[38] Tsang, E., *Foundations of Constraint Satisfactions*, Academic Press, 1993.

[39] Vempaty, N. R., "Solving Constraint Satisfaction Problem Using Finite Automata,'' *Proceedings of Tenth National Conference on Artificial Intelligence*, pp.453-458, 1992.

[40] Yang, C. C., Marefat, M. M., and Ciarallo, F. W., "Tolerance Analysis and Synthesis by Interval Constraint Network," *Proceedings of IEEE International Conference on Robotics and Automation*, Albuquerque, NM, April, 1997.

[41] Yang, C. C., Marefat, M. M., and Ciarallo, F. W., "Tolerance Design by Interval Constraint Networks," *IIE Transactions (Design and Manufacturing)*, submitted.

[42] Yang, C. C., Marefat, M. M., and Ciarallo, F. W., "Constraint Networks with Integer, Real, and Interval Values for Robotics and Mechanical Design," *Applied Artificial Intelligence*, submitted.