MANAGING TEMPORAL CONSTRAINTS WITH
UNCERTAINTY

A Thesis
Submitted to the Faculty of Graduate Studies and Research
In Partial Fulfillment of the Requirements
for the Degree of
Master of Science
In
Computer Science
University of Regina

by
Jia Liu
Regina, Saskatchewan
April, 2008

©Copyright 2008: Jia Liu
NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.
Jia Liu, candidate for the degree of Master of Science in Computer Science, has presented a thesis titled, Managing Temporal Constraints with Uncertainty, in an oral examination held on March 14, 2008. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner: Dr. Mohamed El-Darieby, Faculty of Engineering

Supervisor: Dr. Malek Mouhoub, Department of Computer Science

Committee Member: Dr. Boting Yang, Department of Computer Science

Committee Member: Dr. Samira Sadaouli, Department of Computer Science

Chair of Defense: Dr. Kamal Benslama, Department of Physics
Abstract

Representing and reasoning about time is fundamental in many real world applications, such as scheduling, planning, database design and molecular biology. Temporal Constraint Satisfaction Problems (TCSPs) are problems involving temporal information. The temporal information in TCSPs can be symbolic or numeric. The Interval Algebra (IA) network is a typical model for handling symbolic temporal information. The TemPro model is a typical model for handling symbolic and numeric temporal information. Both symbolic and numeric temporal information may be uncertain in the real world. In this thesis, the IA network is extended to represent uncertainty in symbolic temporal information while the TemPro model is extended to handle uncertainty in numeric temporal information. In both the IA network and the TemPro model, probable worlds are constructed to represent uncertain temporal problems. A new branch-and-bound algorithm is proposed to find the most robust solution. The TemPro model is also extended to handle uncertainty in a dynamic environment for Composite and Conditional Temporal Constraint Satisfaction Problems (CCTCSPs). Experimental tests are performed on randomly generated temporal problems with uncertainty. The results demonstrate the efficiency of our methods.
Acknowledgement

My foremost thanks go to my supervisor, Dr. Malek Mouhoub. He introduced and helped me to start my graduate student life in computer science. His insight and suggestions have helped me to shape my research skills. Without his patience and encouragement that carried me on through difficult times, this thesis would not be possible. His valuable feedback contributed greatly to this dissertation.

I am grateful to the financial support received from the Faculty of Graduate Studies and Research and the Department of Computer Science. Not only did I receive the funding to support my study and life, but I have also gained valuable working experience for my future career.

I also thank Dr. Samira Sadaoui and Dr. Boting Yang for reviewing my thesis.

Last but not least, I thank my grandparents, my parents and my wife for always being there when I needed them most, and for supporting me through all these years.
**Acronym List**

1. **CCSP** Conditional Constraint Satisfaction Problem  
2. **CCTCSP** Composite and Conditional Temporal Constraint Satisfaction Problem  
3. **CSP** Constraint Satisfaction Problem  
4. **DCSP** Dynamic Constraint Satisfaction Problem  
5. **IA** Integral Algebra  
6. **ICSP** Interactive Constraint Satisfaction Problem  
7. **MCSP** Mixed Constraint Satisfaction Problem  
8. **OCSP** Open Constraint Satisfaction Problem  
9. **PCSP** Probabilistic Constraint Satisfaction Problem  
10. **SCSP** Stochastic Constraint Satisfaction Problem  
11. **SOPO** Set of Possible Occurrences  
12. **STP** Simple Temporal Problem  
13. **STPU** Simple Temporal Problem with Uncertainty  
14. **TCSP** Temporal Constraint Satisfaction Problem
# Table of Contents

Abstract................................................................................................................. i

Acknowledgement ................................................................................................. ii

Acronym List ............................................................................................................. iii

Table of Contents ...................................................................................................... iv

List of Figures ........................................................................................................... ix

1. INTRODUCTION ................................................................................................. 1

   1.1 Problem and Motivation ................................................................................... 1

   1.2 Contributions .................................................................................................... 2

   1.3 Organization of the Thesis .............................................................................. 3

2. BACKGROUND IN CSPs AND TCSPs .............................................................. 5

   2.1 CSPs .................................................................................................................. 5

      2.1.1 Definition ................................................................................................... 5

      2.1.2 Examples ................................................................................................... 5

      2.1.3 Solutions to a CSP .................................................................................... 9

   2.2 Symbolic TCSPs ............................................................................................... 9

   2.3 Numeric TCSPs ............................................................................................... 12

      2.3.1 Arithmetic inequations for representing time ........................................... 12
6. CCTCSPs WITH UNCERTAINTY ........................................................................................................... 72

6.1 Composite Variables and Conditional Constraints ................................................................. 72

6.2 Composite Variables with Uncertainty ..................................................................................... 74

6.3 Conditional Constraints with Uncertainty ................................................................................ 76

6.4 Combining All Uncertain Factors ............................................................................................ 78

6.5 Conclusion .................................................................................................................................. 86

7. EXPERIMENTS .............................................................................................................................. 87

7.1 Instances of Symbolic TCSPs and Numeric TCSPs ................................................................. 87

7.2 Generating TCSPs with Uncertainty .......................................................................................... 89

7.2.1 Generating TCSPs with uncertainty in the IA network ....................................................... 89

7.2.2 Generating TCSPs with uncertainty in the TemPro model ............................................... 90

7.3 Testing Environment ................................................................................................................ 91

7.4 Experiments on Symbolic TCSPs with Uncertainty ................................................................. 92

7.4.1 Experiments on generating and loading the composition table ....................................... 92

7.4.2 Experiments on path consistency check ........................................................................... 93

7.4.3 Experiments on the branch-and-bound algorithm for symbolic TCSPs with uncertainty ......................................................................................................................... 94

7.5 Experiments on Numeric TCSPs with Uncertainty ................................................................. 96

7.5.1 Experiments on dividing domains into non-overlapping areas .......................................... 96

7.5.2 Experiments on the branch-and-bound algorithm for solving numeric
TCSPs with uncertainty ................................................................. 97

7.6 Experiments on CCTCSPs with Uncertainty ................................. 99

7.7 Summary of Experimental Results .............................................. 102

7.8 Conclusion .................................................................................. 102

8. CONCLUSIONS AND FUTURE WORK ........................................ 104

Bibliography ................................................................................... 108
List of Figures

Figure 2.1  The eight queen puzzle and one possible solution  6
Figure 2.2  The map coloring problem  7
Figure 2.3  The constraint graph of the map coloring problem  8
Figure 2.4  Allen algebra primitives  10
Figure 2.5  An example of symbolic TCSPs  11
Figure 2.6  One solution for a symbolic TCSP  12
Figure 2.7  An example of STPs  13
Figure 2.8  The SOPO of a given event  14
Figure 2.9  The TemPro representation of a scheduling problem  15
Figure 2.10  One possible solution to the task scheduling problem  16
Figure 3.1  An optimal conditional solution table  22
Figure 4.1  An example of an IA network with uncertainty  28
Figure 4.2  An example of the path-consistency check  34
Figure 4.3  The temporal relations among three events  34
Figure 4.4  The algorithm of path consistency check  35
Figure 4.5  The composition table for Allen primitives [2]  37
Figure 4.6  The algorithm of generating composition of disjunctive constraints  38
Figure 4.7  The pseudo code for solving symbolic TCSPs with uncertainty  46
Figure 5.1 Dividing domains into non-overlapping areas ........................................ 53
Figure 5.2 An incorrect way of dividing domains into non-overlapping areas ........ 54
Figure 5.3 Another way of dividing with mixed INF and SUP ................................. 54
Figure 5.4 The algorithm of dividing domains into non-overlapping areas ............ 56
Figure 5.5 The numeric to symbolic conversion [32]............................................. 61
Figure 5.6 The algorithm of the numeric to symbolic conversion ............................. 62
Figure 5.7 An example of the arc-consistency check [32] ..................................... 63
Figure 5.8 The algorithm of the arc-consistency check [31] .................................. 65
Figure 5.9 Pseudo code for solving symbolic TCSPs with uncertainty .................. 70
Figure 6.1 An example of CCTCSPs [35] ............................................................... 74
Figure 6.2 An uncertain conditional constraint ...................................................... 77
Figure 6.3 The pseudo code for solving CCTCSPs with uncertainty ................... 86
Figure 7.1 An example of a random numeric assignment .................................... 88
Figure 7.2 Extending the assignment to form a TCSP .............................................. 88
Figure 7.3 The generated IA network .................................................................. 89
Figure 7.4 Generating and loading the full composition table ............................... 92
Figure 7.5 Table of the test results on path consistency check (in seconds).......... 94
Figure 7.6 Table of the test results on uncertain symbolic TCSPs (M=2, in seconds)95
Figure 7.7 Table of the test results on uncertain symbolic TCSPs (M=3, in seconds)96
Figure 7.8 Test of dividing uncertain domains into non-overlapping areas .......... 97
Figure 7.9  Table of the test results on uncertain numeric TCSPs (R=2, in seconds). 98
Figure 7.10  Table of the test results on uncertain numeric TCSPs (R=3, in seconds). 99
Figure 7.11  Table of the results of test 1 on uncertain CCTCSPs (in seconds)........ 101
Figure 7.12  Table of the results of test 2 on uncertain CCTCSPs (in seconds)........ 101
1. INTRODUCTION

1.1 Problem and Motivation

A Constraint Satisfaction Problem (CSP) [25,30,44] is a general model for many problems in the real world. A CSP is defined as a tuple <X, D, C>. X is a set of variables, D is a finite set of possible values for each element X_i in X and C is a set of constraints restricting the values that the variables can simultaneously take.

A Temporal Constraint Satisfaction Problem [9,31] is a special type of a CSP that handles temporal information. There are many applications of TCSPs, such as scheduling [39], planning [21], database design [38] and molecular biology [18]. In [31], the variables in TCSPs are events which are associated with temporal intervals. Domains in TCSPs are the possible positions of temporal intervals for each event. Constraints in TCSPs specify the possible temporal relationships among events. Solving a TCSP is to find an assignment of one temporal interval to each event, in such a way that every constraint is satisfied.

There are two ways of representing temporal information, symbolic and numeric. In symbolic TCSPs, only qualitative information of constraints is provided. In numeric TCSPs, quantitative information of events and constraints is provided.

Furthermore, there are many uncertain factors in the real world temporal problems. For
example, when we are trying to schedule a plan for traveling, the aircraft we take may not arrive at the destination on time. When we are allocating tasks among processors, some processors may break down. With these uncertain factors, some solutions may be better suited for the environment than others. However, all the solutions are treated equally in the traditional TCSPs without uncertainty. Our goal is to find the most robust solution that is associated with the highest probability to satisfy all the constraints in the uncertain TCSPs.

Composite and Conditional Temporal Constraint Satisfaction Problems (CCTCSPs) [35] are TCSPs in a dynamic environment. A composite variable is defined in a domain of TCSP events. Conditional constraints are constraints that can activate inactive variables in CCTCSPs. Uncertainty also exists in composite variables and conditional constraints. The existence of some component events in composite variables may be associated with probabilities and the activation of some inactive variables may be associated with conditional probabilities. Our goal is also to find the most robust solution which is associated with the highest probability to satisfy all the constraints in the uncertain CCTCSPs.

1.2 Contributions

The contributions in this thesis are summarized below.
1. A generator for generating instances of symbolic TCSPs with uncertainty is implemented.

2. A new algorithm for generating the full composition table of disjunctive constraints is proposed and implemented.

3. A new algorithm for solving symbolic TCSPs with uncertainty is proposed and implemented.

4. A generator for generating instances of numeric TCSPs with uncertainty is proposed and implemented.

5. A new algorithm for dividing possible domains into non-overlapping areas is proposed and implemented.

6. A new algorithm for solving numeric TCSPs with uncertainty is proposed and implemented.

7. A new algorithm for solving CCTCSPs with uncertainty is proposed and implemented.

8. Experiments are conducted to test the efficiency of the algorithms listed above.

1.3 Organization of the Thesis

This thesis is organized as follows. Chapter 2 introduces traditional CSPs and TCSPs through several examples. It also discusses two types of TCSPs, symbolic TCSPs and
numeric TCSPs. Chapter 3 includes a review of the previous work on handling uncertainty in CSPs and TCSPs. Representations of uncertain factors in several previous models are introduced, such as probabilistic CSP [12], mixed CSP [13] and simple temporal problems with uncertainty [49]. Chapter 4 describes my research on symbolic TCSPs with uncertainty. The IA network is extended to handle uncertain constraints in symbolic TCSPs. A new method for computing robustness of solution is presented. At the end, a branch-and-bound algorithm for solving symbolic TCSPs with uncertainty is proposed. Chapter 5 gives a detailed description of my research on numeric TCSPs with uncertainty. The TemPro model is extended to handle uncertain domains in numeric TCSPs. An algorithm for dividing domains into non-overlapping areas is proposed. Probable worlds are then constructed by combining these areas. At the end, a new algorithm for solving numeric TCSPs with uncertainty is presented. Chapter 6 describes my research on CCTCSPs with uncertainty. The TemPro model is extended to handle uncertainty in composite variables and conditional constraints. The method of computing robustness of solutions in uncertainty CCTCSPs is also described. A new algorithm for solving uncertain CCTCSPs is presented. Chapter 7 presents the experiments. The experimental results on solving both symbolic TCSPs with uncertainty and numeric TCSPs with uncertainty are presented. Chapter 8 provides conclusions and possible future research work.
2. BACKGROUND IN CSPs AND TCSPs

In this chapter, I will discuss the background of CSPs and TCSPs. First, I will introduce the definition of CSPs through several examples. Then, I will describe symbolic TCSPs and numeric TCSPs. At the end, I will introduce a hybrid model, TemPro, for handling both symbolic and numeric information in TCSPs.

2.1 CSPs

2.1.1 Definition

A CSP[25,30,44] is a general model for problems in the real world. A CSP is defined as a tuple \( (X, D, C) \). \( X = \{x_1, x_2, ..., x_n\} \) is a set of variables. \( D = \{D_1, D_2, ..., D_n\} \) is a set of domains in which \( D_i \) is a finite set of possible values for \( x_i \). \( C \) is a set of constraints restricting the values that the variables can simultaneously take.

2.1.2 Examples

A classical example of CSP is the eight queen puzzle (first posted by Max Bezzel in 1848) (See Figure 2.1 [42]). It is a problem of putting eight queens on an 8×8 chessboard such that none of them is able to capture any other using the standard chess queen’s moves. The solution requires that no two queens share the same row, column or diagonal.
Assuming one queen in each column, the eight queen puzzle can be defined as a CSP.

**Variable Set**

\{Q_a, Q_b, Q_c, Q_d, Q_e, Q_f, Q_g, Q_h\}

\(Q_i\) is the position of queen in the row \(i (i = a, b, ..., h)\)

**Domain Set**

\{D_a, D_b, D_c, D_d, D_e, D_f, D_g, D_h\}

\(D_i = \{1, 2, 3, 4, 5, 6, 7, 8\} (i = a, b, ..., h)\)

**Constraints**

- \(Q_i \neq Q_j\), no two queens are in the same row
- \(|Q_i - Q_j| \neq |i - j|\) no two queens are in the same column

![Figure 2.1 The eight queen puzzle and one possible solution [53]](image)
Another example of classical CSPs is the map coloring problem [33]. The goal of this problem is to color a map so that adjacent countries do not have the same color. (See Figure 2.2) It can also be presented as a tuple $<X, D, C>$ in the following way.

**Variable Set**

$\{C_1, C_2, C_3, C_4, C_5, C_6\}$ represents six countries

**Domain Set**

$\{D_1, D_2, D_3, D_4, D_5, D_6\}$, where $D_i = \{\text{Red, Blue, Green}\}$

**Constraints**

No neighboring countries have the same color.

$\{C_1 \neq C_2, C_1 \neq C_5, C_1 \neq C_5, C_2 \neq C_5, C_2 \neq C_6, C_3 \neq C_5, C_3 \neq C_6, C_4 \neq C_7, C_5 \neq C_6\}$

![Figure 2.2 The map coloring problem [33]](image)

The map coloring problem can also be represented as a constraint graph (See Figure 7)
2.3), in which each node represents a country and each edge represents the constraint between two corresponding countries.

One more example of CSPs is the scheduling of crew in airline companies. Suppose there are $m$ pilots and $m$ airlines. Due to experience and preference, each pilot is only available for certain airlines. This problem can be represented as a CSP as follows.

**Variable Set**

$\{L_1, L_2, \ldots, L_m\}$ represents $m$ airlines

**Domain Set**

$\{P_1, P_2, \ldots, P_n\}$, where $P_i = \{\text{available pilots for airline } i\}$

**Constraints**

Each airline must have exactly one pilot.
2.1.3 Solutions to a CSP

For the eight queen puzzle, one solution is given in Figure 2.1. The solution is one position for each queen so that none of them is able to capture any other using the standard chess queen's moves. More generally, a solution for a CSP is mapping from each variable to one value in its domain so that all the constraints are satisfied.

The solution given in Figure 2.1 can be seen as such an assignment of positions for each queen as \( \{Q_a = 2, Q_b = 4, Q_c = 6, Q_d = 8, Q_e = 3, Q_f = 1, Q_g = 7, Q_h = 5\} \). One possible solution for the map coloring problem in Figure 2.2 is \( \{C_1 = b, C_2 = g, C_3 = g, C_4 = r, C_5 = r, C_6 = b\} \).

In classical CSPs, no solution is preferred to another one. In some problems such as the map coloring problem, only one solution is needed. In some problems such as the eight queen puzzle, people might be interested in getting all the possible solutions.

2.2 Symbolic TCSPs

TCSPs [31] are a specific type of general CSPs. All the variables in TCSPs are temporal events. The constraints in TCSPs can be either symbolic or numeric. Allen Algebra [2] (also called Integral Algebra, or IA) is the most commonly used symbolic representation of temporal information. In Allen Algebra, thirteen basic temporal relations between events are defined as Allen primitives. (See Figure 2.4 [2,32]) The
constraints in IA are disjunctions of Allen primitives. Events and constraints are defined as follows [2].

**Event**

_Couple (p, I)_ where _p_ is a proposition and _I_ is the interval where _p_ is true

**Constraints**

\[ R(ev_1, ev_2) = I_1 \lor r_1 \lor r_2 \lor \ldots \lor r_n I_2 \text{ (r_i are basic Allen relations)} \]

Consider the following example [2,32].

**Example 2.1 A Symbolic TCSP**

_Fred was reading the paper while eating his breakfast. He put the paper down and drank the last of his coffee. After breakfast he went for a walk._

<table>
<thead>
<tr>
<th>Relation</th>
<th>Symbol</th>
<th>Inverse</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>X precedes Y</td>
<td>P</td>
<td>P-</td>
<td>X _ Y</td>
</tr>
<tr>
<td>X equals Y</td>
<td>E</td>
<td>E</td>
<td>X _ Y</td>
</tr>
<tr>
<td>X meets Y</td>
<td>M</td>
<td>M-</td>
<td>X _ Y</td>
</tr>
<tr>
<td>X overlaps Y</td>
<td>O</td>
<td>O-</td>
<td>X _ Y</td>
</tr>
<tr>
<td>X during Y</td>
<td>D</td>
<td>D-</td>
<td>X _ Y</td>
</tr>
<tr>
<td>X starts Y</td>
<td>S</td>
<td>S-</td>
<td>X _ Y</td>
</tr>
<tr>
<td>X finishes Y</td>
<td>F</td>
<td>F-</td>
<td>Y _ X</td>
</tr>
</tbody>
</table>

*Figure 2.4 Allen algebra primitives [32]*
The problem in the above example can be presented in a constraint graph (See Figure 2.5 [2,32]).

For symbolic TCSPs, the constraint graph is called IA network [2]. In the IA network, each node represents the time interval of one event. The edges between nodes are disjunctive constraints of Allen primitives. For example, the constraint between the event Paper (reading paper) and the event Coffee (drinking coffee) is DOS. Therefore, this constraint can be represented as $R (\text{Paper, Coffee}) = \text{DURING} \lor \text{OVERLAP} \lor \text{START}$. The relation between the event Paper and the event Coffee can be any one of the three possible Allen primitives.

In order to solve symbolic TCSPs in the IA network, we have to find an assignment of all the temporal intervals so that every constraint between two events is satisfied. A possible solution for the above problem is given in Figure 2.6 [2,32].

![Diagram](image-url)  
Figure 2.5  An example of symbolic TCSPs [32]
2.3 Numeric TCSPs

2.3.1 Arithmetic inequations for representing time

In symbolic TCSP models such as IA networks, only symbolic information is specified in the problem. On the other hand, we can have numeric information for events and constraints.

There are two types of numeric representation of temporal information. The first one is arithmetic inequations. In arithmetic inequations, constraints between events can be defined as

\[ A - B \geq d \]  
\[ (A, B \text{ are time points of events and } d \text{ is a time interval}). \]

Simple Temporal Problem (STP) [9] is a numeric TCSP in which arithmetic inequations are used. A STP is a pair \( \langle X, C \rangle \), where \( X_i \in X \) are the time points of events and \( C_i \in C \) are binary constraints of the following form

\[ X_j - X_i \in [a_{ij}, b_{ij}] \]

Example 2.2 Simple temporal problem

The event set is defined as \( \{E_i, E_2, E_3\} \). The set of time points of these events is defined
as \{X_{11}, X_{12}, X_{21}, X_{22}, X_{31}, X_{32}\}. \{X_{11}, X_{12}\} are the start and end points of \(E_1\). \{X_{21}, X_{22}\} are the start and end points of \(E_2\). \{X_{31}, X_{32}\} are the start and end points of \(E_3\). The set of constraints is defined as \(\{X_{12} - X_{11} > 13, X_{22} - X_{21} > 6, X_{32} - X_{31} > 10, X_{21} - X_{11} > 35, X_{22} - X_{32} > -6\}\) (See Figure 2.7).

A solution for an STP is an assignment of the time points that satisfies all the arithmetic inequations between these time points. One possible solution for the above example is \(\{X_{11} = 16, X_{12} = 33, X_{21} = 56, X_{22} = 70, X_{31} = 42, X_{32} = 60\}\).

### 2.3.2 TemPro: A hybrid model for symbolic and numeric information

In the TemPro model, each event is represented as a Set of Possible Occurrences (SOPO) [31,34]. A SOPO is represented as a tuple \([\text{INF}, \text{SUP}, \text{DURATION}, \text{STEP}]\). \text{INF} is the earliest start time of the event. \text{SUP} is the latest finish time of the event.
DURATION is the length of the event, measured in the number of time units. STEP is the length for one time unit (See Figure 2.8 [34]).

Consider the following example [34].

**Example 2.3 Task scheduling problem in TemPro model**

Three tasks $T_1$, $T_2$, $T_3$ are processed by a mono processor machine $M_1$. Task $T_4$ must be processed before $T_1$ and $T_2$. $T_4$ is processed by another mono processor machine $M_2$. Each task is characterized by its processing time, its earliest start time and its latest end time.

For example, $T_1$ takes three hours, and $T_1$ must be started after 10:00 and finished before 15:00. All of the characteristics of the four tasks are listed below.

$T_1$: 3h, 10:00, 15:00

$T_2$: 3h, 20:00, 24:00

$T_3$: 4h, 7:00, 12:00

$T_4$: 1h, 9:00, 11:00

The TemPro model for this problem is shown in Figure 2.9 [34].
Figure 2.9 The TemPro representation of a scheduling problem [34]

As shown in Figure 2.9, the TemPro model includes both symbolic and numeric information. SOPOs provide numeric information for events, while the constraints between events are symbolic constraints, which are disjunctions of Allen primitives. For example, the constraint between $T_1$ and $T_2$ is $P \lor P^-$, which means $T_1$ should be processed either before $T_2$ or after $T_2$.

The solution in the TemPro model is an assignment of one temporal interval in SOPOs of each event. Besides, all the constraints must be satisfied. In other words, the relation between the assigned intervals for any two events must be one of the Allen primitives in the corresponding disjunctive constraint. One possible solution to the above example is shown below (See Figure 2.10).
In this solution, $T_1$ is assigned an interval of (12, 15) and $T_2$ is assigned an interval of (20, 23). $T_1$ is finished by 15:00 while $T_2$ is started from 20:00. Thus, $T_1$ is processed before ($P$) $T_2$.

2.4 Conclusion

In this chapter, I reviewed the general CSPs and TCSPs. I also introduced the symbolic and numeric representation of time in TCSP. In symbolic TCSPs such as the IA network, only symbolic information regarding the constraints is given. Each constraint is a disjunction of basic Allen primitives. In numeric TCSPs, numeric information regarding events and constraints is given. In STPs, arithmetic inequations are used to represent the constraints between time points of events. TemPro model is a hybrid framework for handling both symbolic and numeric information. In the TemPro model, SOPOs are used...
to represent domains of events while disjunctive constraints of Allen primitives are used to represent constraints.
3. CSPs AND TCSPs WITH UNCERTAINTY

In the previous chapter, I introduced the traditional CSPs and TCSPs. All the components in the traditional CSPs are pre-defined and certain. In this chapter, I will examine CSPs and TCSPs with uncertainty, which involves uncertain components in CSPs and TCSPs such as uncertain variables, uncertain domains and uncertain constraints.

3.1 Motivation for Handling Uncertainty in CSPs and TCSPs

A Classical CSP assumes that all the components in a problem are completely known before the problem is modeled and solved. However, it has been observed that such assumption does not hold in many situations, since many real world applications occur in uncertain environments.

In order to solve problems in uncertain environments, many frameworks [3,10-14,23,27,40,41,48,49] for modeling uncertain CSPs have been proposed. The uncertain factors in these frameworks include variables, domains and constraints. For example, we want to plan a working schedule for one week. In classical CSPs, working tasks are predetermined at the time of planning. However, some unexpected events may emerge from time to time in the real world.
The main objectives of solving CSPs and TCSPs in uncertain environments include predicting future changes, reacting rapidly to these changes and finding robust solutions. Several frameworks of handling CSPs and TCSPs with uncertainty will be examined in the following section.

3.2 Previous Work on Uncertain Constraints

3.2.1 Uncertain constraints: Probabilistic CSP

The probabilistic CSP (PCSP) [12] is used to model the situation where constraints are uncertain. In PCSPs, each constraint is associated with a probability of its existence in the real world. More formally, a PCSP consists of a set of constraints $C = \{C_1, \ldots, C_m\}$ and a set of probabilities $p_i$, which are the probabilities of $C_i$ to be existent in the real problem $P_{real}$.

- $Pr(C_i \in P_{real}) = p_i$
- $Pr(C_i \in P_{real}) = 1 - p_i$

With uncertain constraints, more than one instance of CSPs may exist for each PCSP. Each instance contains some uncertain constraints.

In PCSPs, it is assumed that all the constraints are independent of each other. Thus, the joint probability can be computed as follows [12].

$$Pr(C_i \in P_{real} \quad and \quad C_j \in P_{real}) = Pr(C_i \in P_{real}) \times Pr(C_j \in P_{real})$$
The probability of one specific instance to be the actual problem can be computed as follows [12].

\[
\Pr(P_j) = \Pr(P_j = P_{real}) = \prod \{p_i \mid C_i \in P_j\} \times \prod \{1 - p_i \mid C_i \in \mathcal{C}\} 
\]

(\mathcal{C} defines the set of constraints)

The probability that \(s\) satisfies the actual constraints is computed as follows [12].

\[
\Pr(s \in Sols(P_{real})) = \prod \{1 - p_i \mid C_i \in \text{viols}(s)\} = \sum \{\Pr(P) \mid P \in 2^\mathcal{C}, s \in Sols(P)\} \quad [12]
\]

(\text{viols}(s) is the set of constraints that are violated by \(s\))

Consider the following example for PCSP.

**Example 3.1 Probabilistic CSPs**

There are three constraints \(C = \{C_1, C_2, C_3\}, p_1 = 0.5, p_2 = 0.6, p_3 = 0.7\)

- \(Pr(\text{no constraints}) = (1 - 0.5) \times (1 - 0.6) \times (1 - 0.7) = 0.06\)
- \(Pr(\{C_1\}) = 0.5 \times (1 - 0.6) \times (1 - 0.7) = 0.06\)
- \(Pr(\{C_2\}) = (1 - 0.5) \times 0.6 \times (1 - 0.7) = 0.09\)
- \(Pr(\{C_3\}) = (1 - 0.5) \times (1 - 0.6) \times 0.7 = 0.14\)
- \(Pr(\{C_1, C_2\}) = 0.5 \times 0.6 \times (1 - 0.7) = 0.09\)
- \(Pr(\{C_1, C_3\}) = 0.5 \times (1 - 0.6) \times 0.7 = 0.14\)
- \(Pr(\{C_2, C_3\}) = (1 - 0.5) \times 0.6 \times 0.7 = 0.21\)
- \(Pr(\{C_1, C_2, C_3\}) = 0.5 \times 0.6 \times 0.7 = 0.21\)

If we have a solution \(s\) which satisfies \{no constraints\}, \{C_1\}, \{C_2\} and \{C_1, C_2\}, the probability of \(s\) to satisfy the actual world can be computed as follows.
\[ Pr(\text{s solves the actual problem}) \]

\[ = Pr(\text{no constraints}) + Pr(\{C_1\}) + Pr(\{C_2\}) + Pr(\{C_1, C_2\}) = 0.30 \]

The main goal of solving probabilistic CSPs is to find a solution with a maximal probability solving the actual problem.

### 3.2.2 Uncertain domains: Mixed CSP

Mixed CSP (MCSP) [13] is proposed to model decision problems in uncertain environments. In MCSP, variables are divided into two categories. One category is decision variables that are controllable by users and the other category is environmental variables which are uncontrollable by users. An example of MCSP is given below.

#### Example 3.2 A MCSP [13]

The decision variable set \( X = \{X_1, X_2, X_3\} \).

The domains of decision variables are \( D_1 = D_2 = D_3 = \{1, 2, 3\} \).

The environmental variable set \( A = \{\lambda_1, \lambda_2\} \).

The domains of environmental variables are \( L_1 = L_2 = \{a, b, c, d\} \).

\( K \) contains only one constraint: \( C_0: \lambda_1 \neq \lambda_2 \) \( C \) contains three constraints \( C_1, C_2, C_3 \).

\( C_1: \langle \lambda_1, X_1 \rangle = \langle a, 1 \rangle, \langle a, 2 \rangle, \langle b, 1 \rangle, \langle b, 3 \rangle, \langle c, 2 \rangle, \langle d, 3 \rangle \)

\( C_2: \langle \lambda_2, X_2, X_3 \rangle \)

\[ = \langle a, 1, 1/2/3 \rangle, \langle a, 2, 2/3 \rangle, \langle a, 3, 3 \rangle, \langle b, 1, 2/3 \rangle, \langle b, 2, 3 \rangle, \langle c, 3, 1/2/3 \rangle, \langle d, 2, 1/2/3 \rangle \]
In the above example, \( X_1, X_2, X_3 \) are decision variables and \( \lambda_1, \lambda_2 \) are environmental variables. \( K \) is the set of constraints on environmental variables. Only one constraint \( C_0 \) is in \( K \). \( \text{Poss}(P) \) is used to represent all the possible instances of environmental variables. For the above example, we have

\[
\text{Poss}(P) = \{a,b\}, \{a,c\}, \{a,d\}, \{b,a\}, \{b,c\}, \{b,d\}, \{c,a\}, \{c,b\}, \{c,d\}, \{d,a\}, \{d,b\}, \{d,c\}
\]

Thus, solving a MCSP is to find a conditional solution table such that each possible instance of environmental variables corresponds to some of the solutions in the table. \( \text{Good}(P) \) is used to represent all the possible worlds which are supported by at least one solution. Assume one conditional solution table \( T \) covers a subset \( W_s \) of \( \text{Poss}(P) \). If \( W_s = \text{Good}(P) \), \( T \) is called optimal. If \( W_s = \text{Poss}(P) \), \( T \) is called complete.

The following is an optimal conditional solution table [13] for example 3.2.

![Conditional Solution Table](image)

**Figure 3.1** An optimal conditional solution table [13]

### 3.2.3 Stochastic CSP

Stochastic constraint satisfaction problem (SCSP) [14,52] is proposed to model uncertain worlds in a similar way as MCSPs. As in MCSPs, variables in SCSPs are
divided into controllable (decision) variables and uncontrollable (stochastic) variables. The main difference between SCSPs and MCSPs is that a probability distribution is associated with the domain of each stochastic variable in SCSPs.

In a one stage SCSP, the decision variables are determined before the stochastic variables are given values. A one stage SCSP is satisfiable if and only if there are values for the decision variables so that, given random values for the stochastic variables, the probability that all the constraints are satisfied equals or exceeds some threshold [52].

For multiple stage SCSPs, a decision policy is a tree with nodes labeled with variables, starting with the first variable labeling the root and ending with the last variable. The decision variables will have only one child for the decision value determined by the decision policy, and the stochastic variables will have one child for each possible output. For a SCSP, the expected value of a policy (satisfaction value) is the sum of objective valuations of each leaf node weighted by their probabilities [52]. The goal is to find an optimal solution with a maximum satisfaction value.

Forward checking has been applied to solve SCSPs [52]. All the variables are ordered at first. Then, values are assigned to these variables one by one. For decision variables, we will try each value in the domain and return the maximum value. For stochastic variables, we will return the weighed sum of all the answers to the sub-problems. The main goal of solving stochastic SCSPs is to make a series of decisions and maximize the degrees of satisfaction.
3.2.4 STP with uncertainty

Temporal problems, as general CSPs, are often emerging in uncertain environments. In the real world, there are many uncontrollable factors in TCSPs. In the Simple Temporal Problem with uncertainty (STPU) [49], there are uncertain duration of events and uncertain temporal constraints. The usual notion of consistency is replaced by the notion of controllability [49].

There are three levels of controllability - strong, weak and dynamic. A strongly controllable STPU has a fully robust solution, a dynamically controllable STPU has a flexible solution, and a weakly controllable STPU has a contingent solution. In other words, strong controllability suits cases where the situation is totally unknown; weak controllability suits cases where the situation is totally known after the decisions are made; dynamic controllability suits cases where the situation is only partially known. It has been proved that the most important of these controllability levels, dynamic controllability, is tractable in polynomial time. Furthermore, the relationship among these three kinds of controllability is listed as follows [49].

\[
\text{strong controllability} \Rightarrow \text{dynamic controllability} \Rightarrow \text{weak controllability}
\]

3.2.5 Dynamic CSP, Conditional CSP, and Open CSP

Dynamic constraint satisfaction problem (DCSP) [27] is a sequence of CSPs, each one resulting from some changes of the previous one. These changes are primarily due to
uncertain environments. Variables, domains and constraints in a CSP may change from time to time. More formally, a DCSP is a sequence \{P_0, P_1, ..., P_n\} where \(P_i\) is a CSP \((1 < i < n)\) and \(C_{ai}\) is a set of added constraints, \(C_{ri}\) is a set of removed constraints, such that \(C_i \subseteq P_i\) and \(P_i = P_{i-1} + C_{ai} - C_{ri}\).

Conditional CSP (CCSP) [41] is a special case of the DCSP. In CCSPs, the set of constraints is divided into compatibility constraints and activity constraints. Compatibility constraints are classical constraints. Activity constraints define the conditions of activation of the some optional variables. CCSPs can be considered as a particular case of DCSPs where all the possible changes are defined by activity constraints.

The Open Constraint Satisfaction Problem (OCSP) [11] and Interactive Constraint Satisfaction Problem (ICSP) [23] are proposed to handle problems in an open environment. In OCSPs, the allowed values in domains, as well as the constraints, are not completely known at the beginning. Solving OCSPs or ICSPs involves obtaining values for all the variables one by one. These values are acquired online when no solution has been found with the currently known domain values and constraints. The acquisition process stops when a solution is found.

The main method to solve the DCSPs is solution reuse techniques. The basic idea of solution reuse is to develop a solution for a new CSP from the solution for the original CSP. Some solution reuse method such as tree search-based method [20], local
search-based method [26], solution perturbation-based method [15] and variables
un-assignment and reassignment-based methods [47] can be applied.

3.3 Conclusion

In this chapter, I examined the motivation for handling uncertainty in CSPs and TCSPs.
Several frameworks dealing with uncertainty were discussed. In PCSPs, each uncertain
constraint is associated with a probability to be existent in the real world. In MCSPs,
uncertain domains of uncontrollable variables are taken in consideration. In SCSPs,
uncontrollable variables are given a specified probability distribution for their possible
values. In STPU, uncertainty is focused on linear time constraints. In DCSPs and OCSPs,
solution reuse is emphasized in order to deal with changing problem specifications. By
comparing these frameworks, I found that the common goal of them is to find a robust
solution in real uncertain and dynamic environments. In the next three chapters, I will
propose new models for representing and solving uncertainty in symbolic TCSPs,
numeric TCSPs and CCTCSPs.
4. SYMBOLIC TCSPs WITH UNCERTAINTY

In this chapter, I will discuss how to handle symbolic TCSPs with uncertainty. First, I will propose a framework for representing uncertainty in the IA network. Then, I will define the robustness criteria for comparing solutions. At the end, an algorithm for searching the most robust solution in the IA network with uncertainty will be proposed.

4.1 Uncertainty in Symbolic TCSPs

The IA network is the most used symbolic TCSP model. It consists of a tuple \( <E, R> \) where \( E \) is a set of events and \( R \) is a set of binary constraints between events. A relation \( r \in R \) represents the relative position between two events and is expressed by the disjunction of Allen primitives [2]. For instance, the relation \( R = M \lor O \) between two events \( e_1 \) and \( e_2 \) represents the fact that \( e_1 \) either meets or overlaps the event \( e_2 \). One particular relation called universal relation (or identity relation) and denoted by \( I \) corresponds to the disjunction of thirteen primitives. This relation expresses the fact that there is no constraint between the two involved events. Solving the IA network consists of assigning to each disjunctive relation one of its primitives such that all the relations are consistent together. The consistency can be verified by a path consistency check algorithm that I will discuss in section 4.5.1.
In PCSP [12], constraints can be certain or uncertain. Each uncertain constraint is associated with a probability of its existence. For instance, \( Pr(C) = 0.7 \) means that the constraint \( C \) has a 70\% chance to exist. In the case of the IA network, the uncertainty is associated with every Allen primitive within the given uncertain constraints, rather than the constraints themselves. More formally, a disjunctive relation \( R \) is defined by the disjunction \( r_1 \lor r_2 \lor \ldots \lor r_n \), and we define the probability of \( r_i \) as follows.

- \( Pr(r_i \in P_{\text{real}}) = p_i \): the probability that \( r_i \) exists within \( R \)
- \( Pr(r_i \notin P_{\text{real}}) = 1 - p_i \): the probability that \( r_i \) does not exist in \( R \)

For example, there are two tasks \( S_1 \) and \( S_2 \) and two processors \( P_1 \) and \( P_2 \). \( S_1 \) and \( S_2 \) take the same processing time. \( P_1 \) must handle \( S_1 \) first then \( S_2 \). \( P_2 \) must handle \( S_1 \) and \( S_2 \) only in a parallel way. If only \( P_1 \) works well, the constraint between \( S_1 \) and \( S_2 \) is meet. If only \( P_2 \) works well, the constraint between \( S_1 \) and \( S_2 \) is equal. If both \( P_1 \) and \( P_2 \) work well, the constraint is \( \text{meet} \lor \text{equal} \) (See Figure 4.1).

![Figure 4.1 An example of an IA network with uncertainty](image-url)
4.2 Probabilities of Uncertain Constraints

The assumption for our uncertain IA network is that the probabilities of basic primitives in constraints are independent of each other. This assumption is due to the fact that the external uncertain environmental factors are usually independent of each other, such as the working conditions of $P_1$ and $P_2$ in the above example. With this assumption of independence, the probabilities of possible constraints can be calculated by the joint probability formula.

Note that we can compute the probability of a temporal constraint from the probabilities of its primitives. We will consider here the case where $C$ involves one or two primitives. The probability of a constraint with more than two primitives can be computed in the same manner. Here we simply use $Pr(r_i)$ to represent $Pr(r_i \in P_{real})$

Example 4.1 One certain primitive

There is only one Allen primitive meet ($M$) in the disjunctive constraint, and this primitive is certain in the real world. More formally,

$$C = M. \Pr(C) = \Pr(M) = 1.0$$

The solution to this problem must satisfy the relationship $M$. The probability of the universal relation $I$ corresponds to the probability that $C$ does not exist. $\Pr(C) = \Pr(C \neq I)$. It is thus computed as follows.

- $\Pr(C = M) = 1.0$
• \( Pr (C = I) = 1 - Pr (C) = 0. \)

**Example 4.2 Only one uncertain primitive**

There is only one possible primitive \( M \) in \( C \), and this primitive exist in the real world with a given probability. \( C = M. \) \( Pr(M) = 0.8. \) There are two possible situations.

• \( Pr (C = M) = 0.8 \)

• \( Pr (C = I) = 1 - Pr (C) = 0.2 \)

**Example 4.3 One certain primitive and one uncertain primitive**

There are two primitives \( meet (M) \) and \( start (S) \) in the disjunctive constraint, and the probability of one of the primitives is 1.0. \( C = M \lor S. \) \( Pr (M) = 1.0, \) \( Pr (S) = 0.8. \) There are three possible situations.

• \( Pr (C = M) = Pr (M) \times (1 - Pr (S)) = 1.0 \times 0.2 = 0.2 \)

• \( Pr (C = M \lor S) = 1.0 \times 0.8 = 0.8 \)

• \( Pr (C = I) = 1 - Pr (C) = 1 - Pr (C = M) - Pr (C = M \lor S) = 0 \)

**Example 4.4 Two uncertain primitives**

There are two primitives \( M \) and \( S \) in the disjunctive constraint, and the probability of each primitive is less than 1. \( C = M \lor S. \) \( Pr (M) = 0.6, \) \( Pr (S) = 0.8. \) There are four possible situations.

• \( Pr (C = M) = Pr (M) \times (1 - Pr (S)) = 0.6 \times 0.2 = 0.12 \)

• \( Pr (C = S) = (1 - Pr (M)) \times Pr (S) = 0.4 \times 0.8 = 0.32 \)

• \( Pr (C = M \lor S) = Pr (M) \times Pr (S) = 0.6 \times 0.8 = 0.48 \)
Pr (C = I) = 1 - Pr (C) = 1 - (Pr (C = M) - Pr (C = S) - Pr (C = M V S))

= 1 - 0.92 = 0.08

As we increase the number of uncertain basic primitives in one constraint, the number of situations will also be increased. In all cases, the sum of probabilities of different situations must always be 1.

4.3 Probable Worlds

A probable world of a given IA network P corresponds to replacing each uncertain constraint of P with one of its possible situations.

Example 4.5 Probable world

Let us consider an IA network with two uncertain constraints \( C_1 \) and \( C_2 \), and two certain constraints \( C_3 \) and \( C_4 \).

- \( C_1 = M \lor S \) where \( Pr (M) = 0.2 \) and \( Pr (S) = 0.5 \)
- \( C_2 = O \lor F \) where \( Pr (O) = 0.3 \) and \( Pr (S) = 0.6 \)
- \( C_3 = P \)
- \( C_4 = P - \lor O - \)

Both \( C_1 \) and \( C_2 \) have four possible situations (see the above example 4.4). One probable world can be defined, for example, by the following set of constraints \{M, O \lor F, P, P - \lor O - \}. The total number of probable worlds will be number of combinations of all
the constraints. In the above example 4.5, this number is $4 \times 4 = 16$. In general, the total number of probable worlds can be computed as follows.

$$\prod_{C_i \in \text{proble}m} \text{dim}(C_i)$$

where $\text{dim}(C_i)$ is the number of different situations of $C_i$

Every probable world represents one combination of possible constraints. In classical symbolic TCSPs, there is only one certain world which consists of all the certain constraints. However, in symbolic TCSPs with uncertainty, there is more than one probable world.

### 4.4 Robustness of Solutions

Since there is more than one probable world in symbolic TCSPs with uncertainty, there are many solutions that satisfy various probable worlds. For example, if we have three probable worlds $W_1$, $W_2$, $W_3$. $W_1$ has two solutions $\{S_1, S_2\}$, $W_2$ has three solutions $\{S_3, S_4, S_5\}$, and $W_3$ has only one solution $\{S_6\}$. Let $S$ be the set of solutions to all the possible worlds. $S = \{S_1, S_2, S_3, S_4, S_5, S_6\}$. In classical TCSPs, all of these solutions are treated equally, i.e., none of these solutions is associated with a higher priority than any other. However, in symbolic TCSPs with uncertainty, these solutions are different in terms of their robustness. The goal of solving symbolic TCSPs with uncertainty is to find the most robust solution, which is a problem of optimization.

We define the degree of robustness of one specific solution as the probability of this
solution to satisfy all the constraints in the actual world. More formally, it can be defined as follows.

\[ \text{Robustness}(S) = \sum_{W \in W_S} \Pr(W) \text{ where } W_S = \{ \text{probable worlds} \mid S \text{ satisfies } W \} \]

\( W_S \) is called the support world set of solution \( S \)

The probability of each probable world can be computed by the product of each constraint in the IA network, assuming that each constraint is independent of each other. More formally, it can be computed as follows.

\[ \Pr(W) = \prod_{C_j \in W} \Pr(C_j) \text{ where } C_j \text{ represent uncertain constraints in the probable world } W \]

The most robust solution to a given problem is the solution with the highest degree of robustness. In other words, this solution has the highest probability to satisfy the uncertain problem.

### 4.5 Finding the Most Robust Solution

#### 4.5.1 Path consistency check

For the IA network, a solution consists of one assigned primitives for each disjunctive constraint. (See section 2.2) Path consistency check \([1,2,4,7,24,29,46]\) is the sufficient and necessary condition for these assigned Allen primitives to be consistent in the IA network. The general idea behind path consistency check is as follows. Choose any three vertices \( i, j \) and \( k \) in a constraint network, the labels on the edge \((i, j)\) and \((j, k)\) potentially
constrain the label on the edge \((i, k)\) that completes the triangle [46]. Consider the following temporal constraint graph. (See Figure 4.2)

![Figure 4.2 An example of the path-consistency check](image)

In Figure 4.2, there are three events \(A, B\) and \(C\). The constraint between \(A\) and \(B\) is \textit{during} \(\lor\) \textit{equal}. The constraint between \(A\) and \(C\) is \textit{start} and the constraint between \(C\) and \(B\) is \textit{finish}. Considering the \textit{start} constraint between event \(A\) and event \(C\) and the \textit{finish} constraint between \(C\) and \(B\), the only possible constraint between \(A\) and \(B\) is \textit{during}. (See Figure 4.3)

![Figure 4.3 The temporal relations among three events](image)

The following Figure 4.4 is the pseudo code for path consistency check [25,31].
Function DPC (listOfHardConstraints)

1. PC ← false
2. L ← {(i, j)| 1 ≤ i < j ≤ n } in listOfHardConstraints
3. while (L is not empty) do
4. select and delete an (i, j) from L
5. for k ← 1 to n, k # i and k # j do
6. t ← C_{ik} C_{ij} C_{jk}
7. if (t ≠ C_{ik}) then
8. C_{ik} ← t
9. C_{ki} ← INVERSE(t)
10. L ← L ∪ {(i, k)}
11. t ← C_{kj} \cap C_{ki} \cdot C_{ij}
12. if (t ≠ C_{kj}) then
13. C_{kj} ← t
14. C_{jk} ← INVERSE(t)
15. L ← L ∪ {(k, j)}

Figure 4.4 The algorithm of path consistency check

In path consistency check, INVERSE, JOIN (\cap) and COMPOSE (\cdot) are the three major
operations. We use a bit vector to represent the disjunctive constraint between two events, where each bit represents one basic Allen primitive. If one Allen primitive is in the constraint, its corresponding bit will be 1, otherwise it will be 0. Thus, the JOIN ($\cap$) operation of two constraints is simply a bitwise AND operation.

For constraint with only one primitive, the process of COMPOSE operation is illustrated above (See Figure 4.4) and the composition of basic Allen primitives is listed in Figure 4.5 [2].
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;</td>
<td>&gt;</td>
<td>d</td>
<td>di</td>
<td>o</td>
<td>oi</td>
<td>m</td>
<td>mi</td>
<td>s</td>
</tr>
<tr>
<td>&lt;</td>
<td>no info</td>
<td>&lt; o m d s</td>
<td>&lt;</td>
<td>&lt; o m d s</td>
<td>&lt;</td>
<td>&lt; o m d s</td>
<td>&lt;</td>
<td>&lt; o m d s</td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td>no info</td>
<td>&gt; o m d s</td>
<td>&gt;</td>
<td>&gt; o m d s</td>
<td>&gt;</td>
<td>&gt; o m d s</td>
<td>&gt;</td>
<td>&gt; o m d s</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>d</td>
<td>no info</td>
<td>&lt; o m d s</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>o m d f</td>
<td>&gt;</td>
<td>d</td>
<td>&gt;</td>
</tr>
</tbody>
</table>

Figure 4.5  The composition table for Allen primitives [2]
For disjunctive constraints, the result of the \textit{COMPOSE} operation is the disjunction of each primitives in both constraints. We built and saved a table of all the possible composition, which is a $2^{13} \times 2^{13}$ matrix. The algorithm for generating the table is shown below. (See Figure 4.6)

\begin{verbatim}
Function DisjunctiveCompose (Constraint C_1, Constraint C_2)
1. C = true;
2. for each primitive r_1 in C_1 do
3. for each primitive r_j in C_2 do
4. C = C \cup (r_1 . r_j)
5. return C
\end{verbatim}

Figure 4.6 The algorithm of generating composition of disjunctive constraints

For the \textit{INVERSE} operation, a table of inverse mapping was build and stored in a file. For a single primitive constraint such as \textit{meet} (m), the reverse relation is simply \textit{met_by} (m-). For a disjunctive constraint such as \textit{meet\_start} (m\_s), the reverse constraint is \textit{met\_by\_started\_by} (m\_s-). For thirteen Allen primitives, there are $2^{13}=8192$ possible disjunctive constraints between two events. Thus, there are 8192 entries in the reversion table.

Every time the program is initialized, the \textit{REVERSE} table, the \textit{JOIN} table and the \textit{COMPOSE} table will be loaded. In the process of path consistency check, the result of
4.5.2 Pre-processing the problem

In symbolic TCSPs with uncertainty, some constraints are certain. These certain constraints are called hard constraints. Before we try to solve the problem, path consistency check can be applied to these hard constraints. There are two effects of path consistency check in the pre-processing phase.

Firstly, if path consistency check fails for the hard constraints, there is no solution for each probable world. Thus, we should stop the search and return a result of no solution.

Secondly, by running path consistency check, we can eliminate some inconsistent primitives in the original problem, which will reduce the number of probable worlds. For example, after path consistency check, one previously disjunctive uncertain constraint now contains only one basic primitive. Then all the probable worlds that do not contain that primitive will be inconsistent. Therefore, we can save time by ignoring these inconsistent worlds since there are no solutions for inconsistent worlds.

4.5.3 Baseline algorithm

Using the definition of robustness and probabilities in previous sections, a baseline
algorithm can be intuitively constructed as follows.

1. Get all the probable worlds by enumerating possible combination of uncertain constraints.
2. For each probable world, search for one solution. These solutions form a solution space of solutions.
3. Calculate the robustness of the specific solution by summing up the probabilities of worlds it satisfies.
4. Choose the solution with the highest degree of robustness.

The main problem of the baseline algorithm is efficiency. Probable worlds and solutions may overlap each other. And this interrelationship is not considered in the baseline algorithm. Thus, we can make improvements to the baseline algorithm by using this interrelationship in the following section.

4.5.4 Minimal set of probable worlds

In the uncertain IA network, many probable worlds can be generated by different possible combinations of primitives. The baseline algorithm does not consider the interrelationship between these worlds. However, the interrelationship between worlds can be used to improve the efficiency of the baseline algorithm.

Let us consider two probable worlds $W_A$ and $W_B$. Assume that all the constraints of both probable worlds are identical except one uncertain constraint $C$ between two given
events \( e_i \) and \( e_j \). \( C = M \lor S \) belongs to the world \( W_A \) and \( C = M \) belongs to world \( W_B \). It is obvious that every solution to world \( W_B \) is also a solution to world \( W_A \). More formally, if \( S_A \) is the set of solutions of world \( W_A \) and \( S_B \) is the set of solutions of world \( W_B \), we can deduce \( S_A \subseteq S_B \).

The solution set of probable worlds overlap each other to a large extent. The final solution for a TCSP must be a specific assignment of one primitive in each disjunctive constraint. Consider the following minimal set of probable worlds.

**Example 4.6 Minimal set of probable worlds**

An uncertain IA network consists of two uncertain constraints \( \{C_1, C_2\} \) and two certain constraints \( \{C_3, C_4\} \).

- \( C_1 = M \lor S \) where \( Pr (M) = 0.2 \) and \( Pr (S) = 0.5 \)
- \( C_2 = O \lor F \) where \( Pr (O) = 0.3 \) and \( Pr (S) = 0.6 \)
- \( C_3 = P \)
- \( C_4 = P - \lor O - \)

Given the above definition, we can generate four probable worlds.

- \( W_1 = \{C_1=M, C_2=O, C_3, C_4\} \)
- \( W_2 = \{C_1=M, C_2=F, C_3, C_4\} \)
- \( W_3 = \{C_1=S, C_2=O, C_3, C_4\} \)
- \( W_4 = \{C_1=S, C_2=F, C_3, C_4\} \)

If \( S \) is a solution, the assigned Allen primitive in \( S \) for each uncertain constraint must
be one of the disjunctive primitives. Since the subset \{W_1, W_2, W_3, W_4\} contains all the possible combinations of primitives, \(S\) must be the solution for one probable world in this set. Therefore, we can search the solution space of this subset instead of the entire set of probable worlds. \{W_1, W_2, W_3, W_4\} is called the minimal set of probable worlds.

More formally, suppose set \(\Psi\) is a subset of probable worlds. The solution set of its member probable worlds in \(\Psi\) is equal to the solution set of all probable worlds. If there is no further subset of \(\Psi\) satisfying this condition, \(\Psi\) is called the minimal set of probable worlds.

4.5.5 Branch and bound algorithm

Solving the IA network requires a backtrack algorithm of exponential time cost \([7,8,17,19,22,36,37,45,50]\). In order to overcome this difficulty in practice, constraint propagation based on path consistency check can be used before and during the search in order to detect later failure early. If path consistency fails then the IA network is not consistent and there is thus no need to proceed with the search. (See section 4.5.1) In the case where path consistency check is successful, the hard constraints of the resulting network will have less Allen primitives.

With the IA network with uncertainty, the goal is to find a solution with the highest degree of robustness. In order to achieve this, we propose the following branch-and-bound algorithm for finding the most robust solution.
1. Perform path consistency to the sub-network containing only the hard constraints.
   If the sub-network is not path consistent return that the IA network is not consistent.

2. For each uncertain constraint, sort its primitives by the decreasing order of their probabilities.

3. Following the forward check principle [19], pick a constraint, assign to it one of its Allen primitives and run the path consistency check algorithm on the sub-network containing the newly assigned constraint and the non assigned ones. If path consistency fails then assign another primitive to the current constraint or backtrack to the previous assigned constraint if the current constraint does not have any available primitives left. If the path consistency check algorithm is successful, select another constraint and redo the same process until all the constraints are assigned in which case we obtain a solution otherwise return that the IA network is not consistent. If a solution is obtained, compute its robustness (the probability of the complete assignment it satisfies) and assign the result to the lower bound ($LB$).

4. The rest of the search space is then systematically explored as follows. Each time the current constraint is assigned a primitive, if this constraint is uncertain then an overestimation of the robustness of any possible solution following this decision is computed and used as an upper bound ($UB$). If $UB < LB$, then the current
uncertain constraint is assigned another value or backtrack to the previous constraint if all the primitives have been explored. The overestimated robustness is the product of the probabilities of all the assigned primitives multiply the product of the max probabilities of the uncertain constraints that are not yet assigned. The max probability of a non assigned uncertain constraint corresponds to the largest probability of its primitives.

The pseudo code is shown in Figure 4.7.

// Problem is the symbolic TCSP with uncertainty
// EvtLst is the list of events
// UCLst is the list of uncertain constraints
// FCLst is the list of fixed constraints
// Graph is the constraint graph of the this TCSP

Function SolveUncertainSymbolicTCSP (Problem, EvtLst, FCLst, UCLst, Graph)

1. curSol = NULL // curSol is the current solution
2. lowerBound = 0
3. upperBound = 0
4. oldGraph = graph
5. if DPC (oldGraph) then // check path consistency
6. return NO_SOLUTION
7. sortPrimitivies(UCLst) // sort all primitives in the UCLst by their probabilities

8. while (notAllConstraintsExplored(Problem)) // if not all scenarios visited

9. pushScenario(oldGraph) // save the current scenario

10. curConstraint = removeOneConstraint(FCLst, UCLst, newGraph)
   // removing one constraint from either FCLst or UCLst, forming the newGraph

11. assignOnePrimitive (curConstraint, curSol)
   // pick one primitive in the constraint

12. if (DPC (newGraph)) = false then

13. PopScenario(newGraph) // backtrack to the previous scenario

14. goto 8

15. if (NumOfPrimitives(curConstraint)) > 0 then // still primitives left

16. assignAnotherPrimitive (curConstraint)

17. else

18. popScenario(newGraph) // no primitives left, backtrack

19. goto 8

20. if (completeSol(curSol)) then // if one solution is found

21. newRobustness = calculateRobustness (curSol)

22. if (newRobustness>lowerBound) then

23. lowerBound= newRobustness // update the upper bound

24. else

25. }

26. else
27. popScenario(newGraph) // backtrack for more solutions
28. goto 8
29. else // the solution is not complete
30. // estimate the maximum robustness
31. upperBound = estimateRobustness (newGraph, curSol)
32. if (upperBound < lowerBound) then
33. popScenario(oldGraph) // backtrack
34. goto 8
35. end while

Figure 4.7 The pseudo code for solving symbolic TCSPs with uncertainty

4.6 Conclusion

In this chapter, I examined the uncertainty in the IA network. Uncertain constraints are used to construct probable worlds, which are possible instances of the real world uncertain problems. I proposed a new method for comparing robustness of solutions in the IA network with uncertainty. I also proposed the concept of minimal set of probable world which covers the solution set of all probable worlds. Path consistency check was applied during the backtrack search of solution for the IA network. At the end, I proposed a new branch-and-bound algorithm to find the most robust solution in the IA network with uncertainty.
5. NUMERIC TCSPs WITH UNCERTAINTY

In this chapter, I will discuss how to handle numeric TCSPs with uncertainty. The TemPro model will be extended for handling uncertainty in the real world. The criteria for comparing robustness of solutions will be given. And the algorithm for finding the most robust solution will be presented at the end.

5.1 TemPro Model

In Chapter 4, I presented a new framework for handling uncertainty in the IA network (symbolic TCSP). In the IA network, only qualitative information of events is provided. However, in real world applications, quantitative information is often provided as well. In 2.3, several methods for representing numeric information are introduced, such as the arithmetic inequations and the TemPro model [31,34]. The TemPro model can process both symbolic and numeric information while arithmetic equations can only process the numeric information. Most of the previous research on solving numeric TCSPs is based on arithmetic equations such as the STP with uncertainty. However, my work is based on the TemPro model in order to be applicable for more problems.

The main difference between numeric TCSPs and symbolic TCSPs is the form of solutions. In symbolic TCSPs such as the IA network, the solution is an assignment of
one primitive for each disjunctive constraint. In numeric TCSPs, however, the solution is the time points for each event, such as the start time and the end time. Whether the model is numeric or symbolic depends on the form of the solution. Thus, the TemPro model is a framework for numeric TCSPs even though the constraints between events are the same disjunctive constraints of Allen primitives as in symbolic TCSPs.

5.2 Uncertainty in the TemPro Model

5.2.1 Uncertainty of domains

In the STPU, the uncertainty is mainly focused on the duration of some uncertain events. In addition to duration, there could also be some other uncertainty for one event, such as the earliest start time (INF) and latest finish time (SUP). In uncertain and dynamic environment, the INF and SUP of an event are affected by some external factors. Consider the following example.

Example 5.1 Uncertain INF and SUP of an event

Everyday Bob takes bus to his company. Occasionally, the bus is full and he has to call a taxi. The bus arrives at the company at 8:30 and the taxi arrives at the company at 8:20. Bob normally finishes his job in 12:00. However, his manager Tom may want to talk with him after 12:00, which will take 60 minutes. Bob doesn't know whether Tom wants to talk with him before 12:00.
In the above example, the earliest start (INF) of event workInOffice depends on which vehicle Bob takes. Suppose 8:00 is the zero point in the time line. If he takes the bus, the INF for workInOffice would be 30. If he takes taxi, the INF for workInOffice would be 20.
The latest finish (SUP) of event workInOffice is also associated with some uncertainty. If Tom doesn’t want to talk with him, the SUP for workInOffice would be 240. Otherwise, the SUP of workInOffice would be 300.

With the uncertainty in domains of events, both the INF and SUP are no longer single time points, but probability distributions of several possible time points. For example, there are three possible earliest start time points 20, 30 and 40 for a single event $E_i$. These earliest start time points are caused by some external factors, as the INF of the event workInOffice in the above example. The priori probability of each earliest start can be computed by statistics or estimated by induction. Assume $E_i$ has a length of 15 and the step is 1 unit. The probabilities of these three INF are listed as follows.

- $Pr \{E_i = [20, SUP, 15, 1]\} = 0.2$
- $Pr \{E_i = [30, SUP, 15, 1]\} = 0.3$
- $Pr \{E_i = [40, SUP, 15, 1]\} = 0.5$

The uncertainty of the latest finish (SUP) can be presented in a similar way. Assume that three possible latest end time points 45, 60 and 80 exist for $E_i$. The probabilities of them can be described in a similar form.

- $Pr \{E_i = [INF, 45, 15, 1]\} = 0.4$
• $Pr \{E_1 = [INF, 60, 15, 1]\} = 0.1$

• $Pr \{E_2 = [INF, 80, 15, 1]\} = 0.5$

With the addition of uncertain $INF$ and $SUP$, the domain of one event is no longer a certain time frame, but rather a dynamic time window. For the previous example of event $E_1$, we have 3 different $INF$ and 3 different $SUP$. Therefore, there are totally 9 (3×3) possible domains.

- [20, 45, 15, 1], [20, 60, 15, 1], [20, 80, 15, 1]
- [30, 45, 15, 1], [30, 60, 15, 1], [30, 80, 15, 1]
- [40, 45, 15, 1], [40, 60, 15, 1], [40, 60, 15, 1]

More formally, the set of domains of event $E$ can be defined as

$$D = [INF_i, SUP_j, DUR, STEP]$$

where $INF_i$ and $SUP_j$ are possible values for $INF$ and $SUP$, respectively

5.2.2 Continuous and discrete probabilities

The earliest start ($INF$) and the latest finish ($SUP$) in a SOPO define the time domain for a specific event. In the previous section, we assumed that the probability distribution for $INF$ and $SUP$ are discrete. The discrete probability distribution is applied to handle uncertainty caused by discrete external factors. For example, in Example 5.1, Full and empty are the only two discrete possible statuses of the bus. The actual status of the bus affects the $INF$ of the event workInOffice.
In some real world applications, however, the determinant external factor may be a
continuous variable. For example, the bus may arrive at the office within a range between
8:25 and 8:35 with a continuous probability distribution, which affects the corresponding
event workInOffice. Since the TemPro model can only handle discrete time points, we
have to discreterize the continuous distribution.

First, the continuous distribution is divided into several areas in its range. The
mid-points of each area are used as the representative values for that area. Then,
cumulative probabilities are calculated of the continuous distribution in each area. Each
cumulative probability will be associated with the corresponding mid-point to generate a
discrete distribution.

5.3 Uncertain Domains and Probable Worlds

5.3.1 Calculation of probabilities of uncertain constraints

Given the probabilities of uncertain INF and SUP, the probability of each possible
domain (SOPO) for an event can be calculated. In the above example, we have two sets
of INF and two sets of SUP. Assume the INF and SUP is caused by independent external
factors, we can calculate the probabilities of the SOPO for $E_1$.

- $Pr (E_1 = [20, 45, 15, 1])$
  
  $= Pr (E_1 = [20, SUP, 15, 1]) \times Pr (E_1 = [INF, 45, 15, 1]) = 0.2 \times 0.4 = 0.08$
\begin{itemize}
  \item \( \Pr(E_{i} = [20, 60, 15, 1]) = \Pr(E_{i} = [20, \text{SUP}, 15, 1]) \times \Pr(E_{i} = [\text{INF}, 60, 15, 1]) = 0.02 \)
  \item \( \Pr(E_{i} = [20, 80, 15, 1]) = \Pr(E_{i} = [20, \text{SUP}, 15, 1]) \times \Pr(E_{i} = [\text{INF}, 80, 15, 1]) = 0.01 \)
  \item \( \Pr(E_{i} = [30, 45, 15, 1]) = \Pr(E_{i} = [30, \text{SUP}, 15, 1]) \times \Pr(E_{i} = [\text{INF}, 45, 15, 1]) = 0.12 \)
  \item \( \Pr(E_{i} = [30, 60, 15, 1]) = \Pr(E_{i} = [30, \text{SUP}, 15, 1]) \times \Pr(E_{i} = [\text{INF}, 60, 15, 1]) = 0.03 \)
  \item \( \Pr(E_{i} = [30, 80, 15, 1]) = \Pr(E_{i} = [30, \text{SUP}, 15, 1]) \times \Pr(E_{i} = [\text{INF}, 80, 15, 1]) = 0.15 \)
  \item \( \Pr(E_{i} = [40, 45, 15, 1]) = \Pr(E_{i} = [40, \text{SUP}, 15, 1]) \times \Pr(E_{i} = [\text{INF}, 45, 15, 1]) = 0.20 \)
  \item \( \Pr(E_{i} = [40, 60, 15, 1]) = \Pr(E_{i} = [40, \text{SUP}, 15, 1]) \times \Pr(E_{i} = [\text{INF}, 60, 15, 1]) = 0.05 \)
  \item \( \Pr(E_{i} = [40, 80, 15, 1]) = \Pr(E_{i} = [40, \text{SUP}, 15, 1]) \times \Pr(E_{i} = [\text{INF}, 80, 15, 1]) = 0.25 \)
\end{itemize}

The sum of these probabilities is equal to 1. More formally, the probability of each SOPO is calculated with the following formula.

\[
\Pr\{E = [\text{INF}_{i}, \text{SUP}_{j}, \text{DUR}, \text{STEP}]\} = \Pr\{\text{INF} = \text{INF}_{i}\} \times \Pr\{\text{SUP} = \text{SUP}_{j}\}
\]

\[
\sum_{i,j} \Pr\{E = [\text{INF}_{i}, \text{SUP}_{j}, \text{DUR}, \text{STEP}]\} = 1.0
\]

5.3.2 Dividing domains into non-overlapping areas

In the above example, there are nine possible domains (SOPOs) for \( E_{i} \). However, these domains are not separately distributed along the time line. Some of them such as \([30, 60, 15, 1]\) and \([40, 80, 15, 1]\) overlap each other.

Actually, with three \( \text{INF} \) and three \( \text{SUP} \) for event \( E_{i} \), we can order them as follows.

\begin{itemize}
  \item \( \text{INF}_{1} < \text{INF}_{2} < \text{INF}_{3} \)
  \item \( \text{SUP}_{1} < \text{SUP}_{2} < \text{SUP}_{3} \)
\end{itemize}
Then we can divide the possible domains of $E_i$ into 5 non-overlapping areas.

- $A_1: [INF_3, SUP_1]$
- $A_2: [INF_2, INF_3 + DUR]$
- $A_3: [INF_1, INF_2 + DUR]$
- $A_4: [SUP_1 - DUR, SUP_2]$
- $A_5: [SUP_2 - DUR, SUP_3]$  (See Figure 5.1)

![Figure 5.1 Dividing domains into non-overlapping areas](image)

By dividing the entire possible domains into non-overlapping areas, a solution contains an event in one area will not be contained in any other areas. Furthermore, these non-overlapping areas cover all the possible assignment of each event in the final solution. Therefore, we can consider the assignment of events in each area.

In order to calculate the probabilities of solutions, these non-overlapping areas must also be not overlapped by the original possible SOPOs. In other words, for each possible $SOPO_i (i=1\sim9)$ in the above example, $A_i (i=1,2,3,4,5)$ must be either included or excluded in $SOPO_i$. Consider the following example of an incorrect way of dividing domains into non-overlapping areas. (See Figure 5.2)
Figure 5.2 An incorrect way of dividing domains into non-overlapping areas

It divides the domains into two non-overlapping areas \([INF_1, SUP_1]\) and \([SUP_1-DUR, SUP_1]\). Since the area \([INF_1, SUP_1]\) overlaps the original possible SOPO \([INF_2, SUP_2]\), this dividing strategy is not suitable for computing the probabilities of solutions.

In the above example, \(SUP (SUP_1, SUP_2, SUP_3)\) are greater than \(INF (INF_1, INF_2, INF_3)\). In some other cases, \(INF\) and \(SUP\) may be mixed on the time line. (See Figure 5.3)

Figure 5.3 Another way of dividing with mixed \(INF\) and \(SUP\)

The dividing algorithm is shown below. (See Figure 5.4)
Function DivideAreas (ListofSOPoS)

1. Sort INF and SUP. \( \text{INF}_1 < \text{INF}_2 < \ldots < \text{INF}_m, \text{SUP}_1 < \text{SUP}_2 < \ldots < \text{SUP}_n \)

2. \( \text{AreaList} \leftarrow \text{empty} \)

3. \( \text{L} \leftarrow \text{empty} \) // L contains minimal areas

4. for each SOPO \( S \) in ListofSOPoS do

5. if no other INF or SUP are in \([\text{INF}_S, \text{SUP}_S]\)

6. \( \text{L} \leftarrow \text{L} \cup \{S\} \)

7. end for

8. \( \text{RemainList} \leftarrow \text{the combinations of} \ [\text{INF}, \text{SUP}] \text{ except} \ [\text{INF}_1, \text{SUP}_n] \)

9. \( \text{CurrentCoverage} \leftarrow \text{empty}; \)

   // \text{CurrentCoverage} is the union of current generated areas

   // e.g. we have two areas [25,26], [22,27], then it will be [22,25]

10. while \( \text{L} \) is not empty do

11. \( \text{NextDomainLength} \leftarrow \text{positive infinity} \)

12. \( \text{NextDomain} \leftarrow \text{empty} \)

13. for each SOPO \( R \) in RemainList do

14. if \( R \geq \text{CurrentCoverage} \text{ and } \text{length}(R) < \text{NextDomainLength} \) then

15. \( \text{NextDomainLength} = \text{length}(R) \)

16. \( \text{NextDomain} = R \)

17. end for
18. delete $R$ from $RemainList$

19. $NewArea = R - CurrentCoverage - S$

20. $L = L \cup NewArea$


22. end while

23. $NewArea = [\text{INF}, \text{SUP}] - CurrentCoverage$

24. if $NewArea$ is not empty then

25. $AreaList = AreaList \cup NewArea$ // adding the last domain

Figure 5.4 The algorithm of dividing domains into non-overlapping areas

5.3.3 Probabilities of areas

In Figure 5.1, we defined five non-overlapping areas. There is one begin time point and one end time point for each area. ($A_i = \{A_i\text{-begin}, A_i\text{-end}\}$) For each of its possible SOPO ($\text{INF}_i, \text{SUP}_j$), if $\text{INF}_i < A_i\text{-begin} < A_i\text{-end} < \text{SUP}_j$, then $A_i \in (\text{INF}_i, \text{SUP}_j)$, which implies that the solution in $A_i$ is also a solution in ($\text{INF}_i, \text{SUP}_j$). Since $A_i$ must be contained in either one or more SOPO, we can use the following formula to calculate the probability of area $A_i$ to be included in the actual domain for the corresponding uncertain event.

$$Pr (A_i \text{ is included in the actual domain}) = \sum Pr (\text{INF}_i, \text{SUP}_j) \text{ where } A_i \in (\text{INF}_i, \text{SUP}_j)$$

For example, $A_i$ is the subset of nine possible SOPO (see Figure 5.1), so the probability
of \( A_1 \) to be included in the actual domain is 1.0. In other words, if we can find a solution in \( A_1 \), the probability of this solution to be falling in the actual domain will be 1.0. \( A_2 \) is the subset of \([INF_2, SUP_1], [INF_2, SUP_2] \) and \([INF_2, SUP_3] \). So the probability of \( A_2 \) to be included in the actual domain is 0.12 + 0.03 + 0.15 = 0.30.

### 5.3.4 Probable worlds

Given the possible domains of each event, we can define probable worlds in numeric TCSPs as combinations of domains for each event. For events with certain \( INF \) and \( SUP \), the domains are just the given \( INF \) and \( SUP \). For events with uncertain \( INF \) and \( SUP \), however, one of the possible domains will be picked up.

\[
Probable\ World = \{D_1, D_2, \ldots, D_n, \{D_{\text{certain}}\}, \{C\}\}
\]

where \( C \) is the constraint and \( D_i \) is the specific domain of each uncertain event.

Consider the following example.

#### Example 5.2 Probable world in the TemPro model

We have three events \( E_1, E_2 \) and \( E_3 \). \( E_1 \) and \( E_2 \) are traditional events with certain \( INF \) and \( SUP \). \( E_3 \) and \( E_4 \) are events with uncertain \( INF \) and \( SUP \). The SOPO of them are given below.

- \( E_1 = [20, 60, 15, 1] \)
- \( E_2 = [30, 70, 25, 1] \)
- \( E_3 = [INF_{E_3}, SUP_{E_3}, 40, 1] \)
\[ Pr (INF_{E3} = 10) = 0.4, Pr (INF_{E3} = 20) = 0.6, \]
\[ Pr (SUP_{E3} = 100) = 0.3, Pr (SUP_{E3} = 120) = 0.7 \]

- \( E4 = [45, SUP_{E4}, 35,1] \)

\[ Pr (SUP_{E4} = 90) = 0.2, Pr (SUP_{E4} = 130) = 0.8 \]

Here we use the non-overlapping areas generated by the dividing algorithm in 5.3.2 to build probable worlds. The total number of probable worlds is the product of the number of generated non-overlapping areas of each uncertain constraint. For example, in the above Example 5.2, we have three areas for \( E3 \) and two areas for \( E4 \). So the total number of probable worlds is \( 3 \times 2 = 6 \). The set of these six probable worlds is called set of basic probable world. More formally, suppose the number of certain events is \( m \) while the number of uncertain events is \( n \).

\[
\text{Set of basic probable world} = \{[INF, SUP]_i, Area_j \mid 1 < i < m, 1 < j < n \}
\]

\([INF, SUP]_i \) is one SOPO of certain events and \( Area_j \) is one area of uncertain events.

One probable world \( WP \) in the set of basic probable world is \( \{E_1, E_2, E_3 = [10, 60], E_4 = [55, 80]\} \). In this world, we use the area \( Area_{E3} [10, 60] \) for \( E_3 \) and the area \( Area_{E4} \) for \( E_4 \). Their probabilities are listed below. (See 5.3.3 for the formula of calculation)

\[ Pr (Area_{E3} \text{ is included in the actual domain of } E_3) = 0.4 \]

\[ Pr (Area_{E4} \text{ is included in the actual domain of } E_4) = 0.8 \]

Then the probability of \( WP \) containing the real domains of \( E_3 \) and \( E_4 \) is

\[ Pr (WP \text{ is included in the actual domains of both } E_3 \text{ and } E_4) \]
= \Pr(\text{Area}_{E_3} \text{ is included in the actual domain of } E_3) \\
\times \Pr(\text{Area}_{E_4} \text{ is included in the actual domain of } E_4) \\
= 0.4 \times 0.8 = 0.32

Therefore, if we find a solution Sol to this probable world WP, the probability of Sol to fall in the real domain of E_3 and E_4 will be 0.32. Since the actual problem is determined by uncertain events, 0.32 can also be viewed as the probability of Sol to satisfy every constraint in the real world.

5.4 Robustness of Solutions

Given the set of basic probable worlds, our goal is to find the most robust solution. We define the robustness of a solution as the probability of this solution to satisfy every constraint in the real world. This probability can be illustrated as follows. Suppose we have a numeric TCSP P with uncertainty. If P happens n times and one of the solutions satisfies the problem in 0.6 \times n times, the probability of P satisfying the real world is 0.6.

By using non-overlapping areas to generate probable worlds, the solutions for each probable world do not overlap each other. The solution to one probable world is not the solution for any other probable worlds. So we can consider the solution to each probable world and calculate the robustness of each solution. More formally, it can be calculated in the following formula.
Robustness of $s = Pr (W_i$ is included in the actual domains for all the uncertain events)$

where $s$ satisfies $W_i$ and $W_i$ is in the set of probable worlds

For example, if we have four different worlds $\{W_1, W_2, W_3, W_4\}$ and four corresponding solutions $\{S_1, S_2, S_3, S_4\}$. The probability of each world is listed below.

- $Pr (W_1$ is included in the actual domain for all the uncertain constraints) = 0.6
- $Pr (W_2$ is included in the actual domain for all the uncertain constraints) = 0.5
- $Pr (W_3$ is included in the actual domain for all the uncertain constraints) = 0.8
- $Pr (W_4$ is included in the actual domain for all the uncertain constraints) = 0.2

The robustness of $S_1, S_2, S_3, S_4$ is 0.6, 0.5, 0.8, 0.2, respectively. $S_3$ is the most robust solution due to its highest probability to be the solution for the real problem. The main task of solving numeric TCSPs with uncertain events is to find the solution with the highest degree of robustness.

5.5 Solving the Problem

5.5.1 Removing inconsistent areas

In 5.3.2 we discussed the algorithm for dividing domains into non-overlapping areas. However, we didn’t consider the validity of all the generated areas. According to the values of the lower boundary and the upper boundary, there may be some inconsistent areas. For example, we have an area of an uncertain event $E_1$ $[A_1$-begin, $A_1$-end] with a
probability of 0.8 to be included in the actual domain. The length of the event is specified by $DURATION$. If $A_t\text{-end} - A_t\text{-begin} < DURATION$, there will be no solutions in this domain even if it is associated with a probability of 0.8. Therefore, all the probable worlds that contain this area for event $E_i$ should be removed from the set of basic probable worlds. By removing inconsistent areas, the number of corresponding probable worlds can be reduced exponentially, which will improve the efficiency of the solving algorithm.

5.5.2 Numeric to symbolic conversion

The TemPro model handles both symbolic information and numeric information. The general idea of the conversion is to use numeric information to remove inconsistent Allen primitives in the disjunctive constraints. In the following example (See Figure 5.5 [32]), the original constraint between John and Mary is $ESS\text{-}M$. However, their domains imply that $E$, $S$ and $S'$ are not possible.

![Figure 5.5 The numeric to symbolic conversion [32]](image-url)
The algorithm for numeric to symbolic conversion is given in [31]. (See Figure 5.6)

**Function** \( \text{Num2Symb} (e_i, e_j) \)

1. if \( \text{INF}_i > \text{SUP}_j \) then \( r_{ij} \leftarrow \text{P} \)
2. if \( \text{SUP}_j < \text{INF}_j \) then \( r_{ij} \leftarrow \text{P} \)
3. if \( \text{DUR}_i < \text{DUR}_j \) then remove \( \{E, S-, F-, D-\} \) from \( r_{ij} \)
4. if \( \text{DUR}_i > \text{DUR}_j \) then remove \( \{E, S, F, D\} \) from \( r_{ij} \)
5. if \( \text{DUR}_i = \text{DUR}_j \) then remove \( \{D, D-, S, S-, F, F-\} \) from \( r_{ij} \)
6. if \( \text{INF}_i + \text{DUR}_i > \text{SUP}_j - \text{DUR}_j \) then remove \( \{M, B\} \) from \( r_{ij} \)
7. if \( \text{SUP}_j - \text{DUR}_j < \text{INF}_j + \text{DUR}_i \) then remove \( \{M-, B-\} \) from \( r_{ij} \)
8. if \( \text{INF}_j > \text{SUP}_j - \text{DUR}_j \) then remove \( \{E, B, M, S, S-, O, D-\} \) from \( r_{ij} \)
9. if \( \text{INF}_j + \text{DUR}_j > \text{SUP}_j \) then remove \( \{E, B, M, F, F-, D\} \) from \( r_{ij} \)
10. if \( \text{SUP}_j < \text{INF}_j + \text{DUR}_j \) then remove \( \{F, F-\} \) from \( r_{ij} \)
11. if \( \text{SUP}_j - \text{DUR}_i < \text{INF}_j \) then remove \( \{S, S-, E\} \) from \( r_{ij} \)

Figure 5.6 The algorithm of the numeric to symbolic conversion

The conditions for removing inconsistent symbolic primitives (See Figure 5.6) are only the necessary conditions, but not the sufficient conditions. The cost of removing all inconsistent primitives is exponential. Thus, this algorithm only removes some of the obvious inconsistent primitives. However, even removing some inconsistent primitives
will substantially decrease the time complexity of the solving algorithm. The conversion algorithm will be used in the search for the most robust solution, which we will discuss in section 5.5.4.

5.5.3 Arc consistency check and path consistency check

In 5.5.2, we used numeric information to remove inconsistent symbolic relations. Similarly, the symbolic information can be used to remove inconsistent domains of events. This process is called arc-consistency check. (See Figure 5.7 [32])

In Figure 5.7, the constraint between John and Wendy is $S$ (start). For the domain $[0, 30]$ for event John, there is no possible interval in the domain of Wendy that satisfies the constraint. So $[0, 30]$ is removed from the domain of John. Some other intervals are also removed from the domain for the same reason.

![Figure 5.7 An example of the arc-consistency check [32]](image)

63
Several algorithms for arc-consistency check have been proposed [5,6,28,51]. The most used AC-3 arc-consistency check algorithm in TCSPs is listed below. (See Figure 5.8 [31])

**Function AC 3.1 ()**

// sopo: an array of SOPOs

1. \( Q \leftarrow \{(i, j) \mid (i, j) \in R\} \)
2. \( AC \leftarrow \text{true} \)
3. While \( Q \neq \text{Nil} \) Do
4. \( Q \leftarrow Q \setminus \{(x, y)\} \)
5. If REVISE\((x, y)\) then
6. If \( \text{Dom}(x) \neq \emptyset \) then
7. \( Q \leftarrow Q \cup \{(k, x) \mid (k, x) \in R \land k \neq y\} \)
8. else
9. return \( AC \leftarrow \text{false} \)

**Function REVISE\((x, y)\)**

// compatible: checks if two intervals are compatible

// regarding the symbolic relation they share

1. \( \text{REVISE} \leftarrow \text{false} \)
2. For each interval $a \in sopo[x]$ Do

3. If $\neg$-compatible $(a, b)$ for each interval $b \in sopo[y]$ then

4. remove $a$ from $sopo[x]$

5. REVISE $\leftarrow$ true

Figure 5.8 The algorithm of the arc-consistency check [31]

The constraints in the TemPro model are disjunctive constraints of Allen primitives, which are the same as in the IA network. Thus, path consistency check can be used to remove inconsistent relations, as in symbolic TCSPs with uncertainty. (See 4.5.1)

5.5.4 Branch and bound algorithm

The intuitive method is sorting all the probable worlds by their probabilities. Then we try to find a solution to the most probable world. If a solution is found, it will be the most robust solution. If no solution is found, we switch to the next most probable world. A more effective algorithm is to use branch-and-bound through the search for the most robust solution. The framework of the algorithm is shown below.

1. Perform path consistency check for the symbolic constraints in the problem. (See Figure 4.4) If the problem is inconsistent, return $NO\_SOLUTION$.

2. For all the constraints involving certain events, perform the numeric to symbolic conversion (See Figure 5.5) to remove inconsistent Allen primitives.

3. Divide the uncertain domains of events into non-overlapping areas as in 5.3.2.
For each event with uncertain domains, maintain a list of areas $L = \{A_1, A_2...A_m\}$.

Set the lower bound $LB = 0$.

4. Remove those areas inconsistent with the problem, i.e., $Area_{end} - Area_{begin} < DURATION$.

5. Calculate the probability of each area to be included in the actual domain of each uncertain event.

6. Pick one uncertain event. For one uncertain event, select an area from $L$. An overestimation of the robustness of any possible solution falling in the current assigned areas is calculated and used as the upper bound $UB$ (see below). If $UB < LB$, select another area for the current uncertain event or backtrack to the previous uncertain event if all the areas for this event have been explored.

7. Perform arc-consistency check (See 5.5.3) for current uncertain events with assigned domains. If the AC3.1 check fails, select another area for the current uncertain event or backtrack to the previous uncertain event if all the areas for this event have been explored.

8. For all the constraints related to this uncertain event, perform numeric to symbolic conversion by using the currently assigned area.

9. Repeat 6 ~ 8 until all the uncertain events have been assigned areas.

10. If every uncertain event has been assigned a specific area, try to find a solution for the problem. Calculate the robustness of the solution. If it is larger than $LB$, 

66
update LB with the value. If there is no solution or the probability is less than LB, backtrack and go to step 4.

11. If all the areas of uncertain events have been tried and no solution is found, return NO_SOLUTION.

The overestimation of the robustness is the product of probabilities of already assigned areas and the maximum probabilities of areas for all the non-assigned uncertain events.

For example, there are four uncertain events \{E_1, E_2, E_3, E_4\} and there are two assigned areas, \(A_1\) for \(E_1\), \(A_2\) for \(E_2\). The probabilities of \(A_1\) and \(A_2\) are

\[
\begin{align*}
\text{Pr} (A_1 \text{ is included in the actual domain of } E_1) &= 0.8 \\
\text{Pr} (A_2 \text{ is included in the actual domain of } E_2) &= 0.6
\end{align*}
\]

And we have three areas for event \(E_3\) and two areas for event \(E_4\). For event \(E_3\),

\[
\begin{align*}
\text{Pr} (A_{3,1} \text{ is include in the actual domain of } E_3) &= 0.3 \\
\text{Pr} (A_{3,2} \text{ is include in the actual domain of } E_3) &= 0.2 \\
\text{Pr} (A_{3,3} \text{ is include in the actual domain of } E_3) &= 0.9
\end{align*}
\]

For event \(E_4\),

\[
\begin{align*}
\text{Pr} (A_{4,1} \text{ is include in the actual domain of } E_4) &= 0.5 \\
\text{Pr} (A_{4,2} \text{ is include in the actual domain of } E_4) &= 0.8
\end{align*}
\]

We use \(\text{Pr}(A_i)\) here to simplify \(\text{Pr}(A_i \text{ is included in the actual domain})\). The overestimation of the robustness of solutions in any probable worlds that contain \(A_1\) and \(A_2\) can be calculated as follows.
\[ UB = Pr(A_1) \times Pr(A_2) \times \text{max}(Pr(A_3.1), Pr(A_3.2), Pr(A_{3.12})) \times \text{max}(Pr(A_{4.1}), Pr(A_{4.2})) \]
\[ = 0.8 \times 0.6 \times 0.9 \times 0.8 = 0.3456 \]

The pseudo code is shown in Figure 5.9.

// Problem is the numeric TCSP with uncertainty, FCEvtLst is the list of fixed events
// UCEvtLst is the list of uncertain events, Graph is the constraint graph of the this TCSP

**Function** SolveUncertainNumericTCSP (Problem, FCEvtLst, UCEvtLst, Graph)

1. \( \text{curSol} = \text{NULL} \) // \( \text{curSol} \) is the current solution
2. \( \text{lowerBound} = 0 \)
3. \( \text{upperBound} = 0 \)
4. \( \text{oldGraph} = \text{graph} \)
5. if DPC (oldGraph) then// check path consistency
6. return NO_SOLUTION
7. for each \( e_i \) in FCEvtLst
8. for each \( e_j \) in FCEvtLst
9. \( \text{Num2Sym}(e_i, e_j) \) // perform the numeric to symbolic conversion
10. end for
11. \( \text{areaList} = \text{divideAreas}(FCEvtLst) \) // divide uncertain events into non-overlapping areas
12. for each area in areaList
13. if \( \text{SUP}(area) - \text{INF}(area) < \text{DURATION}(area) \) then
14. remove(area) // remove the area from the areaList
15. probListAreas = calculateProbability(areaList)
16. end for
17. while (notAllEventsExplored(Problem)) // if not all scenarios visited
18. pushScenario(oldGraph) // save the current scenario
19. event = removeNextEvt(UCEvtLst, newGraph)
20. // removing one event from the list of uncertain events, forming newGraph
21. if (noAreas(event)) then // no areas left, backtrack
22. popScenario(oldGraph)
23. goto 17
24. assignNextArea (event, newGraph)
25. // pick one area as the domain for this event, forming the new graph
26. if AC3.1(newGraph) then
27. popScenario(oldGraph)
28. goto 17
29. upperBound = estimateRobustness (newGraph)
30. if (upperBound < lowerBound) then
31. popScenario(oldGraph) // backtrack, cutting branches
32. goto 17
33. `constraintList = allRelatedConstraints(event);`

34. // `constraintList` is the list of all constraints related to this event

35. `for each e_i in constraintList`

36. `for each e_j in constraintList`

37. `Num2Sym(e_i, e_j) // perform the numeric to symbolic conversion`

38. `end for`

39. `if allAreasAssigned() then // if one area is specified for each event`

40. // using backtrack search to find a solution

41. `if (BTNumericTCSP(newGraph, curSol)) then // if one solution is found`

42. `newRobustness = calculateRobustness(newGraph, curSol)`

43. `if newRobustness < lowerBound then`

44. `lowerBound = newRobustness`

45. `else`

46. `popScenario(oldGraph)`

47. `goto 17`

48. `end while`

Figure 5.9   Pseudo code for solving symbolic TCSPs with uncertainty

5.6  Conclusion

In this chapter, I proposed a new method to handle uncertain domain in the TemPro
network. An algorithm for dividing domains into non-overlapping areas is proposed. I also examined the calculation of the robustness of solutions in different probable worlds. At the end, I described a branch-and-bound algorithm for searching the most robust solution in the TemPro model with uncertainty. In the algorithm, I applied several techniques to improve the efficiency of the searching algorithm such as numeric to symbolic conversion, arc-consistency check and path-consistency check.
6. CCTCSPs WITH UNCERTAINTY

In this chapter, I will introduce the composite and conditional constraints, which is an extension of the TemPro model to handle temporal problems in a dynamic environment where the possible changes are known. Then I will discuss how to handle the uncertainty of composite variables and conditional constraints in the TemPro model.

6.1 Composite Variables and Conditional Constraints

In numeric TCSPs, each variable is an event which can be described as SOPO \([\text{INF}, \text{SUP}, \text{DUR}, \text{STEP}]\). In chapter 5, the TemPro model was extended to handle uncertain \(\text{INF}\) and uncertain \(\text{SUP}\). Besides single events in classical numeric TCSPs, there are composite variables and conditional temporal constraints in the real world [16,42,43]. In [35], the TemPro model was extended to handle CCTCSP in a dynamic environment. Here is an example of CCTCSP [35].

Example 6.1 Composite variables and conditional constraints

John, Mike and Lisa are going to see a movie on Friday. John will pick Lisa up and Mike will meet them at the theater. If John arrives at Lisa's before 7:30, then they will stop at a convenient store and 15 minutes otherwise. There are three different shows playing: Movie_1, Movie_2 and Movie_3. If they finish the movie by 9:15, they will stop at a
Pizza place 10 minutes after the end of the movie and will stay there for 30 minutes. John leaves home between 7:00 and 7:20. Lisa lives far from John (15 minutes driving). Mike leaves home between 7:15 and 7:20 and it takes him 20 minutes to get to the theater. Movie 1, movie 2 and movie 3 start at 7:30, 7:45 and 7:55 and finish at 9:00, 9:10 and 9:20 respectively.

This example can be presented in Figure 6.1[35]. In the example, the event Watch_movie is a composite variable, which consists of three separate events Movie 1, Movie 2 and Movie 3. For example, the relation between the event Mike (Mike arrives at the theater) and the composite variable Watch_movie is before meet.

There are also some conditional constraints in Example 6.1. For example, the constraint between JPL and John_Lisa_Store (John and Lisa stop at a convenience store). The event John_Lisa_Store will only happen when JPL happens before 7:30. Otherwise, another event John_Lisa (John and Lisa go directly to the theater) will be activated.

The algorithm for solving CCTCSPs is given in [35]. The general idea of the algorithm is to pick one of the events in the composite variables and activate some conditional constraints in the backtrack search. In the above example, when it picks [0, 15] for event JPL during the search, it will activate the variable John_Lisa_Store and ignore John_Lisa. When the search comes to the composite variable Watch_movie, it will pick one movie, e.g., Movie 1, and continue to the next variable. If the search cannot find a solution, it will backtrack to the previous event.
6.2 Composite Variables with Uncertainty

In CCTCSPs, the component events in a composite variable are certain. For the

Figure 6.1 An example of CCTCSPs [35]
Watch_Movie event in Example 6.1, three component events Movie, Movie2 and Movie3 are available. Therefore, a consistent solution may contain any one of these three movies. However, in some applications, there may be some uncertainty for these events. For example, if Movie is a very popular movie, it would be likely that no tickets are available for Movie.

Watch_Movie = \{Movie, Movie2, Movie3\}. Assume the probability of no tickets for Movie is 0.4. In other words, the probability of Movie to be available is only 0.6. More formally, \( X = \{X_1, X_2, ..., X_m\} \) is a composite variable. \( X_i \) is one component event in the composite variable. Some components in composite variables are uncertain.

- \( Pr(X_i \in X) = p \) is the probability of \( X_i \) to be existent in the composite variable
- \( Pr(X_i \notin X) = 1 - p \) is the probability of \( X_i \) to be absent in the composite variable

Given the probabilities of the uncertain events in composite variables, the probability of a specific solution satisfying the composite variable can be computed. Consider the following example.

**Example 6.2 Uncertain composite variable**

The above example consists of only one composite variable Movie. Movie = \{Movie, Movie2, Movie3\}, \( Pr(Movie) = 0.6, Pr(Movie2) = 0.3, Pr(Movie3) = 1.0 \). All the other events are common events with certain INF and SUP. Assuming no conditional constraints, three are solutions \( S_1, S_2, S_3 \), which includes Movie, Movie2, and Movie3, respectively. The robustness of these three solutions can be calculated as follows.
• Robustness(S1) = Pr (S1 is the solution to the real world) = 0.6

• Robustness(S2) = Pr (S2 is the solution to the real world) = 0.3

• Robustness(S3) = Pr (S3 is the solution to the real world) = 1.0

In chapter 5, the uncertainty of numeric TCSPs results from the uncertain domains of the event, i.e., the uncertain INF and SUP. For composite variables, the uncertainty results from the component events in composite variable. Uncertainty is also represented as a discrete probability distribution as in chapter 4 and chapter 5.

6.3 Conditional Constraints with Uncertainty

The conditional constraints in CCTCSPs are also fixed conditions. In the above example 6.1, if John_Pick_Lisa is [0, 15], the event John_Lisa_Store will be activated. This condition can be represented as a conditional probability as follows.

\[ Pr (\text{John and Lisa go to the store} \mid \text{John pick Lisa at 7:10}) = 1 \]

However, in the real world, the conditional probability is not always 1. Consider the following example for a conditional constraint.

Example 6.3 Uncertain conditional constraint

Bob is going to see a movie on Monday. He can arrive at the theater in 20 minutes from home. The movie starts at 9:00. The movie is a very popular one. If he arrives at the theatre before 7:30, the probability of getting one ticket is 0.9. If he arrives at the theatre
after 7:30, the probability of getting one ticket is 0.2. This conditional constraint can be seen in Figure 6.2

![Diagram of uncertain conditional constraint]

The conditional probabilities of these uncertain conditional constraints are listed below.

- \( Pr (Get\_Ticket \mid Reach\_Theatre\_Early) = 0.9 \)
- \( Pr (No\_Ticket \mid Reach\_Theatre\_Early) = 0.1 \)
- \( Pr (Get\_Ticket \mid Reach\_Theatre\_Late) = 0.2 \)
- \( Pr (No\_Ticket \mid Reach\_Theatre\_Late) = 0.8 \)

If this is the only uncertain event in a CCTCSP, we can compute the robustness of each solution. Assume there are two solutions \( S_1 \) and \( S_2 \). [1, 21] for the event
Bob_Goto_Theatre is in both $S_1$ and $S_2$. $S_1$ activates the event *Watch_movie* while $S_2$ activates event *Go_Home*. The robustness of $S_1$ and $S_2$ can be calculated as follows.

- $\text{Robustness}(S_1) = Pr (S_1 \text{ is the solution to the real world}) = 0.9$
- $\text{Robustness}(S_2) = Pr (S_2 \text{ is the solution to the real world}) = 0.1$

All else equal, solution $S_1$ has a higher value of robustness than $S_2$.

### 6.4 Combining All Uncertain Factors

For uncertain composite variables, the uncertainty results from the uncertain existence of events in composite variables. For uncertain conditional constraints, the uncertainty results from uncertain conditions. This uncertainty can be combined with the uncertainty of INF and SUP as discussed in chapter 5. In order to solve them altogether, priori probabilities are used for uncertain domains and uncertain composite variables while conditional probabilities are used for uncertain conditional constraints.

In section 5.4, we proposed the method for computing robustness of solution for numeric TCSPs with uncertain INF and SUP. The uncertainty of composite variables and conditional constraints can also be integrated into the computation of robustness.

More formally, a CCTCSP can be represented as a tuple $<E, D, C>$ where $E$ is the set of events, $D$ is the set of domains of events and $C$ is the set of constraints. $E = \{E_{CD}, E_{UD}, E_{COM,CD}, E_{COM,UD}\}$. $E_{CD}$ is the set of events with fixed domains while $E_{UD}$ is the set of events with uncertain INF and/or SUP. $E_{COM,CD}$ is the set of composite variables with no
uncertainty. $E_{COM\_UD}$ is the set of composite variables with uncertainty. $C = \{\text{CFIXED}, \text{CCON\_FIXED}, \text{CCON\_UNFIXED}\}$. $\text{CFIXED}$ is the set of ordinary constraints. $\text{CCON\_FIXED}$ is the set of conditional constraints with no uncertainty. $\text{CCON\_UNFIXED}$ is the set of conditional constraints with uncertainty. A specific solution $S$ must pick one event for each composite variable and activate one condition for each conditional constraint. And $S$ must also select one certain START and END value for every event. The overall value of robustness can be computed by the following formula.

Formula 6.1

\[
\prod_i \Pr(E_{UD}) \times \prod_j \Pr(E_{COM\_UD}) \times \prod_k \Pr(E_{COND\_UNFIXED} \mid S)
\]

- $\Pr(E_{UD})$ are the probabilities of one uncertain event to fall in the actual domain
- $\Pr(E_{COM\_UD})$ are the probabilities of existing in the actual domain of the assigned events in one composite variable specified by $S$
- $\Pr(E_{COND\_UNFIXED})$ are the conditional probabilities of the uncertain constraints activated by the values of events specified by $S$

With the robustness that involves all kinds of uncertainty together, we can modify the branch and bound algorithm in 5.5.4 to solve CCTCSPs with uncertain domains, uncertain composite variables and uncertain conditional variables. The only difference here is the processing of composite variables and conditional constraints. The branch-and-bound algorithm can be created based on the backtrack search algorithm for CCTCSPs [35]. The branch-and-bound algorithm for CCTCSPs with uncertainty is
1. Perform path consistency check for the symbolic constraints in the problem. (See Figure 4.4) If it is inconsistent, return NO SOLUTION.

2. For all the constraints involving certain events, perform numeric to symbolic conversion (See Figure 5.5) to remove inconsistent Allen primitives.

3. Divide the uncertain domains of events into non-overlapping areas as in 5.3.2. For each event with uncertain domains, maintain a list of areas \( L = \{A_1, A_2...A_m\} \). Set the lower bound \( LB = 0 \).

4. Remove those areas inconsistent with the problem, i.e., \( Area_{\text{end}} - Area_{\text{begin}} < DURATION \).

5. Calculate the probability of each area to be included in the actual domain of each uncertain event.

6. Pick one uncertain event or one uncertain composite event or an activation event in uncertain conditional constraints
   a) For an event \( E_{UD} \) with uncertain domains, select an area from \( L \). An overestimation of the robustness of any possible solutions falling in the current assigned areas is calculated and used as the upper bound \( UB \) (see below). If \( UB < LB \), select another area for the current uncertain event or backtrack to the previous uncertain event if all the areas for this event have been explored.
b) For an uncertain composite event $E_{COM,UD}$ with uncertain component events, select one event in $E_{COM,UD}$. An overestimation of the robustness of any possible solutions containing this event is calculated and used as the upper bound $UB$ (see below). If $UB < LB$, select another component event in the current composite event or backtrack if all the component events in this uncertain composite event have been explored.

c) For an event $E$ which is the activation event of an uncertain conditional constraint, select one possible outcome event. An overestimation of the robustness of any possible solutions containing this event is calculated and used as the upper bound $UB$ (see below). If $UB < LB$, select another outcome event for the current uncertain conditional constraint or backtrack if all the outcome events in this uncertain conditional constraints have been explored.

7. For all the constraints related to this uncertain event, perform numeric to symbolic conversions by using the currently assigned area (from 6.a) or the currently assigned event (from 6.b and 6.c)

8. Perform arc-consistency check (See 5.5.3). If the AC3.1 check fails, select another area for the current uncertain event, or select another event in the uncertain composite variables or select another event in the uncertain conditional constraints. If all the areas or events have been explored, backtrack to the
previous uncertain event, composite variable or conditional constraint.

9. Repeat 6 ~ 8 until all the uncertain events have been assigned areas and all the uncertain composite variables and conditional constraints have been assigned specific events.

10. If every uncertain event has been assigned a specific area, try to find a solution for the problem using a backtrack search. Calculate the robustness of the solution. If it is larger than $LB$, update $LB$ with the value. If there is no solution or the robustness is less than $LB$, backtrack and goto step 4.

11. If the areas of all the uncertain events have been tried and no solution is found, return $NO\_SOLUTION$.

The overestimation of the robustness is the product of three probabilities. The first probability is the product of the probabilities of the assigned areas and the maximum probabilities of areas for the non-assigned uncertain events. The second probability is the product of the probabilities of the assigned events in the uncertain composite events and the maximum probabilities of the non-assigned uncertain composite events. The third probability is the product of the conditional probabilities of the already assigned uncertain conditional constraints and the maximum conditional probabilities of the non-assigned uncertain conditional constraints. (See Formula 6.1)

The pseudo code is shown in Figure 6.3.
// Problem is the CCTCSP with uncertainty

// FCEvtLst is the list of fixed events

// UCEvtLst is the list of uncertain events

// UCComEvtLst is the list of uncertain composite variables

// UCCondLst is the list of uncertain conditional constraints

// Graph is the constraint graph of the this TCSP

**Function** SolveCCTCSP (Problem, FCEvtLst, UCEvtLst, UCComEvtLst, UCCondLst, Graph)

1. curSol = NULL // curSol is the current solution

2. lowerBound = 0

3. upperBound = 0

4. oldGraph = graph

5. if DPC (oldGraph) then // check path consistency

6. return NO_SOLUTION

7. for each \( e_i \) in FCEvtLst

8. for each \( e_j \) in FCEvtLst

9. Num2Sym(e_i, e_j) // perform the numeric to symbolic conversion

10. end for

11. areaList = divideAreas(FCEvtLst) // divide uncertain events into non-overlapping areas
12. for each area in areaList
13. if SUP(area) − INF(area) < DURATION(area) then
14. remove(area) // remove the area from the areaList
15. probListAreas = calculateProbability(areaList) // calculate the probabilities of areas
16. end for
17. while (notAllUncertaintyExplored(P)) // if not all scenarios visited
18. pushScenario(oldGraph) // save the current scenario
19. event = removeNextEvt (UCEvtLst, UCComEvtLst, UCCondLst, newGraph)
20. // removing one event from the list of events with uncertain INF/SUP, or one
21. // uncertain composite event, or one uncertain conditional constraints
22. if (noAvailableValues(event)) then // no areas left, backtrack
23. popScenario(oldGraph)
24. goto 17
25. // event with uncertain INF/SUP
26. // pick one area as the domain for this event, forming the new graph
27. if (isUncertainINFSUP(event)) then
28. assignNextArea (event, newGraph)
29. else if (isUncertainCompositeEvent(event)) then
30. assignNextComponent (event, newGraph)
else if (isUncertainConditionalConstraints(event)) then
    assignNextConditionalConstraint(event, newGraph)
    upperBound = estimateRobustness (newGraph, curSol)
    if (upperBound < lowerBound) then
        popScenario(oldGraph) // backtrack, cutting branches
        goto 17
    end if
    constraintList = allRelatedConstraints(event);
    // constraintList is the list of all constraints related to this event
    for each ei in constraintList
        for each ej in constraintList
            Num2Sym(ei, ej) // perform the numeric to symbolic conversion
        end for
    end if
    if allAreasAssigned() then // if one area is specified for each event
        // using backtrack search to find a solution
        if (BTCCTCSP(newGraph, curSol)) then
            newRobustness = calculateRobustness(newGraph);
            if newRobustness < lowerBound then
                lowerBound = newRobustness // update the robustness
            else
                popScenario(oldGraph)
            end if
        end if
50. goto 17
51. end while

Figure 6.3 The pseudo code for solving CCTCSPs with uncertainty

6.5 Conclusion

In this chapter, I proposed a new method to handle uncertainty in composite variables and conditional constraints in CCTCSPs. The uncertainty of composite variables is represented as a probability of each component event within it. The uncertainty of conditional constraint is represented as a conditional probability of each condition. I also described how to calculate the robustness of solutions of CCTCSPs with uncertain composite variables and uncertain conditional constraints. At the end, I proposed the algorithm for integrating uncertain domains (INF, SUP), uncertain composite variables, and uncertain conditional constraints altogether in the TemPro model.
7. EXPERIMENTS

In this Chapter, I will present the experimental results of solving symbolic TCSPs and numeric TCSPs with uncertainty. These uncertain temporal problems are generated randomly. Experimental results will be used to evaluate the efficiency of the algorithms I proposed.

7.1 Instances of Symbolic TCSPs and Numeric TCSPs

In [31,34], TCSPs presented in the TemPro model are generated randomly following the specified parameters of events and constraints, which include the number of events, the upper limit of time for events and the maximum number of Allen primitives in disjunctive constraints. By different combinations of these parameters, temporal problems of different complexity can be generated. The construction of a TCSP starts from a random assignment of interval for each event. The TCSP is then generated by extending this assignment.

For example, we want to generate a TCSP with three events $E_1$, $E_2$ and $E_3$. $E_1$ is given an interval of $(10, 20)$, i.e., $E_1$ starts from 10 and ends at 20. $E_2$ is given an interval of $(15, 18)$ and $E_3$ is given an interval of $(20, 30)$. Thus, the duration of $E_1$ is 10, the duration of $E_2$ is 20 and the duration of $E_3$ is 3. The relation before $E_1$ and $E_2$ is $D$ (During_By), the relation between $E_1$ and $E_3$ is $S$ (Start), and the relation between $E_2$ and $E_3$ is $M$ (Meet).
With the assignment of time intervals, a TCSP can be generated by adding some other possible domains and Allen primitives. The assigned intervals are extended to form SOPOs for events while the relations are extended to form disjunctive constraints. The TCSP in Figure 7.2 is one example of extending the assignment in Figure 7.1.

Figure 7.2 Extending the assignment to form a TCSP
The generated TCSP is represented in the TemPro model. In order to generate the IA network, we can simply ignore the numeric information about temporal intervals and keep the symbolic information about the disjunctive constraints. The IA network for the above example is shown in Figure 7.3.

![Figure 7.3 The generated IA network](image)

### 7.2 Generating TCSPs with Uncertainty

#### 7.2.1 Generating TCSPs with uncertainty in the IA network

The uncertainty in the IA network is the uncertain existence of Allen primitives in disjunctive constraints. Thus, we can add probabilities to each primitive in some constraints. For example, in Figure 7.3, the constraint between $E_1$ and $E_3$ is $D\lor O \lor S$. We can assign probabilities of 0.8, 1.0 and 0.5 for $D$, $O$ and $S$, respectively.

There are two determinant factors in generating uncertain TCSPs in the IA network. One is the number of uncertain constraints in the problem. In the above example, if we
set the constraint between \( E_1 \) and \( E_3 \) to be the only uncertain constraint, this number is one. The other is the number of uncertain Allen primitives in one uncertain constraint. In the above example for \( D-\lor O\lor S \) between \( E_1 \) and \( E_3 \), \( O \) and \( S \) are both associated with probabilities of less than 1.0 while \( D- \) is associated with a probability of 1.0. \( O \) and \( S \) are both uncertain Allen primitives while \( D \) is a certain Allen primitive. Thus, the number of uncertain primitives is two.

7.2.2 Generating TCSPs with uncertainty in the TemPro model

As we discussed in section 5, the uncertain factors in the TemPro model are uncertain \( INF \) and uncertain \( SUP \). In section 7.1, we have described how to generate TCSPs in the TemPro model. The uncertainty of \( INF \) and \( SUP \) can be added to the original TCSPs to form uncertain temporal problems. In the above example in Figure 7.2, the SOPO of event \( E_1 \) is \([8, 22, 10, 1]\). In order to generate uncertain TCSPs in the TemPro model, some other \( INF \) and \( SUP \) can be added. For example, we can add two more \( INF \) and one more \( SUP \) to the original problem. A random discrete probability distribution is assigned to these \( INF \) and \( SUP \). The new uncertain domain of \( E_1 \) is shown as follows.

- \( \text{Pr}(INF = 8) = 0.5, \text{Pr}(INF = 10) = 0.3, \text{Pr}(INF = 6) = 0.2 \)
- \( \text{Pr}(SUP = 22) = 0.1, \text{Pr}(SUP = 25) = 0.9 \)

The other events in the TemPro model can be added with uncertainty in a similar manner. The two determinant factors in uncertain TCSPs in the TemPro model are the
number of events with uncertain domains and the number of uncertain $INF$ and $SUP$ in one event.

In [34], an algorithm for generating instances of TCSPs in the TemPro model is given. However, all the instances generated there are TCSPs without uncertainty. There are no existing algorithms that can generate TCSPs with the uncertainty defined in this thesis. I designed and implemented three algorithms to incorporate random probability distributions into symbolic TCSPs with uncertainty, numeric TCSPs with uncertainty and CCTCSPs with uncertainty. In symbolic TCSPs with uncertainty, the probability distributions of uncertain Allen primitives are generated. In numeric TCSPs with uncertainty, the probability distributions of uncertain INF and uncertain SUP are generated. In CCTCSPs with uncertainty, the probability distributions composite events and the conditional probability distributions of uncertain conditional constraints are generated. These algorithms can be modified and used by other people who want to generate their own TCSPs with uncertainty.

7.3 Testing Environment

All of the programs are written in C++ under the Microsoft Visual Studio 2003. The operating system is Microsoft Windows XP. The CPU used in the testing environment is Intel Pentium Core Duo Q6600 2.4GHz and the memory used is 2 Gigabytes.
7.4 Experiments on Symbolic TCSPs with Uncertainty

7.4.1 Experiments on generating and loading the composition table

The composition table of Allen primitives is used in path consistency check during the search for the most robust solution in the IA network with uncertainty. The composition table of two Allen primitive is given in Figure 4.5. In section 4.5.1, I proposed an algorithm for generating the composition table of disjunctive Allen constraints. The composition table is generated once and then saved in a file. During the initialization of the solving algorithm, the composition table is loaded into memory and ready for further reference. The running time of generating the composition table and loading the table is given in Figure 7.4.

In Figure 7.4, the running time of generating the composition table is about 42 minutes. Although it takes 42 minutes to generate the table, it is generated only once. The loading time of the table is just about 4.2 seconds, which can be ignored compared with the algorithm for solving symbolic TCSPs.

<table>
<thead>
<tr>
<th>Generating the table</th>
<th>38 minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loading the table</td>
<td>4.2 seconds</td>
</tr>
</tbody>
</table>

Figure 7.4 Generating and loading the full composition table
7.4.2 Experiments on path consistency check

As we discussed in section 4.5.1, the path consistency is the major operation in solving TCSPs in the IA network. By using the full composition table of Allen primitives, the efficiency of path consistency check in our algorithm is improved substantially. The running time of path consistency check is given below (See Figure 7.5)

As shown in the table, the time of path consistency check increases as the number of events increases. For the experiments, we set the number of maximum Allen primitives in disjunctive constraints to be 13. In other words, all of the 13 possible Allen primitives can be existent in one constraint. This setting will generate the most complex symbolic TCSPs for the given number of events. From the table, we find the running time of one path consistency check for 100 events is only 0.305 seconds, which is quite short. Even for 200 events, path consistency check is only 2.587 seconds, which is reasonable for practical use.
<table>
<thead>
<tr>
<th>$N$</th>
<th>Time of path consistency check</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.0280</td>
</tr>
<tr>
<td>60</td>
<td>0.0883</td>
</tr>
<tr>
<td>80</td>
<td>0.192</td>
</tr>
<tr>
<td>100</td>
<td>0.305</td>
</tr>
<tr>
<td>120</td>
<td>0.602</td>
</tr>
<tr>
<td>150</td>
<td>0.992</td>
</tr>
<tr>
<td>180</td>
<td>1.63</td>
</tr>
<tr>
<td>200</td>
<td>2.59</td>
</tr>
</tbody>
</table>

$N$ – Number of events

Figure 7.5  Table of the test results on path consistency check (in seconds)

7.4.3 Experiments on the branch-and-bound algorithm for symbolic TCSPs with uncertainty

In section 4.5.5, I proposed a new branch-and-bound algorithm to solve the IA network with uncertainty. The results of experimental tests are given below. (See Figure 7.5 and Figure 7.6)

As shown in Figure 7.5 and Figure 7.6, the running time of finding the most robust solution increases as the number of uncertain events increase. The time also increases as
the maximum number of uncertain primitives in a constraint increases. When the maximum number of uncertain primitives in one constraint (M) is 2 and the ratio of uncertain constraints (UC) is 10%, the problems including 200 events can be solved in 30 minutes. When the maximum number of uncertain primitives in one constraint (M) is 3 and the ratio of uncertain constraints (UC) is 10%, the problems including 200 events can still be solved in 40 minutes. Therefore, my algorithm is efficient for most real world applications.

<table>
<thead>
<tr>
<th></th>
<th>UC=1%</th>
<th>UC=5%</th>
<th>UC=10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=40</td>
<td>1.403</td>
<td>2.675</td>
<td>3.675</td>
</tr>
<tr>
<td>N=60</td>
<td>6.961</td>
<td>8.935</td>
<td>18.93</td>
</tr>
<tr>
<td>N=80</td>
<td>15.31</td>
<td>19.32</td>
<td>52.32</td>
</tr>
<tr>
<td>N=100</td>
<td>42.00</td>
<td>50.98</td>
<td>130.9</td>
</tr>
<tr>
<td>N=120</td>
<td>97.83</td>
<td>135.3</td>
<td>240.3</td>
</tr>
<tr>
<td>N=150</td>
<td>200.3</td>
<td>288.8</td>
<td>477.8</td>
</tr>
<tr>
<td>N=180</td>
<td>370.9</td>
<td>495.3</td>
<td>945.3</td>
</tr>
<tr>
<td>N=200</td>
<td>602.4</td>
<td>714.6</td>
<td>1543</td>
</tr>
</tbody>
</table>

N - Number of events    UC - the ratio of uncertain constraints

Maximum number of uncertain primitives in one constraint (M) = 2

Figure 7.6 Table of the test results on uncertain symbolic TCSPs (M=2, in seconds)
<table>
<thead>
<tr>
<th>N</th>
<th>UC=1%</th>
<th>UC=5%</th>
<th>UC=10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>N = 40</td>
<td>2.259</td>
<td>3.103</td>
<td>4.689</td>
</tr>
<tr>
<td>N = 60</td>
<td>8.513</td>
<td>11.93</td>
<td>20.35</td>
</tr>
<tr>
<td>N = 80</td>
<td>18.59</td>
<td>33.86</td>
<td>61.68</td>
</tr>
<tr>
<td>N= 100</td>
<td>63.15</td>
<td>95.98</td>
<td>143.8</td>
</tr>
<tr>
<td>N= 120</td>
<td>125.8</td>
<td>162.3</td>
<td>206.8</td>
</tr>
<tr>
<td>N= 150</td>
<td>250.4</td>
<td>450.3</td>
<td>623.9</td>
</tr>
<tr>
<td>N= 180</td>
<td>480.8</td>
<td>884.6</td>
<td>1403</td>
</tr>
<tr>
<td>N= 200</td>
<td>943.1</td>
<td>1366</td>
<td>1946</td>
</tr>
</tbody>
</table>

N - Number of events  
UC - the ratio of uncertain constraints

Maximum number of uncertain primitives in one constraint (M) = 3

Figure 7.7 Table of the test results on uncertain symbolic TCSPs (M=3, in seconds)

7.5 Experiments on Numeric TCSPs with Uncertainty

7.5.1 Experiments on dividing domains into non-overlapping areas

In section 5.3.2, I proposed a new algorithm for dividing uncertain domains into non-overlapping areas. These non-overlapping areas are then used to construct the basic set of probable worlds. The test result of this dividing algorithm is shown below (See Figure 7.8)
As shown in Figure 7.8, the time cost of the dividing algorithm is less than 0.00001 seconds even with 50 possible INF and 50 possible SUP. The dividing algorithm is executed only once for each uncertain event in the problem. Therefore, the time cost for dividing uncertain domains into non-overlapping areas can be ignored.

7.5.2 Experiments on the branch-and-bound algorithm for solving numeric TCSPs with uncertainty

In section 5.5.4, I proposed a new branch-and-bound algorithm for solving uncertain TCSPs in the TemPro model. The algorithm is used for finding the most robust solution. The results of experimental tests are given below. (See Figure 7.9 and Figure 7.10)

As shown in the test results, the running time of finding the most robust solution increases as the number of uncertain events increases. The time also increases as the maximum number of uncertain INF and SUP increases. When the maximum number of uncertain INF and the maximum number of uncertain SUP are both 2, our algorithm is efficient for problems including 180 events of which 15% are events with uncertain domains. When the maximum number of uncertain INF and the maximum number of
uncertain SUP are both 3, our algorithm is still efficient for problems including 180 events of which 15% are events with uncertain domains.

<table>
<thead>
<tr>
<th></th>
<th>UE=5%</th>
<th>UE=10%</th>
<th>UE=15%</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=40</td>
<td>4.563</td>
<td>7.346</td>
<td>18.91</td>
</tr>
<tr>
<td>N=60</td>
<td>6.376</td>
<td>15.30</td>
<td>39.04</td>
</tr>
<tr>
<td>N=80</td>
<td>13.67</td>
<td>40.01</td>
<td>112.35</td>
</tr>
<tr>
<td>N=100</td>
<td>22.84</td>
<td>73.09</td>
<td>204.7</td>
</tr>
<tr>
<td>N=120</td>
<td>45.67</td>
<td>163.5</td>
<td>506.9</td>
</tr>
<tr>
<td>N=150</td>
<td>73.54</td>
<td>308.9</td>
<td>1037</td>
</tr>
<tr>
<td>N=180</td>
<td>145.0</td>
<td>735.2</td>
<td>2930</td>
</tr>
<tr>
<td>N=200</td>
<td>261.4</td>
<td>1652</td>
<td>&gt; 1 hour</td>
</tr>
</tbody>
</table>

N - Number of events  
UE - the ratio of events with uncertain domains

Maximum number of uncertain INF in one constraint (R_{INF}) = 2

Maximum number of uncertain SUP in one constraint (R_{SUP}) = 2

Figure 7.9  Table of the test results on uncertain numeric TCSPs (R=2, in seconds)
<table>
<thead>
<tr>
<th>N</th>
<th>UE=5%</th>
<th>UE=10%</th>
<th>UE=15%</th>
</tr>
</thead>
<tbody>
<tr>
<td>N = 40</td>
<td>5.720</td>
<td>18.36</td>
<td>77.39</td>
</tr>
<tr>
<td>N = 60</td>
<td>8.269</td>
<td>26.58</td>
<td>122.6</td>
</tr>
<tr>
<td>N = 80</td>
<td>22.84</td>
<td>84.33</td>
<td>431.8</td>
</tr>
<tr>
<td>N= 100</td>
<td>42.41</td>
<td>204.1</td>
<td>652.9</td>
</tr>
<tr>
<td>N= 120</td>
<td>78.23</td>
<td>162.3</td>
<td>915.6</td>
</tr>
<tr>
<td>N= 150</td>
<td>257.5</td>
<td>1362</td>
<td>&gt; 1 hour</td>
</tr>
<tr>
<td>N= 180</td>
<td>530.6</td>
<td>3141</td>
<td>&gt; 1 hour</td>
</tr>
<tr>
<td>N= 200</td>
<td>1072</td>
<td>&gt;1 hour</td>
<td>&gt;1 hour</td>
</tr>
</tbody>
</table>

N - Number of events    UE - the ratio of events with uncertain domains

Maximum number of uncertain INF in one constraint ($R_{INF}$) = 3

Maximum number of uncertain SUP in one constraint ($R_{SUP}$) = 3

Figure 7.10 Table of the test results on uncertain numeric TCSPs (R=3, in seconds)

### 7.6 Experiments on CCTCSPs with Uncertainty

In section 6.4, I proposed a new branch-and-bound algorithm for solving uncertain CCTCSPs in the TemPro model. The algorithm is used for finding the most robust solution to CCTCSPs with uncertainty. Some events and constraints are modified
randomly in the generated TCSPs to build uncertain composite variables and uncertain conditional constraints. The results of experimental tests are given below. (See Figure 7.11, Figure 7.12) In each of these tests, the number of event is 100, the ratio of events with uncertain domains is 15% ($UE = 15\%$), the maximum number of uncertain $INF$ is 3 and the maximum number of uncertain $SUP$ is 3 ($R_{INF} = 3, R_{SUP} = 3$). In the first test, both the number of component variables in uncertain composite variables and the number of outcome variables in uncertain conditional constraints are 2. In the second test, both the number of component events in uncertain composite variables and the number of outcome events in uncertain conditional constraints are 3. $#CCON\_UNFIXED$ is the number of uncertain conditional constraints and $#ECOM\_UD$ is the number of uncertain composite variables.

As shown in the test results, the running time of finding the most robust solution increases as the number of uncertain composite events increases or the number of uncertain conditional constraints increases. The time also increases as the number of component variables in each uncertain composite variable increases or the number of outcome events in each uncertain conditional constraint increases. Our algorithm is efficient for problems of 100 events in which there are 32 uncertain composite variables and 32 uncertain conditional constraints.
Number of component events in $E_{COM,UD}$ and $C_{CON,UNFIXED}$ are both 2

Figure 7.11 Table of the results of test 1 on uncertain CCTCSPs (in seconds)

<table>
<thead>
<tr>
<th>#C_{CON,UNFIXED}</th>
<th>#E_{COM,UD}</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>168.2</td>
<td>180.0</td>
<td>237.6</td>
<td>308.6</td>
<td>444.9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>194.6</td>
<td>221.4</td>
<td>282.4</td>
<td>392.5</td>
<td>499.1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>264.1</td>
<td>295.8</td>
<td>350.0</td>
<td>430.8</td>
<td>578.5</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>356.0</td>
<td>400.3</td>
<td>436.1</td>
<td>485.6</td>
<td>696.4</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>481.5</td>
<td>523.5</td>
<td>591.9</td>
<td>723.5</td>
<td>906.3</td>
<td></td>
</tr>
</tbody>
</table>

Number of component events in $E_{COM,UD}$ and $C_{CON,UNFIXED}$ are both 3

Figure 7.12 Table of the results of test 2 on uncertain CCTCSPs (in seconds)

<table>
<thead>
<tr>
<th>#C_{CON,UNFIXED}</th>
<th>#E_{COM,UD}</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>235.2</td>
<td>286.3</td>
<td>361.9</td>
<td>564.7</td>
<td>599.3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>304.9</td>
<td>375.1</td>
<td>476.2</td>
<td>606.5</td>
<td>750.8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>385.6</td>
<td>495.5</td>
<td>548.8</td>
<td>693.7</td>
<td>954.0</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>572.0</td>
<td>647.3</td>
<td>772.9</td>
<td>835.2</td>
<td>1258</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>633.8</td>
<td>796.9</td>
<td>988.5</td>
<td>1350</td>
<td>2459</td>
<td></td>
</tr>
</tbody>
</table>
7.7 Summary of Experimental Results

The results in section 7.4, 7.5 and 7.6 show the running time of my algorithms for solving symbolic TCSPs with uncertainty, numeric TCSPs with uncertainty and CCTCSPs with uncertainty, respectively. The results show that these algorithms can find the most robust solution in these uncertain TCSPs with a number of 200 variables in just one hour. The size of 200 is large enough for real world applications of small sizes and middle sizes. For example, in most automatic manufacturing lines, the number of machines and tasks will not exceed 200. Once the problem is constructed, the search for the most robust solution is performed only once. This solution can then be saved and referred to later. Therefore, the running time of the search, which is less than one hour, is short compared with the subsequent long period through which this solution can be used. Furthermore, the running time of my algorithm is short compared with the existing result for solving TCSPs or CCTCSPs without uncertainty as in [34,35].

7.8 Conclusion

In this chapter, I performed experimental tests on the algorithms for solving symbolic TCSPs with uncertainty in the IA network, and the algorithms for solving numeric TCSPs with uncertainty in the TemPro model as well as algorithms for solving CCTCSPs with uncertainty. The uncertain TCSPs were generated by randomly adding uncertainty to
traditional TCSP instances. The experimental results showed that my algorithms can be used to solve uncertain temporal problems in reasonable time when the number of uncertain factors is large. I will improve the efficiency of my algorithms in the future to make them suitable for more complex problems.
8. CONCLUSIONS AND FUTURE WORK

In this thesis, I presented a comprehensive investigation on uncertainty in symbolic and numeric TCSPs. First, I reviewed the concepts of CSPs and TCSPs and several frameworks for handling uncertainty in TCSPs. Then, I proposed a new algorithm for solving symbolic TCSPs with uncertainty. I also proposed a new algorithm for solving numeric TCSPs with uncertainty. At the end, I proposed a new algorithm for solving composite and conditional TCSP with uncertainty. I implemented these algorithms and provided the experimental results for them.

For symbolic TCSPs, I extended the IA network to handle the uncertainty in the disjunctive constraints of Allen primitives. Each primitive in an uncertain constraint is associated with a probability, which indicates the likelihood of the primitive to exist in the real world. Probable worlds are used to represent different combinations of uncertain constraints. A minimal set of probable worlds is constructed to cover all the solutions to the IA network with uncertainty. Robustness is defined as the probability of one specific solution to satisfy all the constraints in the real world, which is equal to the total probability of all the worlds it satisfies. I also proposed and implemented a new branch-and-bound backtrack algorithm to find the most robust solution to the symbolic TCSPs with uncertainty.
For numeric TCSPs, I extended the TemPro network to handle uncertain domains of some events. The domain of each event in the TemPro model is defined by its earliest start time (INF) and latest finish time (SUP). In numeric TCSPs with uncertainty, there are several different INF and different SUP for some events. Each INF and SUP is associated with a probability, which indicates its likelihood to be in the real world. A possible domain of one event consists of one specific INF and one specific SUP. Some of these possible domains overlap other domains. I proposed an algorithm for dividing all the possible domains into non-overlapping areas. The probabilities of non-overlapping areas are calculated with the probabilities of the component INF and SUP. By combining the non-overlapping areas, non-overlapping probable worlds are constructed. The probabilities of probable worlds are calculated using the probabilities of their component areas. Then, the robustness of a possible solution is calculated as the total probability of all the worlds it satisfies. I also proposed and implemented a new branch-and-bound backtrack algorithm to find the most robust solution to the numeric TCSPs with uncertainty.

Composite variables and conditional constraints in CCTCSPs are used for modeling TCSPs in a dynamic environment. Composite variables (events) in CCTCSPs are events which contain several component events. In CCTCSPs with uncertainty, some of these component events are associated with probabilities to be in the real world. Conditional constraints are constraints in which specific values of an event will activate another event.
In CCTCSPs with uncertainty, one specific value may activate several possible outcome events. Each of these conditional constraints is associated with a conditional probability, which indicates the probability of the specific activation. With uncertain composite events and uncertain conditional constraints, different probable worlds can be constructed. The probability of each probable world is calculated using the probabilities of the uncertain composite events and the conditional probabilities of uncertain conditional constraints. The probability of a possible solution is equal to the total probability of all the worlds it satisfies. I also proposed and implemented a new branch-and-bound backtrack algorithm to find the most robust solution to CCTCSPs with uncertainty.

To show the efficiency of my algorithms, I designed several experiments. In these tests, a number of classical symbolic TCSPs, numeric TCSPs and CCTCSPs with no uncertainty were first generated by a TCSP generator. Then, I implemented a program to randomly incorporate uncertain factors and probabilities into these original problems in order to generate TCSPs and CCTCSPs with uncertainty. The running time of solving these uncertain problems by my algorithms were recorded. The test results showed that my algorithms are efficient for relatively complex TCSPs and CCTCSPs with uncertainty.

In the future, I will try to optimize the branch-and-bound algorithms proposed in this thesis. The optimization will be done in both the design of the algorithms and the implementation of programming source codes. Some heuristics such as the most
constrained variables and the least constrained values will be applied in the branch-and-bound algorithm. I will also try to use more complex probability models for representing uncertain environments, such as the interrelationships between uncertain factors. The occurrences of uncertain factors in the current algorithm are independent of each other. In some probability models such as the Bayesian network, the probabilities of uncertain factors are interrelated. Other uncertain factors, such as duration of events in the TemPro model may also be explored.

Another possible research direction is to apply the proposed algorithms to real world applications. In a highly automatic manufacturing factory, for example, numerous machines and working tasks can be modeled into events in TCSPs. Some external factors, such as power breaking or machine malfunctioning can be incorporated into uncertainty in TCSPs. An important concern in the real world applications is the accuracy of the user input. The probability distributions can be provided by end-users, either from experience or from statistics of historical data. In order to estimate the sensitivity of the algorithms to the accuracy of user input, stress testing and hypothesis testing can be performed by constructing various scenarios. The proposed algorithm in this thesis can be further extended to find a set of robust solutions, of which the total probability exceeds a pre-specified threshold.
Bibliography


[32] Mouhoub, M., "Temporal reasoning (CS820 lecture notes chapter 8)" 2007,  
http://www2.cs.uregina.ca/~mouhoubm/.

[33] Mouhoub, M., "Constraint satisfaction (CS820 lecture notes chapter 4)" 2007,  
http://www2.cs.uregina.ca/~mouhoubm/.

[34] M. Mouhoub, F. Charpillet and J. P. Haton, "Experimental analysis of numeric and symbolic constraint satisfaction techniques for temporal reasoning,"  


