Ind. JNT
(Indian Journal in Number theory)
Received: *13.10.2015*
Published: *30.11.2017*
Year: *2017*, ***Pages:*** *111-117*
*Original Article***

REPRESENTATION OF THE LARGE SET COVERING PROBLEM

Mohite S. D[1], Kakde R. V.[2]

[1] Research Scholar- Mathematics, JJTU, Jhunjhunu, Rajasthan

[2] Dept. of Mathematics, RSGM College, Loha, Nanded (Maharashtra)

[1] *mohiteshital@gmail.com*

Abstract- If a finite set E and a family $F=\{E_1 \ldots Em\}$ of subsets of E such that F covers E, the famous unicost set covering problem (USCP) is to determine the smallest possible subset of F that also covers E. We study in this paper a variant, called the Large Set Covering Problem (LSCP), which differs from the USCP in that E and the subsets E_i are not given in extension because they are very large sets that are possibly infinite. All concepts presented in the paper are illustrated on the k-colouring problem which is formulated as a constraint satisfaction problem.

Key-words: Set covering, Constraint satisfaction, k-colouring.

INTRODUCTION

Let E be a set of n elements, and let $E_1, \ldots, Em$ be m subsets of E such that $\bigcup_{i=1}^{m} E_i = E$. The unicost set covering problem (USCP) is to determine a subset $I \subseteq \{1,\ldots,m\}$ of minimum size such that $\bigcup_{i \in I} E_i = E$. This is a famous NP-hard problem [8]. In this paper, we study a variant of the USCP, called the *Large Set Covering Problem* (LSCP), which differs from the USCP in that E and the subsets E_i are not given in extension because they may be very large, and possibly infinite sets. In order to determine whether a given element $e \in E$ belongs to a subset E_i, we assume we are given a procedure Is-

ELEMENT (e, i) that returns value "true" if and only if $e \in E_i$. Also, in order to extract elements from E, we assume that given any weighting function ω that assigns a weight $\omega(i)$ to each set E_i, we can use a procedure MIN_WEIGHT(ω) that returns an element $e \in E$ such that $\sum_{e \in E_i} \omega(i)$ is minimum. A subset I of $\{1 \ldots m\}$ such that $\bigcup_{i \in I} E_i = E$ is called a *cover*. The LCSP is to determine a minimal (inclusion wise) cover. We also consider the problem, called *minimum* LSCP which is to determine a minimum cover. All concepts and techniques described in this paper are illustrated on the *k*-colouring problem.

FINDING IRREDUCIBLE INFEASIBLE SETS IN INCONSISTENT CONSTRAINT SATISFACTION PROBLEMS

The constraint satisfaction problem (CSP) [7,9] is defined over a constraint network, which consists of a finite set of *variables,* each associated with a *domain* of values, and a set of *constraints*. A constraint specifies, for a particular subset of variables, a set of incompatible combinations of values for these variables. A *solution* of a CSP is an assignment of a value to each variable from its domain such that all constraints are satisfied. A CSP is *consistent* if it has at least one solution; otherwise it is *inconsistent* (or unsolvable, or overconstrained, or infeasible). While the CSP is to determine whether a solution exists, related problems are to find one or all solutions, and to find an optimal solution relative to a given cost functions. For example Max-CSP is the problem of determining an assignment of a value to each variable from its domain such that as many constraints as possible are satisfied. Constraint satisfaction provides a convenient way to represent and solve problems where mutually compatible values have to be assigned to a predetermined number of variables under a set of constraints. Numerous applications arise in a variety of disciplines including machine vision, belief maintenance, temporal reasoning, graph theory, circuit design, and diagnostic reasoning [9].

A well-known example of a constraint satisfaction problem is the *k-colouring problem*, where the task is to colour, if possible, a given graph G with at most *k* colours, such that any two adjacent vertices have different colours. If such a colouring exists, then G is said *k-colourable*. A constraint satisfaction formulation of this problem associates the vertices of the graph with variables, the set of possible colours $\{1,\ldots,k\}$ is the domain

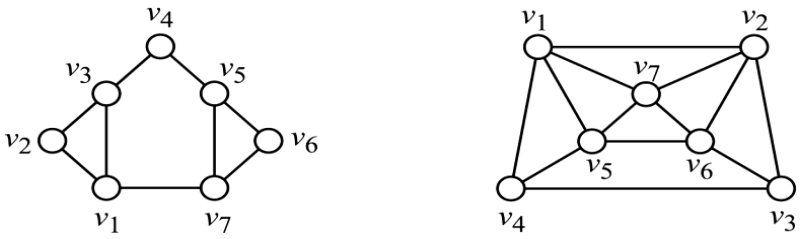
of each variable, and the inequality constraints between adjacent vertices are the constraints of the problem.

For a given CSP, a *partial* assignment is an assignment of a value from its domain to some variables, but not necessarily all. When all variables get a value, the assignment is said *complete*. We say that a partial assignment *satisfies a constraint* if it can be extended to a complete assignment that satisfies this constraint. A partial assignment is *legal* if it satisfies all constraints.

A subset S of constraints of a CSP is infeasible if no complete assignment satisfies all constraints in S simultaneously, otherwise it is feasible. Following the terminology in [2, 3, 4, 15] a subset of constraints is called an *irreducible infeasible set* (IIS) *of constraints* if it is infeasible, but becomes feasible when any one constraint is removed. Similarly, a subset V of variables of a CSP is infeasible if there is no legal partial assignment of the variables in V, otherwise it is feasible. We define an *irreducible infeasible set* (IIS) *of variables* as any subset of variables which is infeasible, but becomes feasible when any one variable is removed.

For illustration of these concepts, consider the graph represented in Figure 1.a. It is 3-colourable, but not 2-colourable. The CSP corresponding to the 2-colouring problem on this graph has $V=\{v_1,\ldots,v_7\}$ as variable set (vertex set). To each edge (v_i, vj), we associate a constraint denoted (i,j) and imposing that v_i and v_j must receive different colours. Hence, the constraint set C is $\{(1,2),(1,3),(1,7),(2,3),(3,4),(4,5),(5,6),(5,7),(6,7)\}$. This CSP has four IISs of constraints,

$\{(1,2),(1,3),(2,3)\}$, $\{(5,6),(5,7),(6,7)\}$, $\{(1,3),(1,7),(3,4),(4,5),(5,7)\}$ and $\{(1,2),(1,7),(2,3),(3,4),(4,5),(5,6),(6,7)\}$ and three IISs of variables, $\{v_1,v_2,v_3\}$ $\{v_5,v_6,v_7\}$ and $\{v_1,v_3,v_4,v_5,v_7\}$.



1.a 1.b

Figure 1

Notice that if the variables of a CSP define an IIS of variables, then the constraints involving these variables do not necessarily define an IIS of constraints. As an example, consider the CSP associated with the 3-colouring problem for the graph in Figure 1.b. The vertex set V of the graph is an IIS of variables since the graph is not 3-colourable while the removal of any vertex produces a 3-colourable graph. However, the edge set is not an IIS of constraints since the removal of the edge linking v_1 to v_2 gives a graph that is still not 3-colourable.

Exhibiting an IIS of constraints or variables can be very useful in practice, especially IISs of small size. For example, when solving a timetabling problem, it often happens that there is no solution satisfying all constraints. An IIS represents a part of the problem that gives a partial explanation to this infeasibility. An IIS can therefore be very useful to the person in charge of building the timetable since he gets an idea on which data should be changed in order to get a solvable problem. Determining IISs of constraints or variables can also be very helpful to prove the inconsistency of a CSP. Indeed, IISs contain typically a small amount of constraints and variables when compared to the original problem, and a proof of inconsistency is therefore possibly easier to obtain on an IIS rather than on the original problem. To illustrate this, consider once again the k-colouring problem. Suppose that no heuristic algorithm is able to determine a k-colouring of the considered graph G. One may then suspect that G is not k-colourable. To prove it, it is sufficient to exhibit a partial subgraph G' (obtained by removing edges) or an induced subgraph G'' (obtained by removing vertices and all edges incident to these vertices) which is not k-colourable but which becomes k-colourable as soon as any edge of G' or any vertex of G'' is removed. The edges of the partial subgraph G' correspond to an IIS of constraints while the vertices of the induced subgraph G'' form an IIS of variables. If G' and G'' have fewer edges and vertices than G, then instead of proving that G is not k-colourable, it is hopefully easier to prove that G' or G'' is not k-colourable [6].

Crawford [5] and Mazure et al. [8] have designed algorithms to determine infeasible subsets of constraints for the Satisfiability problem. A good review on the detection of IISs of constraints in linear programs is given in [3]. A review of the theory and history of IISs in other types of mathematical programs is given in [4].

The problems of finding IISs of constraints and variables in an inconsistent CSP are two special cases of the LSCP. Indeed, given an inconsistent CSP, define as an element of E any complete assignment. To each constraint c_i (i=1,…,m) of the CSP let us associate the subset E_i of E containing all assignments that violate c_i. A set of constraints is infeasible if and only if it covers E. Hence, finding an IIS of constraints is equivalent to solving the LSCP. Procedure IS-ELEMENT (e, i) simply determines if the complete assignment e violates constraint i. Procedure MIN_WEIGHT (ω) returns a complete assignment e that minimizes the sum of the weights of the constraints violated in e. In the case of the k-colouring problem, define a *conflicting edge* as an edge having both endpoints with the same colour. Procedure MIN_WEIGHT(ω) returns a colouring that minimizes the sum of the weights of the conflicting edges. Notice that the problem solved by the procedure MIN_WEIGHT corresponds to Max-CSP when all weights equal 1.

Similarly, given an inconsistent CSP, define as an element of E any legal partial assignment. To each variable v_i of the CSP let us associate the subset E_i of E containing all legal partial assignments in which v_i has no value (i.e., v_i is not instantiated). A subset of variables is infeasible if and only if it covers E. Hence, here again, finding an IIS of variables is equivalent to solving the LSCP. Procedure IS-ELEMENT (e, i) returns "true" if and only if variable i is not instantiated in the partial assignment e. Procedure MIN_WEIGHT (ω) returns a legal partial assignment e that minimizes the sum of the weights of the variables that are not instantiated in e. In the case of the k-colouring problem, procedure MIN_WEIGHT (ω) returns a partial colouring without conflicting edges that minimizes the sum of the weights of the uncoloured vertices.

At this point it is important to observe that procedure MIN_WEIGHT typically solves an NP-hard problem, both in the case of searching for an IIS of constraints or an IIS of variables. While Section 3 contains exact algorithms for the LSCP, Section 4 will be devoted to the analysis of the heuristic algorithms obtained by replacing procedure MIN_WEIGHT by a heuristic version.

There are some cases where MIN_WEIGHT can be implemented in an efficient way. For illustration we mention a CSP which appears in the context of an interactive decision support problem [1]. A consistent CSP with a set C of constraints is considered, where the solutions represent the catalog of a company, i.e., all the variants of the articles

produced in that company. When configuring a product, the user of the decision support system specifies a series of additional constraints C_1, $C_{2,...}$ concerning the features of the product he is interested in. At some iteration p, this set of additional constraints may become infeasible (i.e., $C \cup \{C_1,..,C_{p-1}\}$ is feasible while $C \cup \{C_1,...,C_p\}$ is not). In order to guide the user, the system should provide a minimal subset S of $\{C_1,...,C_p\}$ such that $C \cup S$ is already infeasible. Set S is called an *explanation* in that it explains why constraint C_p generates inconsistency. Thanks to sophisticated data-structures, Amilhaste et al. [1] have developed an efficient procedure that implements procedure MIN_WEIGHT: the procedure provides a solution that violates no constraints in $C \cup \{C_p\}$ and a minimum weighted subset of constraints in $\{C_1,..,C_{p-1}\}$ for any weighting of the constraints. The procedure is used to determine a maximum subset F of $\{C_1,...,C_{p-1}\}$ such that $C \cup F \cup \{C_p\}$ is feasible. The algorithms presented in this paper make it possible to generate explanations while this was an open problem in [1].

CONCLUSION

In this paper, we have introduced a new problem, called the Large Set Covering Problem (LSCP) which can be seen as a variant of the well known unicost set covering problem. The specificity of the LSCP is that the set E to be covered and the subsets E_i (i=1,...,m) are not given in extension, while we are given a procedure, called MIN_WEIGHT, that can extract elements from E. There are at least two motivations for searching IISs of constraints or variables – and, in both cases, preferably IISs of reduced size. First, in many real-life applications, an IIS of constraints makes it easier to understand the cause of infeasibility and, eventually, may help the user to decide which constraints he should relax in his problem. Secondly, the identification of an IIS can be a powerful tool when no exact algorithm is able to prove the infeasibility of a CSP.

References:

[1] J. Amilhastre, H. Fargier, P. Marquis, "Consistency restoration and explanations in dynamic CSPs - Application to configuration", in: "Artificial Intelligence", vol. 135, no. 2002, pp. 199-234, 2002.

[2] W.B. Carvier, Systems of Linear Inequalities. *Annals of Mathematics* 23 (1921), pp 212-220.

[3] J.W. Chinneck, Finding a Useful Subset of Constraints for Analysis in an Infeasible Linear Program. *INFORMS Journal on Computing*, 9/2 (1997), pp 164-174.

[4] J.W. Chinneck, Feasibility and Viability, in T. Gal and H.J. Greenberg, eds., Recent Advances in Sensitivity Analysis and Parametric Programming, (Kluwer Academic Publishers, Boston, 1997) pp 14:2-14:41.

[5] J.M. Crawford, Solving Satisfiability Problems Using a Combination of Systematic and Local Search, in *Working* notes of the DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability, Rutgers University, NJ (1993).

[6] F. Herrmann and A. Hertz, "Finding the chromatic number by means of critical graphs. *ACM Journal of Experimental Algorithmics* 7/10 (2002), pp 1-9.

[7] A.K. Mackworth, Constraint satisfaction, in S.C. Shapiro ed., Encyclopedia on Artificial Intelligence (John Wiley \& Sons, NY, 1987), pp 205-211.

[8] B. Mazure, L. Saïs and É. Grégoire, Boosting complete techniques thanks to local search methods. *Annals of Mathematics and Artificial Intelligence* 22 (1998), pp 319-331.

[9] E.P.K. Tsang, Foundations of Constraint Satisfaction (Academic Press, London, 1993)

[10] J. Van Loon, Irreducibly Inconsistent Systems of Linear Inequalities. *European Journal of Operational Research* 8 (1981), pp 283-288.

***