Implementation of Strategies for Solving Constraint Satisfaction Problems

Zewar Fadhlulddin Hasan

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
July 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Nazim Mahmudov
Chair, Department of Mathematics

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Rashad Aliyev
Supervisor

Examining Committee

1. Prof. Dr. Rashad Aliyev
2. Assoc. Prof. Dr. Benedek Nagy
3. Asst. Prof. Dr. Ersin Kuset Bodur

ABSTRACT

This thesis investigates the different strategies for solving constraint satisfaction problems. The basic properties of constraint satisfaction problems are discussed. The different types of constraints are given. The constraint graph and cryptarithmetic constraint satisfaction problems are discussed. Such search techniques as backtracking search, local search, and constraint propagation for solving constraint satisfaction problems are presented. The forward checking in constraint satisfaction problems is used. Some constraint satisfaction problems such as map-coloring problem, cryptarithmetic problem, n-queens problems and Sudoku problem are solved.

Keywords: Constraint satisfaction problem, Constraint graph, Backtracking search, n-queens problem, Local search, Constraint propagation, Forward checking

ÖZ

Bu tez kısıtlama memnuniyeti sorunlarını çözmek için farklı stratejiler araştırıyor. Ayrıca, kısıtlama memnuniyeti problemlerinin temel özellikleri tartışılır. Bunun yanında, kısıtlamaların farklı türleri verilmiştir. Kısıtlama grafiği ve cryptarithmetic kısıtlama memnuniyeti sorunları tartışılır. Geriye arama, yerel arama ve kısıtlama memnuniyeti problemlerinin çözümü için kısıtlama yayılma gibi arama teknikleri sunulmuştur. Kısıtlama memnuniyeti problemlerinde ileri kontrol yöntemi kullanılır. Harita renklendirme problemi, cryptarithmetic problemi, n-vezir problemi gibi bazı kısıtlama memnuniyeti problemleri ve Sudoku problemi çözülür.

Anahtar Kelimeler: Kısıtlama memnuniyeti sorunu, Kısıtlama grafiki, Geri İzleme arama, n-vezir problemi, Yerel arama, Kısıtlama yayılımı, İleri kontrol

ACKNOWLEDGMENT

First of all, I thank God for every minute of my life in this world. I would like to express my appreciation and gratitude to my supervisor, Prof. Dr. Rashad Aliyev, for his support during my work on this thesis. I thank him for every good advice, explanation and instruction at any stage of this thesis.

I thank God for a wonderful family I have. I want to express my appreciation and gratitude to my gorgeous parents, my father Fadhlulddin and my mother Fewziya Abdulrazaq for supporting me in each step I take. The special thanks to my brother Karwan for being by my side and helping me. I thank all my sisters, brothers, nephew, relatives and friends. Finally, I am grateful to every person supporting or helping me during my study.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Artificial Intelligence (AI) is a special area of computer science that aims to create intelligent machines. AI has become an basic part of the technology industry. Alan Turing is a founder of AI, and in 1950 he made a simple machine to solve the mathematical problems (called the Turing Machine). Based on this idea, Turing wondered that if a computer's reply was indistinguishable from a human, then the computer could be considered as a thinking machine [1-2]. AI is used in logistics, data mining, medical diagnosis, industry and many other areas [2].

The concept of constraint satisfaction problem (CSP) was proposed in 1970s. CSP is defined by a relation on a subset of the set of variables. CSP is ubiquitous in our everyday lives, in academic and industrial worlds. CSP is a mathematical problem defined as a set of objects whose state must satisfy a number of constraints. CSP has been used in many fields to represent various problems like N-queen game, the graph-coloring problem, design issues such as VLSI, the industrial application such as scene analysis and interpretation, planning, scheduling, and allocation of resources.

A set of variables and a set of constraints are components of CSP. A finite and discrete domain of values is associated with each variable [3-4].

A CSP can be considered as a search problem with the following incremental formulation:

- Initial state: no value is assigned to the variables, i.e. empty assignment {};

- Successor function: the value is assigned to one of the unassigned variables with the condition that there is no violation of constraints. In other words, no conflict should take place between unassigned variable and the variables which have been assigned previously;

- Goal: the process of assigning the values to the variables is complete, i.e. all the variables have values;

- Past cost: every step should be with a constant cost which is irrelevant.

In order to solve a CSP, the following two steps should be performed:

- constraint propagation: the arc consistency is propagated on the graph. It is intended to constrain values and determine the possible inconsistencies in order to make sure that each arc is consistent. The values that can never be the parts of any solution should be eliminated;

- searching is carried out to explore the valid assignments of values to variables.

There are different types of constraints in CSPs:

- Unary constraints consist of a single variable, and an arc representing this constraint is originated and terminated at the same node;

- Binary constraints consist of pair of variables. These constraints can be represented as a graph in which each node depicts a variable, and each arc depicts a constraint between two neighbor variables (pair of variables);

- Higher-order constraints consist of three or more variables to be represented by the constraint hypergraph;
- Soft constraints can represent the preference of one variable over another one by indicating the costs assigned to each of these variables. Soft constraints are useful in constraint optimization problems.

Maximum constraint satisfaction problem (Max-CSP) and weighted constraint satisfaction problem (WCSP) are two main types of CSP.

While solving Max-CSP it is necessary to take into consideration that some constraints can be violated during the process that solution of the problem is found. In Max-CSP all the constraints are to be equally important, and the purpose is to find the assignment which is able to maximize the satisfied constraints. Several real world problems can be formulated as Max-CSP, for example, planning, scheduling, warehouse location problem and max-cut problems. In order to solve the Max-CSP, a value for each variable from each domain must be assigned in such a way that the maximum of constraints is satisfied. However, a high complexity in a Max-CSP is requiring a combination of heuristics and combinatorial methods of search to be solved in an appropriate time. Formally speaking, a Max-CSP is difficult and interesting problem for mathematicians, operational researchers and computational scientists [5].

There are various search algorithms in AI, and the backtracking algorithm is the basic uninformed search algorithm for solving CSP. This algorithm uses constraint propagation technique. In this algorithm, the variables and their values are chosen by applying the heuristic approach. The variables are represented in some order. The unassigned variable is chosen and the value to this variable is given which should be consistent with all the variables. The values should be assigned to the variables one-by-one. This process continues until no more assignment is made, i.e. every variable has a value, then it is decided that the solution is found or the backtracking to the initial variable is performed.

The backtracking algorithm is effective in many coloring (for example, map coloring) problems.

One of the disadvantages of the backtracking algorithm is that sometimes exploring the entire search space can't be successful for the real-world CSPs.

A distributed CSP can be considered as a general framework for dealing with problems in multi-agent systems. In a distributed CSP the variables and constraints are distributed among multiple agents, and the agents must communicate with each other to satisfy all constraints. The agents must assign values to their variables so that all the constraints are satisfied [6-7]. The multi-agent systems can be used for distributed interpretation problems, distributed resource allocation problems, distributed scheduling problems etc.

Chapter 2

REVIEW OF EXISTING LITERATURE ON CONSTRAINT SATISFACTION PROBLEMS

In [8] the expansion of the constraint satisfaction problem (CSP) universally quantified variable called a Quantified Constraint Satisfaction Problem (QCSP) is discussed. The algorithm of QCSP with discrete non-Boolean domains is given. The problem is solved by expanding the search algorithm from standard CSPs to QCSP. The generalization of CSPs to QCSP increases the expressiveness of the framework, but at the same time the complexity of the reasoning task raises from NP-complete to PSPACE-complete. PSPACE-complete problems are modeled for the implementation in game playing and planning. This algorithm shows how the value can be interchangeably exploited in QCSPs.

[9] is about probabilistic algorithm for solving k-SAT and CSP. The algorithm randomly creates an initial task and then leads by those conditions that are not satisfied to select random literal from such a condition and try to find a satisfying task by flipping the conformable bit. If a satisfying task is not available after using O(n) steps, so it will start again. This is the best known algorithm of 3-SAT known until now.

The combination of global and local searches in which the abstraction of a constraint satisfaction problem is created by local search, and afterwards the global search tries to exploit it [10]. It is proved that the cluster-based abstraction provides better results in searching problems.

In [11] a machine called Turing machine with atoms (TMA) is determined, and this determination is represented as CSP. The determination problem is solved by TMA to characterize the classes of structures to make available from words with atoms.

The Distributed Partial Constraint Satisfaction Problem (DPCSP) as a new framework for transaction with over-constrained cases in distributed CSPs is presented in [12]. It is foreseen DPCSP to have the great possibility in different applications. Two new algorithms called the Iterative Distributed Breakout (IDB) and the Synchronous Branch and Bound (SBB) are discussed. Both algorithms are intended for solving Distributed Maximal Constraint Satisfaction Problems (DMCSPs) which belong to an important category of DPCSP. Both algorithms are tested. The result shows that SBB is better when the optimal solution is looked for, and IDB is more suitable for the cases when one wants to get an optimal solution satisfying all constraints.

In [13] proposed filtering techniques are necessary to solve the constraint satisfaction problems. These techniques can sharply minimize the search particularly on hard and large problems. One of the most useful filtering techniques is arc consistency, because this technique easily removes the values which are suitable for neither solution. The filtering techniques which are more meaningful than arc consistency in cases when the set of constraints do not change, are studied and compared.

A technique which is based on simple structure with a minimal space requirement is offered in [14], and it is capable of bounding the worst-case performance better than pseudo-tree search.

Many techniques exist for solving the constraint satisfaction problem. The modified branch and bound algorithm is proposed in [15] to show a solution of a constraint satisfaction problem in a problem of a map coloring. The adjacent areas in the map must not be of the same color. The branch and bound algorithm uses back jumping when it faces a dead-end in the search, and the variable ordering is used to help the searching process. In comparison with backjumping, the branch and bound algorithm shows better results using the technique of variable ordering.

The tractable classes of CSPs are investigated in [16]. In order to solve constraint satisfaction problems, the complexity of generic algorithms are considered. Measuring of complexity of generic algorithms is done by a new parameter using of which derives a new complexity. The classical types of algorithms in polynomial time are used to solve tractable classes of constraint satisfaction problems.

The extraction of domain values with interchangeable nature is an actual problem, and is used for constraint satisfaction problems. In [17] the basis for the extraction process is developed. The known constraint satisfaction algorithms can be adapted to a proposed basis. The domain values can be exploited by a proposed backtrack procedure. The experimental results show that a proposed approach is accurate for a certain types of problems.

In [18] it is determined that the constraint satisfaction problem is defined in first order. The characterizations of structures of first-order definability of constraint satisfaction problem are discussed. The first-order definability of constraint satisfaction problem leads to nondeterministic polynomial time (NP)-complete problem. The polynomial-time algorithm is given which is used to determine the core structures.

Some solutions of constraint satisfaction problem require to make preferences among them, and this type of CSP is called weighted CSP (WCSP) which is discussed in [19]. One of the complete techniques used to solve such problems is a bucket technique. The bound for the optimal solution is calculated by implementation a heuristic method which is applied in case when the memory for the application of bucket elimination is very high. A memetic algorithm for weighted CSP is presented which provides better solutions in large instances than classical algorithms. The hybrid form of bucket elimination (BE) and mini-bucket (MB) with memetic algorithm is presented.

In case constraint satisfaction problem has any solution, this is NP-complete task. There are many filtering techniques, and one of the useful of them is arc-consistency. In [20] the dynamic CSP is defined, and in order to achieve arc-consistency in dynamic CSP, the algorithm is proposed. The advantages of the proposed algorithm are given.

In [21] the proposed method reuses the previous solution of constraint satisfaction problem by local changing to produce a new solution for CSP. The idea and algorithm are given. The experimental results are compared to classical backtrack and dynamic backtracking methods, and show the effectiveness of this method.

In [22] the constraint satisfaction problem is solved by DNA Computing. A new algorithm is suggested for above problem that uses JOIN operation which is applied using biochemical DNA manipulations. The EXTRACT manipulation can produce two types of errors called false negative and false positive errors. The affect of such errors to the propose algorithm is analyzed. The error probabilities can be decreased using technique whish was proposed by Karp.

The combination of base algorithms which are described in terms of forward and backward moves is given in [23]. The forward move of one algorithm can be successfully combined with backward move of another algorithm and versus visa. The combination of tree search algorithms can be useful for some other algorithms.

Chapter 3

BASIC PROPERTIES OF CONSTRAINT SATISFACTION PROBLEMS

3.1 Basic definitions of CSP

CSP is defined as a set of variables $A_1, A_2, A_3, \dots, A_m$, and a set of constraints $B_1, B_2, B_3, \dots, B_n$. There is a nonempty domain D_i of possible values for each variable A_i. Every constraint B_j limits the values that variables can take, for example, $A_1 \neq A_2$. A state of the problem is defined as an assignment of values to some or all variables. An assignment does not break the constraints. A complete assignment for every variable is defined, and a solution to CSPs is a complete assignment that pleases all the constraints.

Below the main components involved in a CSP are introduced:

1) Variables: $A = A_1, A_2, A_3, \dots, A_m$.
2) Domains: Y, integers.
3) Constraints: $B_1, B_2, B_3, \dots, B_n \subseteq \mathrm{Y}^n$. .
4) Problem: Find $r = A_1, A_2, A_3, \dots, A_m \in$ Y^n such that $r \in \mathrm{B}_i$, for all $1 \leq \mathrm{i} \leq \mathrm{n}$.

There should be the connections between variables, for example: $d \neq e$, $d > e$, $d + e < f$. The following properties are defined:

Constraint C_{def} ... between variables $, e, f, ...$;

$C_{def} \subseteq D_d \times D_e \times D_f \times ...$ (a subset of all tuples)

Assume there are three variables $\{a, b, c\}$, and for the variable a we have the set of values $\{1,2,3,4\}$, and for variable b we have the set of values $\{2,3,4\}$ and for variable c we have a value $\{7\}$:

The constraint $a \neq b$ is defined as $\{(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\}$.

The constraint $a = b$ is defined as $\{(2,2), (3,3), (4,4)\}$.

The constraint $a > b$ is defined as $\{(3,2), (4,2), (4,3)\}$.

The constraint $a + b < c$ is defined as $\{(1,2), (1,3), (1,4), (2,2), (2,3), (2,4), (3,3)\}$.

CSP can be classified with:

1) Discrete variables:

- Finite domains: size $r \rightarrow O(r^m)$ complete assignments.

- Infinite domains: strings, integers etc.

The variable like starting day of a job is an example of discrete variable.

2) Continuous variables:

The class schedule, the airline schedule are the examples of continuous variables.

3.2 Types of constraint satisfaction problem

The following types of CSP are known:
1) A unary CSP: consists of a single variable. The following samples are related with the unary CSP:

$$P \neq \mathrm{red};$$

$$D(X): X = 2;$$

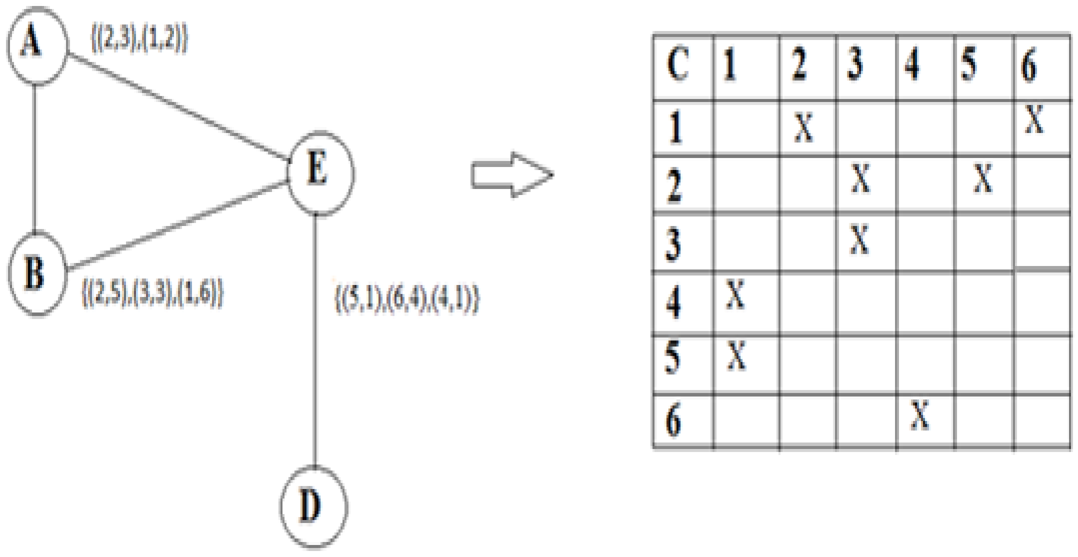
$$D(Y): Y > 5.$$

2) A binary CSP: all constraints are binary. Every unary (non-binary) CSP can be converted into a binary CSP by entering an additional variable. A binary CSP can be represented as a constraint graph that has a node for each variable and an arc between two nodes. The binary constraints are the edges between nodes.

3.3 Constraint Graph

It is beneficial to imagine a constraint satisfaction problem (CSP) as a constraint graph. The constraint graph is the graph represents the constraint, which communication between variables in the problem. This graph relies heavily on the impersonation that picks for a specific problem. Constraint graph has a node for each variable and an arc between two nodes. The constraints are the edges between every node. Constraint graphs are more beneficial when whole constraint propagation is performed by arc consistency.

In Figure 1 the binary constraint graph and the matrix for binary constraints are represented.



C	1	2	3	4	5	6
1		X				X
2			X		X	
3			X			
4	X					
5	X					
6				X		

Figure 1: Binary constraint graph and matrix for binary constraint

Variables: $\{\mathrm{A}, \mathrm{B}, \mathrm{D}, \mathrm{E}\}$.

Constraints: B $(\mathrm{A}, \mathrm{E}) = \{(2, 3), (1, 2)\}$.

$\mathrm{B}(\mathrm{B}, \mathrm{E}) = \{(2, 5), (3, 3), (1, 6)\}$.

$\mathrm{B}(\mathrm{E}, \mathrm{D}) = \{(5,1), (6, 4), (4, 1)\}$.

Map coloring is another famous problem of CSP. The problem can be defined in the following form:

Variables (countries): {Germany (GE), Poland (PL), Czech Republic (CZ), Austria (AT), Switzerland (SUI), Hungary (HU), Slovakia (SK), Italy (IT), Slovenia (SL)}.

Domain (colors): {red, green, blue, yellow}.

Constraints: the adjacent areas must have different colors (e.g.: color of (GE) ≠ color of PL)).

A CSP is solved if the complete assignment satisfying all the constraints is determined.

Figure 2 shows the initial (uncolored) and final (colored) map coloring CSP problem.

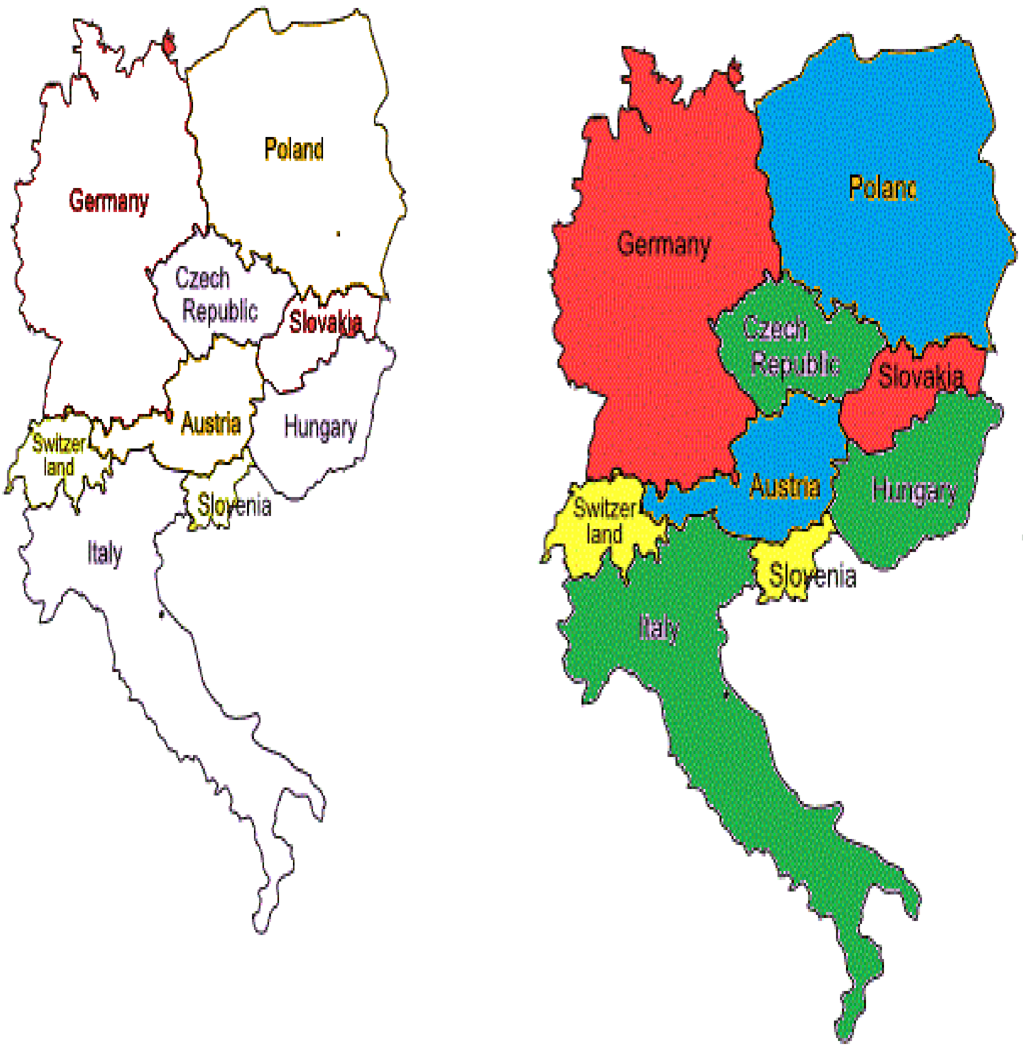


Figure 2: Initial (uncolored) and final (colored) map coloring CSP problem

The map coloring problem can also be represented as a constraint graph. The nodes of the graph represent the variables, and the arcs represent the binary constraints or constraints.

Formulation for constraint graph:

- Node = variable;

- Arc = constraint;

- Initial state: no variable has a color;

- Successor state: select a value to one of the variables without a color or value;

- Goal: all variables have a color or value.

The Figure 3 shows the constraint graph of the uncolored map.

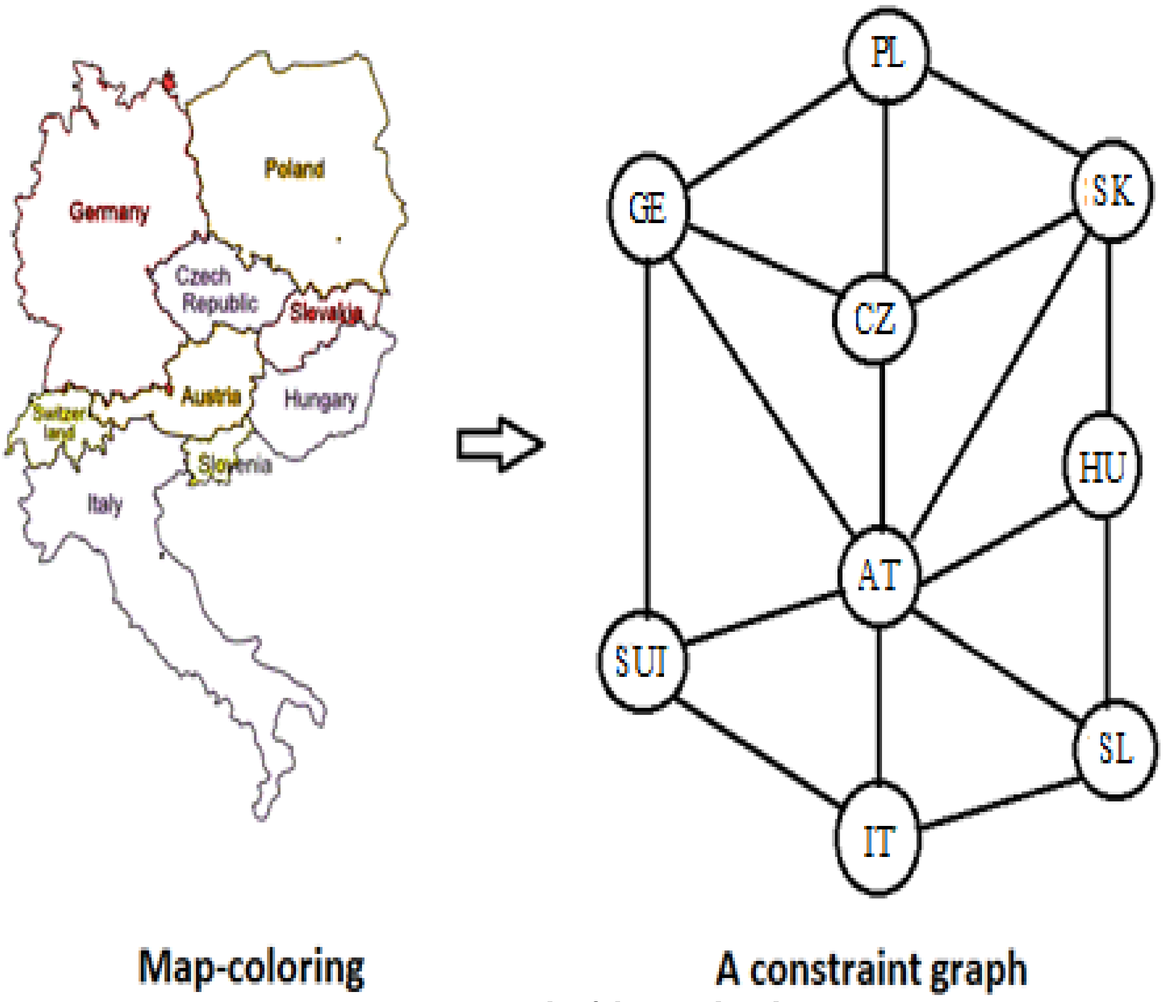


Figure 3: Constraint graph of the uncolored map

Another example for map-coloring is the following: color this map by using three colors (green, red, yellow), and no adjacent areas have the same color:

Variables: A, B, C, D, E, F, G.

Domains: $\{\text{green}, \text{red}, \text{yellow}\}$.

Constraints: $\mathrm{A} \neq \mathrm{B}, \mathrm{A} \neq \mathrm{C}, \mathrm{B} \neq \mathrm{D}, \mathrm{B} \neq \mathrm{C}, \mathrm{C} \neq \mathrm{D}, \mathrm{C} \neq \mathrm{E}, \mathrm{C} \neq \mathrm{F}, \mathrm{D} \neq \mathrm{F}, \mathrm{E} \neq \mathrm{F},\ \mathrm{G} \neq \mathrm{F}$.

Solution:$\mathrm{A} = \text{red}, \mathrm{B} = \text{green}, \mathrm{C} = \text{yellow}, \mathrm{D} = \text{red}, \mathrm{E} = \text{red}, \mathrm{F} = \text{green}, \mathrm{G} = \text{yellow}$ (Figure 4).

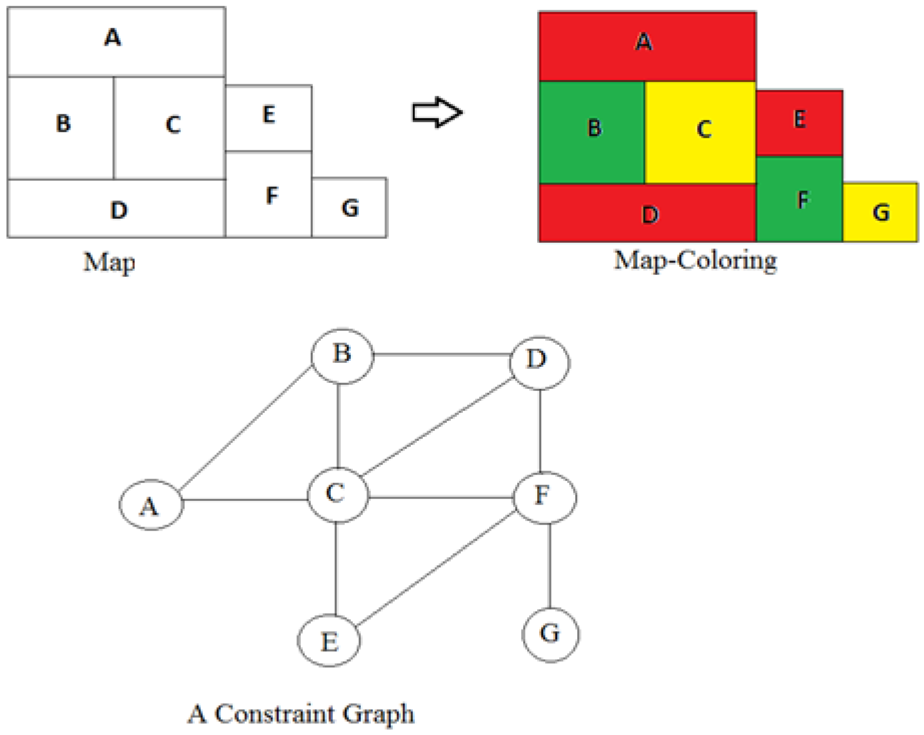


Figure 4: Map coloring and corresponding constraint grap

3) Higher-order constraints: consist of three or more variables to be represented by the constraint hypergraph. In the figure 5 the hypergraph coloring is represented.

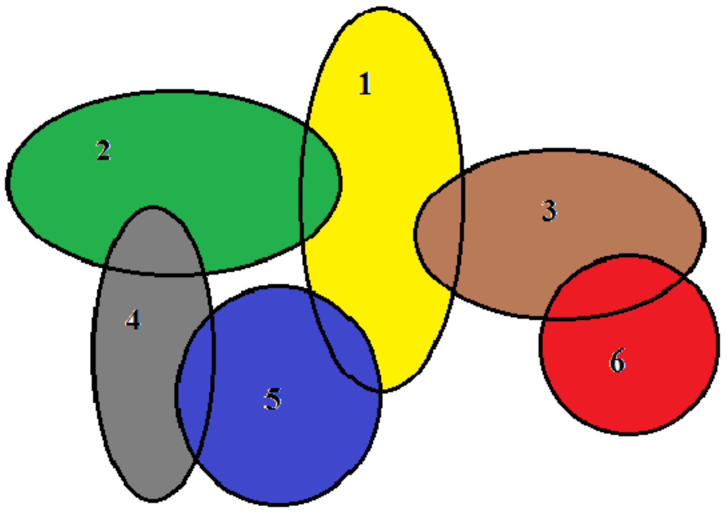


Figure 5: Hypergraph coloring

$K = \{1, 2, 3, 4, 5, 6\}, H\ (K) = \ (K, B).$

$B = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{1,2\}, \{1,3\}, \{1,5\}, \{2,4\}, \{3,6\}, \{4,5\},$

$\{1,2,3\}, \{1,2,4\}, \{1,2,5\}, \{1,3,5\}, \{1,3,6\}, \{1,4,5\}, \{2,4,5\}, \{1,2,3,5\}\}.$

3.4 Cryptarithmetic problem

In cryptarithmetic problem, each letter stands for a different digit (or each letter could be one digit), 0 through 9. The purpose of the problem is to determine which digit represents which letter.

Consider the following cryptarithmetic problem:

$$\begin{array}{r} \text{TOM} \\ \underline{+\ \text{NAG}} \\ \text{GOAT} \end{array}$$

Formalization of cryptarithmetic problem as a CSP:

Variables: $\{T, O, M, N, A, G)$

Domain: {0,1,2,…,9}

Number of variables: 6

Constraints:

$B1 =: \{T, O, M, N, A, G) \in R^6 \mid 0 \leq T, \ldots, G \leq 9\}$

$B2 = \{(T, O, M, N, A, G) \in R^6 \mid 100 * T + 10 * O + M +$

$100 * N + 10 * A + G =$

$1000 * G + 100 * O + 10 * A + T =$

$B3 = \{(T, O, M, N, A, G) \in R^6 \mid T \neq 0\}$

$B4 = \{(T, O, M, N, A, G) \in R^6 \mid N \neq 0\}$

$B5 = \{(T, O, M, N, A, G) \in R^6 \mid T, \ldots, G \text{ all different digits}\}$

Solution: $(7, 0, 6, 3, 5, 1) \in R^6$

```
  706
+ 351
 1057
```

Another cryptoarithmetic problem with one more variable is as follows:

```
  HERE
+  SHE
 COMES
```

Formalization of this problem as a CSP:

Variables: $\{H, E, R, S, C, O, M\}$

Number of variables: 7

Constraints:

$$B1 = \{(H, E, R, S, C, O, M) \in R^7 \mid 0 \leq H, \dots, M \leq 9\}$$

$$B2 = \{(H, E, R, S, C, O, M) \in R^7 \mid 1000 * H + 100 * E + 10 * R + E + 100 * S + 10 * H + E = 10000 * C + 1000 * O + 100 * M + 10 * E + S\}$$

$$B3 = \{(H, E, R, S, C, O, M) \in R^7 \mid H \neq 0\}$$

$$B4 = \{(H, E, R, S, C, O, M) \in R^7 \mid S \neq 0\}$$

$$B5 = \{(H, E, R, S, C, O, M) \in R^7 \mid H, \dots, M \text{ all different digits}\}$$

Solution: $(9, 4, 5, 8, 1, 0, 3) \in R^7$

$$\begin{array}{r} 9454 \\ \underline{+\ 894} \\ 10348 \end{array}$$

The following is a timetable scheduling example. There are three computer courses and three instructors who are able to teach these courses. The main problem is to arrange the courses and classes in order that the same teacher will not teach more than one course at the same time.

Time:

Course 1: from 8:30am – 9:30am

Course 2: from 9:00am – 10:00am

Course 3: from 9:30am – 10:30am

The constraints for instructors are:

Teacher A can teach course 2, and course 3.

Teacher B can teach course 1, course 2, and course 3.

Teacher C can teach course 2.

By considering above constraints, it is necessary to decide which courses should be taught by the instructors A, B, and C.

The solution of the problem can be in the following form:

Course 1: {B};

Course 2: {A, B, C};

Course 3: {A, B}.

The following courses must not be arranged at the same time:

Course 1≠Course 2;

Course 2≠Course 3.

The following shows which course(s) can be taught by which instructor(s) (according to the constraints):

Course 1: {B};

Course 2: {A, C};

Course 3: {A, B}.

There are some possible solutions for this problem, and below two of these solutions are given:

First solution:

Course 1= B;

Course 2= A;

Course 3= B.

Second solution:

Course 1= B;

Course 2= C;

Course 3= A.

Chapter 4

SEARCH TECHNIQUES FOR SOLVING CONSTRAINT SATISFACTION PROBLEMS

4.1 Backtracking search

It is an algorithm to find all solutions for computational problems. After trying all possible variables in depth-first search we jump to the next variable. If a solution is not found, it is necessary to come back and take another way until the correct solution is found. The backtracking search is useful for solving CSPs, for example, n-queens problem, Sudoku, crosswords, and many others.

Solving a constraint satisfaction problem commonly means that one or more different ways to assign the value to each of the variables must be found so as no constraint is violated. Backtracking crosses the space of fractional solutions in a depth-first search, and is a regular search step for solving constraint satisfaction problems.

The backtracking algorithm usually considers the variables in some finite order. Beginning with the first variable, the backtracking algorithm assigns a temporary value for every variable in the role so long as every assigned value is found to be regular with values assigned in the last. In case the algorithm marks a variable for whatever none of the values in accordance with the precedent assignments, a dead-end happens and a backtracking takes place.

The algorithm halts when the solution is found, or no solution can be found after the backtracking process has been unsuccessful.

4.2 n-queens problem

This is a problem of positioning n chess queens on n × n chessboard in which no any two queens would be able to attack each other. There must be no any two queens positioned in the same diagonal, row, and column. Otherwise, if any two queens are in the same diagonal, row, and column, they will threaten each other.

We need to formulate the constraint satisfaction problem which is given below:

Variables: V_i for each row i;

Domain: $X_i = \{1, 2, 3, \ldots, n\}$;

Constraints: $V_i \neq V_j$.

$$V_i - V_j \neq i - j$$

$$V_i - V_j \neq j - i$$

For n-queens, there are n^n different formations.

For 4-queens, there are 256 different formations.

For 8-queens, there are 16777216 different formations.

For 16-queens, there are 18446744073709551616 different formations.

4.3 Search for 4-queens problem

A trick: create step by step and use one queen for each step.

States (nodes) compatible to configurations of 0, 1, 2, 3, 4 queens;

Links (operators) compatible with the addition of a queen;

Initial state: no queen put on the board;

The components of 4-queens problem are defined below:

Variables: represent the rows $\{\mathrm{r1}, \mathrm{r2}, \mathrm{r3}, \mathrm{r4}\}$.

Values: represent the columns $\{\mathrm{c1}, \mathrm{c2}, \mathrm{c3}, \mathrm{c4}\}$.

Constraints: constraint relations (allowed combinations). All possible position to put queens without attacking each other:

$$\mathrm{B}(\mathrm{r1}, \mathrm{r2}) = \{(1,3), (3,1), (1,4), (4,1), (2,4), (4,2)\}$$

$$\mathrm{B}(\mathrm{r1}, \mathrm{r3}) = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$$

$$\mathrm{B}(\mathrm{r1}, \mathrm{r4}) = \{(1,2), (1,3), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4), (4,2), (4,3)\}$$

$$\mathrm{B}(\mathrm{r2}, \mathrm{r3}) = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$

$$\mathrm{B}(\mathrm{r2}, \mathrm{r4}) = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$$

$$\mathrm{B}(\mathrm{r3}, \mathrm{r4}) = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$

Variables: the rows $\{1, 2, 3, 4\}$

Values: the columns $\{\mathrm{x1}, \mathrm{x2}, \mathrm{x3}, \mathrm{x4}\}$

For each column there are 4 possibilities to put queen:

$$x1 = \{1, 2, 3, 4\}$$

$$x2 = \{1, 2, 3, 4\}$$

$$x3 = \{1, 2, 3, 4\}$$

$$x4 = \{1, 2, 3, 4\}$$

- First step:

The first queen is put onto the cell 2 in column x1. Afterward the following cells are excluded for the next queens to make the first queen unattacked.

$x1 = 2,$ so eliminate $\{x1 = 1, x1 = 3, x1 = 4, x2 = 1, x2 = 2, x2 = 3, x3 = 2,$ $x3 = 4,\ x4 = 2\}$ (Figure 6) (red circles mean that no queen is permitted to be put onto).

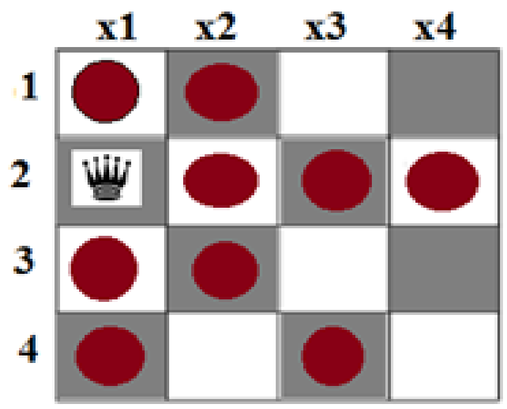


Figure 6: First step in 4-queens problem with one queen on chessboard and unattacked cells

- Second step:

After the first queen is put, there is only cell [4] in column x2 to put the second queen. So second queen is put onto the cell [4], and so the following cells must be excluded for the next queens to make the first two queens unattacked.

$\mathrm{x1} = 2,\ \mathrm{x2} = 4,$ so eliminate $\{\mathrm{x1} = 1, \mathrm{x1} = 3, \mathrm{x1} = 4, \mathrm{x2} = 1, \mathrm{x2} = 2, \mathrm{x2} = 3, \mathrm{x3} = 2, \mathrm{x3} = 3, \mathrm{x3} = 4,\ \mathrm{x4} = 2, \mathrm{x4} = 4\}$ (Figure 7).

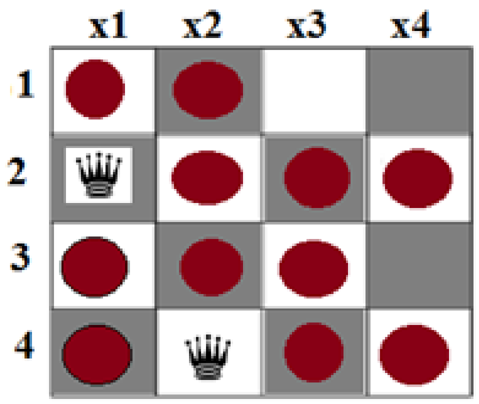


Figure 7: Second step in 4-queens problem with two queens on chessboard and unattacked cells

- Third step:

There is only cell 1 of the column x3 to put the third queen. So the third queen is put on. After the third queen is put, the following cells must be excluded for the last fourth queen:

$\mathrm{x1} = 2,\ \mathrm{x2} = 4, \mathrm{x3} = 1,$ so eliminate $\{\mathrm{x1} = 1, \mathrm{x1} = 3, \mathrm{x1} = 4, \mathrm{x2} = 1, \mathrm{x2} = 2, \mathrm{x2} = 3, \mathrm{x3} = 2, \mathrm{x3} = 3, \mathrm{x3} = 4,\ \mathrm{x4} = 1, \mathrm{x4} = 2, \mathrm{x4} = 4\}$ (Figure 8).

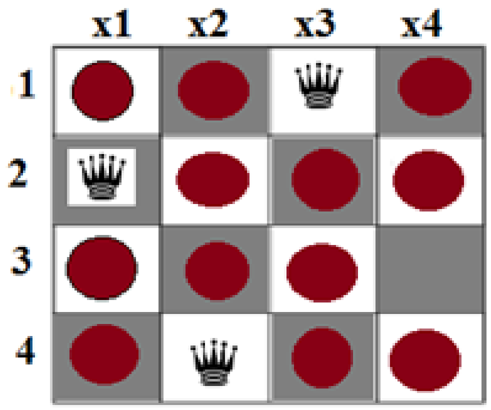


Figure 8: Third step in 4-queens problem with three queens on chessboard and unattacked cells

- Fourth step:

The only cell to put the last fourth queen on is the cell 3 of the column x4. So the final form of one of the possible solutions of 4-queens problem is illustrated in Figure 9.

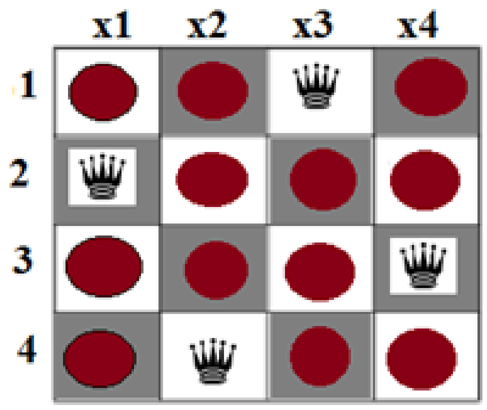


Figure 9: One of the possible solutions of 4-queens problem

The Figure 10 represents the sequence of steps (from starting to target) for a possible solution of 4-Queens problem.

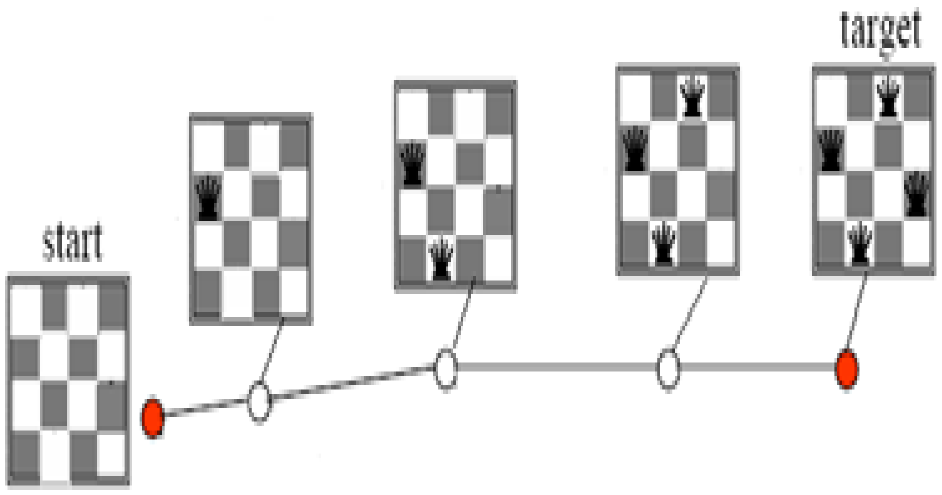


Figure 10: The sequence of steps for the possible solution of 4-queens problem

Finding all the solutions of n-queens problem can be reached just by using backtracking.

The backtracking search can be also used to solve the 8-queens problem, where the problem is to arrange eight chess queens on a chessboard so that no queen attacks any another queen. So, the arrangement of queens should be carried out in the form that no queen will be attacked by any another queen on the chessboard. It can be easily calculated that there are 4,426,165,368 (8^8) possible ways to arrange queens on 8×8 chessboard, but there are only 92 possible solutions of this problem. In backtracking, any incomplete solution that contains any two queens attacking each other, should be rejected and it is needed to go back to take another way in order to get the correct solution.

Below the backtracking algorithm for n-queens problem is presented:

1) At the first column and first row start with the first queen.

2) Determine which position the first queen will attack.

3) Go on with the second queen to be put at first row and second column.

4) Continue until a correct position is found. Then move to step 8.

5) If there is no permissible position on chessboard to put queen, reject to put.

6) Go to the previous queen.

7) Go back to step 4.

8) Go on with the next queen.

9) Determine which position will be attacked.

10) Stop if it is the last queen. If it is not, go back to step 3.

This is the advantage of a forward checking that no domain-specific initial state, or successor function or testing of a goal are required to perform the task.

4.4 Formalization of 8-queens problem

The 8-queens problem can be formulated as follows:

State: Arrangement of 8 queens on a chessboard.

Initial state: a chessboard is empty.

Successor function: put queen in any cell of a chessboard.

Goal: all eight queens on a chessboard and no queen attacks any another queen on the chessboard.

First, there are eight possible cells to put the queen in the first row. It is shown in Figure 11.

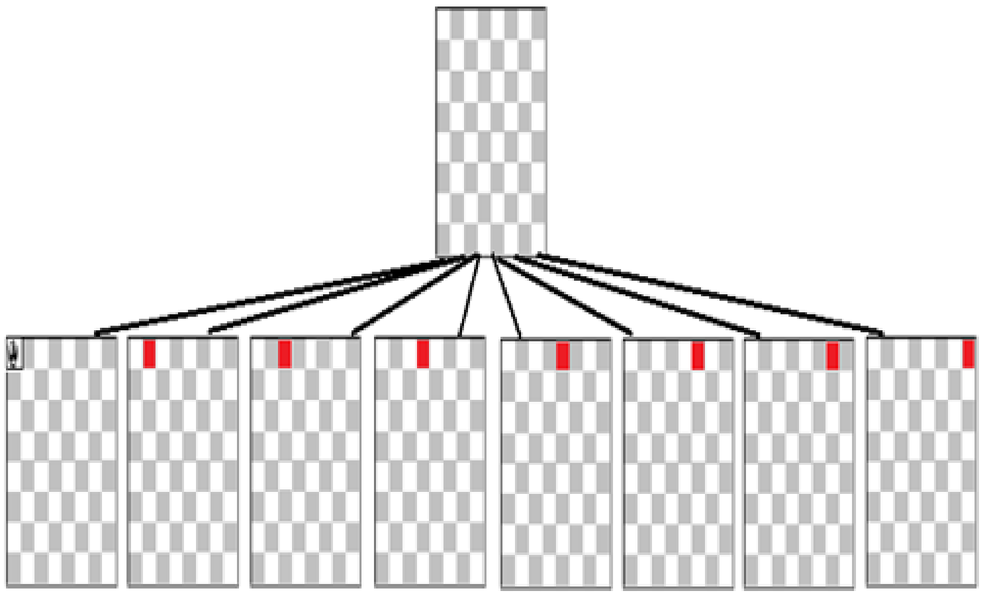


Figure 11: The possible arrangements for first chess queen in 8-queens problem

In the next stage, for each queen, there are many cells to be put, but the second queen must be put on those cells which will not be attacked by any another queen (Figure 12).

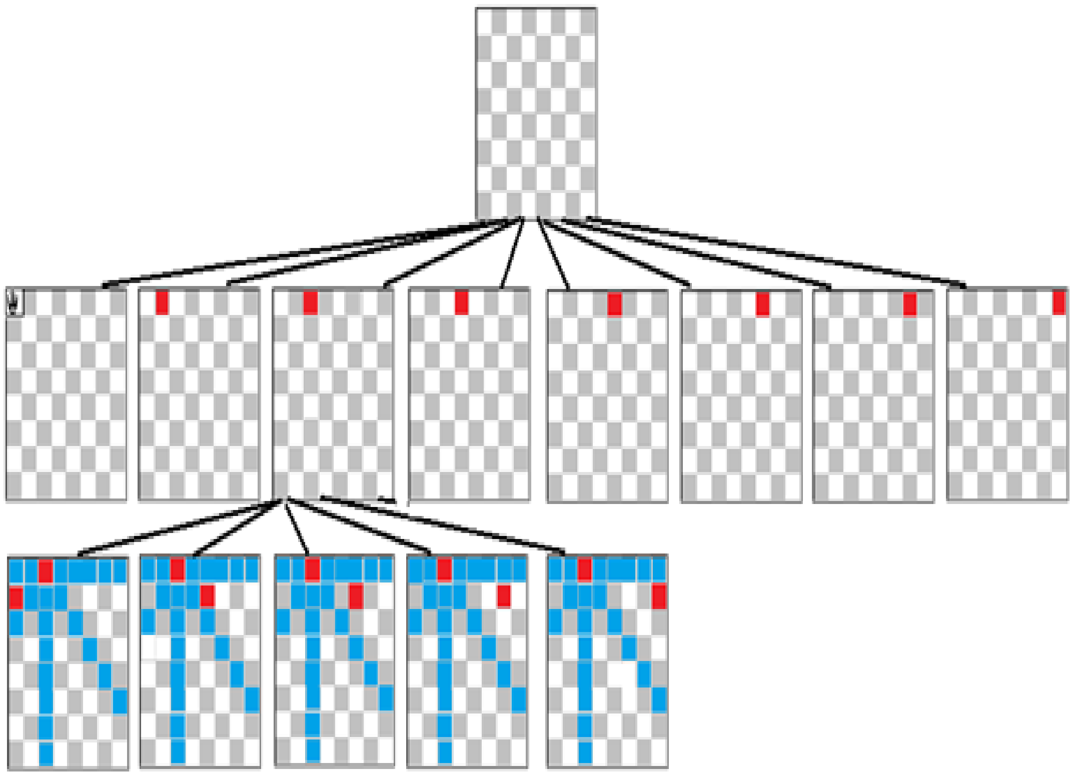


Figure 12: The possible arrangements for two chess queens in 8-queens problem

A possible position for the third queen can be arranged on a chessboard in the form illustrated in Figure 13:

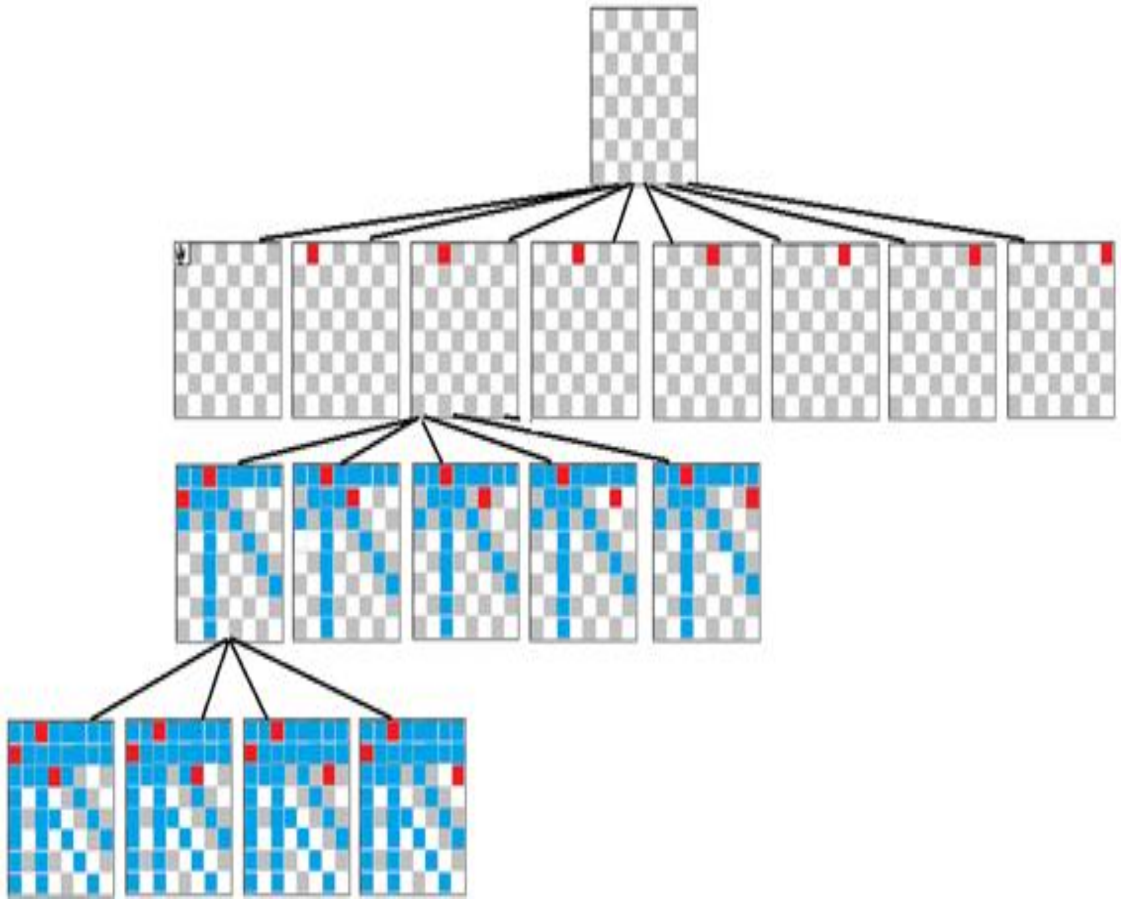


Figure 13: The possible arrangements for three chess queens in 8-queens problem

In Figure 14, one of the possible solutions of 8-queens problem is represented. It is easy to see that none of any 8 queens is attacked by any another queen. The solution of the problem can be only reached by using backtracking search.

4.5 Local search

The local search is a helpful way for some optimization problems and consistency.

The local search techniques are unfinished satisfiability algorithms which can get a solution of a problem.

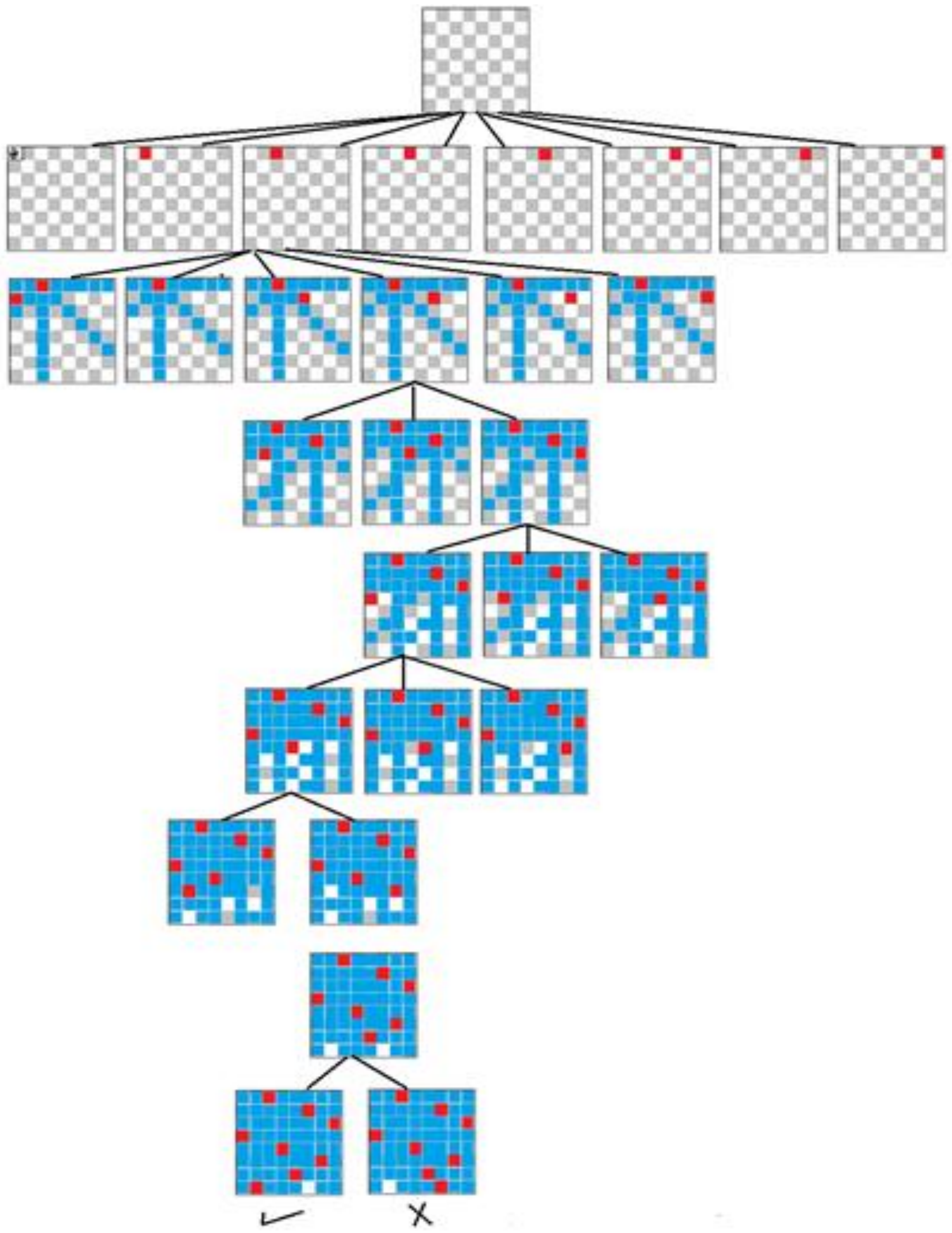


Figure 14: One of the possible solutions of 8-queens problem

The idea of a local search is looking into the range of full tasks of values to variables, and neighbors of an existing node are comparable variable tasks. It is required to go on from one node to any another node agreeing to a function.

4.6 Constraint propagation

The constraint propagation methods are used to adjust a constraint satisfaction problem (CSP). The main idea of using constraint propagation is to find a feasible area of a point meaning that all the variables must satisfy the constraints. In another way, they are methods that impose a form of consistency which is a requirement concerning the consistency of a set of constraints and/or variables. The constraint propagation has different uses. First, it becomes a problem that is similar but easier to be solved. Second, it might prove satisfiability of problems. This is not ensuring to happen in general. Every time it happens for some certain kinds of problems and for some forms of constraint propagation. The famous constraint propagation technique is the AC-3 algorithm, which imposes an arc consistency**.**

4.7 Forward checking

It is similar to backtracking, but also deals with the future variable. Forward checking is always choosing those variables that haven't chosen yet.

For solving CSP, there is a forward checking algorithm which is an effective alternative to a backtracking. The forward checking is closely regarding to backmarking which is a vastly used improvement of normal backtracking. The backmarking helps understand a lately enter advance to the forward checking algorithm which is called a minimal forward checking.

On the other hand, a forward checking it totally different from backtracking, and it is easier than backtracking. Because forward checking is checking the same nodes as backtracking, but a loss of time occurs to assign next nodes that might has never been used. If there's no either useful value for present variable, back-track occurs to the past variable to define another value. A useful value for the present variable is a value that is not signed useless from the allocated variables.

The constraint propagation is a heavier-duty copy of a forward checking. Nevertheless, whole arc consistency graph traverse is not needed to execute forward checking. Finally, the formation of the constraint graph can be utilized to facilitate the solution operation.

Unfortunately, forward checking does not provide the premature disclosure for all the mistakes. In particular, it does not reveal fail between any two unassigned variables.

4.8 Sudoku as a CSP

Sudoku is a puzzle which is based on logic, and the numbers are placed in a particular order.

A Sudoku is a matrix of 9×9 grid, divided into nine 3×3 sub-grids, and each sub-grid must contain the digits from 1 to 9, and also no column, row, and block must contain repeating digits from 1 to 9 (it must be distinct in every vertical line, horizontal line and 3×3 square). Firstly, some cells of a matrix 9×9 are filled with some digits, and the target is to fill the empty cells. It is required to use all possible outcomes of Sudoku until the solution reached.

The formalization of a Sudoku game is as follows:

Variable: $\mathrm{Y11}, \mathrm{Y22}, \ldots\ldots, \mathrm{Y99}$.

Domain: $\{1, 2, \ldots, 9\}$.

Constraints:

Column constraints: $\{\mathrm{Y11} \neq \mathrm{Y12}, \mathrm{Y11} \neq \mathrm{Y13}, \ldots, \mathrm{Y11} \neq \mathrm{Y19}\}$

Row constraints: $\{\mathrm{Y11} \neq \mathrm{Y12}, \mathrm{Y11} \neq \mathrm{Y13}, \ldots, \mathrm{Y11} \neq \mathrm{Y19}\}$

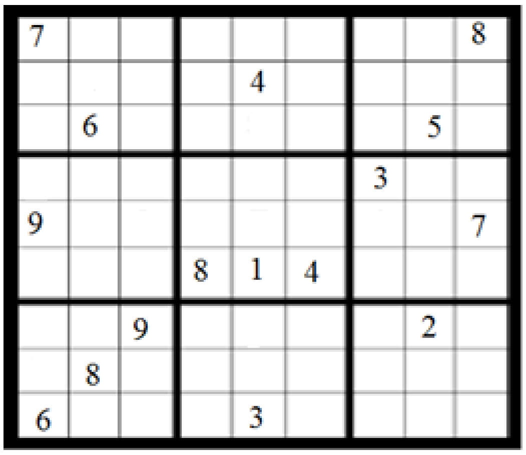
Block constraints: $\{\mathrm{Y11} \neq \mathrm{Y12}, \mathrm{Y11} \neq \mathrm{Y13}, \ldots, \mathrm{Y11} \neq \mathrm{Y33}\}$

There are several types of Sudoku:

1. Mini Sudoku;
2. Alphabetical Sudoku;
3. Akshara Sudoku**;**
4. Hypersudoku.

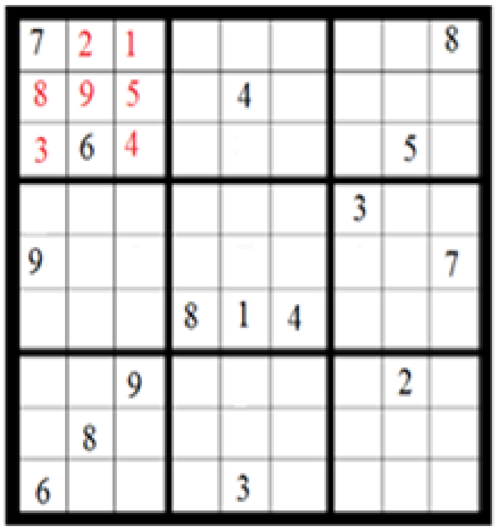
An example of Sudoku problem as a CSP is given below: some cells are filled with digits from 1 to 9 (Figure 15).

One of 3×3 grids is filled with red numbers, and this grid contains all the numbers from 1 to 9 with no repetition of any number in each grid, column and row (Figure 16).



7								8
				4				
	6						5	
						3		
9								7
			8	1	4			
		9					2	
	8							
6				3				

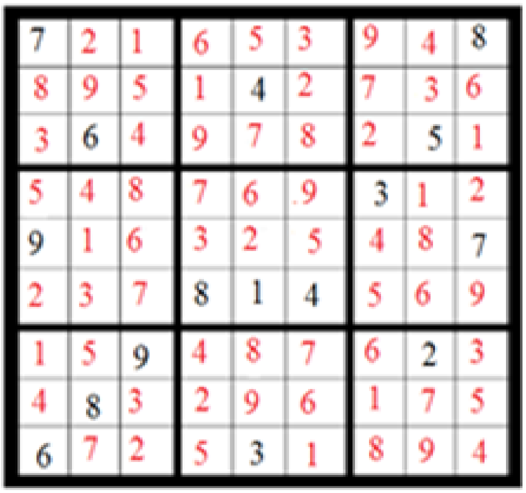
Figure 15: The first state in Sudoku game filled with digits from 1 to 9



7	2	1						8
8	9	5		4				
3	6	4					5	
						3		
9								7
			8	1	4			
		9					2	
	8							
6				3				

Figure 16: The second state in Sudoku game without repetition of digits in the first grid

The complete solution to Sudoku game is illustrated in Figure 17.



7	2	1	6	5	3	9	4	8
8	9	5	1	4	2	7	3	6
3	6	4	9	7	8	2	5	1
5	4	8	7	6	9	3	1	2
9	1	6	3	2	5	4	8	7
2	3	7	8	1	4	5	6	9
1	5	9	4	8	7	6	2	3
4	8	3	2	9	6	1	7	5
6	7	2	5	3	1	8	9	4

Figure 17: Complete solution to Sudoku game

Chapter 5

CONCLUSION

In this thesis, some strategies for solving constraint satisfaction problems are described. These problems have several possible solutions, and each problem has a different way to be solved. Constraint satisfaction problem is defined by a relation on a subset of the set of variables. In other words, the constraint satisfaction problem consists of a set of variables, a set of the values domain, and a set of constraints. In addition, the CSP can be viewed as a search problem with the initial state, successor function, goal, and past cost.

A unary constraint, binary constraint, higher-order constraint and soft constraint are types of constraint satisfaction problem. CSPs are used in many fields such as graph-coloring problem, optimization problems, n-queens problem, Sudoku and many others. There are some search techniques such as backtracking search, local search, and constraint propagation for solving CSP. The backtracking search is used to solve the n-queens problem.

REFERENCES

[1] Dobrev, D. (2005). A Definition of Artificial Intelligence. *Mathematica Balkanica, New Series Volume 19,* pp. 67-73.

[2] Tim, M. (2008). Artificial Intelligence: A Systems Approach. *Infinity Science Press LLC.*

[3] Ghedira, K. (2013). Constraint Satisfaction Problems: CSP Formalisms and techniques. *Wiley-ISTE,* p. 240.

[4] http://en.wikipedia.org/wiki/Constraint_satisfaction_problem.

[5] Ettaouil, M., Loqman, C., Haddouch, K., & Hami, Y. (2013). Maximal Constraint Satisfaction Problems Solved by Continuous Hopfield Networks. Wseas Transactions on Computers, Issue 2, Volume 12, pp. 29-40.

[6] Bessiere, C., Maestre, A., Brito, I., & Meseguer, P. (2005). Asynchronous Backtracking without Adding Links: A new Member in the ABT Family. *Artificial Intelligence,* Volume 161, Issues 1-2, pp. 7-24.

[7] Yokoo, M., Durfee, E., Ishida, T., & Kuwabara, K. (1998). Distributed constraint satisfaction problem: Formalization and Algorithms. *IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No. 5,* pp. 673–685.

[8] Mamoulis, N., & Stergiou, K. (2004). Algorithms for Quantified Constraint Satisfaction Problems. *In Proceedings of CP-2004,* pp. 752-756.

[9] Schoning, U. (1999). A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems. *FOCS '99 Proceedings of the 40th Annual Symposium on Foundations of Computer Science,* pp. 410-414.

[10] Susan L., & Xingjian Li. (2009). Cluster Graphs as Abstractions for Constraint Satisfaction Problems. *ARA, AAAI.*

[11] Klin, B., Lasota, S., Ochremiak, J., & Torunczyk, S. (2014). Turing Machines with Atoms, Constraint Satisfaction Problems, and Descriptive Complexity. *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS).*

[12] Hirayama, K., & Yokoo, M. (1997). Distributed Partial Constraint Satisfaction Problem. *Principles and Practice of Constraint Programming,* pp. 222-236.

[13] Debruyne, R., & Bessiere, C. (1997). Some Practicable Filtering Techniques for the Constraint Satisfaction Problem. *In Proceedings of IJCAI'97,* pp. 412-417.

[14] Roberto J. B., & Daniel P. M. (1995). On the Space-Time Trade-off in Solving Constraint Satisfaction Problems. *In Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95},* p.p. 558-562.

[15] Mohamed, A., Yusoff, M., Mohtar, I., A., Mutalib, S., & Rahman, S., A. (2008). Constraint Satisfaction Problem Using Modified Branch and Bound Algorithm. *Wseas Transactions on Computers, Issue 1, Volume 7.*

[16] El Mouelhi, A., Jegou, P., Terrioux, C., & Zanuttini, B. (2012). On the Efficiency of Backtracking Algorithms for Binary Constraint Satisfaction Problems. *International Symposium of Artificial Intelligence and Mathematics (ISAIM).*

[17] Haselbock, A. (1993). Exploiting Interchangeabilities in Constraint Satisfaction Problems. *In Proceedings of IJCAI-93,* pp. 282-287.

[18] Larose, B., Loten, C., & Tardif, C. (2007). A characterisation of First-Order Constraint Satisfaction Problems. *LMCS 3 (4:6).*

[19] Jose E., Cotta, C., & Antonio J. (2009). Solving Weighted Constraint Satisfaction Problems with Memetic/Exact Hybrid Algorithms. *Journal of Artificial Intelligence Research, Volume 35,* pp. 533-555.

[20] Bessière, C. (1991). Arc-Consistency in Dynamic Constraint Satisfaction Problems. *AAAI'91 Proceedings of the ninth National conference on Artificial intelligence - Volume 1*, pp. 221-226.

[21] Verfaillie, G., & Schiex, T. (1994). Solution Reuse in Dynamic Constraint Satisfaction Problems. *AAAI-94 Proceedings,* pp. 307-312.

[22] Dantsin, E., & Wolpert, A. (2002). Solving Constraint Satisfaction Problems with DNA Computing. *COCOON 2002, LNCS 2387,* pp. 171-180.

[23] Prosse, P. (1993). Hybrid Algorithms for the Constraint Satisfaction Problem. *Computational Intelligence, Volume 9, Number 3,* pp. 268-299.