ALGORITHMIC COMPLEXITY OF SOME CONSTRAINT SATISFACTION PROBLEMS

by

Daya Ram Gaur

B. Tech. Computer Science and Engineering
Institute of Technology, Banaras Hindu Univ.
Varansi, India.

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
April 1995

ISBN 0-612-06663-0

APPROVAL

Name:	Daya Ram Gaur
Degree:	Master of Science
Title of thesis:	Algorithmic Complexity of some Constraint Satisfaction Problems

Examining Committee: Dr. Pavol Hell
Chair

Dr. William S. Havens

Dr. Binay Bhattacharya

Dr. Lou Hafer

Dr. Peter Van Beek

Date Approved: November 8, 1994

Abstract

Constraint networks are a simple knowledge representation model, useful for describing a large class of problems in planning, scheduling and temporal reasoning. A constraint network is called decomposable if any partial solution can be extended to a global solution. A constraint network is called minimal if every allowed 2-tuple of assignments can be extended to a global solution.

Much of the existing research in the field has been aimed at identifying restrictions on constraint networks such that the resulting network is minimal or decomposable or both. In this thesis we will examine issues related to minimal networks. We will address the complexity issues related to minimal networks. We will show that determining whether a given constraint network is minimal is NP-Complete. We also show that given a 2-tuple finding a solution which contains this edge is NP-Complete in a minimal network.

We show that there exists a greedy algorithm for finding a solution to a subclass of minimal network. The recognition problem for this class is of the same complexity as the recognition problem for decomposable networks.

We use a result of Feige and Lovasz to show that there exists another class of constraint satisfaction problems for which determining the satisfiability is polynomial. The recognition problem for this class is also NP-Complete.

Next we address the weighted constraint satisfaction problem. In the weighted constraint satisfaction problem, associated with each assignment is a cost. The goal is to find a consistent assignment with minimum cost. We will show that for minimal graphs and 0/1 weights the weighted constraint satisfaction problem is NP-Complete.

For my mother

Chandrawati Sharma

and my father

Pandit Salig Ram

Acknowledgements

I would like to thank Bill Havens for introducing me to the field of constraint logic programming. Without his continuing support it would have been an impossible task. Lou Hafer for all the time he spent reading many drafts of the thesis. Readability of this thesis is due to his patience and the valuable suggestions.

I would also like to thank other members of my committee, Binay Bhattacharya and Pavol Hell for valuable suggestions and for reading and correcting my thesis. I would like to thank the faculty and the staff of the school and the members in the Intelligent Systems Lab for creating an environment conducive to research.

Finally I would like to acknowledge the fellow graduate students, Pinaki Mitra, Roman Bacik and Sridhar Hannenhalli for the fruitful discussions. I am grateful to Bill Havens and the School of Computing Science for the financial support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Constraint networks are a simple knowledge representation model, useful for describing a large class of problems in planning, scheduling, natural language understanding, image recognition, scene analysis [Mon74, Wal72, Wal75b, Wal75a], representation of physical systems [Bob84, J.84, JS84], temporal reasoning [DMP91, VK86], and specification of software systems [Bal85, BGW82, Luc85], to name a few. A Constraint Network can be viewed as a graph consisting of nodes and arcs. The nodes represent the variables and the arcs can be viewed as binary relations specifying the mutually consistent assignments between the nodes. A solution to such a network is an assignment of values to all the nodes which satisfies all the constraints simultaneously. The notion is easily extensible to constraint graphs in which the relations between the nodes are $n - ary$. In this thesis, however, we will work with binary constraints.

Waltz [Wal72, Wal75b, Wal75a] proposed an algorithm for consistent labelling of bidimensional scenes. He examined each pair of nodes linked by a line segment and eliminated the inconsistent values. This operation is called $arc-consistency$ [Mac77]. Arc consistency is central to many constraint processing algorithms. Montanari first introduced the idea of constraint networks [Mon74] and proposed the path consistency algorithm. Mackworth [Mac77] improved on the complexity of Montanari's algorithm by keeping track of modified constraints to avoid rechecking. In [MF85] Mackworth and Freuder discuss the complexity of some polynomial arc-consistency algorithms. Seidel [Sei81] gave an algorithm for constraint satisfaction based on *dynamic programming* which runs in $O(m * D^{f+1})$, where m is the number of constraints, D is the maximum size of the domains of the variablea and f is an integer whose value depends on the structure of the problem.

As many of the NP-complete problems [GJ79], such as graph coloring, are constraint satisfaction problems, the constraint satisfaction problem is NP-complete. Since the constraint satisfaction problem belongs to the class of hard problems it is conjectured that it is impossible to find a polynomial time algorithm for it. One approach can be to find the subclasses of constraint satisfaction problems which are polynomially solvable. A second approach can be to give good approximation algorithms for the general constraint satisfaction problem.

1.2 Definitions

Definition 1 Chromatic number: *The* chromatic number (χ) *of a graph is defined as the minimum number of colors required to color the vertices of the graph such that no two adjacent vertices get the same color.*

Definition 2 *Maximum Clique: A* maximum clique *is a complete subgraph of a given graph with maximum size.*

We denote the maximum clique size by ω. K_i represents a complete graph over i vertices.

Definition 3 Perfect Graph *A graph G is* perfect *if every vertex induced subgraph G_A has the property that $\omega(G_A) = \chi(G_A)$.*

In a perfect graph a maximum clique can be found in polynomial time [GLS88].

Definition 4 Constraint Satisfaction Problem (CSP): *Let X be the set of variables and D be the set of domain values. Relations $R_{i,j}$ between variables x_i and x_j define, the mutually consistent values. A* solution *to a CSP is an instantiation of all the variables such that relations between all the pairs of variables are satisfied.*

A CSP can be represented as a 3-tuple (X, D, R) where X is the set of variables, D is the set of associated domains and R is the set of binary relations over domains specifying the consistent values.

A CSP can also be visualized as a graph problem. From a given CSP the constraint graph can be constructed by introducing a node for each assignment of a domain value to a variable. Consistent instantiations are connected by an arc. Finding a solution to the CSP amounts to finding a maximum clique in the constraint graph. We will refer to the graph generated from the constraint satisfaction problem as the *constraint graph.*

Definition 5 Path Consistency: *A constraint network is* path consistent *if and only if, for every 3-tuple of variables x_i, x_j, x_k, the following holds: for every instantiation of x_i and x_j that satisfies the relation $R_{i,j}$, there exists an instantiation of x_k such that the relations $R_{j,k}$ and $R_{i,k}$ are satisfied.*

Definition 6 k-consistent network: *A constraint network is* k-consistent *if and only if, given an instantiation of any $k-1$ variables satisfying all the direct relations between those variables, there exists an instantiation of the k^{th} variable such that the k values taken together satisfy all the relations between them.*

Definition 7 Strongly k-consistent: *A constraint network is* strongly k-consistent, *if it is consistent for $1 \leq i \leq k$.*

Definition 8 Decomposable Network: *A constraint network is called* decomposable *if any partial solution can be extended to a complete solution.*

Definition 9 Minimal Network: *A constraint network is called* minimal *if any allowed 2-tuple of assignments can be extended to a complete solution.*

Let the graph associated with minimal constraint network be called a minimal constraint graph. In graph theoretic terms a minimal network can be characterised as: every edge in the minimal graph participates in a maximum clique, where a maximum clique is a solution to the constraint satisfaction problem. Since this thesis is concerned solely with constraint networks and constraint graphs, the word constraint will be dropped whenever the meaning is clear from the context. An important point to be kept in mind while reading this thesis is that we distinguish constraint networks from constraint graphs. The phrase *constraint graph* is being used in the non-traditional sense. Whenever we use the word constraint graph we refer to the product graph as described in section 2.1 and figure 2.1. By *Constraint Network* we mean the graph which is defined over variables in the constraint satisfaction problem where two variables are linked by an edge if there is a binary constraint over the two variables.

1.3 Overview of the thesis

In chapter 2 we will introduce some definitions and show the reduction of the constraint satisfaction problem to a graph problem. The reduction will show that the constraint satisfaction problem can be formulated as finding a maximum clique in a graph.

We study minimal graphs and show that they are not necessarily perfect. We will address the complexity of recognizing minimal graphs, finding minimal subgraphs of a general graph, and finding a solution to a minimal graph. The notion of minimal networks is interesting to study because it is the natural generalization of *decomposable networks*, which admit backtrack free solution. The solution to a decomposable network can be computed by a greedy algorithm; the structure of the constraints guarantees the correctness of the algorithm.

In chapter 3 we use a result of Fiege and Lovasz to show that there exists a class of constraint satisfaction problems for which satisfiability can be determined in polynomial time. This is exactly the class of constraint networks for which the associated graph has chromatic number χ equal to the size of the maximum clique ω. This raises the question of recognizing graphs for which the former property is true. We show that it is NP-complete to determine whether $\omega = \chi$ is true for any graph.

We then address the weighted constraint satisfaction problem. In the weighted constraint satisfaction problem, associated with each assignment is a cost. The goal

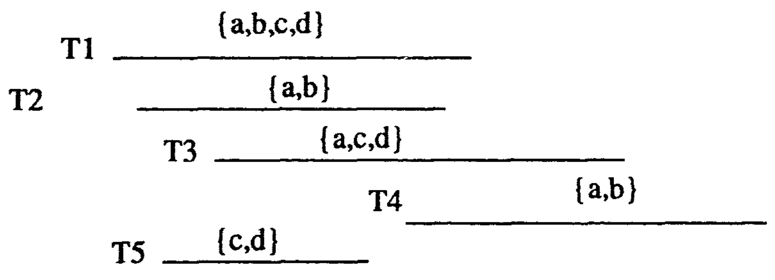


Figure 1.1: A schedule for a simple resource allocation problem

is to find a consistent assignment with minimum cost. We will show that for minimal graphs and 0/1 weights the weighted constraint satisfaction problem is NP-Complete.

We conclude with a summary of results and open problems in the last chapter.

1.4 Related Work

There has been a considerable amount of effort expended on identifying the classes of constraint networks which are easily satisfiable. These classes are obtained either by restricting the type of constraints or the topology of the network.

1.4.1 Resource Allocation Problem

The resource allocation problem is an example where the constraints are of special type as well as the network has a special structure. In particular the constraints are all disequalities and the network is an interval graph.

In a resource allocation problem, we have a set of *tasks*. Associated with each *task* is a set of allowable resources. The problem is to find an assignment of a resource to each task such that no two tasks which overlap share a resource in common. We assume that the execution times for each task have already been set.

Figure 1.1 shows a resource allocation problem comprising of five tasks $T_1, \ldots, T_5$. The resources which can be used to perform a task are specified by a set. For example

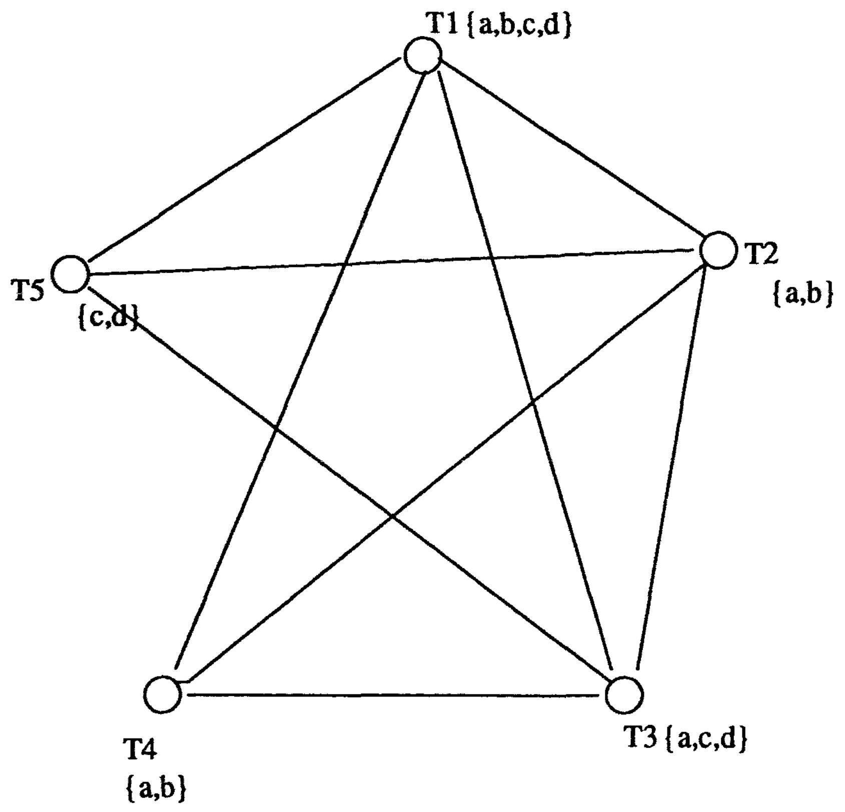


Figure 1.2: Corresponding Graph coloring problem

T_1 can use any of the resources $\{a,b,c,d\}$. A resource allocation problem can be expressed as a graph coloring problem defined as follows:

Let T be the set of tasks and $\forall i\ C_i$ is the set of resources used by task T_i. Define $G = (V, E)$ to be a graph such that $V = T$ and $\forall i \neq j : T_i \cap T_j \neq \emptyset$ *implies* $(T_i, T_j) \in E$.

Given G and associated colors with each node (C_i *here*) finding a valid coloring is equivalent to solving the resource allocation problem, where a valid coloring is an assignment of colors to the nodes from the list of permitted colors such that no two adjacent nodes have the same color. Figure 1.2 shows the graph coloring problem corresponding to the resource allocation problem shown in Figure 1.1.

The graph defined by the resource allocation problem is an *interval graph* [Gol80]. Coloring of interval graphs when the colors are not uniformly available is known to be NP-Complete [AS87], whereas the regular coloring of interval graphs (when all the colors are available to all the nodes) can be accomplished in linear time [Gol80].

Choueiry and Faltings in [CF94] describe how the properties of interval graphs can be exploited to simplify the problem. They group successive tasks which can be executed by the same resource. The $VAD - heuristic$ described in [CF94] performs such grouping. The $VAD - heuristic$ delays the assignment of the most constrained resource.

Let $G = (V, E)$ be an interval graph corresponding to some resource allocation problem. $U \subset V$ is called an independent set if no two members of U share an edge. A *covering* of G by independent sets is a collection of sets $(U_1, \ldots, U_m) | \bigcup_{i=1}^{m} U_i = V$.

It is well known that a covering of an interval graph can be obtained in linear time [Gol80]. Furthermore this covering is the smallest possible. Such a covering groups tasks into sets which can have the same resource. If all the resources are available to every task then such a covering indeed gives a coloring.

Relationship to k-consistency

If the constraint graph is an interval graph then the variables can be grouped into maximal cliques such that no two variables get the same color. We can then rewrite the coloring problem using higher order constraints. We call the higher order operator *all_different(List)*. It takes as an argument a list of domain variables and finds an instantiation for each variable in the argument list such that no two variables in the *List* get the same value.

The all_different formulation for the problem shown in Figure 1.2 is:

solve:-

$$all\_different(T_1, T_2, T_3, T_4),$$
$$all\_different(T_1, T_2, T_3, T_5).$$

As mentioned previously, if the constraint graph is an interval graph then we can find all the maximal cliques in polynomial time [Gol80]. This grants us the liberty of rewriting the problem using higher order constraints. In general we cannot solve the conjunction of these higher order constraints efficiently.

As we will see ahead: a single all_different constraint can be solved efficiently. This provides us with a way to define local-consistency stronger than arc-consistency for binary networks. By making the network locally-consistent (with respect to the higher order constraint) we hope to prune more search space.

Observation:

- Let n be the maximum cardinality of the argument lists to all_different constraints describing a problem. If we make the network n consistent (in the Freuderian sense) then the network is globally consistent.

Let k be the size of the maximum intersection of these cliques. In [DAGJ95] it is shown that if the constraint network is $n - ary\ k + 1 - consistent$ then there exists a backtrack free search.

1.4.2 all_different constraint solver

In [Reg94] Regin describes an algorithm for making constraints of difference (all_different constraint) consistent. First we define a few terms:

Definition 10 all_different constraint: *is one which has inequality as the constraint on every pair of variables in the constraint.*

Definition 11 all_differentCSP: *is a constraint satisfaction problem in which all the constraints are of the type* all_different.

Definition 12 Consistent_all_different constraint: *is an all_different constraint such that for every allowed domain value in a variable there exists a globally consistent assignment for the other variables.*

Definition 13 Consistent_all_differentCSP: *is an all_differentCSP such that for any variable every allowed domain value is consistent with all the all_different constraints involving the variable.*

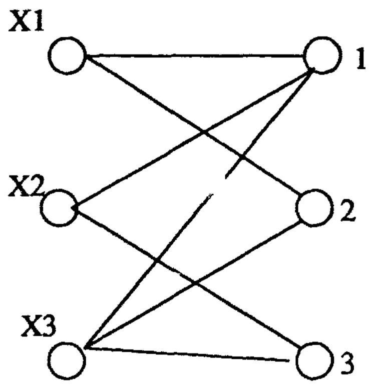


Figure 1.3: Bipartite Graph

Let C be *all_different*(X_1, X_2, X_3) where the domain of X_1 is $\{1,2\}$, the domain of X_2 is $\{1,3\}$ and the domain of X_3 is $\{1,2,3\}$. Let X and D denote the union of all the variables and the domains respectively . We define B to be a bipartite graph over nodes (X, D). (x_i, d_j) is an edge in B if $d_j \in$ *domain of* x_i. The bipartite graph corresponding to constraint C is shown in Figure 1.3.

Definition 14 Matching: *A subset of edges is called a matching if no two edges share a vertex in common.*

A matching with maximum cardinality in a graph is called a *maximum matching.*

A solution to a constraint C is a maximum matching in B. Furthermore a maximum matching in B is a solution to C. By definition if C is consistent then every edge in B belongs to a maximum matching. Maximum matching in bipartite graphs can in computed in $O(n^{1.5}\sqrt{(}m/\log n))$ [ABMP91].

Given an all_differentCSP we can make it consistent by removing all the values which do not belong to a maximum matching in some constraint C. Since each value can be removed at most once, the number of calls to the matching subroutine is bounded by the number of domain values.

In general it is hard to find the maximal cliques describing the constraint satisfaction problem. In the case of resource allocation problems, the underlying constraint graph is an interval graph, therefore we can compute all the all_different constraints in polynomial time and use higher order consistency to prune more search space.

1.4.3 Minimality related to the Topology of a constraint network

Next we will discuss the classes which are obtained by restricting the topology. Montanari [Mon74] showed that if the constraint network is a tree, path consistency ensures that the network is minimal.

Definition 15 Ordered Constraint Network: *An ordered constraint network has its nodes arranged in a linear order.*

Definition 16 Width at a node *in an ordered constraint network is the number of links that lead back from that node to previous nodes in the ordering.*

Definition 17 Width of an ordering *is the maximum width at the nodes.*

Definition 18 Width of a constraint network *is the minimum width over all the orderings of the network.*

Freuder [Fre82, Fre85] related the *width* of a network to the level of local consistency required to ensure a backtrack free solution. The following theorem is from [Fre82].

Theorem 1 *(Freuder) Given a constraint satisfaction problem:*

(1) A search order is backtrack-free if the level of strong consistency is greater than the width of the corresponding ordered constraint network.
(2) There exists a backtrack-free search order for the problem if the level of strong consistency is greater than width of the constraint network.

It can be seen that the width of a tree is 1. Therefore by the previous theorem $2-$ *consistency* would give us a backtrack-free search in trees.

Dechter and Pearl [DP88] provide an adaptive scheme where the level of consistency is adjusted on a node by node basis. Freuder [Fre90] generalizes the previous result (Theorem 1) on trees to $k-trees$.

1.4.4 Minimality related to the type of constraints

Let $\leq$ be a partial order on the set D_i of values of the variable X_i. Moreover we impose a lattice structure with inf and sup operations on D_i. We treat $r \leq s$ as being equivalent to $x_{i,r} \leq x_{i,s}$. A total relation R_{ij} between the sets D_i and D_j is called *monotone* if:

(i) if $R_{ij,rs} = 1$ and $t \geq r$ then $R_{ij,ts} = 1$ and conversely
if $R_{ij,rs} = 1$ and $t \leq s$ then $R_{ij,rt} = 1$
(ii)if $R_{ij,ps} = 1, R_{ij,qs} = 1$ and $r = inf(p,q)$ then $R_{ij,rs} = 1$ and
if $R_{ij,rp} = 1, R_{ij,rq} = 1$ and $s = sup(p,q)$ then $R_{ij,rs} = 1$.

A *monotone* relation is shown in Figure 1.4.

Montanari [Mon74] showed that if the relations are monotone then path consistency ensures that the network is minimal and decomposable.

The following theorem from [Dec90] relates the size of the domains of the variables and the level of local consistency required to ensure a decomposable network.

Theorem 2 *(Dechter)Any k-valued r-ary constraint network that is strongly (k(r-1)+1) consistent is globally consistent. In particular, any k-valued binary constraint network that is strongly (k+1) consistent is globally consistent.*

If we have a bi-valued binary network, ensuring 3-consistency would guarantee a globally consistent network, though this might not be the fastest way to solve bi-valued binary networks, as bi-valued binary networks are equivalent to 2-Sat which can be solved in linear time.

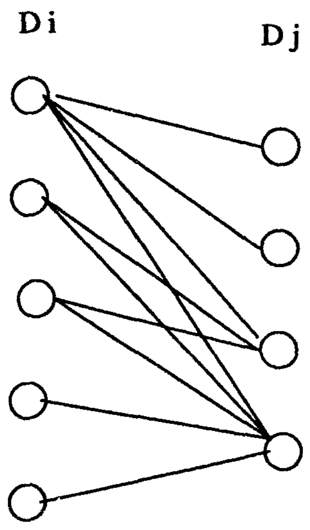


Figure 1.4: A monotone relation

Definition 19 Functional Relation R: *A binary relation represented in matrix form is functional if and only if there is at most one* 1 *in each row and in each column of* R.

Deville and Van Hentenryck [DVH91] show that if the relations are monotone and functional then arc consistency itself guarantees satisfiability. Van Beek in [VB92] generalized the previous result to show that if a matrix representation of the relation is *row-convex* then path consistency ensures that the network is both minimal and decomposable.

1.4.5 Some Graph theoretic problems

For related definitions the reader is referred to section 1.2.

Zykov [Zyk49] defined a graph G to be $k-saturated$ if it does not contain a $k+1$ clique, but every graph obtained from G by adding an edge contains a clique of size $k+1$. Hajnal [Haj65] studied $k-saturated$ graphs.

Suppose that the constraint graph is $k-saturated$ and the degree of each vertex is $\leq n-2$, where n is the number of vertices in the graph. A maximum clique in such a constraint graph can be computed as follows: let (a, b) be a pair of vertices such that a is not connected to b. If we add the edge (a, b) to the constraint graph then we get a clique of size $k+1$, therefore the common neighbours of a and b have to be a $k-1$ clique. Thus the common neighbours of a and b with either a or b form a clique of size k.

From this it follows that every node in a $k-saturated$ constraint graph belongs to a maximum clique. Clearly, $k-saturated$ graphs are properly contained in minimal graphs. Observe that the assumption about the degree of each vertex is natural. We assume that the domain of each variable has more than one element, which implies that each row in the constraint graph has at least two nodes. As all the nodes in a row form an independent set, the degree of each node cannot be equal to the number of the vertices.

Let $K_n(i), i = 1 \ldots n$, be n complete graphs of n vertices. Assume that every two of the $K_n(i)'s$ have at most one vertex in common. The following question has been posed in [Erd81]:

Prove that the graph $\cup_{i=1}^{n} K_n(i)$ has chromatic number n.

Clearly the above stated class of graphs is a subclass of minimal networks. The problem is still open to the best of my knowledge. This shows how hard it is to study the structure of the minimal graphs.

Chapter 2

Minimal Networks

2.1 Introduction

To facilitate the discussion we will reintroduce some of the terms.

Definition 20 Constraint Satisfaction Problem (CSP): *Let X be the set of variables and D be the set of domain values. Relations $R_{i,j}$ between variables x_i and x_j define, the mutually consistent values. A* solution *to a CSP is an instantiation of all the variables such that relations between all the pairs of variables are satisfied.*

A CSP can be represented as a 3-tuple (X, D, R) where X is the set of variables, D is the set of associated domains and R is the set of binary relations over domains specifying the consistent values.

A CSP can also be visualized as a graph problem. From a given CSP the constraint graph can be constructed by introducing a node for each assignment of a domain value to a variable. Consistent instantiations are connected by an arc. Finding a solution to the CSP amounts to finding a maximum clique in the constraint graph. We will refer to the graph generated from the constraint satisfaction problem as the *constraint graph.* Figure 2.1 is a constraint graph for some constraint satisfaction problem.

All the $x_{i,j}$ in row i correspond to the values x_i can take. For example variable x_1 has four domain values $d_{11}, d_{12}, d_{13}, d_{14}$. The corresponding assignments are denoted by the variables $x_{11}, x_{12}, x_{13}, x_{14}$. There is an edge connecting two nodes if and only

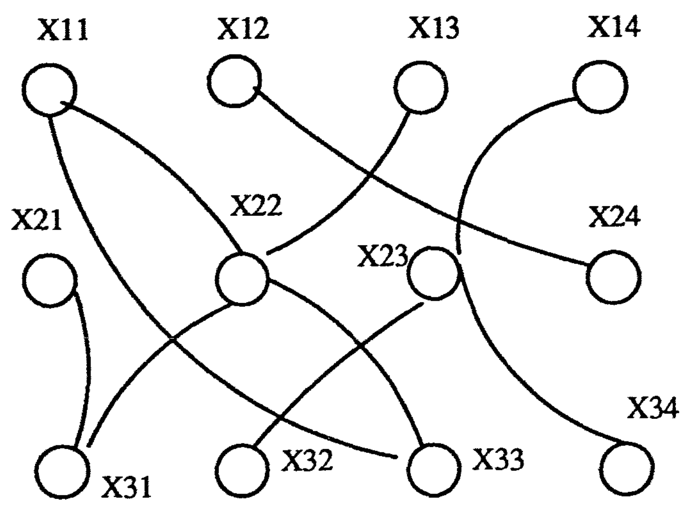


Figure 2.1: CSP

if the assignments are mutually compatible. By definition for all j, k, $x_{i,j}$ is not connected to $x_{i,k}$. If we visualize the nodes of the constraint graph as a matrix of rows and columns then the previous statement asserts that are no edges in between elements of a row. All the edges are from one row to another. If the CSP is satisfiable, then there is a clique of size $\mid X \mid$ in the graph G. If the CSP is not satisfiable then there does not exist a clique of size $\mid X \mid$. Furthermore only one element from each row can participate in the formation of the clique. The CSP shown in Figure 2.1 has a unique solution x_{11}, x_{22}, x_{33}.

The transformation described above can be carried out in polynomial time.

Definition 21 Path Consistency: *A network is* path consistent *if and only if, for every 3-tuple of variables x_i, x_j, x_k, the following holds: for every instantiation of x_i and x_j that satisfies the relation $R_{i,j}$, there exists an instantiation of x_k such that the relations $R_{j,k}$ and $R_{i,k}$ are satisfied.*

Definition 22 k-consistent network: *A network is* k-consistent *if and only if, given an instantiation of any $k-1$ variables satisfying all the direct relations between those* *variables, there exists an instantiation of the k^{th} variable such that the k values taken together satisfy all the relations between them.*

Definition 23 Strongly k-consistent: *A network is* strongly k-consistent, *if it is consistent for $1 \leq i \leq k$.*

Definition 24 Minimal network: *A network is called* minimal *if every consistent 2-tuple of instantiated variables can be extended to a complete instantiation of the variables.*

Definition 25 Decomposable network: *A strongly n-consistent network is said to be* decomposable. *Decomposable networks can be instantiated without backtrack. A strongly n-consistent network is also minimal. The converse is not true. We assume that n is the number of variables in the corresponding CSP.*

Figure 2.2 shows the minimal constraint graph for a CSP which is not decomposable. The partial solution defined over nodes x_{22}, x_{31}, x_{41} cannot be extended to a clique of size four, whereas each edge participates in a clique of size four. The network is 1,2,3-consistent, but not 4-consistent.

In this chapter we will examine problems related to minimal constraint graphs. Recall that by constraint graphs we refer to the graphs which are derived from the constraint satisfaction problem as specified in section 2.1 (Figure 2.1 shows an example of a constraint graph).

In the traditional usage of the term minimal graph we assume that the size of the maximum clique in the constraint graph is equal to the number of variables in the corresponding CSP (i.e, the CSP has a solution). If ω denotes the size of the maximum clique in a graph and χ is its chromatic number then the previous statement asserts that for minimal graphs $\omega = \chi$. We would relax this definition to take into consideration the minimal graphs for which ω may not be equal to χ. If $\omega \neq \chi$ for a minimal graph then the corresponding CSP does not have a solution. If $\omega \neq \chi$ for a minimal graph then the term ***solution*** means a maximum clique in the constraint graph. In the corresponding CSP it would be the maximum consistent set of variables

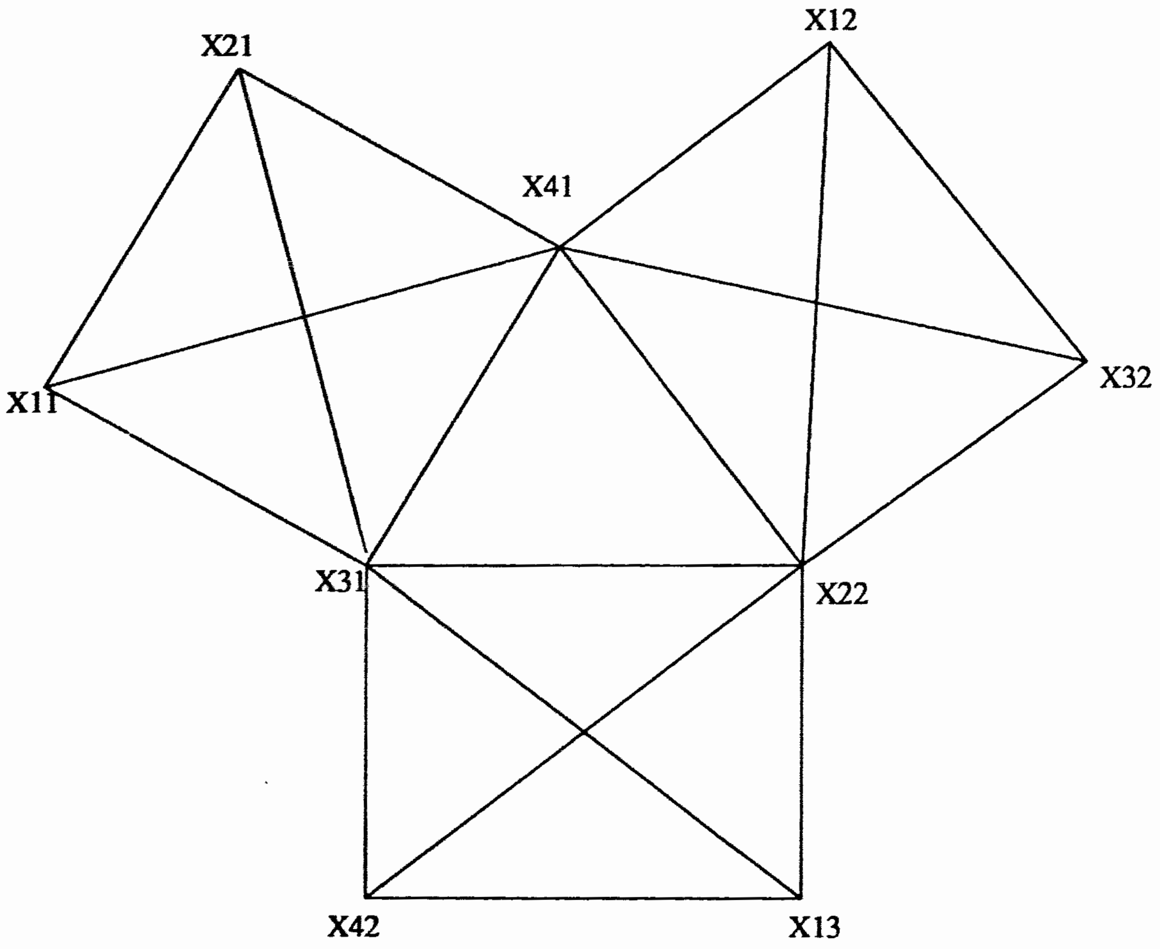


Figure 2.2: Minimal Network

which can be assigned a value [1]. In this framework we can talk about unsatisfiable *CSPs* and computing maximally consistent assignment sets to them.

In the next theorem we will show the equivalence of the two definitions. With this equivalence we will able to prove properties of minimal graphs without having to worry about the size of the maximum clique.

Observe that our definition properly contains the minimal graphs for which $\omega = \chi$. Let us assume that $\omega \neq \chi$ for the minimal graphs under consideration.

Theorem 3 *If we can find a maximum clique in minimal graph for which $\omega = \chi$ then we can find a maximum clique in minimal graph for which $\omega \neq \chi$.*

Proof: Let G_a be the minimal graph under consideration. We can assume that $\omega \neq \chi$ for $G_a = (V_a, E_a)$. From this graph we will construct another minimal graph $G_b = (V_b, E_b)$ such that $\omega = \chi$ for G_b. Let the maximum clique size in G_a be n. Vertices of G_b are defined as two tuples $V_b = \cup_{v \in V_a}(v, i)$ $i = 1..n$. Two vertices $(v_1, i), (v_2, j) \mid i \neq j$ belonging to G_b are connected if (v_1, v_2) is an edge in E_a. This states that the edges of G_a give rise to edges in G_b. Observe that if $i = j$ then we are not putting any edges in G_b. G_b can again be visualized as comprising of n rows, where each row contains $\mid V_a \mid$ number of vertices. There are no edges in a row. Edges are between rows only. G_b so constructed is minimal because every edge in G_b belongs to a clique of size n such that there is a vertex from each row in the clique. As the rows are independent sets we can cover G_b by n independent sets therefore the chromatic number of G_b is also n. If we can find a maximum clique in G_b we can find a maximum clique in G_a. This follows from the fact that the first element in the 2-tuple defining the vertex in G_b is a vertex in G_a. For any two edges in the clique in G_b there has to be an edge between the corresponding vertices in G_a (by construction). □

Theorem 3. tells us that finding maximum cliques in minimal graphs with $\omega = \chi$ is as hard as finding cliques in minimal graphs with $\omega \neq \chi$. From now on minimal graphs may or may not have maximum clique size equal to the chromatic number.

In this chapter we will address the following problems related to minimal graphs.

[1]The reader is cautioned that the solution to a minimal graph with $\omega \neq \chi$ is a maximum consistent set of variables which are assigned some value even though the *CSP* as a whole might be unsatisfiable.

Problem 1.

INSTANCE M: Given a constraint graph.

QUESTION: Is the constraint graph minimal?

Problem 2.

INSTANCE S: Given a minimal constraint graph G and an edge e in G.

QUESTION: Find a maximum clique in G which contains the given edge e.

We will show that M is NP-Complete: the reduction follows from the problem of finding a $k-$ *clique*. It is interesting that problem S is also NP-Complete.

Next we define S_a to be the problem of finding a maximum clique in the minimal graph [2].

Problem 3.

INSTANCE S_a: Given a minimal constraint graph G.

QUESTION: Find a maximum clique in G.

Let us define $P2$ as the problem of finding the minimal constraint graph in a graph G. This can be restated as : given a constraint graph G, does there exist an edge induced subgraph of G such that the induced subgraph is minimal for some clique size k.

Problem 4.

INSTANCE $P2$: Constraint Graph $G = (V, E)$.

QUESTION: Does there exist $E_1 \subset E$ such that $G_1 = (V, E - E_1)$ is minimal and has a clique of size k?

We will show that there exists a sub-class of minimal graphs for which S_a can be solved in polynomial time. The recognition problem for this sub-class is of the same complexity as the recognition problem for decomposable graphs. The recognition problem for decomposable graphs is known to be in $Co-NP$. Next, we will show that the problem $P2$ is NP-Complete.

Definition 26 Relational Matrices: *A binary relation between two variables x_i and x_j can be represented in a* $(0,1)$ *matrix form with* $\mid D_i \mid$ *rows and* $\mid D_j \mid$ *columns by*

[2]Note that if $\omega \neq \chi$ for the minimal graph then the corresponding CSP is unsatisfiable. The maximum clique in the minimal graph gets mapped onto the maximum consistent set of variables which can be assigned a value.

imposing an ordering on the domains of the variables. A 0 in row a and column b means that the pair (a, b) such that $\{a \in D_i, b \in D_j\}$ is not permitted. A relational matrix *describes all the pairs of relations between values of the variables.*

For example, the relational matrix for variables x_1 and x_2 from Figure 2.1 is

$$\left|\begin{array}{ccccc} & x_{21} & x_{22} & x_{23} & x_{24} \\ x_{11} & 0 & 1 & 0 & 0 \\ x_{12} & 0 & 0 & 0 & 1 \\ x_{13} & 0 & 1 & 0 & 0 \\ x_{14} & 0 & 0 & 1 & 0 \end{array}\right| \tag{2.1}$$

Definition 27 Row-Convex: *A 0/1 matrix M is said to be* row-convex *if all the rows have the embedded ones property, i.e. each row of M can be expressed by the regular expression* 0*1*0*.

Van Beek [VB92] showed that if the relational matrix of a constraint satisfaction problem is row-convex then path consistency guarantees a minimal and decomposable network. Before describing his theorem, we will illustrate the relational matrix for a complete constraint satisfaction problem.

Given a graph $G = (V, E)$ and k colors, the coloring problem is to determine a unique assignment of colors to the vertices of G such that no two adjacent vertices get the same color.

Consider the 3-color problem shown in Figure 2.3. There are three vertices in the graph. The colors are designated by labels a, b, c. The binary relation between the two variables x_1 and x_2 can be written as

$$\left|\begin{array}{cccc} & a & b & c \\ a & 0 & 1 & 1 \\ b & 1 & 0 & 1 \\ c & 1 & 1 & 0 \end{array}\right| \tag{2.2}$$

where the rows correspond to the domain of variable x_1 and the columns correspond to the domain of variable x_2. Since there is an edge between x_1 and x_2 they

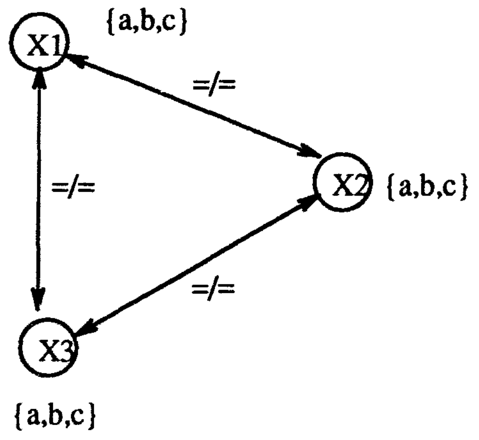


Figure 2.3: 3-colorability

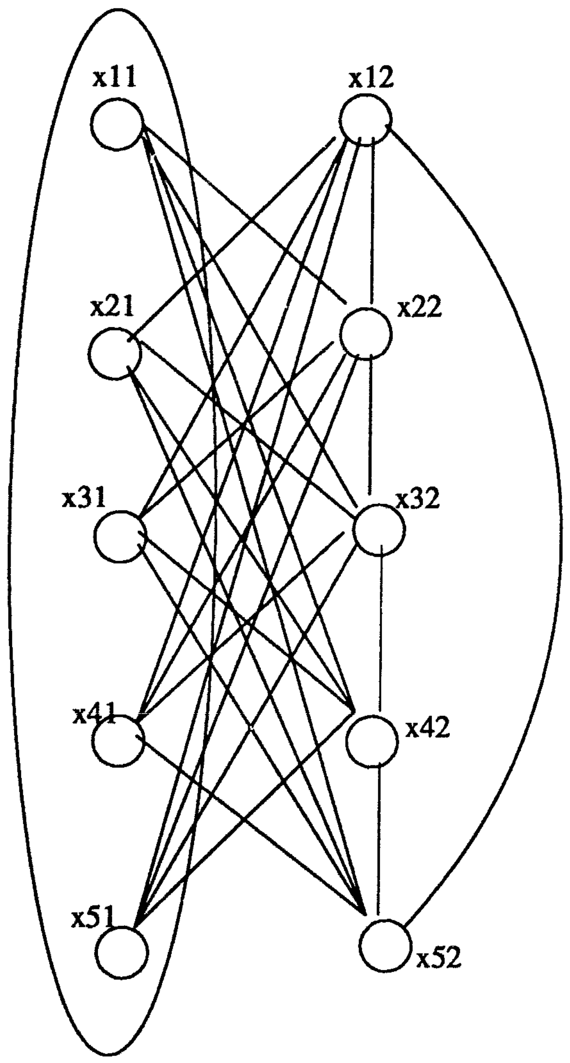
cannot have the same colors, hence the diagonal entries are zero. If two vertices are not connected then the relational matrix corresponds to a matrix with all ones.

The relational matrix corresponding to Figure 2.3 is given by the following matrix:

$$\left|\begin{array}{cccc} & x_1 & x_2 & x_3 \\ x_1 & R_{1,1} & R_{1,2} & R_{1,3} \\ x_2 & R_{2,1} & R_{2,2} & R_{2,3} \\ x_3 & R_{3,1} & R_{3,2} & R_{3,3} \end{array}\right| \tag{2.3}$$

Theorem 4 (Van Beek) *Let L be a set of $0,1$ matrices closed under composition, intersection, and transposition such that each element of L is row-convex. Let R be a binary constraint network with all the relations taken from L. The path consistency algorithm will correctly determine the minimal network of R. Further the minimal network will be decomposable.*

If the constraint graph is a perfect graph then a maximum clique (and hence a solution to the CSP) can be found in polynomial time [GLS88]. Unfortunately, even when all relations are row-convex, the constraint graph is not necessarily perfect. Figure 2.4 shows a constraint graph for which all the binary relations are row-convex but



All relations are row-convex but not Perfect
x11-x51 form a clique (not shown)

Figure 2.4: Row-Convex Relations but not Perfect

the graph is not perfect, as the subgraph induced over vertices $(x_{12}, x_{22}, x_{32}, x_{42}, x_{52})$ is an odd cycle, which has $\omega = 2$ and $\chi = 3$.

2.2 Minimal Graphs

Recall that a network is *minimal* (with $\omega = \chi$) if each pair of values allowed by the constraints participates in at least one consistent instantiation. Let $(x_i, x_j) \in R_{ij}$. If the network is minimal (with $\omega = \chi$) then every edge (x_i, x_j) in the constraint graph can be extended to a maximum clique of size n.

Let $G = (x_{ij}, E)$ be a constraint graph, where $i = 1 \ldots n$ and $j = 1 \ldots m$. n is the number of variables in the constraint satisfaction problem and m is the size of the domain. For simplicity we assume that all the domains have equal size.

In this section we will assume that $\omega = \chi$ for G. Showing that the restriceted version of the problem is NP-Complete implies the NP-Completeness of the general problem (where ω may not be equal to χ).

We will now show that the problem M (Is G minimal?), is NP-Complete.

Observation 1. G is minimal (in the strong sense)[3] if and only if G can be covered by maximum cliques of size n.

The forward direction follows from the definition of minimal networks. The reverse implication follows because, if G can be covered by cliques of size n, then each edge participates in a clique of size n. Now we know that determining whether a network is minimal is equivalent to asking whether G can be covered with maximum cliques of size n. Note that covering does not exclude the possibility of nodes being shared between the cliques.

Before we give the proof we will describe *Turing reduction.*

Turing Reduction from a search problem Π to a search problem Π' is an algorithm A that solves Π by using a hypothetical subroutine S for solving Π' such that, if S were a polynomial time algorithm for Π', then A would be polynomial time algorithm for Π.

[3]G is minimal in the strong sense if the CSP associated with is satisfiable, this implies that $\omega = \chi$ for G.

The problem which we use for the reduction is:

INSTANCE k-clique: Graph $G = (V, E)$.
QUESTION: Does G have a clique of size k?

NP-completeness of this problem is explained in the next paragraph.

Given an arbitrary graph and a number k, finding a clique of size k is NP-Complete. If this were not the case then we could find the maximum clique by executing an oracle for each k in decreasing order and stopping when the output is indeed a clique, thereby solving an NP-Complete problem.

Theorem 5 *Problem M is NP-Complete.*

Proof: We will give a *Turing reduction* to show that M is as hard as any NP-Complete problem. In particular we will show that M can be used to find the k-clique in a graph (which is known to be NP-Complete).

The construction for the Turing reduction is: for each edge e in the graph G we generate a new graph by covering all the edges in G except e by cliques of size n, where n is the number of the vertices in G. This covering is accomplished by adding a clique over $(n-2)$ vertices for each edge (a, b) and connecting a and b to all the vertices in the clique of size $(n-2)$. It is to be noted that for each edge we introduce a new K_{n-2}.

Now we ask the question whether all the generated graphs are minimal. If for one instance the answer is affirmative it implies that G contains a clique of size n. This follows from the fact that the edge which is not covered by the introduced clique is covered by a clique in G. If answers to all the instances are no, then we repeat the whole operation with cliques of size $n-1$. We continue this until we get an affirmative answer.

We will illustrate the process by an example. In Figure 2.5: the first column corresponds to the graph for which we are trying to find the k-clique. The second and the third columns correspond to generated graphs: for each edge e in the graph G we generate a new graph by covering all the edges in G except e by cliques of size

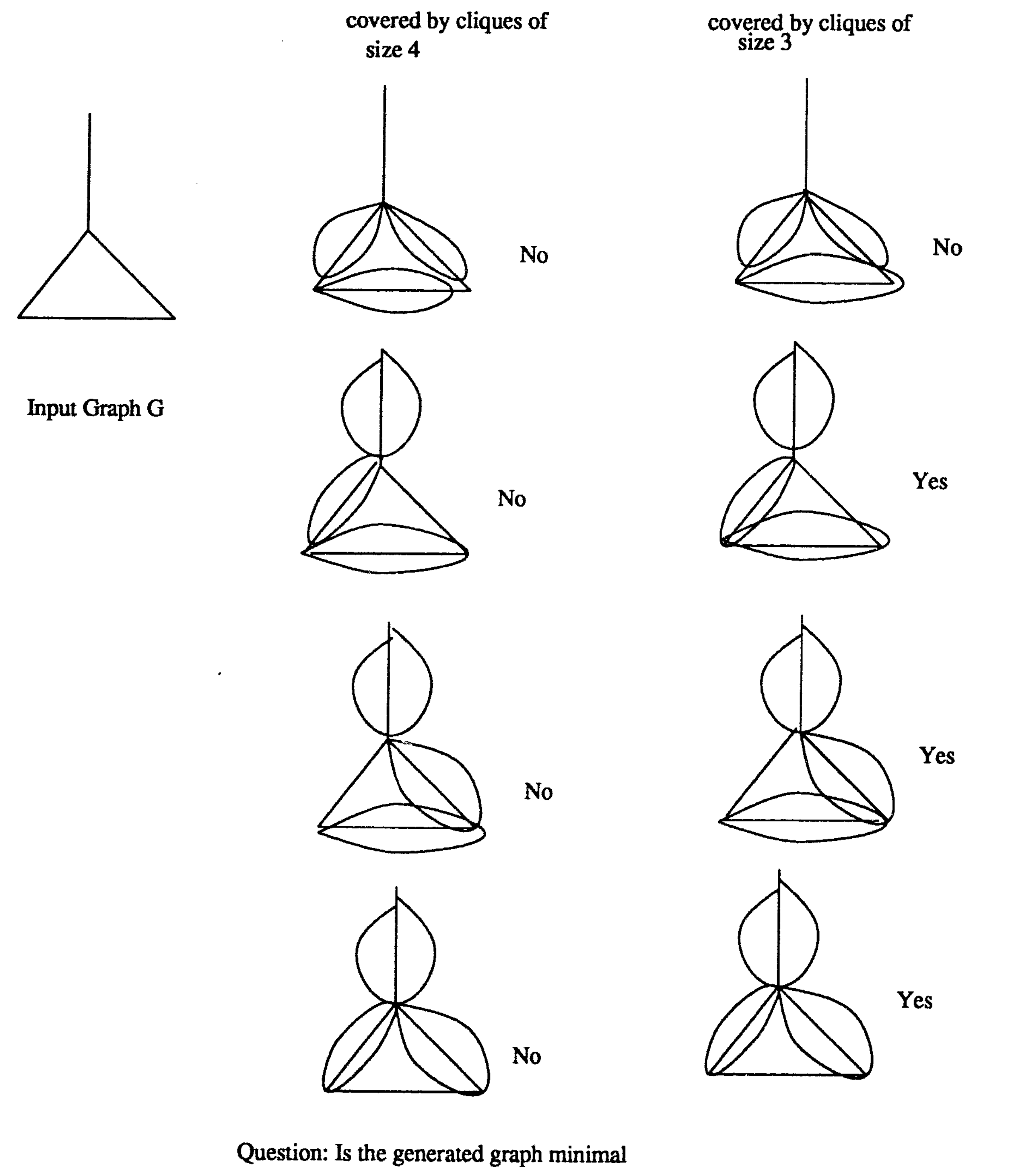

covered by cliques of size 4
covered by cliques of size 3
Input Graph G
No
No
No
Yes
No
Yes
No
Yes
Question: Is the generated graph minimal

Figure 2.5: Reduction

n, where n is the number of the vertices in G. The loops in Figure 2.5 denote the cliques being added to every edge. For each edge e we have a graph such that all the edges except e are in a clique of size 4 and 3 respectively. Now for graphs in column 2 and 3 we ask the question whether they are minimal. If the answer for one of the graphs is "yes" then we stop. An affirmative answer means that there is a clique of size 3 in the original graph. This follows from the fact that there is an edge which is not covered by an introduced maximum clique but the network is minimal therefore it has to be covered by a clique in the original graph.

As shown by the previous example, the number of graphs generated is bounded by $n * \binom{n}{2}$. This establishes that M is at least as hard as the *k-clique* problem. Membership in NP can be verified easily. The input is a set of sets of edges covering G. We have to verify that all the edges in G are covered and each set is a clique of size n. □.

2.2.1 Complexity of finding a minimal constraint graph

In this section we look at the problem of finding the minimal constraint graph in a given constraint graph G. The problem can be rephrased as: given a constraint graph G, is there a subset of edges of G, whose removal from G will result in G being minimal.

INSTANCE P2: Constraint Graph $G = (V, E)$.

QUESTION: Does there exist $E_1 \subset E$ such that $G_1 = (V, E - E_1)$ is minimal and has a clique of size k?

Corollary 1 *P2 is NP-Hard.*

Proof: Let us suppose that we can solve $P2$ in polynomial time. We will show that using the algorithm to solve $P2$, M can be also be solved in polynomial time, thereby showing that $P2$ is at least as hard as M.

For a graph G we find the set of edges whose removal from G makes G minimal. If the edge set being removed is empty we say G was minimal otherwise G is not minimal. Using $P2$ we can determine whether a graph is minimal, which is an NP-Complete problem. Therefore $P2$ is NP-Hard. □

2.3 Finding a clique in a minimal graph

In this section we would like to show that given a minimal graph and an edge e, it is NP-Complete to find a solution [4] which contains e. Showing that a restricted version of the problem described above is NP-Complete will suffice for our purpose, as it implies that the general version of the problem has to be at least as hard as the restricted version. The restriction which we will use in our proof is the class of constraint satisfaction problems which are satisfiable. This implies that the minimal graphs associated with the *CSPs* have $\omega = \chi$. We will also show that there exists a greedy algorithm to compute a maximum consistent instantiation for a subclass of minimal networks.

In this section we assume that the minimal graphs have $\omega = \chi$.

Let $G = (V, E)$ be the minimal graph associated with the given constraint satisfaction problem $CSP = (X, D, R)$. We can assume that all the domains have the same cardinality m. Let the number of variables in the CSP be n. The minimal graph corresponding to the CSP is defined by $V = \{\cup_{i=1..n, j=1..m} x_{ij}\}$, $\forall i, j, k, l \ \ (x_{ij}, x_{kl}) \in E \ implies \ (d_j, d_l) \in R_{ik}$. This follows from Observation 1.

One interesting observation is that if we delete a row from G, the induced subgraph remains minimal.

Lemma 1 *Let G_i be the graph obtained by deleting the i^{th} row from G. If G was minimal then G_i is also minimal* [5].

Proof: Let e be an edge in G_i. Since e is in a clique of size n in G, e is also in a clique of size $n-1$ in G_i. This is true for every edge in G_i. Therefore every edge in G_i is in a clique of size $n-1$ and $n-1$ is the size of the maximum clique. □

2.3.1 Finding a solution containing a given edge

In this section we will show that given a minimal graph G and an edge belonging to G, the problem of finding a maximal clique of size n which contains the given edge is

[4] Again the word solution is being used in the sense of maximum consistent solution, the CSP does not have to be satisfiable.

[5] We assume that $\omega = \chi$ for G.

NP-Complete.

The problem which we use for the reduction is: find a clique of size m in a graph G.

Theorem 6 *Given a minimal graph $G = (V, E)$ and a specified edge $e \in E$, finding a maximum clique containing e is NP-Complete.*

Proof: Let $G_a = (V_a, E_a)$ be an arbitrary graph. We will show how to construct G. We introduce vertices a, b and edge $e_1 = (a, b)$ and connect it to every vertex in G_a. This is achieved by connecting the vertices a and b of e_1 to every vertex of G_a. Now on each edge $g \neq e_1$ in G constructed so far we put a clique of size $m + 2$, where m was the maximum clique size in G_a. This is achieved by introducing a clique of size m for each g and connecting the vertices of g to all the vertices of the introduced clique.

G constructed in this fashion is minimal with maximum clique size $m + 2$. Now if we can find a maximum clique containing e_1 in G then we can find the maximum clique in G_a, because e_1 belongs to all the maximum cliques. We know that finding a maximum clique even when the size is given is NP-Complete, therefore the problem is NP-Hard. Verification can clearly be done in polynomial time. □

2.3.2 A sub-class of minimal graphs

Definition 28 Maximal clique: *Given G and a clique K in G we call K maximal if K is not contained in any other clique of bigger size.*

We do not put any restrictions on the size of the maximal clique. Which means that size of the maximal clique may be equal to the size of the maximum clique.

Definition 29 A Sub-Maximum Maximal Clique *is a maximal clique with size strictly less than the maximum clique size.*

We define $\mathcal{MD}$ as the class of graphs for which there exists an edge e which does not belong to any sub-maximum maximal clique. For this class of graphs the greedy algorithm will work from the given edge.

Definition 30 nbhd(a): *Nbhd(a) of a vertex a in a graph G is the subgraph induced by the set of vertices which are adjacent to a.*

Observe that by definition a vertex a is not in the $nbhd(a)$.

Lemma 2 *$\mathcal{MD}$ is the class for which there exists an edge such that the common neighbourhood induced subgraph is decomposable.*

Proof: Suppose that $e = (a, b)$ is the edge which does not belong to any sub-maximum maximal clique. Let $G = nbhd(a) \cap nbhd(b)$ be the neighbourhood induced subgraph over a, b. Let us look at the neighbourhood induced subgraph over the vertices a, b. We claim that G is decomposable. If this is not the case then there exists a sub-maximum maximal clique in G. Since e is contained in every partial clique, therefore e would be in a sub-maximum maximal clique also. This leads to a contradiction.

If there is no edge with the desired property, then neighbourhood induced subgraphs over all the edges will not be decomposable. $\square$

2.4 Discussion

We conjecture that finding a maximum clique in a constraint network which can be covered by maximum cliques is polynomial (problem S_a) [6].

Conjecture 1: Let G be a constraint graph such that each edge of G participates in a maximum clique. There exists an edge of G which does not participate in any *sub-maximum maximal* clique.

Let us describe a *greedy* algorithm for finding a maximal clique in such graphs. We start with a node (corresponding to the assignment of a value to the variable), and delete all the nodes not connected to the chosen one. Out of the remaining we pick another node. Repeat the delete operation for the new node and continue. By induction it follows that at all times the new node being added will be connected to all the previous ones. Therefore it will be in a clique. When there are no more nodes

[6]For the recent developments look at Chapter 5.

to be added we cannot extend the clique any more therefore it is maximal. It can be seen that the algorithm presented above is polynomial.

If conjecture 1 is true then the maximum clique can be found by running the greedy algorithm for each edge. Assuming the hypothesis the correctness of the algorithm is easy to verify. Since the algorithm returns a maximal clique, for one edge it will return the maximum clique (because by assumption there is one edge which does not participate in any sub-maximum maximal clique).

It was proposed by VanBeek [VB93] that we should verify conjecture 1 on the n-queens problem, as it is believed that the constraint graph for the n-queens problem are minimal for $n >= 10$. We coded the greedy algorithm to find a solution to the n-queens problem. The greedy algorithm was not able to find a solution for $n = 16$. We then tried to verify the minimality of the constraint graph for $n = 16$.

Proposition 1 *The constraint graph for the 16-queens problem is not minimal.*

Proof: Let us label the squares on the $16X16$ board from 0..255 in the increasing order of rows and columns. Let us place the first queen on the square numbered 2 and the second queen on the square numbered 16. This placement is mutually consistent and if the corresponding constraint graph is minimal then there should be a solution containing it. By exhaustive search we found out that there is no solution when the queens are placed in the above mentioned positions. $\Box$.

Chapter 3

$\omega = \chi$

3.1 Introduction

In this chapter we will study the weighted constraint satisfaction problem and the constraint satisfaction problem restricted to the class of graphs $(\omega = \chi)$.

In chapter 2. we were unable to determine the complexity of finding a clique in a given minimal graph. A variant to the original question can be the weighted minimal graph satisfiability problem defined as follows:

INSTANCE P1: Let graph $G = (V, E)$ be minimal with integral weights on the vertices and maximum clique size m.

QUESTION: Does there exist a clique with cost m.

We will show that the weighted minimal graph satisfiability problem is NP-complete. A result of Feige and Lovasz [FL92] implies that there exists a class of graphs $(\omega = \chi)$ for which the satisfiability problem can be solved in polynomial time. This raises the question of recognizing graphs for which the former property is true. We show that it is NP-complete to determine whether $\omega = \chi$ is true for any graph.

3.2 Class of graphs for which $\omega = \chi$

Let $\mathcal{C}$ be the class of networks for which size of the maximum clique is equal to the chromatic number [1]. In [FL92] Feige and Lovasz showed the maximum clique problem for graphs in $\mathcal{C}$ is solvable in polynomial time.

Let G_1 and G_2 be two graphs. We write $G_1 \rightarrow G_2$ if there is a homomorphism of G_1 into G_2, where a homomorphism is defined to be a mapping of vertices of G_1 into vertices of G_2 such that adjacent nodes are mapped to adjacent nodes. If we choose G_1 to be a clique over k vertices and G_2 be any graph then $G_1 \rightarrow G_2$ if and only if G_2 has a k-clique. Another example can be: G_1 is k-colorable, choose G_2 as k-clique. Figure 3.1 shows two pairs of graphs. In the first pair there exists a homomorphism from G_1 to G_2. In the second pair a homomorphism from $G3$ to G_4 does not exist. The sets in G_2 correspond to the mappings of the vertices of G_1.

Now we play a two prover and one verifier game. The two provers want to convince a verifier that G_1 and G_2 are homomorphic. The protocol is simple: the two provers agree on a mapping and the verifier asks each of them the image of a node of G_1. The nodes are picked by the verifier at random. The criteria for acceptance are: if the verifier asks each prover for the image of the same node of G_1 then he should get the same node of G_2; if he asks for adjacent nodes of G_1 then he should get adjacent nodes in G_2. Observe that if the provers have a deterministic strategy then the verifier rejects one of the pairs by a probability of $1/n^2$ if the two graphs are not homomorphic. Feige and Lovasz proved that even if the provers use a randomized strategy the probability that the verifier would accept the answers will be 1 iff the two graphs are homomorphic for certain classes of graphs. If the graphs are not homomorphic and there exists a randomized strategy which will convince the verifier that the graphs are homorphic then the strategy is called a *hoax*.

We will describe the procedure from [FL92] to determine the whether a graph belongs to the class $\mathcal{C}$. Since the chromatic number is equal to the size of the maximum clique, we have:

$$K_n \rightarrow G \rightarrow K_n$$

[1] *Perfect graphs* belong to this class

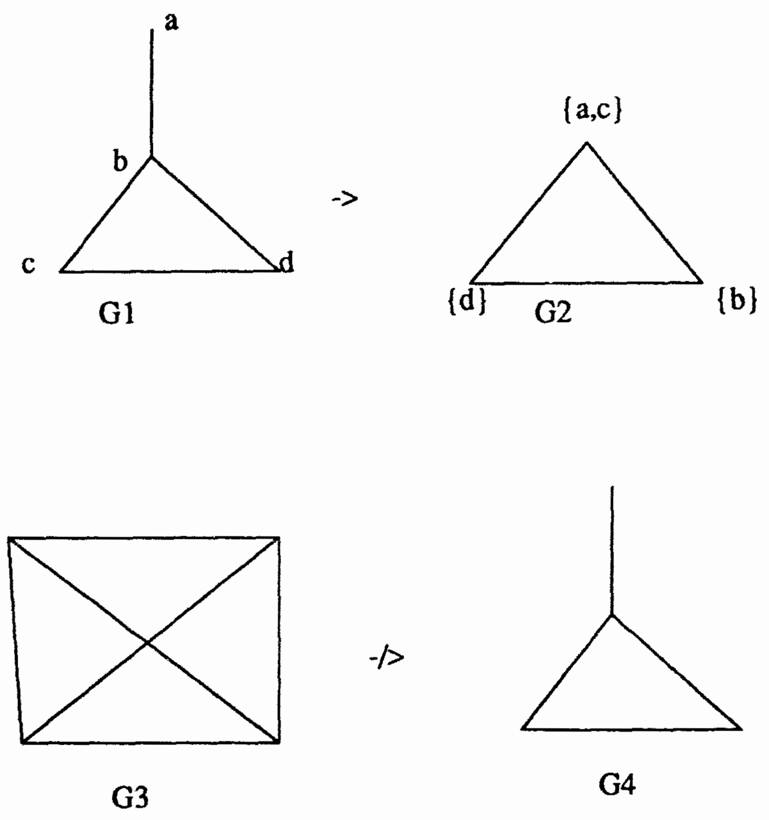


Figure 3.1: Examples of Homomorphic and Non-Homomorphic pairs

	1	*2*	*3*	*4*
1	0	1	0	1
2	1	0	1	0
3	0	1	0	1
4	1	0	1	0

Table 3.1: Incidence Matrix A of G in Figure 3.2

	1	*2*
1	0	1
2	1	0

Table 3.2: Incidence Matrix B of K_2 in Figure 3.2

where K_n is the complete graph on n vertices. Let A be the incidence matrix of G and B be the incidence matrix of K_n. We generate a matrix C from A and B as follows:

- Replace all the diagonal entries in A by an identity matrix of size $n * n$.
- A 1 in A is replaced by B and a 0 is replaced by a matrix with all ones except on the diagonal.

It is to be noted that the incidence matrix of a clique has all ones except on the diagonal, so in this restricted case a 1 and 0 are replaced by the same matrix.

We will illustrate the construction of matrix C from A and B by means of an example shown in Figure 3.2.

Table 3.1 shows the incidence matrix A of graph G shown in Figure 3.2.

Table 3.2 shows the incidence matrix of K_n in Figure 3.2. Table 3.3 shows the matrix C generated from A and B.

Let P be another matrix with dimensions equal to C. Blocks of entries of P can be indexed by two indices i, j denoting the particular block in C obtained by substitution of 1 or 0 in matrix A by a $n*n$ matrix. The entries $p_{su,tw}$ in the submatrices correspond to the probabilities that the provers when given vertices s, t will return vertices u, w.

Let L be the following program:

$$\textit{maximize} \quad C.P$$

	11	*12*	*21*	*22*	*31*	*32*	*41*	*42*
11	1	0	0	1	0	1	0	1
12	0	1	1	0	1	0	1	0
21	0	1	1	0	0	1	0	1
22	1	0	0	1	1	0	1	0
31	0	1	0	1	1	0	0	1
32	1	0	1	0	0	1	1	0
41	0	1	0	1	0	1	1	0
42	1	0	1	0	1	0	0	1

Table 3.3: Matrix C for the Figure 3.3

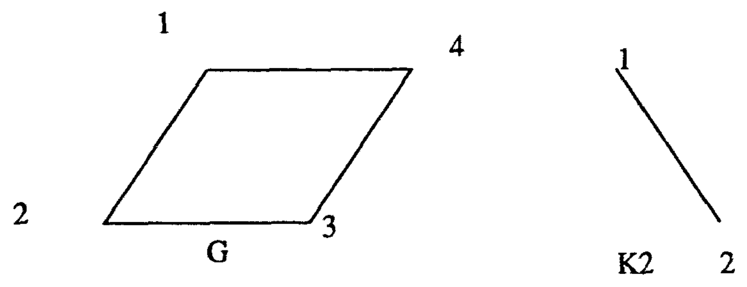


Figure 3.2: Example for generation of matrix C

$$\sum_{i,j} P_{si,tj} = 1 \quad \forall \quad s,t$$

P is symmetric positive semi definite

all the entries of P are positive

Feige and Lovasz [FL92] show that program L can be solved in polynomial time.

A *hoax* is a cheating randomized strategy of provers which cannot be detected by tests available to the verifier. In the game defined in the paper [FL92] the tests are: the entries of each submatrix should sum up to 1 and the whole matrix is positive semi-definite.

Lemma 3 (Feige and Lovasz) *There does not exist a hoax for $K_k \rightarrow K_{k-1}$.*

Let $\mathcal{C}$ be the class of graphs for which $\omega = \chi$. The previous lemma implies that for this class of graphs the value of the objective function *maximize C.P* will be 1 iff the given graph is $k - colorable$.

If we assume that the minimal network is also satisfiable then we get $\omega = \chi$. As a corollary to the previous lemma we get:

Corollary 2 *Determining whether a given minimal constraint network is satisfiable can be accomplished in polynomial time.*

The corollary is not interesting because we know that the minimal graph is satisfiable.

We can relax our definition of minimal graphs, assume that every edge participates in a clique of some fixed size but the clique size is not maximum. Using this definition we will get a class of graphs which are minimal but for which there is no solution to the corresponding constraint satisfaction problem. The previous corollary is applicable to this class also under the assumption that $\omega = \chi$.

Suppose that we are doing backtracking. It is possible to use the test given by Feige and Lovasz to determine whether we should expand a given assignment. If the value of the game is less than 1, then there does not exist a maximum clique. In this case we can fathom the node. As the oracle does not lie when it gives a no answer, the algorithm would be complete.

3.2.1 Complexity of recognizing $\omega = \chi$

In this section we will show that deciding whether a graph G belongs to the class $\mathcal{C}$ is NP-Complete. Recall that $\mathcal{C}$ was defined as the class of graphs for which $\omega = \chi$.

The reduction is from the constraint satisfaction problem. The input will be a constraint satisfaction graph G_{csp} and we will generate a new graph G from G_{csp} such that for G, $\omega = \chi$ iff the CSP is satisfiable. We know that if the CSP is not satisfiable then G_{csp} does not contain a clique of size k. We need a construction which will force G to be $k - colorable$ but the maximum clique size should not exceed the maximum clique size in G_{csp}.

Let K_{k-3} be a clique over $k-3$ vertices. Enclose K_{k-3} in a 5-cycle. Connect every vertex of K_{k-3} to every vertex of the 5-cycle. Let the resulting graph be denoted by G_o. G_o has a maximum clique of size $k-1$ and the minimum number of colors required to color it is k. An example of G_o for $k = 6$ is shown in Figure 3.3.

Theorem 7 *Given G, determining if $G \in \mathcal{C}$, is NP-Complete.*

Proof: We will reduce the constraint satisfaction problem CSP. The CSP is satisfiable iff the corresponding constraint graph G_{csp} has a clique of size k. Let G be defined as the union of G_{csp} and G_o, where G_o is the graph defined previously with clique size $k-1$ and $\chi = k$.

If CSP is satisfiable then G_{csp} has a clique of size k. In this case for G, $\omega = \chi$ ($\chi = k$, by construction). If the CSP is not satisfiable then $\omega \neq \chi$ for G. This follows from the fact that G_{csp} does not contain a clique of size k whereas the chromatic number for G is k. To show the membership in NP we will guess the clique and the coloring. Verification can be done in polynomial time. □

3.3 Weighted minimal networks

Given a constraint network G and weights associated with each node, the question is to find an instantiation with minimum cost. We will show that even if the weights are restricted to integers the problem is NP-complete

INSTANCE P1: Let graph $G = (V, E)$ be minimal with integral weights on the vertices and maximum clique size m.

QUESTION: Does there exist a maximum clique with cost k.

We will reduce $k-$*clique* to $P1$.

Theorem 8 *$P1$ is NP-Complete.*

Proof: We will reduce $k-$*clique* to $P1$. Given an instance $G_2 = (V_2, E_2)$ of $k-$*clique*, we generate an instance $G_1 = (V_1, E_1)$ of $P1$ as follows: For each vertex in

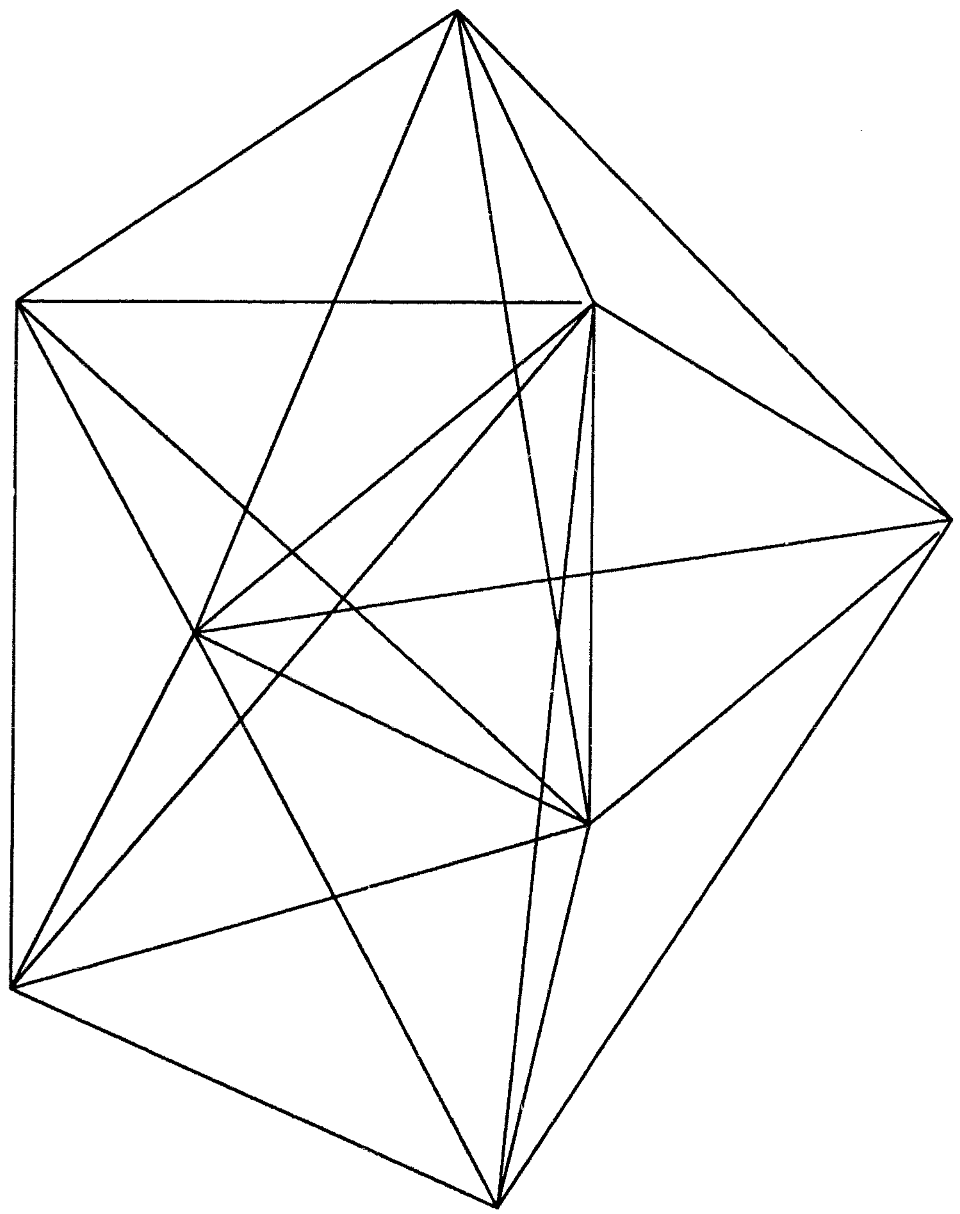


Figure 3.3: The Graph G_o

G_2 we create a complete graph over m vertices in G_1 denoted by $K_m(i)$. The vertices $1..m$ in $K_m(i)$, denoted $K_{m,1}(i), \ldots, K_{m,m}(i)$, are assigned weights $1..m$. For each edge $(u, v) \in E_2$, connect vertex $K_{m,1}(u)$ with vertices $K_{m,j}(v)$ for $j = 1..m-1$. The graph G_1 so generated is minimal. Every edge in G_1 participates in a clique of size m. Now we will show that if G_2 has a clique of size m then there exists a clique in G_1 with weight equal to m, otherwise, the weight of the maximum clique is greater than m.

It can be readily observed that if G_2 has a clique of size m, then $K_{i,1}(i) \in G_1$ *for* $i = 1..m$ form a clique. This clique has cost m and this is the minimal cost. If G_2 does not have a clique of size m, then there does not exist a clique over $K_{i,1}(i)$ for all i. Therefore the miuimum cost is $m + 1$. □

It can be seen that even for 0/1 weight assignments the problem is NP-Complete.

3.4 Conclusions

The weighted minimal network satisfiability problem is NP-complete. We showed that a result of Feige and Lovasz [FL92] implies that there exists a class of graphs ($\omega = \chi$) for which the satisfiability problem can be solved in polynomial time. This raised the question of recognizing graphs for which this property is true. We showed that it is NP-complete to determine whether $\omega = \chi$ is true for any graph.

Chapter 4

Conclusions

4.1 Conclusions

In this thesis we studied complexity issues related to minimal graphs. We showed that determining whether a given constraint graph is minimal is NP-Complete. We also showed that given a 2-tuple finding an instantiation which contains this edge is NP-Complete in a minimal graph.

We characterized a sub-class $\mathcal{MD}$ of minimal graph for which there exists a greedy algorithm for finding an instantiation. The recognition problem for this class is of the same complexity as the recognition problem for decomposable graph. The problem of deciding whether a given graph is decomposable is in $Co-NP$.

We showed that from a result of Feige and Lovasz it follows that that there exists another class ($\omega = \chi$) of constraint satisfaction problems for which determining the satisfiability is polynomial. The recognition problem for this class is also NP-Complete.

Next we addressed the weighted constraint satisfaction problem. In the weighted constraint satisfaction problem, associated with each assignment was a cost. The goal was to find a consistent assignment with minimum cost. We showed that for minimal graphs and 0/1 weights the weighted constraint satisfaction problem is NP-Complete.

4.2 Open Problems

There are two major question which remain unanswered in this thesis.

- Given a graph, what is the complexity of deciding whether it is decomposable.
- Given a minimal graph, can an instantiation be found in polynomial time.

Chapter 5

Addendum

5.1 Conjecture 1.

In chapter 2. we had posed a conjecture that in a minimal graph there always exists an edge which does not belong to any sub-maximum maximal clique. At the IRIS Precarn conference in Toronto (1994), where this work was presented, Jack Snoeyink gave the following counter example to the conjecture, thereby refuting it.

We will give the coverings of the graph by maximal cliques of size 3 and maximum cliques of size 4. Let the vertices of the graph be labelled by numbers 1..10. The tables below give the coverings of the graph.

Covering by maximal triangles

$\{2,3,9\}$
$\{1,3,5\}$
$\{4,3,7\}$
$\{5,6,7\}$
$\{7,8,9\}$
$\{4,2,8\}$
$\{1,6,4\}$
$\{5,10,9\}$
$\{6,8,10\}$

{1,2,10}

Covering by maximum cliques of size 4

{1,2,3,4}
{4,6,7,8}
{1,6,5,10}
{2,8,9,10}
{3,5,7,9}

It can be verified that there is no clique of size 5 in the example. As the graph is minimal with maximum clique size 4 and every edge belongs to a maximum and a sub-maximum maximal clique, conjecture 1. is not true.

The example shown above can be defined as the union of five cliques of size 4, each of chromatic number 4. Whereas the whole graph is not 4 colorable. This construction has been generalized in [BG94] answering a question of Erdos [Erd81].

Bibliography

[ABMP91] H. Alt, N. Blum, K. Melhorn, and M. Paul. Computing a maximum cardinality matching in bipartite graph in time $o(n^{1.5}\sqrt(m/\log n))$. *Information Processing Letters*, 37:237–240, 1991.

[AS87] E. Arkin and E. Silverberg. Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18:1–8, 1987.

[Bal85] R. Balzer. A 15 year perspective on automatic programming. *IEEE Transaction on Software Engineering*, 11:1257–1268, 1985.

[BG94] Roman Bacik and Daya Ram Gaur. When n-colors are not sufficient. *In Preparation*, 1994.

[BGW82] R. Balzer, N. Goldman, and D. Wile. Operational specification as the basis for rapid prototyping. *In Proc. Second Software Engineering Symposium: Workshop on rapid prototyping*, 1982.

[Bob84] D. G. Bobrow. Qualitative reasoning about physical systems: an introduction. *Artificial Intelligence*, 24:1–5, 1984.

[CF94] B. Y. Choeiry and B. Faltings. Interactive resource allocation by problem decomposition and temporal abstractions. *Technical Report No. TR-94/43, EPFL*, 1994.

[DAGJ95] T. Dakic, J. Adhikary, D. R. Gaur, and K. W. Jackson. Backtrack free search for resource allocation problems. *CONSTRAINT-95 workshop on* *constraints in conjunction with FLAIRS 95*, Intelligent Systems Lab, SFU, 1995.

[Dec90] R. Dechter. From local to global consistency. *In Proc. of the Eight Canadian Conference on Artificial Intelligence*, pages 231–237, 1990.

[DMP91] R. Dechter, I Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, pages 61–95, 1991.

[DP88] R. Dechter and J. Pearl. Network based heuristics for solving constraint satisfaction problems. *Artificial Intelligence*, 34:1–38, 1988.

[DVH91] Y. Deville and P. Van Hentenryck. An efficient arc consistency algorithm for a class of constraint satisfaction problems. *In Proc. of the Twelfth International Joint Conference on Artificial Intelligence*, pages 325–330, 1991.

[Erd81] P. Erdos. On the combinatorial problems which i would most like to see solved. *Combinatorica*, 1:25–42, 1981.

[FL92] U. Feige and L. Lovasz. Two-prover one-round proof systems: their power and their problems. *IEEE Annual Symposium on Theory of Computing*, pages 733–745, 1992.

[Fre82] E. G. Freuder. A sufficient condition for backtrack-free search. *J. ACM*, 29:24–32, 1982.

[Fre85] E. G. Freuder. A sufficient condition for backtrack-bounded search. *J. ACM*, 32:755–761, 1985.

[Fre90] E. G. Freuder. Complexity of k-tree structured constraint satisfaction problems. *In Proc. of the Eight National Conference on Artificial Intelligence*, pages 1–4, 1990.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability.* W. H. Freeman and Company, 1979.

[GLS88] M. Grotschel, L. Lovasz, and A. Shrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, 1988.

[Gol80] M. C. Golumbic. *Algorithmic Graph theory and Perfect Graphs.* Academic Press Inc., 1980.

[Haj65] A. Hajnal. A theorem on $k-saturated$ graphs. *Canadian Journal of Math*, 17:720–724, 1965.

[J.84] De Kleer J. How circuits work. *Artificial Intelligence*, 24:205–280, 1984.

[JS84] De Kleer J. and Brown J. S. A qualitative reasoning based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[Luc85] Lucas. On the versatility of knowledge representation. In *Proc. IFIP Working Conference, The role of abstract models in information processing.* North Holland, 1985.

[Mac77] A. Mackworth. Consistency in the network of relations. *Artificical Intelligence*, 8:99–118, 1977.

[MF85] A. Mackworth and E. C. Freuder. The complexity of some polynomial network consistency algorthms. *Artificial Intelligence*, 25(1):65–74, 1985.

[Mon74] U. Montanari. Networks of constraints: fundamental properties and application to picture processing. *Information Science*, 7:95–132, 1974.

[Reg94] Jean-Charles Regin. A filtering algorithm for constraints of difference in csp's. *Proceedings of the twelfth National Conference on Artificial Intelligence*, 1:362–367, 1994.

[Sei81] R. Seidel. A new method for solving constraint satisfaction problems. *In Proc. IJCAI*, pages 338–342, 1981.

[VB92] P. Van Beek. On the minimality and decomposability of constraint networks. *In Proc. of Tenth National Conference on Artificial Intelligence*, pages 447–452, 1992.

[VB93] P. Van Beek. Personal communication. 1993.

[VK86] M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. *In proc. AAAI-86*, pages 377–382, 1986.

[Wal72] D. L. Waltz. Generating semantic descriptions from drawings of scenes with shadows. *MIT Techincal Report AI271*, Nov. 1972.

[Wal75a] D. L. Waltz. Automata theoritic approach to visual processing. In *Applied computation theory.* Engelwood Cliffs, New York, Prentice Hall, 1975.

[Wal75b] D. L. Waltz. Understanding line drawings of scenes with shadows. In *The pyscology of computer vision*, pages 19–91. Mc Graw-Hill, New York, 1975.

[Zyk49] A. A. Zykov. On some properties of linear complexes. *Mat. Sb., N. S. (in Russian)*, 24:163–188, 1949.