A SOLVER FOR INCONSISTENT CONTINUOUS CONSTRAINT SATISFACTION PROBLEMS: *an Application to Automobile Shock-Quality Testing*

RICHARD GARRETT COY

DEPARTMENT OF COMPUTER SCIENCE

APPROVED:

_____
Martine Ceberio, Ph.D., Chair

_____
François Modave, Ph.D.

_____
Leticia Velazquez, Ph.D.

_____
Charles H. Ambler, Ph.D.
Dean of Graduate School

to my

MOTHER, FATHER, and GRAND-PARENTS

with love

A SOLVER FOR INCONSISTENT CONTINUOUS CONSTRAINT SATISFACTION PROBLEMS: *an Application to Automobile Shock-Quality Testing*

by

RICHARD GARRETT COY, B.S.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the degree of

MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT EL PASO

DECEMBER 2005

UMI Number: 1430239

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

ACKNOWLEDGEMENTS

First, I would like to express my greatest gratitude to my advisor Dr. Martine Ceberio of the Computer Science Department at the University of Texas at El Paso. Her time, continuous support, encouragement, and guidance through this endeavor will always be valued. Furthermore, I am very thankful for the knowledge gained from this area of research since it can certainly be applied to many problems that exist in real world applications. In addition, I would also like to thank my committee members Dr. François Modave and Dr. Leticia Velazquez for their suggestions with respect to the organization of this thesis as well as the defense presentation.

Secondly, I would like to thank my mother, Esther Coy and Grand-Parents, Mr. and Mrs. Bruno Vea for their love, support, and tolerance during the time needed to complete this Master's Degree in Computer Science. Also, I am especially grateful to my brother, Eric Coy for his potential ideas that were motivated by his Electrical Engineering experience in Industry. Secondly, I want to express special thanks to my father, Charles I. Coy for his Electrical Engineering influence and zero tolerance for academic failure while growing up, since he was the prime inspiration for pursuing a technical degree.

In addition, I am grateful for the help and continuous email support provided by Luc Jaulin, Xavier Baguenard, and Frédéric Goualard concerning the functionality of their CSP solvers. Through their busy schedule, they always seemed to answer questions in a very promptly manner. I would also like to express my appreciation to MSD Ignitions, a local company that employed me through this endeavor. Due to the

flexibility of my work schedule at their firm, I was able to complete my Master's Degree in a timely manner and gain additional work experience at the same time.

Lastly, I would like to give special thanks to Cynthia S. Campos for always giving me something to look forward too as well as the University of Texas at El Paso Computer Science Department professors for providing the necessary curriculum and tools to be successful in all possible endeavors. There is one final person in the department that I would like to express my appreciation towards, our Graduate Program Coordinator and true friend, Sue-Ann Walker. She knows what I am grateful for!

ABSTRACT

Constraint Programming is an efficient declarative software technology that has attracted attention to researchers because of its potential to solve hard real world Science and Engineering problems. The users simply state their problem and the computer provides them with solutions. In this thesis, we are primarily interested in Constraint Programming where the problem is defined over continuous domains.

In particular, we are concerned about two opposite situations that can surface during *Constraint Solving*, a particular branch of Constraint Programming. Ideally, you would like to determine a unique solution, but because this is not always the case, our research focuses on cases when no distinct solution exists and too many solutions are found. In the case where no solution is found, the model of the problem is said to be inconsistent or incompatible. However, sometimes a solution is necessary and simply stating that no solution exists is not acceptable. As a result, some form of flexibility or relaxation of the original problem can be introduced to the solving process so a weaker solution set can still be reached. Therefore, studies on this flexibility sparked a research area known as flexible constraints. In the latter case, one is to wonder which solution among many is best. Thus, finding the optimal or best solution among the entire solution set can be thought of as a *constrained optimization problem*. Intuitively, the idea is to filter the solution set by adding a criterion over the entire solution set to produce a more suitable solution.

Consequently, the center of our work focuses on both optimization problems and the development of tools used to model flexible constraints. Since we choose to work

with problems defined over continuous domains and these domains cannot be simulated on the computer, then interval methods were chosen to solve this class of problems because intervals guarantee completeness on the solution set.

Concerning flexible constraints and optimization, we have developed a Graphical User Interface that will be used as a tool both to model and solve optimization and inconsistent problems. To prove the feasibility of the optimization algorithms, many optimization problems were solved with the results detailed in this thesis. In addition, as an illustration of a flexible constraint system, we present a novel approach based on flexible constraint methodologies, to solve a parameter estimation problem motivated by the fundamental property of a shock absorber found on all automobiles. By experimentation, not only do we show that this method speeds up the solving process when classical solving approaches are overwhelmed by a huge number of constraints, but a weaker solution set is generated when classical constraint solving methods result in an empty solution set.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

"Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it."
Eugene C. Freuder.

Constraint programming (CP) is the study of computational systems based on constraints. The idea is to solve problems by placing constraints over the problem and then finding solutions that satisfy all the constraints. The earliest ideas of CP originated in the field of Artificial Intelligence dating back to the sixties and seventies. CP is a software technology that has attracted attention to researchers because of its potential to solve hard real world Science and Engineering problems efficiently. For instance, the user simply states the problem and the software application takes care of providing them with solutions.

CP has successfully been applied to wide range of problems such as scheduling [22], [23], [40], [51], natural language processing [10], graphical interfaces [15], business applications [41], molecular biology [2], [18], and parameter estimation [16], [37], [38] to name a few. In particular, many real world engineering problems require finding *all* solutions to systems of linear and non-linear constraints, or searching for all *global* optima to optimization problems. The study of these methods can be categorized as *optimization*. Solutions to optimization problems can be found by using CP solving methodologies.

CP can be broken down into two branches specifically *constraint satisfaction* and *constraint solving*. Most applications, about 95% of them as quoted in [4], deal with problems where the variables domains are defined over a finite set. Problems such as these fall under the category of *constraint satisfaction*. However, in this thesis, we are interested in a more specific class of problems known as *constraint solving*, where the domains of variables are continuous. In other words, most of the time, the domains are defined over subsets of real numbers.

1.1 Constraint Solving Cases

Many real world problems can be represented as a set of constraints, so traditional constraint solving approaches can be applied to determine a solution set. Unfortunately, a unique solution is not always found. Consequently, our work is concerned primarily with two opposite situations occurring in constraint solving namely too many solutions found or no solutions at all. For the case of too may solutions, determining the *best* solution among many with respect to the concerned problem, consists of placing a condition or criterion over the entire solution set. Therefore, the problem is transformed into a *constrained optimization* problem.

In contrast, the original problem may be modeled in an incompatible way leading to an empty solution set. Problems such as these are said to be inconsistent or over-constrained and traditional constraints are not enough to model such problems. However, it might so happen that a solution set is required and it makes no sense to conclude that none exists. Although, redefining the model of the problem might seem like an intuitive approach to generate a solution set, it is debatable as to whether the modification of a

poorly defined model actually results in an improvement. Alternatively, a better approach might be to consider other types of solution sets, namely weaker ones. The intuition behind this concept is a weaker membership to the solution set. Thus more elements of the search space are likely to be considered acceptable solutions.

Weaker solutions can be defined by introducing some form of flexibility to the solving process. For instance, it might consist of ignoring some incompatible constraints. Studies on this flexibility sparked a new area of research known as *flexible constraints*. As a result, the work in this thesis focuses mainly on the development tools used for modeling flexible constraints as well as optimization problems. In this thesis, interval methods were chosen as an approach to solve these problems defined over continuous domains because these domains can not be replicated on the computer [26], [35]. As a result, solutions are intervals that contain the "real" results as opposed to real numbers.

1.2 A Tool for Optimization and Flexible Constraints

Regarding optimization, we developed a total of three algorithms, two for unconstrained optimization and one for constrained optimization. The first algorithm for unconstrained optimization is referred to as the standard algorithm for minimization and the second algorithm is a modification to the standard algorithm by adding a midpoint heuristic. The two unconstrained optimization algorithms use interval methods or more specifically, a *branch* and *bound* algorithm to locate solution sets. The constrained optimization algorithm is based on a *branch* and *prune* global search method. This algorithm is merely an extension of the standard unconstrained optimization algorithm except a constraint solving process is needed since constraints are involved. The

performance of the optimization algorithms were tested by taking many problems from [54] and [55]. These problems were used as benchmark problems because the solutions were known before hand. That way, we could validate that our optimization algorithms were generating the correct solution set.

Concerning flexible constraints and optimization, we developed a Graphical User Interface (GUI) over two platforms, to be used as a tool to model and solve inconsistent and optimization problems. With this tool, the user has the ability to easily manipulate both types of problems. Both GUIs support all of the optimization algorithms presented in this thesis. With respect to flexible constraints, our GUI applies a flexibility framework explained in [6] based on optimization and constraint solving. In this framework, a distance function is defined. Once defined, distance functions are minimized to determine the minimum flexibility needed for a constraint system to be satisfied. However, if distance functions are not allowed, then traditional constraint solving methods are applied to maintain a preference order over the constraints.

1.3 An Application of Flexible Constraints

As an illustration of a flexible constraint framework, a real world engineering problem motivated from the active decay over time of the quality of a shock absorber located on all automobiles was successfully solved using our flexible framework. The overall idea is to collect and store data, while an automobile is in motion, from a shock absorber. By examining the stored data, a smart gauge located somewhere in the interior of the vehicle would notify the driver if the quality of a shock absorber falls below some acceptable tolerance.

To simulate our experiment, we generated sampled shock absorber data at different sample rates using the general equation for Damped Harmonic Motion. As a result, we formed a constraint system from the sampled shock data to determine the unknown parameters of our model. In this case, by using the flexible approach, we accurately estimated the unknown parameter known as the *damping constant*. Once this constant is known, we can easily determine the quality of a shock absorber when compared against some threshold. As a result, we show, by experimentation, that this approach not only speeds up the solving process when the size of the original problem exceeds the capabilities of classical constraint solving methods, but a solution set is generated when classical constraint solving methods result in an empty solution set. The experimental results are detailed in Appendix G.

1.4 Thesis Outline

This thesis is organized as follows: An overview of some preliminary concepts with respect to the research in this thesis is presented in the following chapter. In Chapters 3, a complete overview of interval analysis is given followed by a chapter detailing an algorithm for constraint solving using interval methods. Chapter 5 provides a general description of the most commonly used frameworks to handle flexible constraints. Chapters 6 and 7 formulate the contribution of this thesis. A Graphical User Interface that we developed to model and solve both inconsistent and optimization problems are presented in Chapter 6. Chapter 7 details an application motivated by a real world engineering problem that was successfully solved using our flexible constraint

framework. The remaining two chapters focus on possible extensions of this work and the key points and conclusions presented in this thesis.

CHAPTER 2

PRELIMINARY CONCEPTS

The purpose of this chapter is to provide a general overview of the preliminary concepts emphasized in this thesis. This chapter begins with the definition of a Constraint Satisfaction Problem (CSP) followed by a description of an Optimization Problem. Next, a general overview of a solving process for both CSPs and Optimization Problems is presented. Finally, the chapter is concluded with a brief introduction to Flexible Constraints.

2.1 Constraint Satisfaction Problem

A Constraint Satisfaction Problem can be stated as a problem consisting of a finite number of *variables*, a set of requirements or *constraints* which defines a relationship between variables, and a set of domains for each variable.

Definition 1 (Constraint Satisfaction Problem) A *constraint satisfaction problem* is a triple P = (*V,D,C*), where

- $V = \{v_1, \ldots, v_n\}$ is a finite set of variable(s)
- $D = \{D_1, \ldots, D_n\}$ is a finite set of domain(s), where each domain is a infinite set of values for the corresponding variable.
- $C = \{c_1, \ldots, c_n\}$ is a finite set of constraint(s) restricting the values of the variables $\{v_1, \ldots, v_n\}$ that they can simultaneously take.

A constraint can be defined over continuous or discrete domains referred to as continuous constraints or discrete constraints respectively. A continuous constraint is a

type of constraint where the domain on the variables is continuous. In other words, the variables can take on an infinite number of values that are not even enumerable (continuous domain). For instance, given a constraint $x^2 + y^2 = 1$, where x and y are defined over the interval [-1,1]. This constraint is continuous since the variables x and y can take on all values within the interval [-1,1]. In contrast, a discrete constraint is a type of constraint where the domain of the variables can take on a finite number of values (discrete domain).

A constraint $c \in C$ is said to be *satisfied* if the variables in c contain some value from its domain such that c is true. Those elements or values in the domain that satisfy c are known as *consistent*. Conversely, those elements that violate the constraint are called *inconsistent*. For example, given the constraint $x^2 + y^2 = 1$, where x and y are both defined over the interval [-1,1]. The constraint is satisfied when x = [1,1] and y = [0,0]. If all the constraints in the CSP are satisfied at the same time then this is a solution to the CSP. This process of finding all values for the variables (within a specified domain) that meet or satisfy the set of constraints in the CSP simultaneously is known as *Constraint Solving* when the variables are defined over continuous domains. However, it is referred to as *Constraint Satisfaction* when the variables are defined over discrete domains.

Figure 2-1 illustrates a Constraint Satisfaction Problem defined by the following constraints two constraints: $y + \frac{1}{2}x^2 - 2.0 \le 0$ and $2y + x + 2.0 \ge 0$ over domains $x \in [-1,6]$ and $y \in [-4,1]$. The search space indicated by the rectangle is restricted to only those values in the domains x and y. The solution to the CSP is the shaded area denoted as “solution set”.

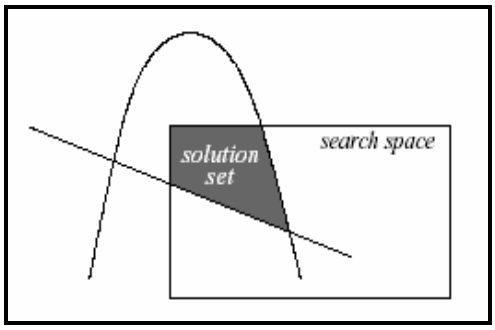

Figure 2-1. A Constraint Satisfaction Problem

2.2 Optimization Problem

Within the exploration of the search space, we may decide to search for the best solution among all possible solutions or sometimes the number of solutions to a CSP is so large that we need to discriminate among them. Such a problem is referred to as an *optimization problem*. A function that is associated with an optimization problem is called the *objective function*. The study of methods to find all optima to optimization problems and all solutions to systems of linear and non linear constraints is known as *optimization*. There are two types of optimization problems namely, unconstrained and constrained optimization.

Problems defined as unconstrained optimization problems seek to find a *global* minimum or maximum of an objective function $f(x_1,...,x_n)$ over variables $x_1,...,x_n$. Before we continue, it is important to mention that a minimum or maximum point can also be *local* (See Figure 2-2).

Definition 2 (Global Minimum/Maximum) Given a function $f(x_1,...,x_n)$ over variables $x_1,...,x_n$.

- A global maximum of f is a point x^* such that $f(x^*) \geq f(x)$ for all x
- A global minimum of f is a point x^* such that $f(x^*) \leq f(x)$ for all x

Definition 3 (Local Minimum/Maximum) Given a function $f(x_1,...,x_n)$ over variables $x_1,...,x_n$.

- A local maximum of f is a point x^+ such that $f(x^+) \geq f(x)$ for all x in a neighborhood of x^+
- A local minimum of f is a point x^+ such that $f(x^+) \leq f(x)$ for all x for all x in a neighborhood of x^+

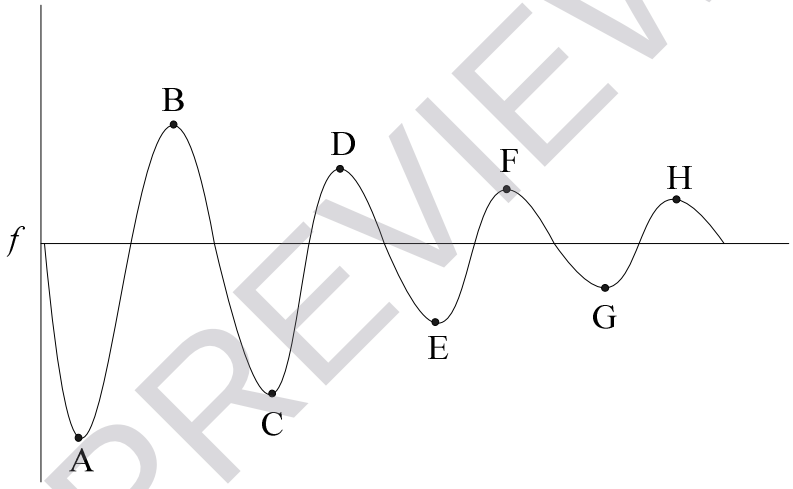

Figure 2-2. Local vs. Global Minimum and Maximum Values

In Figure 2-2, the function f consists of three local maxima points namely, D, F, and H with one global maximum point denoted as B. Similarly, points C, E, and G are local minima, but not the global minimum which is point A. Unconstrained optimization problems that search for only the global minimum of the objective function are referred to as *unconstrained minimization problems.* By simply negating the objective function i.e. $-f(x_1,...,x_n)$, the problem is transformed to an *unconstrained maximum problem*,