A Complete Reified Temporal Logic and Its Applications

Guoxing Zhao

A thesis submitted in partial fulfillment of the requirements of the University of Greenwich for the degree of Doctor of Philosophy

June 2008

The University of Greenwich,
School of computing and Mathematical Science,
30 Park Row, Greenwich, SE10, 9LS

ACKNOWLEDGEMENTS

I am deeply indebted to my supervisors, Dr. Jixin Ma and Dr. Miltos Petridis, for their great interesting, enthusiasm and guidance in my research. They provided me with many helpful comments and suggestions during my studies and on the writing of research papers and this thesis.

I grateful acknowledge Prof. Fuxing Shen (Beijing Normal University), Prof. Rongfang Bie (Beijing Normal University), Prof. Bin Luo (Anhui University) and Prof. Angsheng Li (Institute of Software Chinese Academy of Sciences) for their help and suggestions on my research.

I also would like to thanks my colleges in University of Greenwich and Beijing Normal University for their helps in my research and daily life.

Finally, I want to thanks my parents for their encouragement and support during many years of the study.

ABSTRACT

Temporal representation and reasoning plays a fundamental and increasingly important role in some areas of Computer Science and Artificial Intelligence. A natural approach to represent and reason about time-dependent knowledge is to associate them with instantaneous time points and/or durative time intervals. In particular, there are various ways to use logic formalisms for temporal knowledge representation and reasoning. Based on the chosen logic frameworks, temporal theories can be classified into modal logic approaches (including propositional modal logic approaches and hybrid logic approaches) and predicate logic approaches (including temporal argument methods and temporal reification methods). Generally speaking, the predicate logic approaches are more expressive than the modal logic approaches and among predicate logic approaches, temporal reification methods are even more expressive for representing and reasoning about general temporal knowledge. However, the current reified temporal logics are so complicate that each of them either do not have a clear definition of its syntax and semantics or do not have a sound and complete axiomatization.

In this thesis, a new complete reified temporal logic (CRTL) is introduced which has a clear syntax, semantics, and a complete axiomatic system by inheriting from the initial first order language. This is the main improvement made to the reification approaches for temporal representation and reasoning. It is a true reified logic since some meta-predicates are formally defined that allow one to predicate and quantify over propositional terms, and therefore provides the expressive power to represent and reason about both temporal and non-temporal relationships between propositional terms.

For a special case, the temporal model of the simplified CRTL system (SCRTL) is defined as scenarios and graphically represented in terms of a directed, partially weighted or attributed, simple graph. Therefore, the problem of matching temporal scenarios is transformed into conventional graph matching.

For the scenario graph matching problem, the traditional eigen-decomposition graph matching algorithm and the symmetric polynomial transform graph matching algorithm are critically examined and improved as two new algorithms named meta-basis graph matching algorithm and sort based graph matching algorithm respectively, where the meta-basis graph matching algorithm works better for 0-1 matrices while the sort based graph matching algorithm is more suitable for continuous real matrices.

Another important contribution is the node similarity graph matching framework proposed in this thesis, based on which the node similarity graph matching algorithms can be defined, analyzed and extended uniformly. We prove that that all these node similarity graph matching algorithms fail to work for matching circles.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

c	a constant
f	a function
i, j, k, l, m, n	natural numbers
t	a temporal term
tm	a term
u	a non-temporal term
x,y	variables
A, B	Hermitian matrices
BTK	the temporal argument method in [BTK1991]
CS	a set of constants
CRTL	the complete reified temporal logic
D	the domain of a model
EXPTIME	the exponential time complex class
EDGM	Umeyama's eigen-decomposition graph matching algorithm
FS	a set of functions
G, H	graphs or real matrices
H_t	the minimal tense logic

L	a language
M(i, j)	the (i, j)-th entry of matrix M
M(p, q)	the matrix whose (i, j)-th entry is the (p(i), q(j))-th entry of M
M(:, q)	the matrix whose (i, j)-th entry is the (i, q(j))-th entry of M
M(p, :)	the matrix whose (i, j)-th entry is the (p(i), j)-th entry of M
MBGM	the meta-basis graph matching algorithm
MK	the reified temporal logic approach by Ma and Brian
NP-complete	the non-determined polynomial time complex class
NSGM	the node similarity graph matching algorithms or framework
Perm(n)	the set of all permutation matrices of n elements
PS	a set of predicates
PSPACE-complete	the polynomial space complex class
SCRTL	the simplified complete reified temporal logic
SG(n)	the set of all permutations of n elements
SPGM	the symmetric polynomial graph matching algorithm
STGM	the sort based graph matching algorithm
TM	the hybrid temporal logic approach by Reichgelt
TA	the temporal argument method by Reichgelt
TR	the reified temporal logic approach by Reichgelt
X	a unitary matrix

GLOSSARY

$<B>, <E>, <A>,$	modal operators
$<\underline{B}>, <\underline{E}>, <\underline{A}>$	modal operators
$<G>, <P>, <F>, <H>$	modal operators
$\alpha, \beta, \gamma, \varphi, \phi$	formulae
ε	a real number
λ, η	matrix eigenvalues
$\|\bullet\|_F$	Frobenius norm

This thesis will deal with two different research subjects: reified temporal logic and graph matching. In order to coincide with the notations in both fields, some symbols have double meaning, such as the M stands for a model in temporal logic and a matrix in graph matching. This can be easily distinguished because that the chapter 3 and chapter 4 discuss the reified temporal logic while the chapter 5 and chapter 6 handle the graph matching problems.

Symbols	**Chapter 3 - 4**	**Chapter 5 - 6**
p	a predicate	a permutation vector
I	the interpretation of a model	an identity matrix
M	a model	a matrix
P	a set of predicates	a permutation matrix
P_k	a predicate	a permutation matrix
S	the set of sorts	a similarity matrix
S_1, S_2	scenarios	similarity matrices

T	a set of temporal elements	the transpose of a matrix
U	a set of non-temporal elements	a unitary matrix
V	a set of variables	a unitary matrix

Chapter 1 Introduction

The term *temporal logic* is used to describe any system of rules and symbolism for representing, and reasoning about, propositions qualified in terms of time.

The logical study of time dates back to the days of Aristotle, while modern research on time started since 1960 when Arthur Prior firstly interprets modality as tense [Pri1967, Pri1969]. After that, the temporal logic is extended in many different ways. These extensions can be roughly classified into two groups: extending fundamental logics and extending time primitives. The fundamental logic evolves from prime modal logics to hybrid logics [BT1999, Alt2006], first order logics [TM1989, STL1987, BTK1991] and temporal reifications [Allen1983, TM1989, MK1996]; while the time primitives advance from points [GHR1994] to intervals [All1983, All1984], and finally both points and intervals are taken as time primitives [KM1992].

In applications, the temporal logic is widely used in computer science including: program specification [EGR1994], databases [SJ1999, DDL2002], real time systems [Ost1989, LT2002], distributed systems [CDF1995, DLH1998]; artificial intelligence including natural language processing [Dow1979, Tay1985, Ric1989], planning [CM1998, May2006], case based reasoning [Jac1997, Han2000, JAS2002].

The matching of temporal knowledge is what we shall study in detail in this thesis.

Section 1.1 Motivation: Matching Temporal Knowledge

Object similarity plays an important role in case-based reasoning [Kol1996, Lea1996, and Wat1997], pattern recognition [Gib2004, TK2006], web search engine [Lev2006], and cluster analysis [Rom1989]. Similar objects are usually supposed to have similar properties and solutions. In order to use case based reasoning techniques on temporal knowledge or recognize temporal patterns, one has to explore some kinds of similarity among these temporal knowledge or temporal structures. There are already some approaches for matching temporal knowledge and patterns which are classified here as point based approaches, interval based approaches and point and interval based approaches.

Section 1.1.1 Point based Approaches

Examples of these are that of Nakhaeizadeh [Nak1994], of Branting and Hasting [BH1994], of Jaczynski [Jac1997], and of Hansen [Han2000].

Problems

The underlying time models employed in the above systems are point-based, and therefore, it is required that absolute time points or intervals delimited by a pair of points, must be associated with the time-dependent statement being addressed. However, there are many applications in which there may be just some relative temporal knowledge about the time-depended statements to hand, where their precise time characters such as the exact starting and finishing time are not available (e.g., “John ran 3 miles yesterday morning”, “John arrived at the office before Mary went to home”, etc.).

Section 1.1.2 Interval based Approach

Jaere, Aamodt and Skalle [JAS2002] propose a method for representation and reasoning with temporal case within case based reasoning framework for unwanted events predictions. Based on Allen’s theory [All1983, All1984], time is represented as intervals and time-relations between intervals; and every temporal case is represented by a labeled graph as:

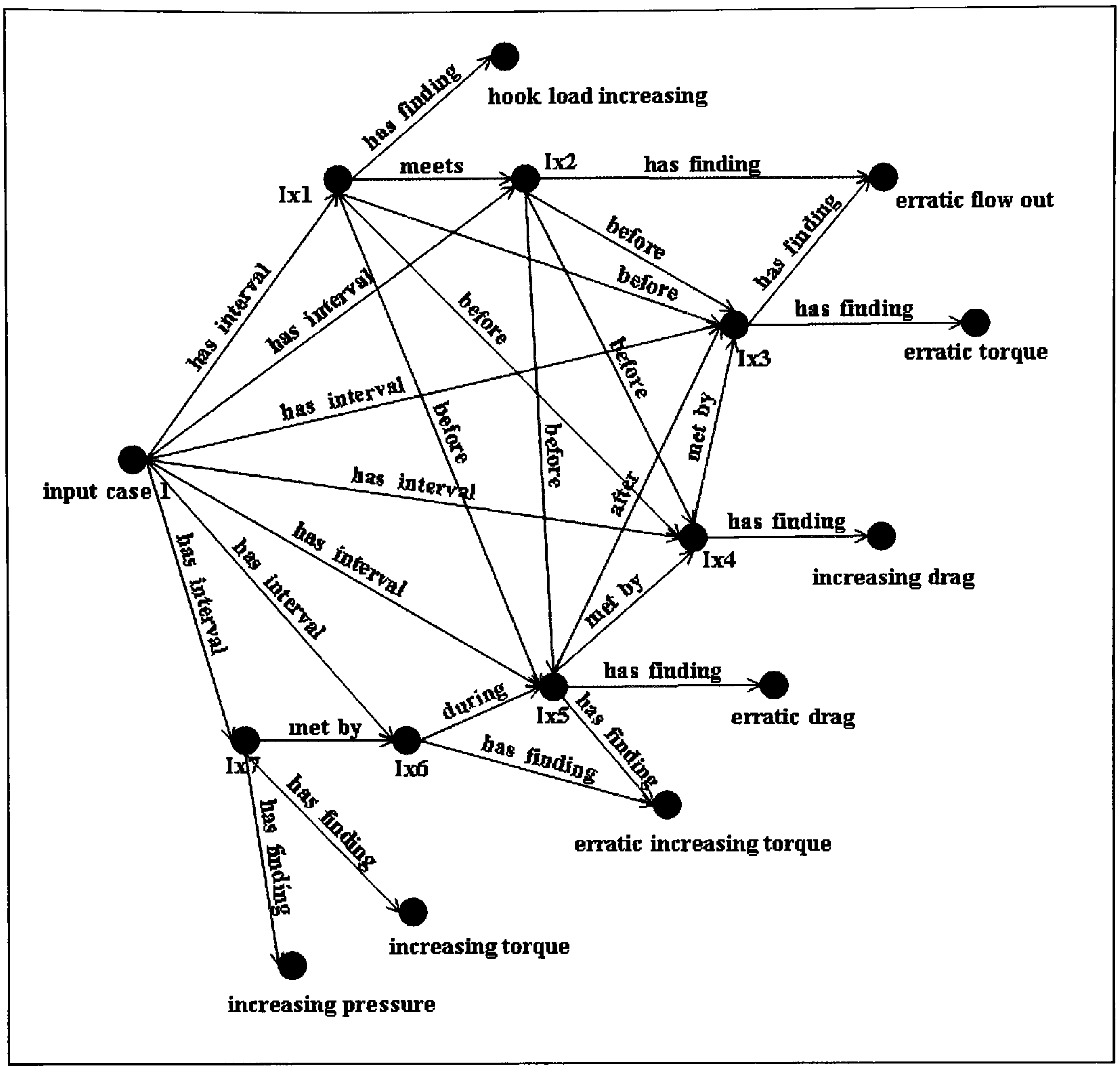


Figure 1.1 A case of drill-sticking [JAS2002]

Figure 1.1 shows a simple case after the raw data has been transformed into qualitative findings. There are seven intervals (named Ix1, Ix2, ..., Ix7) with corresponding time relationships and finding attributes to indicate the symptoms during that time. For example, "hook load increasing meets erratic flow out" is described by "Ix1 has finding hook load increasing", "Ix2 has finding erratic flow out" and "Ix1 meets Ix2".

In addition to the similarity degree for the non-temporal part, an extra temporal similarity measurement named the *temporal path strength* is introduced [JAS2002]. To enable intervals to be related to each other so that a similarity assessment of parameters can be made in order to predict particular states, a dynamic ordering algorithm is developed for matching temporal paths. Such an algorithm requires the corresponding temporal knowledge to be complete for both the input case and the current case.

Problems

The fundamental logic of this approach is based on Allen's interval theory, which itself has been argued lacking clarity of its semantics and completeness [GAL1990].

The matching algorithm proposed in [JAS2002] lacks a theoretical foundation of its effectiveness or generality. So using this framework for other problems, one may need to develop another matching algorithm for the new fields.

Section 1.1.3 Point and interval based Approach

In [MK2003], a framework for *Historical Case-Based Reasoning* which allows the expression of both relative and absolute temporal knowledge, representing case histories in the real world is presented.

The concepts of fluents, elemental cases, and case histories are formally defined. A graphical representation of case histories is also provided, where every case history can be described as a simple attributed graph as figure 1.2 [MK2003].

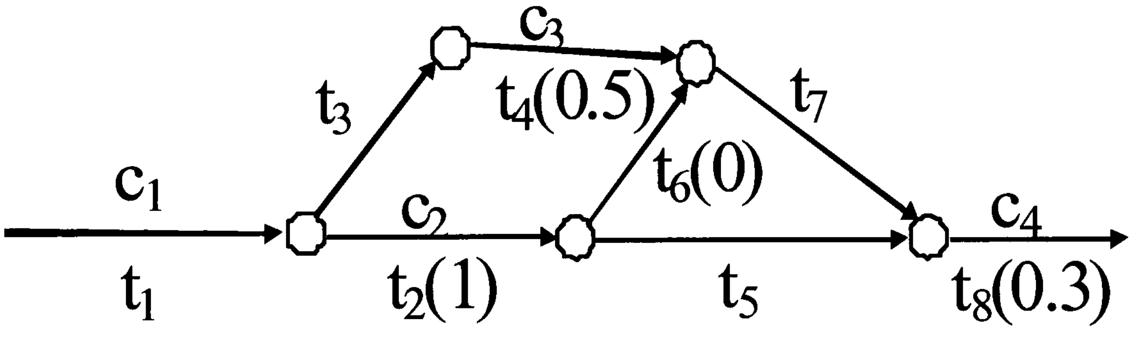


Figure 1.2 A case history.

The formalism is founded on a general temporal theory that accommodates both points and intervals as primitive time elements. A case history is formally defined as a collection of (time-independent) elemental cases, together with its corresponding temporal reference. Case history matching is two-fold, i.e., there are two similarity values that need to be computed: the non-temporal similarity degree and the temporal similarity degree. On the one hand, based on elemental case matching, the non-temporal similarity degree between case histories is defined by means of computing the unions and intersections of the involved elemental cases. On the other hand, by means of the graphical presentation of temporal references, the temporal similarity degree in case history matching is transformed into conventional graph similarity measurement.

Problems

Although this is a general and efficient representation of temporal cases, there are still two problems left.

Firstly, the fundamental temporal theory this representation based on is the temporal theory of Ma and Knight in [MK1994, MK1996], which has not been proved to be complete.

Secondly, an efficient graph matching algorithm has to be provided to make this representation applicable to real problem.

These two problems are what we try to solve in this thesis.

Section 1.2 Object: A Framework to Represent and Match Temporal Scenarios

This thesis tries to accomplish the following two tightly associated goals:

- A sound and complete reified temporal logic system for describing time structure, representing temporal information and reasoning about temporal knowledge.
- A graphical representation of temporal information and an efficient matching algorithm for reasoning of temporal knowledge.

Section 1.3 Outline of the Main Contributions

Aimed at the two goals listed above, the following works have been done:

- Based on the review and comparison of the existing important temporal logic systems, a complete reified temporal logic system is proposed, which has a clear syntax, semantics, sound and complete axiomatic deduction system and enough expressive power to talk about the generalities of the temporal aspect of assertions.
- Umeyama's eigen-decomposition graph matching (EDGM) algorithm [Ume1988] is critically examined and three important constraints are pointed out for matching general graphs by the EDGM algorithm. In order to match arbitrary graphs, a new approximate formula is presented together with theoretical proof of its accuracy. Based on this approximate formula, a unitary-invariant meta-basis graph matching (MBGM) algorithm is proposed as an improvement of the traditional eigen-decomposition method.
- Almohamod's symmetric polynomial transformation graph matching (SPGM) algorithm is critically examined and greatly improved by a new matching algorithm based on vector sort, named as STGM algorithm.
- A node similarity graph matching framework is presented to generally discuss all the node similarity graph matching (NSGM) algorithms. An interesting result shows that all the node similarity based graph matching algorithms fail to work for circles.
- These node similarity graph matching algorithms are applied to match general scenario graphs. The testing result shows that the meta-basis graph matching algorithm is more suitable for scenario graphs.

Section 1.4 Thesis Structure

The rest of this thesis is organized as follow:

In chapter 2, a detailed review will be provided to introduce some current temporal logic systems. They are rationally classified according to fundamental primitive logic and time primitives, and critically examined by theoretical and practical criterions including clarity of definition, soundness and completeness, expressiveness, etc.

In chapter 3, a complete reified temporal logic system CRTL is proposed. This work is done by applying the idea of Reichgelt's reified temporal logic [Rei1989] to reconstruct the reified logic system of Ma [MK1996] using the first order language. A simplified sub system SCRTL is also introduced in this chapter.

In chapter 4, temporal scenarios are introduced together with graphical representations and matrix representations. Some examples are illustrated to show how temporal knowledge is expressed by this framework. Based on this graphical representation, the pattern matching problem of the temporal scenario is transferred into graph matching problem.

In chapter 5, the eigen-decomposition graph matching algorithm and the symmetric polynomial graph matching algorithm are critically examined. The unitary-invariant meta-basis method and sort based method are proposed as improvements to the eigen-decomposition approach and symmetric polynomial transformation approach respectively.

In chapter 6, these node similarity graph matching algorithms are applied to scenario graphs.

Finally, a summary of conclusion and recommendations for future work are presented in chapter 7 and chapter 8.

There are also five appendices for this thesis. These are five of my published papers tightly associated with this research and titled as "A sound and complete reified temporal logic" [ZMS2008], "A Navigation-based Algorithm for Matching Scenario Patterns" [MZH2007], "Matching Case History Patterns in Case-Based Reasoning" [ZLM2006], "Matching Scenarios Patterns by Using Linear Programming" [ZLM2007] and "Using Eigen-decomposition Method for Weighted Graph Matching" [ZLT2007] respectively.

Chapter 2 Literature Review

Since the research associates with two subjects: completeness of temporal logic and graph matching algorithms, this section contains literature reviews in these two fields.

Section 2.1 Review of Temporal Logics

The term temporal logic has been broadly used to cover all approaches to the description of time structure and representation of temporal information within a logical framework. Based on the selections of time primitives, temporal logics can be classified into: point based, interval based, point and interval based; while according to the fundamental logic, temporal logics are grouped as: modal logic approaches and predicate logic approaches.

In this thesis, the main temporal logic systems are classified according to the above classifications; and this research pays more attentions to the three criterions: clarity of definition, "soundness and completeness", expressiveness.

Section 2.1.1 Propositional modal logic approaches

Propositional modal logics follow the traditional modal logic way to define temporal connectives by modal operators.

Point based

These approaches semantically re-interpret the classical possible-worlds by making each possible world represent a different time. It accommodates the concepts of time by means of extending the propositional modal temporal operators such as $<F>\varphi$, $<P>\varphi$, $<H>\varphi$ and $<G>\varphi$, representing that formula φ "will be true", "was true", "will always be true" and "was always true", respectively.

A simple system H_t, named as *Minimal Tense Logic*, was firstly studied by Lemmon and Scott in the 1960s with unpublished work (see in [GHR1994]) and this system was first published by Prior in [Pri1957, Pri1967, Pri1969].

The syntax and semantics of H_t are defined as the same way of classical modal logics. Four axioms and two inference rules are chosen as its deduction system, which are:

1 Axioms:

1.1 $p \rightarrow <H><F>p$

1.2 $p \rightarrow <G><P>p$

1.3 $<H>(p \rightarrow q) \rightarrow <H>p \rightarrow <H>q$

1.4 $<G>(p \rightarrow q) \rightarrow <G>p \rightarrow <G>q$

2 Inference rules:

2.1 Generalization: $\varphi \vdash <G>\varphi$, and $\varphi \vdash <H>\varphi$

2.2 Modus Ponens: $\{\varphi, \varphi \rightarrow \phi\} \vdash \phi$

The H_t system was proved to be sound, complete and computable [GHR1994]. The satisfiability problem for the flow of integer-like time in this logic system has been proved to be NP-complete by Sistla and Clarke [SC1985].

Lots of time properties can be expressed in the H_t system, such as transitivity, linearity, density, finiteness. However, it has been proved that simple irreflexivity cannot be defined by the H_t system [GHR1994].

Soon after its introduction, the basic "<P><F><G><H>" syntax of H_t was extended in various ways, and such extensions have continued to this day. Some important examples are the next operator and *US logic* system which will be introduced below.

The US logic system is proposed by Kamp [Kam1968], which enriched tense logic by the addition of two new binary connectives, the "since" operator <S> and the "until" operator <U>, where <S>pq denotes "q has been true since a time over which p was true" and <U>pq stands for "q will be true until a time over which p is true". The syntax and semantics are defined as the normal modal logic.

Some time afterward, Kamp announced axiomatizability results for the US-tense logics of various classes of linear orders. His completeness proof was (in his own words) "by no means simple", and have never been published, though a manuscript treating certain classes of linear orders is in existence. This work has been revised by Burgess as an axiomatization for the classes of arbitrary linear orders and of dense and discrete orders, with and without first and last elements. He also proved the soundness and completeness of such US-tense logic system using maximal consistent sets [Bur1982].

The satisfiability problem of US temporal logic system for integer-like time flow is proved to be PSPACE-complete [SC1985].

The US tense logic is more expressively powerful than the H_t system. The US logic system has been proved to be expressively complete over integer time and Dedekind complete time [GHR1994], which means that all monadic formulae can be equivalent expressed in US logic system. However, US system is not expressively complete for the rational number time Q. This negative result prompted Jonathan Stavi to develop a fixed point extension of US logic which is expressively complete over the rational numbers Q [Sta1979].

Interval based

A propositional modal logic of time intervals HS was proposed by Halpern and Shoham [HS1986].

In HS system, six modal operators were defined: <B>, <E>, <A>, <$\underline{B}$>, <$\underline{E}$> and

$<\underline{A}>$, which have the following intended readings:

$<B>\varphi$ φ holds at a strict beginning interval of the current one.

$<E>\varphi$ φ holds at a strict end interval of the current one.

$<A>\varphi$ φ holds at an interval met by the current one

$<\underline{B}>\varphi$ φ holds at an interval which has the current one as a beginning interval

$<\underline{E}>\varphi$ φ holds at an interval which has the current one as an ending interval

$<\underline{A}>\varphi$ φ holds at an interval meeting the current one

The formulae of HS are defined as the same way of classical modal logic.

However, the semantics of HS can not follow the traditional modal logic; in fact, it has been carefully redefined based on *interval set*. Given a partial order (or temporal frame) (T, <), the interval set of such frame is defined as the set INT of all closed intervals $[t_1, t_2]=\{t \in T: t_1 \leq t \leq t_2\}$.

The axioms and inference rules of HS system were not provided by Halpern and Shoham in [HS1986], which have been supplemented by Venema in [Ven1990]. Venema represented the interval set on a two-dimensional plane, proposed adequate axioms and inference rules to make the HS system sound and complete over linear temporal flows.

For its computability, HS has been proved to be non-computable. In fact, Halpern and Shoham [HS1986] have proved that the satisfiability problems of HS formula over the natural number, rational number and general linear flow are Π_1^1-complete, r.e.-complete and co-r.e.-complete respectively.

Since every time point p was taken as an interval [p, p] in HS system, point-based modal temporal logics can be directly embedded in the HS system. Allen's 13 temporal relations are also definable by the HS formulae [HS1986]. On the other hand, Venema has proved [Ven1990] that there is no finite functionally complete set of interval tense operators over the dense linear order, which indicates that HS system is not as powerful as first order logic approaches.

Section 2.1.2 Hybrid logic approaches

The term *hybrid logic* refers to a number of extensions to propositional modal logic with more expressive power, though still less than first-order logic. Unlike ordinary modal logic, hybrid logic makes it possible to refer to states (possible worlds) in formulae. This is achieved by a class of formulae called nominals, which are true in exactly one state.

Point based

Blackburn and Tzakova introduced a general *hybrid temporal logic* BT [BT1999], where three new modal operators are defined: current state binder ↓, accessible state binder ⇓ and nominal operator @. A point based temporal logic is introduced based on these modal operators.

For each modal operator, the syntax and semantics were formally defined; axioms and inference rules for the operator were also provided. Blackburn and Tzakova also proved the soundness and completeness of the BT system.

Areces, Blackburn and Marx [ABM2000] have proved that the satisfiability problem of hybrid temporal logic BT is in NP over strict total orders and in EXPTIME over all frames.

It also has been claimed [BT1999] that the hybrid logic system BT is more expressive than the modal temporal logic approaches, where the since and until modal operators can be directly defined on the state binders and nominal operator. The time irreflexivity can be easily formulated by $\downarrow x H \neg x$, which is not definable in modal logic approaches.

Reichgelt [Rei1989] presented a modal temporal logic TM, which is actually a hybrid temporal logic of time points. In TM system, although syntax and semantics are well-defined, it does not have its own axioms and inference rules, so it lacks sound and complete deduction system. TM is essentially a subsystem of BT.

Interval based

Altaf [Alt2006] proposed minimal hybrid logic for intervals, named IHL. IHL has 4 modal operators <D>, <U>, <F>, <P> and nominals, which have the following intended readings:

<D>φ φ holds at an interval during the current one

<U>φ φ holds at an interval which contains the current interval

<F>φ φ holds at an interval after the current one

<P>φ φ holds at an interval before the current one

In Altaf's work [Alt2006], the syntax and semantics of IHL system are formally defined. A tableau system is proposed as the auto theorem proof system for IHL. The soundness and completeness are also theoretically proved. For its computability, the satisfiability problem of IHL formula has been shown to be EXPTIME-complete for minimal interval structures.

Although the hybrid logic approaches are more powerful than propositional modal logic approaches, Areces, Blackburn and Marx [ABM2000] have proved that hybrid logic is the bounded fragment of first order logic, so all the formulae in hybrid logics can be equivalent expressed in first order logic approaches.

Section 2.1.3 Temporal argument approaches

Compared with modal logic approaches using modal operators to express temporal knowledge, predicate temporal logic approaches are normally many-sorted languages including a sort of temporal elements and a sort of non-temporal elements. There are usually three kinds of functions and predicates: (i) temporal predicates that take only temporal terms as arguments to describe temporal relationships; (ii) non-temporal predicates that take only non-temporal terms as arguments to describe non-temporal relationships; (iii) mixed predicates that take both temporal and non-temporal terms as arguments to describe global relationships between temporal and non-temporal terms.

For the temporal argument methods, the temporal dimension is captured by augmenting each time-variable proposition or predicate with an extra argument-place, to be filled by an expression designating a time.

Point-based

Reichgelt [Rei1989] introduced a temporal argument method TA within the many sorted first order logic framework. In the TA system, time structure is simply described by an explicit ordering relation <. And the mixed predicate is just expressed by an additional temporal argument, for example the two-place predicate love(x, y) is redefined as love(x, y, t).

Since TA is proposed within the first order logic framework, its syntax, semantics, axioms and inference rules are automatically inherited from the standard first order logic. Its soundness and completeness also holds without any additional proof.

Interval based

In [BTK1991] Bacchus, Tenenberg and Koomen introduced another temporal argument method BTK within the many sorted first order logic framework. BTK has two sorts ‘u’ and ‘t’ stand for non-temporal and temporal elements. Functions are classified as temporal functions and non-temporal functions according to their range; while predicates are classified as temporal, non-temporal and mixed due to the arguments they takes.

The BTK system describes an interval just by two time points. For example Bob is sleeping during 0:00 to 7:00 is formulated as Sleeping(Bob, 0:00, 7:00). This treatment is indeed point based, where interval is defined by open or close intervals with two points as ends, which may lead to the so-called *Dividing Instant Problem* [All1983, Gal1990, Vil1994], that is the puzzle encountered when attempting to represent what happens at the boundary instant (point) which divides two successive intervals.

In short, since temporal argument approaches try to devise temporal logics by means of simply including time elements as additional arguments to functions and predicates in first order language, the method of temporal argument directly employs the syntax, semantics and the axiomatic system of the standard first order logic, and therefore, the completeness and soundness of temporal argument theories remain as default. Compared with modal temporal logics, on one hand, the method of temporal argument has more expressive power in representing properties of the time itself. For instance, using the method of temporal argument, the irreflexivity of time can be simply characterized by: $\forall t \neg(t<t)$. On the other hand, since the first order logic is not generally computable, it obtains more expressive power and loses some efficiency in the meanwhile. In addition, since time is represented just as an additional argument(s) to functions and/or predicates, neither conceptual nor notational special status to time is accorded in temporal argument approaches. Therefore, it is not expressive enough to talk about the generalities of the temporal aspect of assertions. For example, Shoham [Sho1987] claimed that using the method of temporal argument, one cannot express common-sense knowledge such as "effects cannot precede their causes".

Section 2.1.4 Temporal reification approaches

As an alternative approach, reified temporal logics reify standard propositions of some initial language (e.g., the classical first-order logic or modal logic) as objects denoting propositional terms. Propositional terms are related to temporal objects or other propositional terms through an additional sort of "meta-predicates" [MK1996, Sho1987], such as HOLDS (or TRUE), OCCUR and CAUSE, etc.

Point based

McDermott's logic [McD1982] is probably one of the earliest and most influential formalisms in AI that possess the characteristics of temporal reification. In this system, an infinite collection of states (or points) is introduced as the set of primitive temporal elements, where, for general treatment, states are partially ordered by the "no later than" relation $\leq$. In order to model continuous change, it is assumed that between any two distinct states, there is a continuum of states, so this time structure is actually linear dense order and isomorphic to the real line.

McDermott's dichotomy, i.e., *facts* and *events*, are the two basic entities that are associated with time. A fact is something that may be true in some states and false in others, so fact p is true at time s can be formulated as (T *s p*); on the other hand an event is something happening, for example event e happens during s_1 to s_2 is denoted as OCC(s_1, s_2, *e*).

As pointed out by Shoham [Sho1987], although McDermott gives the semantic of what may be regarded as the propositional theory, some assertions in his logic lack a clear meaning. Thus, it does not enjoy a sound and complete deduction system either.

In addition, as BTK does, McDermott's approach also defined interval by two points, which has been pointed out may cause the Dividing Instant Problem.

Shoham [Sho1987] proposed a reified temporal logic STL, which accommodates a set of temporal constants and variables, as well as a set of non-temporal ones. A syntactic separation is made between temporal and non-temporal terms.

There are two kinds of atomic formulae:

(1) Formulae of temporal relations $tm_1=tm_2$ or $tm_1<tm_2$;

(2) Formulae of propositions TRUE(tm_1, tm_2, r), *r* is an n-ary non-temporal relation symbol.

However, as argued by Bacchus, Tenenberg, Koomen [BTK1991] and Vila [Vil1994], although Shoham himself claims that his logic is a new reified temporal one, it doesn't really deserve the qualification of reification. STL is essentially a temporal argument method which has been embedded in BTK system [BTK1991].

Reichgelt [Rei1989] proposed a reified temporal logic system TR which is actually developed totally within the framework of first order logic, and therefore inherits the syntax, semantics, axioms and inference rules of the first order theory. As a many sorted first order logic, TR does have a clear syntax, semantics and sound and complete axiomatization. But in fact, TR is so complicated that Reichelt hasn't provided adequate axioms to make it a real complete system, since some functions or predicates in TR are not axiomatic defined.

On the other hand, although TR is a truly reified temporal logic which has great expressive power, it is difficult for temporal knowledge representation, because every element has an expression form and a denotation form which have to be carefully distinguished.

Interval based

Allen [All1983, All1984] develops his theory of time and action ITL based on intervals as primitive rather than as derived structure from points. A set of 13 mutually exclusive binary relations between two intervals are introduced, i.e., EQUALS, BEFORE, AFTER, MEETS, MET_BY, OVERLAPS, OVERLAPPED_BY, STARTS, STARTED_BY, DURING, CONTAINS, FINISHES and FINISHED_BY, which may be formally defined in terms of the single primitive relation MEETS [AH1989]. While McDermott's basic entities associated with time is the dichotomy of facts and events, Allen introduces three ontological categories, i.e., *properties*, *events* and *processes*, to time intervals over which they hold or occur. Allen's properties are actually very similar to McDermott's facts with the only difference Allen interprets his properties over intervals, while McDermott interprets facts at points. Allen denotes the assertion that property p holds over interval i by the formula HOLDS(p, i). The assertion that event e occurs over interval i is denoted by Allen with the formula OCCUR(e, i). The most important divergence from McDermott's logic is that Allen takes time intervals as the primitive temporal objects, rather than making them up out of points. An advantage of such an approach is that it excludes the concept of ending-points of intervals, and hence overcomes the Dividing Instant Problem.

However, on one hand, as shown by Galton [Gal1990] in his critical examination of Allen's interval logic, if all intervals are characterized as infinitely decomposable, where time points are entirely excluded, then reasoning correctly about continuous change is inadequate.

Allen provided ITL neither a clearly defined semantics, nor a sound and complete axiomatization. The most important efforts of ITL is taking time intervals as the primitive temporal objects and fully discussing the relations of two intervals.

Point and interval based

Based on the classical first-order logic as the initial language, a truly reified temporal logic (MK) is developed by Ma and Knight [MK1996]. Such a many sorted system recasts various temporal ontology in a general framework. The syntax of RTL consists of: Terms in the reified language are also partitioned into three different types: *temporal terms*, *non-temporal terms* and *propositional terms*. Both temporal and non-temporal terms are defined in the standard first-order way, while propositional terms are defined in the form of standard formulae of the classical first-order language with each predicate being a non-temporal predicate taking only non-temporal terms as arguments. Predicates are distinguished as temporal predicates, non-temporal predicates and meta-predicates.

The MK system is truly a reified temporal logic which has general expressive power for temporal knowledge expression and deduction and allows one to reason about the truth of assertions over time while preserving the first-order structure of the propositions. Most of the significant ideas presented by [McD1982, All1984, Sho1987, Lif1987, and Gal1990], etc., are echoed within the framework.

However, the MK system doesn't enjoy a sound and complete axiomatic deduction system.

In a word, although most modal logic approaches, hybrid logic approaches, and temporal argument methods do have formal definition and complete axiomatization, their expressive power are limited; on the other hand, temporal reification methods are powerful enough to talk about general temporal assertions, they are not clearly defined or axiomatized.

Section 2.2 Review of Graph Matching Algorithm

Graphs are a powerful and versatile tool used for the description of structural objects which has been widely used in mathematics, computer science, artificial intelligence, biology, geography, or even politics, for representing structural objects and concepts. In general, in terms of their graph representation, parts of object can be represented by the vertices whilst the relationships between parts can be represented by the edges. Therefore, the task of calculating the similarity degree between two objects can be simply transferred into the problem of matching the corresponding pair of graphs.

There are a large number of applications of graph matching which have covered many different areas of human social and scientific life including image recognition [IZ1986, WFK1997, LRS1991, CR1992], robot vision [Won1992], chemical structure analysis [RB1979, TA1997], case based reasoning [Poo1993], machine learning [MB1996], videos indexing [SBV2001].

Various algorithms for graph matching problems have been developed, which, according to Gold and Rangarajan [GR1996], can be classified into two categories: (1) search-based methods which rely on possible and impossible pairings between vertices; and (2) optimization-based methods which formulate the graph matching problem as an optimization problem.

In this thesis, we classify the graph matching algorithms a bit different from above categories where traditional graph matching algorithms are classified into three groups:

explicit search methods, implicit search methods and node similarity based methods.

Section 2.2.1 Explicit search methods

Generally speaking, explicit search methods directly search the optimal match among permutation space (or permutation matrices space). Since the size of the search spaces increase exponentially according to the graph size, different kinds of heuristic techniques are developed to reduce the search space to a smaller acceptable size.

A widely known matching algorithm by Ullmann [Ull1976], based on a backtracking procedure with an effective look-ahead function to reduce the search space, is devised for both graph isomorphism and sub-graph isomorphism and is still today one of the most commonly used for exact graph matching because of its generality and effectiveness. As a consequence, many maximum sub-graph based or edit distance based explicit-search graph matching methods exist.

An backtrack search algorithm is presented by McGregor [McG1982] for the problem of finding the maximal common sub-graph of two graphs is described and used for analyzing chemical reactions and enumerating the bond changes which have taken place. In this note the problem is considered of finding the maximal common sub-graph of two given graphs.

Shapiro and Haralick [SH1981] associated the graph matching problem with a brute-force backtracking tree search and proposed corresponding algorithms to make the tree search faster. The similar methods can be found in [TF1983, CYS1996, and ACT1997].

Bunke and Shearer proposed a new distance measure based on the largest common sub-graph of two graphs in [BS1998] and corresponding search algorithm is characterized in [BFG2002]

Another important algorithm is introduced by Messmer and Bunke [MB1998] for error-tolerant sub-graph isomorphism determination. This approach requires exponential offline computational time and only polynomial online computational time. It also inspired the matching algorithms of [SBV2001].

In conclusion, most of these explicit search methods find the optimal solutions, but require exponential calculating time in the worst case. Although different kinds of heuristic techniques are applied, it is difficult to make the algorithm efficient all the time.

Section 2.2.2 Implicit search methods

Implicit-search methods do not search for the optimal match in permutation space; instead, the permutation space is transferred into some other continuous real number space or mixed 0-1 and real number space and meanwhile the graph matching problems is also represented as an optimization among the continuous or mixed space.

A linear programming approach is proposed by Almohamad and Duffuaa [AD1993] for the weighted graph matching problem, where the graph matching problem is formulated in L_1 norm as a linear programming problem with computational complexity $O(n^6L)$. Recently, Justice and Hero [JH2006] developed a binary linear program for computing graph *edit distance* (the minimum operation making two graph isomorphic), together with polynomial time methods for determining upper and lower bounds on the solution of the binary program are derived by applying solution methods for standard linear programming and the assignment problem.

A quadratic programming approach is proposed by Neuhaus and Bunke [NB2007] to computing the edit distance of graphs. Whereas the standard edit distance is defined with respect to a minimum-cost edit path between graphs, the notion of fuzzy edit paths between graphs is presented together with a quadratic programming formulation for the minimization of fuzzy edit costs.

Torsello and Hancock [TH2001, TH2003] transform the tree edit distance problem into a series of maximum weight clique problems and use relaxation labeling to find an approximate solution.

Many neural networks based approaches have been developed using Hopfield networks [MGA1989, FLD1994, CL1994, PG1995, and SY1998] or self-organizing map [XO1990, Sha1995, GB2002].

Some genetic algorithm based methods may be found in [CWH1997, LH1998, BMJ1999, MH2000, and MH2001, WI2006, Auw2007].

The computational complexity of these approaches is tightly dependent on the optimization problem the graph matching problem is reformulated. However, all these optimizations can either be solve in polynomial computational time to reach a local optimal solution or need exponential time to get a global optimal solution. Besides, since these approaches are tightly associated with the specific theories, they lead to another kind of complexity namely programming complexity, which means that although these programs cost only polynomial computational time, they cost lots of time to be designed and implemented.

In addition, since graph matching problems are represented as some optimization problems, it is usually difficult to analyze the reason of the failure cases, in other words, it is not easy to improve it or get a theoretical conclusion of their applicable fields.

Section 2.2.3 Node similarity based methods

The node similarity based methods do not use any search, instead, they simply explore some kind of node similarity between nodes of graph pairs, and get the optimal solutions by matching those similar nodes.

In [Ume1988], Umeyama proposed an *eigen-decomposition based graph matching algorithm* (EDGM) for matching both undirected and directed weighted graphs. The node similarity of two graphs is constructed based on the eigenvector of the adjacency matrices and the optimal solution is gained by applying the *Hungarian algorithm* [Kuh1955, mun1957, AMO1993] on the node similarity matrix.

In [Alm1991], Almohamod presented a *symmetric polynomial transform based graph matching algorithm* (SPGM), where the node similarity of two graphs is constructed by the coefficient of the polynomial transform of the weights of the edges.

In [Kle1999], Kleinberg proposed a *hubs and authorities graph matching algorithm* (HAGM) for internet searching, where the node similarity is based on the idea that two nodes are similarity if their adjacent nodes are similar. An iterative algorithm is provided to calculate such node similarity. This algorithm has been revised in [Zag2003, ZV2007].

In [Wyk2002], Wyk presented several Kronecker product successive projection based graph matching algorithms. The graph matching problem is transferred into the Kronecker Product Graph Matching formulation, based on which several approaches are derived, such as the *least squares Kronecker product graph matching* (LSKPGM) algorithm, the *interpolator-based Kronecker product graph matching* (IBKPGM) algorithm, the *gradient-based Kronecker product graph matching* (GBKPGM) algorithm and the *orthonormal kernel Kronecker product graph matching* (OKKPGM) algorithm.

Although these methods are derived from different theories, they are using the same idea to matching graphs by the nodes similarity. These methods are mostly only applicable for certain kinds of graphs, but they can be easily implemented, analyzed and improved. In addition, most of the node similarity based graph matching algorithms have low computational complexities and are consequently applicable to large size graphs.

Chapter 3 A Complete Reified Temporal Logic

In this chapter, a reified temporal logic with clear syntax and semantics in terms of a sound and complete axiomatic formalism will be proposed which retains all the expressive power of the current temporal reification formulations.

Section 3.1 simply reviews the standard first order logic and many sorted first order logic. Section 3.2 introduces three well-known temporal logic systems, BTK, TR and MK. The complete reified temporal logic CRTL is proposed in section 3.3. And a simplification of CRTL, named SCRTL, is presented in section 3.3.6, which is more applicable for real problems.

Section 3.1 First Order Logic

First-order logic is a formal deductive system used in mathematics, philosophy, linguistics, and computer science which employs a wholly unambiguous formal language interpreted by mathematical structures [Lu1989]. First order logic is a system of deduction extending propositional logic by allowing quantification over individuals of a given domain of discourse.

We follow the definition in [MT1993].

Section 3.1.1 Syntax

Firstly, we have to introduce the first order language.

Definition 3.1: The *alphabet* of a first order logic language consists of the following sets of symbols:

1) Logical symbols: logical implication $\rightarrow$, logical not $\neg$, quantifier for all $\forall$.

2) Equality symbol: $\equiv$

3) Variables: $V=\{x_0, x_1, x_2, \ldots\}$

4) Auxiliary symbols: "(" and ")".

NOTE1: Here only three logical symbols are selected and others are defined in terms of these three.

NOTE2: we use $\equiv$ as the equality symbol of the first order language to distinguish with the equality symbol $=$ of meta-language, the language we used to talk about first order language.

Definition 3.2: The *signature* of a first order logic language is a triple L=<FS, PS>, where

- FS is a countable set (possibly empty), whose elements are function symbols with a non-negative integer rank(f) for each function symbol f.

- PS is a countable set whose elements are predicate symbols with a non-negative integer rank(p) for each predicate symbol p.

It is assumed that the sets FS and PS are disjoint. Functions of rank 0 are called constants and the set of constants is denoted as CS.

Definition 3.3: The *terms* of a first order logic language L (referred as $TERM_L$) are inductively defined as:

1) Every constant and every variable is a term.

2) If $tm_1, tm_2, \ldots, tm_n$ are terms and f is a function of rank n>0, then $f(tm_1, tm_2, \ldots,$

tm_n) is a term.

Intuitively, a term is an expression stands for an element.

Definition 3.4: The *atomic formula* of a first logic language L is inductively defined as:

1) Every predicate of rank 0 is an atomic formula.

2) If tm_1, tm_2, ..., tm_n are terms and p is a predicate of rank n>0, then r(tm_1, tm_2, ..., tm_n) is a atomic formula, and so is $tm_1 \equiv tm_2$.

Definition 3.5: The *formula* of a first logic language L (referred as $Form_L$) is inductively defined as:

1) Every atomic formula is a formula.

2) For any two formulae α and β, $\neg\alpha$, $(\alpha \rightarrow \beta)$ are also formulae.

3) For any formula α and variable x_i, $\forall x_i \alpha$ is also a formula.

NOTE: as mentioned above, $\alpha \vee \beta$ is seen as abbreviation of $\neg\alpha \rightarrow \beta$, $\exists x \alpha$ is short for $\neg \forall x \neg \alpha$, and so on.

The formulae of $FORM_L$ are all the expressions can be talked in the first order logic L. Intuitively, a formula is an expression representing some property of the elements.

In the formulae $\forall x_i \alpha$, α is said to be the *field* of the quantifier $\forall x_i$.

Let x be a variable appear in formula α, obviously, x may appear in several places in α, each appearance is called an *occurrence.*

Definition 3.6: An occurrence of x in formula α is said to be *bounded occurrence* if it appears as the form $\forall x$, or in the field of quantifier $\forall x$. In this case, x is said to be a *bounded variable* of formula α. BV(α) denotes all the bounded variables of formula α.

Definition 3.7: An occurrence of x in formula α is said to be *free occurrence* if it is not bounded. In such case, x is said to be a *free variable* of the formula α. FV(α) denotes all the bounded variables of formula α.

Obviously that a variable x can be both free and bounded occur in formula α, which means BV(α)$\cap$FV(α) may not be empty.

The concept of free and bounded variables is very important for the substitutions of formulae

Definition 3.8: The result of *substituting* term tm for variable x in a term tm' or formula α is recursively defined as:

1) y[tm/x]= if y=x then t else y, when tm' is the variable y.

2) c[tm/x]=c, when tm' is the constant c.

3) f(tm_1, tm_2, ..., tm_n)[tm/x]= f(tm_1[tm/x], tm_2[tm/x], ..., tm_n[tm/x]), when tm' is the term f(tm_1, tm_2, ..., tm_n).

4) p(tm_1, tm_2, ..., tm_n)[tm/x]= p(tm_1[tm/x], tm_2[tm/x], ..., tm_n[tm/x]), when α is the formula p(tm_1, tm_2, ..., tm_n).

5) $(\beta\rightarrow\gamma)$[tm/x]=β[tm/x] $\rightarrow\gamma$[tm/x]

6) $(\neg\beta)$[tm/x]= $\neg$ (β[tm/x])

7) $\forall y\beta$[tm/x]= if x=y then $\forall y\beta$ else $\forall y$(β[tm/x])

Intuitively, only free occurrences of x can be substituted by term tm.

Definition 3.9: It is said that a *term tm is free for x in formula α*, if every occurrence of x in formula α is not in the filed of $\forall$y, where y is a free variable of term tm.

A special case is that variable x is always free for x in any formula α.

For example, let tm=1-y, and α=$\exists$y(x+y=0), the occurrence of x in α is in the field of $\exists y$, so t is not free for x in α. If we substitute x by t in α, we get

$\alpha[tm/x] = \exists y(1-y+y\equiv 0) = \exists y(1 \equiv 0)$

This is absurd in some sense.

Till now, the language of a first order logic system is defined clearly and next step is providing enough axioms and rules to deduct theorems.

Definition 3.10: The *axioms* of a first order logic system consist of the following:

- Propositional axioms:

  Ax1. $\alpha \to (\beta \to \alpha)$

  Ax2. $(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$

  Ax3. $(\neg\beta \to \neg\alpha) \to (\alpha \to \beta)$

- Quantifier axioms:

  Ax4. $\forall x(\alpha \to \beta) \to (\forall x\alpha \to \forall x\beta)$

  Ax5. $\forall x\alpha \to \alpha[tm/x]$, such that t is free for x in α.

  Ax6. $\alpha \to \forall x\alpha$, such that $x \notin FV(\alpha)$.

- Equality axioms:

  Ax7. $tm \equiv tm$

  Ax8. $tm_1 \equiv tm_1' \wedge \cdots \wedge tm_n \equiv tm_n' \to f(tm_1, tm_2, \cdots, tm_n) \equiv f(tm_1', tm_2', \cdots, tm_n')$

  Ax9. $tm_1 \equiv tm_1' \wedge \cdots \wedge tm_n \equiv tm_n' \to p(tm_1, tm_2, \cdots, tm_n) \to p(tm_1', tm_2', \cdots, tm_n')$

- **Generalization:**

Ax10: $\forall x\alpha$, if α is an axiom.

Definition 3.11: A first order logic system L has only *MP-rule*, that is:

MP: $\{\alpha, \alpha \rightarrow \beta\} \vdash \beta$

Definition 3.12: A *formula α is said to be provable by a set of formulae Γ* (referred to $\Gamma \vdash \alpha$), if there exists a finite formula series α_1, α_2, ..., α, where each α_k is in Γ, or a axioms, or is gained by applying the MP-rule for two former formula α_i and α_j (i,j,<k).

This definition is the formal description of what a mathematical proof is. The relation $\vdash$ between a formula set Γ and a formula α is usually called *deducibility relation*. Formulae proved by empty set ($\vdash \alpha$) is called *theorems*.

Section 3.1.2 Semantics

Definition 3.13: Given a first order logic system L, a *Model* is a pair M=<D, I>, where D is a non-empty set called *Domain* (or *Universe*, or *Carrier*), I is an *interpretation function* which interpret the signature of L as:

1) For each function symbol f with rank n, $I(f):D^n \rightarrow D$ is an n-ary function.

2) For each predicate symbol p with rank n, $I(p):D^n \rightarrow \{0, 1\}$ is an n-ary predicate.

3) For each variable x, $I(x) \in D$ is an element of D.

Given a Model M of L, the constants and variables are interpreted directly and the term $f(tm_1, tm_2, \cdots, tm_n)$ is interpreted by

$$I(f(tm_1, tm_2, \cdots, tm_n)) = I(f)(I(tm_1), I(tm_2), \cdots, I(tm_n)).$$

For the interpretation I, the symbol $I_{x=a}$ stands for the same interpretation as I, except the variable x is interpreted as a. And for a model M=<D, I>, the model $M_{x=a}$ is defined as <D, $I_{x=a}$>.

Definition 3.14: Given a model M, the formulae of L are interpreted as:

1) $I(p(tm_1, tm_2, \cdots, tm_n)) = I(p)(I(tm_1), I(tm_2), \cdots, I(tm_n))$

2) $I(\alpha \to \beta) = (1 - I(\alpha)) \times I(\beta)$ and $I(\neg\alpha) = 1 - I(\alpha)$.

3) $I(\forall x \beta) = \inf_{a \in D} \{I_{x=a}(\beta)\}$

Definition 3.15: Given a model M=<D, I> of first order logic L,

1) A formula α is said to be satisfied by M (referred as M $\models \alpha$) if $I(\alpha)=1$.

2) A set of formula Γ is said to be satisfied by M if M $\models \alpha$ for all $\alpha \in \Gamma$

3) A formula α is said to be satisfied by a formula set Γ (referred as $\Gamma \models \alpha$), if for all model M, M $\models \Gamma$ implies M $\models \alpha$.

The relation $\models$ between a formula set Γ and a formula α is usually called *entailment relation*. Formulae can be satisfied by any model (denoted as $\models \alpha$) is called *tautology*.

Section 3.1.3 Soundness and Completeness

In the last two sections, two relations, we introduced two relations, deducibility $\vdash \alpha$ and entailment $\models \alpha$. The equivalence between these two is termed the *soundness* and *completeness*.

Theorem 3.1: Soundness of FOL. If $\Gamma \vdash \alpha$, then $\Gamma \models \alpha$.

Theorem 3.2: Completeness of FOL. If $\Gamma \models \alpha$, then $\Gamma \vdash \alpha$.

Especially, theorems and tautologies are equivalent.

Section 3.1.4 Many Sorted First Order Logic

These are situations in which it is desirable to express properties of structures of different types or sorts. By adding to the formalism of first order logic the notion of type, one obtains a flexible and convenient logic called *many sorted first order logic*, which enjoys the same property as first order logic. [Wan1952, Gil1956]

Definition 3.16: The alphabet of a many sorted first order logic system consists of the following symbols:

1) A countable set $S\cup\{bool\}$ of sorts.

2) Logical Connectives: $\rightarrow$of rank ($bool^2\rightarrow bool$), $\neg$ of rank ($bool\rightarrow$ bool)

3) Quantifiers: for every sort $s\in S$, there is a quantifier $\forall_s$ of rank ($bool\rightarrow bool$)

4) Equalities: for every sort $s\in S$, there is an equality symbol$\equiv_s$.

5) Variables: for every sort $s\in S$, there are countable infinite variables $V_s=\{x_0:s, x_1:s, x_2:s, \ldots\}$.

6) Auxiliary symbols "(" and ")".

The signature also has to be typed.

Definition 3.17: The signature of $S\cup\{bool\}$ ranked first order logic language consists of:

- Function symbols: A set FS of function symbols with a rank function rank: $S^*\rightarrow S$. The function with rank ($e\rightarrow s$) is called a constant of sort s.

- Predicate symbols: A set PS of predicate symbols with a rank function rank: $S^*\rightarrow\{bool\}$. The predicate with rank($e\rightarrow\{bool\}$) is called a *propositional constant.*

All the other definitions of terms, atomic formulae, formulae, axioms and inference rules are the same as FOL.

To define a many sorted logic system, one only has to define the sorts, the signatures and special axioms in that theory.

Definition 3.18: Given a many sorted first order logic L with sort S. A model M is defined as:

1) For each sort $s \in S$, there is a non-empty set M_s as its Domain.

2) Each function symbol f rank $(s_1 \ldots s_n \rightarrow s)$ is interpreted as a function $f_M : M_{s_1} \times \cdots \times M_{s_n} \rightarrow M_s$.

3) Each predicate symbol p with rank $(s_1 \ldots s_n \rightarrow bool)$ is interpreted as a predicate $r_M : M_{s_1} \times \cdots \times M_{s_n} \rightarrow \{0,1\}$.

4) Each variable symbol x:s is interpreted as a element $(x:s)_M$ in M_s.

All the other interpretations of terms, atomic formulae, formulae are the same as FOL.

To define a model for a many sorted first order logic, one just has to define its domain of each sort, and the interpretations of function symbols, predicate symbols and variables.

Theorem 3.3: Soundness and Completeness. $\Gamma \vdash \alpha$, if and only if $\Gamma \models \alpha$. [Wan1952, Gil1956]

The many sorted first order logic is frequently used in the rest of the thesis. Each time we only need to describe the sorts, signature and interpretation of the signature.

Section 3.2 Predicate Temporal Logics

We shall discuss some predicate based temporal logic systems which are tightly associated with our research.

Section 3.2.1 BTK

BTK is a non-reified temporal logic system proposed by Bacchus, Tenenberg and Koomen in [BTK1991]. BTK system is exactly a standard many sorted first order logic with sorts S={t, u}, where t stands for the temporal elements and u stands for the non-temporal elements.

Syntax

The signature of BTK is L_{BTK}=<FS, PS>, where

- FS is the set of function symbols. Every function f $\in$FS has a rank $t^n \times u^m \rightarrow t$ (temporal function) or $t^n \times u^m \rightarrow u$ (non-temporal function).
- PS is the set of predicate symbols. Every predicate p$\in$PS has a rank $t^n \rightarrow \{bool\}$ (temporal predicate), or $u^m \rightarrow \{bool\}$ (non-temporal predicate), or $t^n \times u^m \rightarrow \{bool\}$ (mixed predicate)

All the other definitions of terms, formulae, axioms and rules are in the standard fashion.

The syntax structure of BTK can be illustrated as figure 3.1

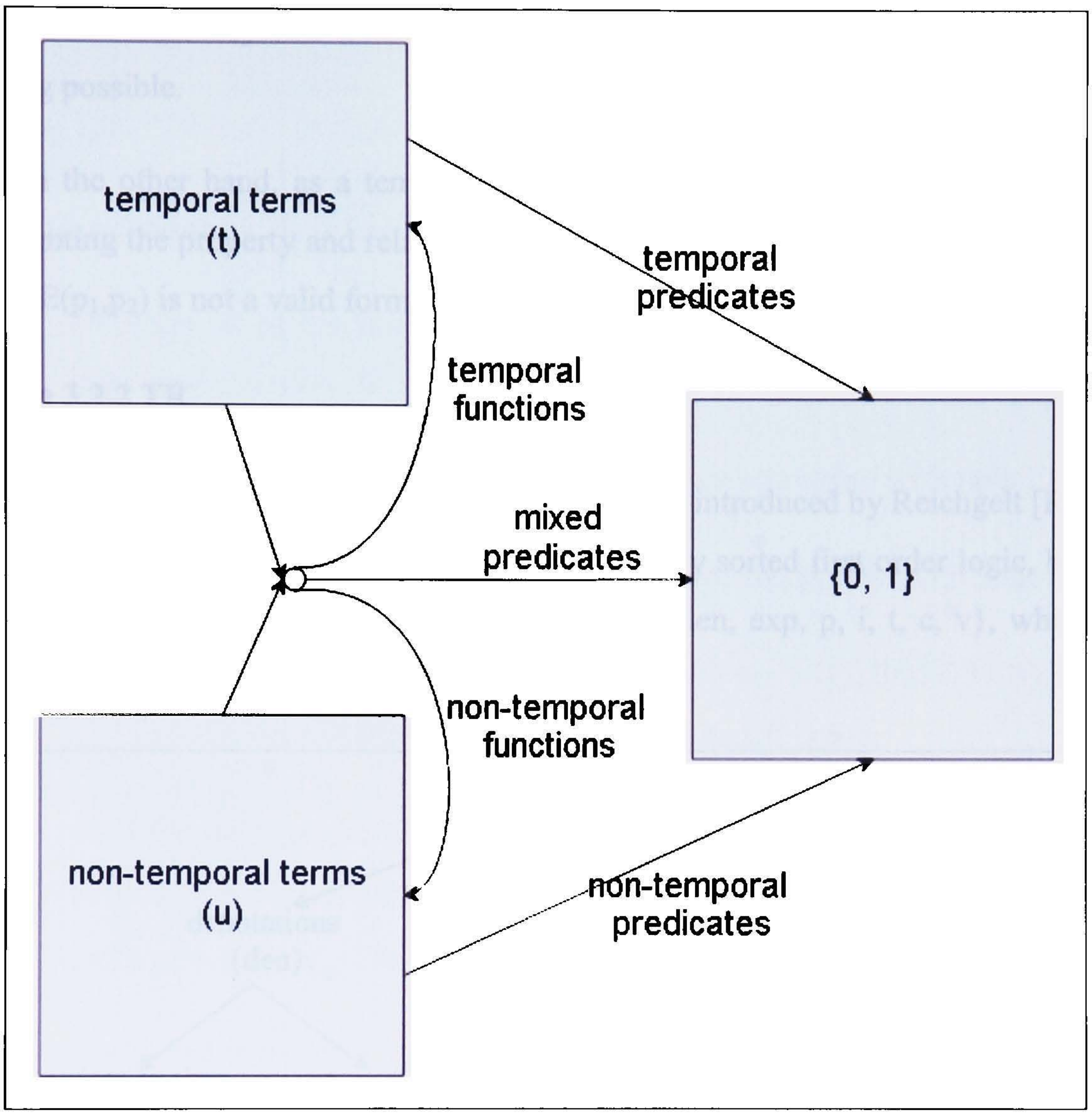


Figure 3.1 Structure of BTK

Semantics

A model of BTK is defined as a triple M=<T, U, I>, where T and U are domains of temporal elements and non-temporal elements respectively. I is a function interpreting the signatures and variables of BTK in the standard way.

Property

As a many sorted first order logic system, BTK has simple, clear syntax, which makes it easy to understand and apply. More important, BTK inherits the sound and complete inference system from first order logic, which makes the automatic theorem proving possible.

On the other hand, as a temporal argument method, BTK has expressive limit in representing the property and relation between the predicates. For example, the sentence CAUSE(p_1,p_2) is not a valid formula in BTK.

Section 3.2.2 TR

The TR system is a reified temporal logic system introduced by Reichgelt [Rei1989]. Similarly as BTK, the TR logic system is also a many sorted first order logic, but more complicated. The sorts of TR are S={den, t-den, d-den, exp, p, i, t, c, v}, which have been organized as Figure 3.2.

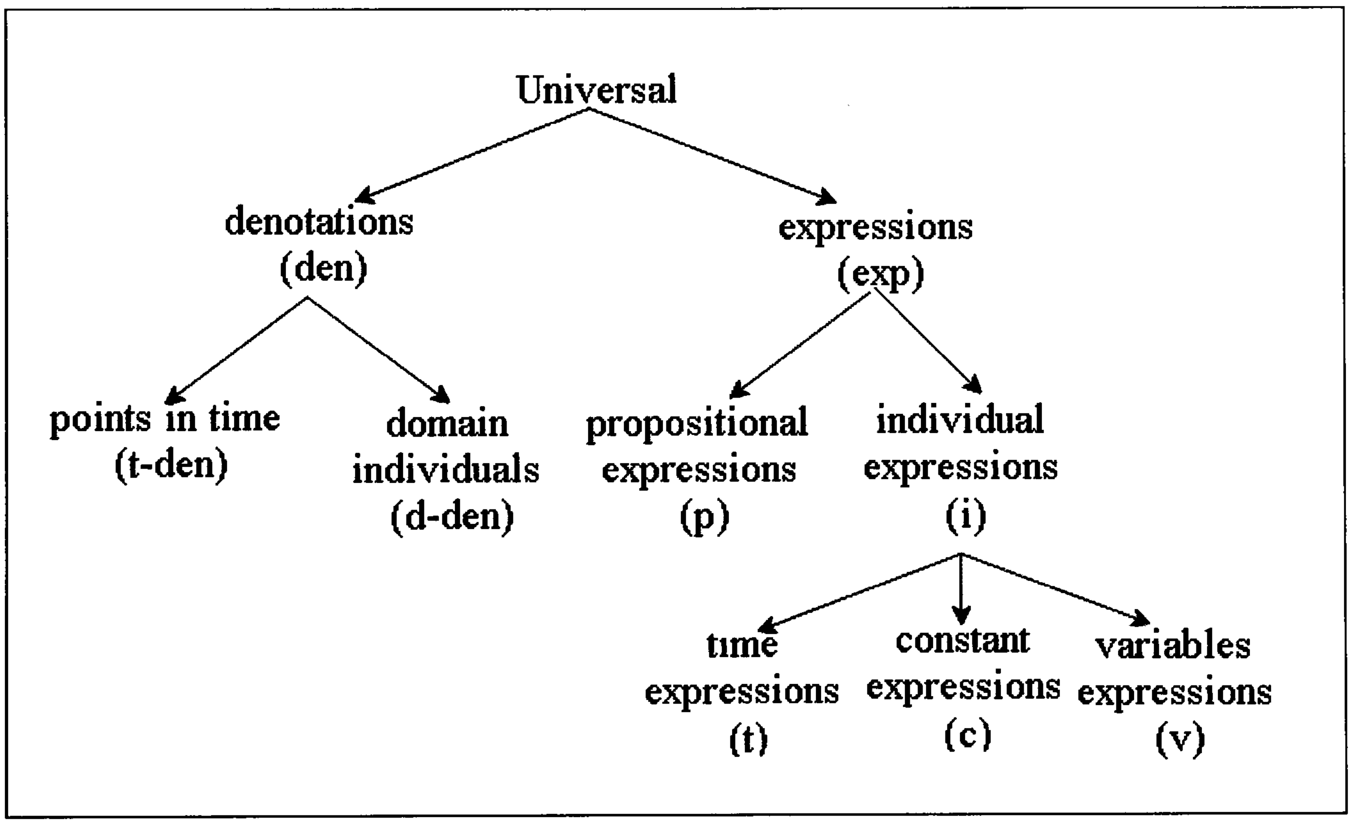


Figure 3.2 Sort hierarchy for TR [Rei1989]

Informally speaking, TR reified both syntax and semantics of temporal argument methods, where expression exp and denotation den stand for the reifications of syntax and semantics respectively. The sub-sorts of den, t-den and d-den, express the temporal domain and non-temporal domain respectively. And sub-sorts of expression exp, p and i, express the formula and terms of temporal argument methods. Time expression t, constant expression c and variable expression v stand for three special kinds of terms.

Syntax

The signature of TR is L_{TR}=<FS, PS>, where

1) FS is the set of function symbols, which can be can be split into three groups: individual function DF, propositional function PF and logical function LF.

   1.1) Each element of DF is an individual function symbol with rank $i^n \rightarrow i$.

   1.2) Each element of PF is a propositional function symbol with rank $i^n \rightarrow p$

   1.3) LF consists of AND, IF, OR with rank $p^2 \rightarrow p$, NOT, PAST, FUTURE with rank p→p and FORALL, THEREIS with rank i×p→p

2) PS is the set of predicate symbols. PS={HOLDS (with rank form×t-den→bool), < (with rank $t\text{-}den^2 \rightarrow bool$), EXISTS (with rank d-den×t-den→bool), T-DEN (with rank t×t-den→bool) and DEN (with rank c×d-den×t-den→bool)}.

Informally speaking, each n-ary function of temporal argument method is still an n-ary function with the same rank; each n-ary predicate of temporal argument method is revised as a propositional function from n terms to propositional expressions; the logical connectives and quantifiers of temporal argument method are revised as logical functions from one propositional expression to another. HOLDS relation capture the meaning of entailment of temporal argument method; $<_{TR}$ indicate the time structure; EXISTS express if an individual exists at a time point. T-DEN interprets every time constant as a time point and DEN interpreting the constant symbol as an individual at some time point.

These complex sorts and signatures make structure of TR very complicated. Intuitively, the structure of TR can be illustrated as figure 3.3.

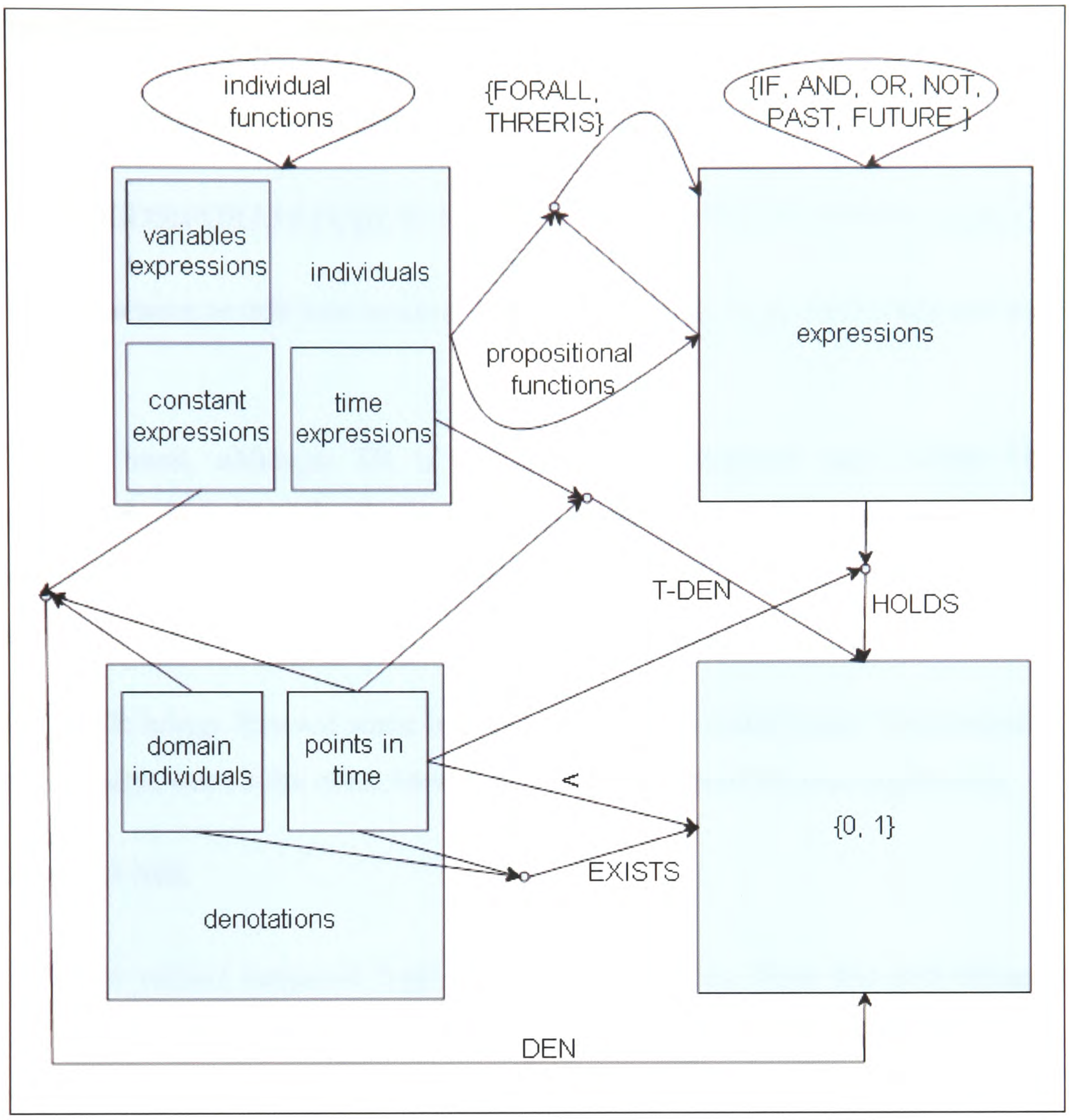


Figure 3.3 Structure of TR

Semantics

Since TR is a many sorted first order logic system, its semantics can simply be provided by following the semantic definition of standard many sorted first order logic.

Property

As a many sorted first order logic, TR should have a clear syntax, semantics and sound and complete axiomatization. But in fact, TR is too complicated to provide adequate axioms to make it a real complete system. For example, to describe the function FORALL clearly, Reichgelt uses the following axiom:

$$\forall p{:}p\forall t{:}t\text{-den}\forall v{:}v\forall d{:}d\text{-den}\forall c{:}c$$

$$\mathrm{HOLDS}(\mathrm{FORALL}(v, p), t)\rightarrow \mathrm{DEN}(c, d, t) \rightarrow \mathrm{HOLDS}(\mathrm{SUBST}(c, v, p), t)$$

Then one have to add new axiom to define SUBST(c, v, p), which has not been done in [Rei1989].

On the hand, although TR is a truly reified temporal logic which has great expressive power, it is difficult for temporal knowledge representation, because every element has an expression form and a denotation form which have to be carefully distinguished.

In all, TR brings forward some brilliant idea to reify both syntax and semantics of an object language, but on the other hand, it is too complicated for real application.

Section 3.2.3 MK

MK is a reified temporal logic system proposed by Jixin Ma and Brian Knight [MK1996]. MK is again a many sorted predicate logic with sort S={t, u, p} stand for temporal terms, non-temporal terms and propositional terms respectively.

Syntax

MK is not given by the standard first order logic language; instead, it is defined by its own syntax and semantics which are very similar to standard first order logic. MK has the following signatures:

- **TC:** a set of temporal individual symbols;
- **TV:** a set of temporal variables;
- **UC:** a set of non-temporal individual symbols;
- **UV:** a set of non-temporal variables;

- **TF**: a set of temporal function symbols;
- **UF**: a set of non-temporal function symbols;
- **LF**: the set of connectives functions {NOT, AND}
- **TP**: a set of temporal predicate symbols;
- **UP**: a set of non-temporal predicate symbols;
- **MP**: a set of meta-predicate symbols.

The terms and formulae of MK are defined in the same way of many sorted first order logic. The structure of MK can be illustrated as figure 3.4

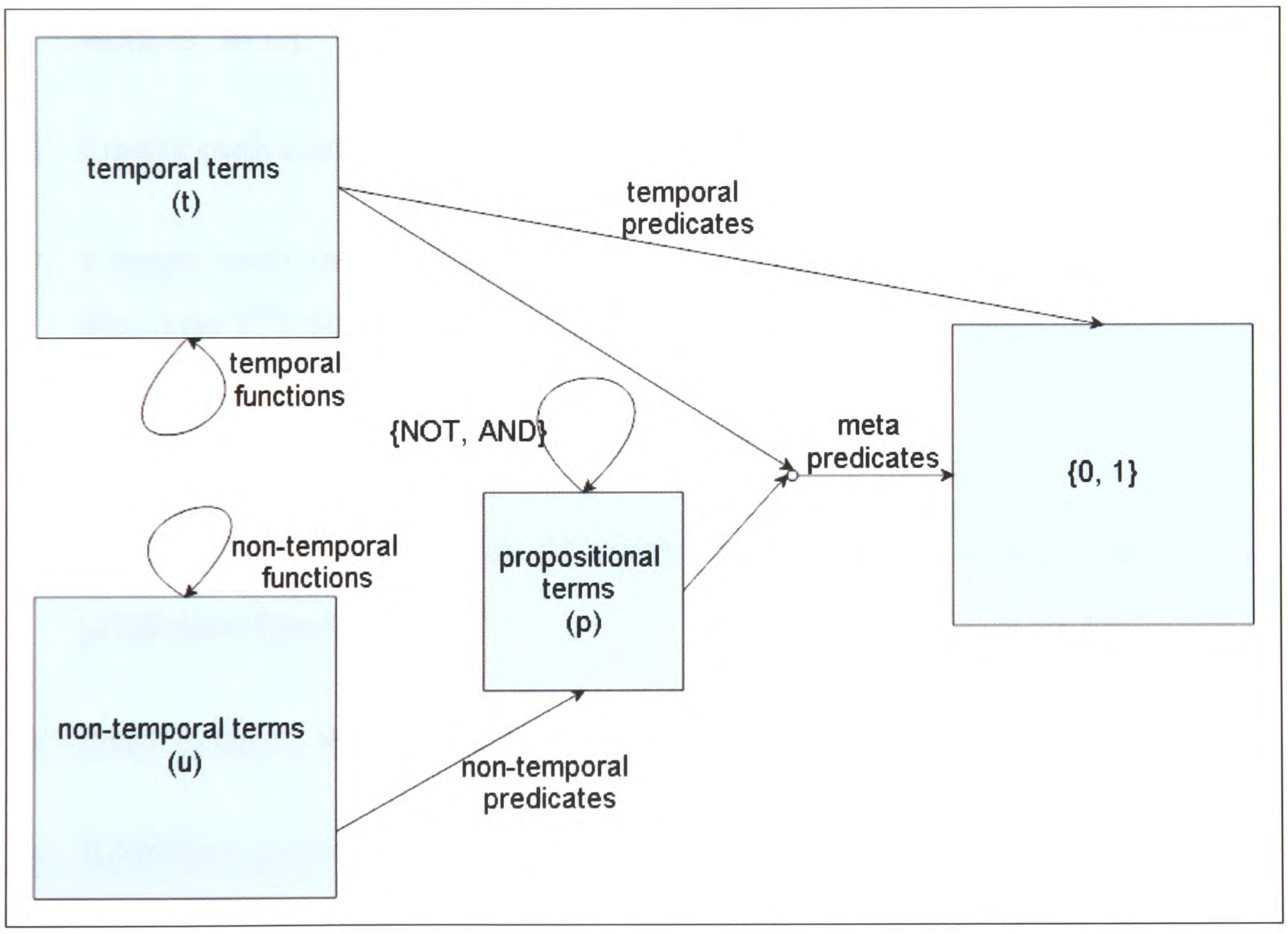


Figure 3.4 Structure of MK

Figure 3.4 shows that the structure of MK is also simple and clear as BTK does.

Semantics

The model of MK is defined as a triple M= <T, U, I>, where T is a nonempty universe of temporal individuals; U is a nonempty universe of non-temporal individuals; and I is an interpretation function, such that:

1) I maps each temporal individual symbol to a member of T;

2) I maps each non-temporal individual symbol to a member of U;

3) I maps each n-ary temporal function symbol tf to an n-ary function I(tf) from T^n to T;

4) I maps each n-ary non-temporal function symbol uf to an n-ary function I(uf) from U^n to U;

5) I maps each n-ary temporal predicate symbol tp to an n-ary relation I(tp) on T^n;

6) I maps each (m+n)-ary meta-predicate symbol mp to a (m+n)-ary predicate I(mp) on $T^m \times (I(Term_{Propositional}))^n$.

In Ma's original work, $I(Term_{Propositional})$ is defined by:

1) Each n-ary non-temporal predicate symbol up is interpreted as an n-ary predicates I(up) on U^n;

2) $I(up(u_1, u_2,\ldots, u_n))=I(up)(I(u_1), I(u_2), \ldots, I(u_n))$

3) $I(AND(p_1, p_2))=I(p_1)\wedge I(p_2)$

4) $I(NOT(p))=\neg(I(p))$

This definition interprets the propositional term as true or false, $I(Term_{Propositional})=\{0, 1\}$, so a meta-predicate is interpreted on $T^m\times \{0, 1\}^n$. We shall revise this work by re-define the interpretation $I(Term_{Propositional})$ by:

1) Each n-ary non-temporal predicate symbol up is interpreted still as the symbol itself.

2) $I(up(u_1, u_2,\ldots, u_n))=I(up)(I(u_1), I(u_2), \ldots, I(u_n))$

3) $I(AND(p_1, p_2))=I(p_1)$ AND $I(p_2)$

4) I(NOT(p))=NOT(I(p))

This definition interprets the propositional terms as *propositional tokens*. And meta-predicates are interpreted on time and propositional tokens, which is more reasonable.

Property

The MK system is truly a reified temporal logic which has general expressive power for temporal knowledge expression and deduction. However, MK system has not got the complete axioms to deduce all the theorems. Although it is believable that the axiomatic system of MK would be very similar to the many sorted first order logic, we will not provides these axioms, instead, we will revise it under the first logic framework.

Section 3.3 The Complete Reified Temporal Logic System CRTL

To overcome the weakness of existing reified temporal logic systems, we proposed a well-defined, completely axiomatized reified temporal logic, named CRTL, which is also simple and applicable for real problems.

Section 3.3.1 Syntax

CRTL is a standard many sorted first order logic with sorts S={t, u, p} stands for temporal terms, non-temporal terms and propositional terms.

Definition 3.19: The signature of the CRTL is a pair L_{CRTL}=<FS, RS>, where

1) FS is the set of function symbols, which can be can be split into four groups: TF,

UF, RF and PF defined as:

1.1) Each symbol tf in TF is a temporal function with rank $t^n \times u^m \rightarrow t$.

1.2) Each symbol uf in UF is a non-temporal function with rank $t^n \times u^m \rightarrow u$

1.3) Each symbol pf in PF is a propositional function with rank $t^n \times u^m \rightarrow p$

1.4) Each symbol lf in LF is a logical function with rank $p^n \rightarrow p$

2) PS is the set of relation symbols which can be split into three groups:

2.1) Temporal predicates TP with rank $t^n \rightarrow$bool

2.2) Non-temporal predicates UP with rank $u^n \rightarrow$bool

2.3) Meta predicates MP with rank $t^n \times p^m \rightarrow$bool.

In some sense, CRTL is an intermix system of BTK, TR and MK. The two kinds of functions, temporal and non-temporal functions, and three kinds of predicates are reserved in CRTL, only the mix-predicates are revised as meta-predicates. TR's reification of the syntax part is used here, where formulae of temporal argument methods are re-interpreted as propositional terms here. The structure of MK system is kept by CRTL with some necessary extensions.

The structure of CRTL can be illustrated as figure 3.5.

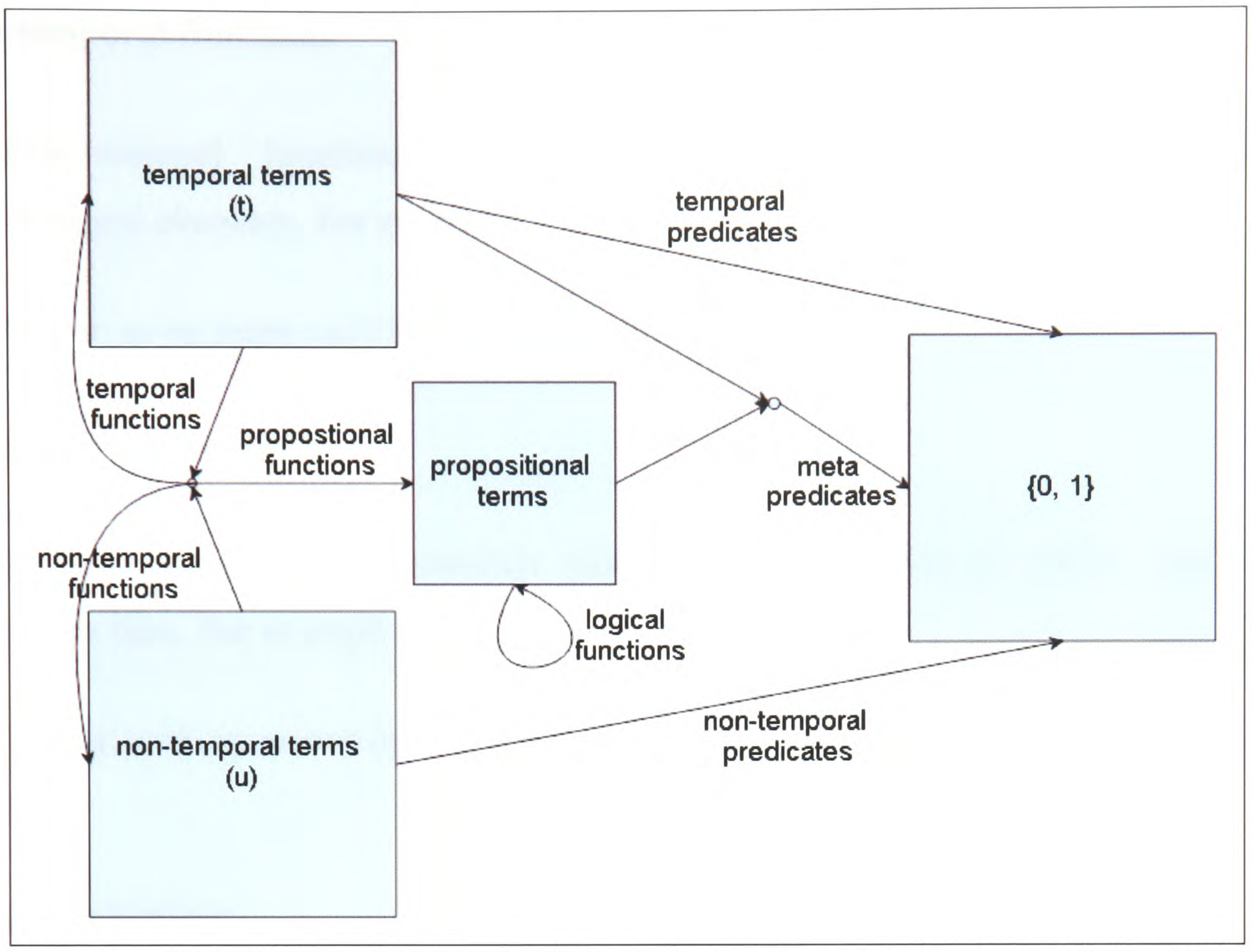


Figure 3.5 Structure of CRTL

Section 3.3.2 Examples of all kinds of functions and predicates

Here we use a simple example to demonstrate all kinds of functions and predicates.

We choose the human society as an example, where non-temporal sort u stands for all the humans and t stands for the time we used in our daily life composed of years, months and days which is formally expressed as integer-like time.

Temporal functions

Temporal functions map temporal or/and non-temporal elements to temporal elements. For example:

Next_day: t→t maps a day to its next day.

Birthday: u→t maps each human to its birthday.

Non-temporal functions

Non-temporal functions map temporal or/and non-temporal elements to non-temporal elements. For example:

Father: u→u maps each human to its father.

Propositional functions

Propositional functions construct some propositional tokens whose truth values depend on time. For example:

Richer: u×u→p means one person is richer than the other, which holds on the certain time.

Logical functions

Logical functions are constructors for complex propositions. For example:

AND: p×p→p is the conjunction of propositional terms.

NOT: p→p is the negation of propositional terms

FORALLu$_j$: p→p, FORALLt$_j$: p→p are the universal quantification of propositional terms.

Temporal predicates

Temporal predicates express the relations of time elements. For example:

MEETS: t×t→{0, 1} denotes the immediate predecessor order relation over time elements:

Non-temporal predicates

Non-temporal predicates express the relations of non-temporal elements. For example:

Older: $u\times u\rightarrow\{0, 1\}$ means one person is older than the other.

Meta-predicates

Meta-predicates express the relations of propositional tokens. For example:

HOLDS: $p\times t\rightarrow\{0, 1\}$ denotes proposition p is true at time t.

CAUSES: $t^3\times p^3 \rightarrow\{0, 1\}$, where CAUSES($t_1$, t, t_2, p_1, p, p_2) denote a causal law, which intuitively states that, under the precondition that proposition p_1 hold true over time t_1, the truth holding of proposition p over time t will cause the truth holding of proposition p_2 over time t_2.

Section 3.3.3 Semantics

The semantics of CRTL follows standard many sorted first order logic semantics. A model M of CRTL is a 4-tuple M=<T, U, P, I>, where T, U and P are non-empty domain of temporal elements, non-temporal elements and propositional terms. I is the interpreting function such that:

1) Each temporal function symbol tf is interpreted as a function $T^n\times U^m\rightarrow T$.

2) Each non-temporal function symbol uf is interpreted as a function $T^n\times U^m\rightarrow U$

3) Each propositional function symbol pf is interpreted as a function $T^n\times U^m\rightarrow P$

4) Each logical function symbol lf is interpreted as a function $P^n\rightarrow P$

5) Each temporal predicate symbol tp is interpreted as a predicate on T^n

6) Each non-temporal predicate symbol up is interpreted as a predicate on U^n

7) Each meta temporal predicate symbol mp is interpreted as a predicate on $T^n\times P^m$

The interpretation is the same as the revised interpretation of MK system, where function symbols are interpreted as temporal or non-temporal functions, predicate symbols are interpreted as temporal, non-temporal and meta-predicates. Meta-predicates take time elements and propositional terms (or propositional tokens) as its argument, and this make it possible to qualification over all propositional terms.

Section 3.3.4 CRTL with Durations

In some circumstance, the problem solution depends on not only the meets relations of time elements, but also their durations. In order to formally define such a system, a new sort d is added to denotes the length metric and a duration assigning function DUR is added to indicate the duration of every time element.

The structure is show as figure 3.6.

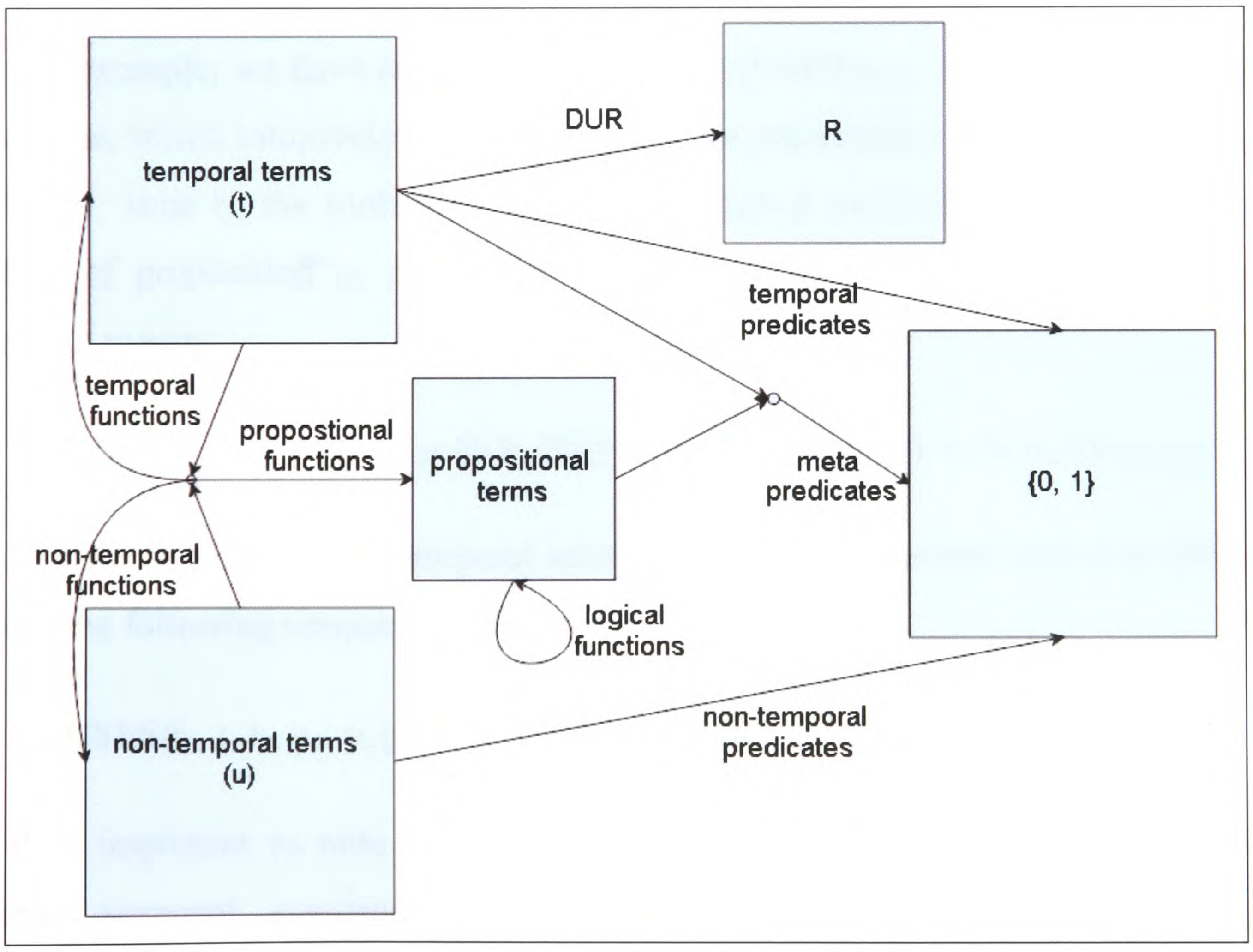


Figure 3.6 Structure of CRTL with duration

The MK system is somehow a reified temporal logic with durations, and now can be formally embedded in the CRTL system.

To reduce complexity, usually the non-negative real or rational numbers are chosen as the domain of the durations. However, this choice is not mandated.

Section 3.3.5 Property

From a theoretical view, CRTL is a many sorted first order logic which enjoys clear definition of syntax and semantics, sound, complete axioms and deduction rules. This makes it feasible to directly use Herbrand principal for theorem determination and auto-proof.

On the other hand, from a practical view, CRTL is an extension of the reified logic system MK, which means CRTL has even more general expressive power for temporal knowledge representation and reasoning.

For example, we have introduced formula CAUSES(t_1, t, t_2, p_1, p, p_2) denoting the causal law, which intuitively states that, under the precondition that proposition p_1 hold true over time t_1, the truth holding of proposition p over time t will cause the truth holding of proposition p_2 over time t_2. This can be formally characterized by the following axiom:

$$\text{CAUSES}(t_1, t, t_2, p_1, p, p_2) \wedge \text{HOLDS}(p_1, t_1) \wedge \text{HOLDS}(p, t) \rightarrow \text{HOLDS}(p_2, t_2)$$

In order to characterize temporal relationships between events and their effects, we impose the following temporal constraints:

$$\text{CAUSES}(t_1, t, t_2, p_1, p, p_2) \rightarrow \text{MEETS}(t_1, t) \wedge (\text{MEETS}(t_1, t_2) \vee \text{Before}(t_1, t_2))$$

It is important to note that above axiom actually specifies the so-called (most) *general temporal constraint* (GTC) [Sho1987]. Such a GTC guarantees the common-sense assertion that “the beginning of the effect cannot precede the beginning of the cause”, namely, there is a time delay between the event and its effect.

Section 3.3.6 The Simplified CRTL System SCRTL

The logic system CRTL describes a framework for temporal knowledge representation and deduction. It mainly aims at the logical completeness and expressive power, in the meanwhile, it lose some efficiency.

SCRTL is a simplified system of CRTL aiming at simplicity and efficiency. The sort of SCRTL is $S=\{t, p, d\} \cup \{bool\}$, where non-temporal terms are no longer considered.

The signature of SCRTL is L=<FS, PS>, where

- FS contains the duration function DUR with the rank t→d and the propositional constants $P_1, P_2, \ldots, P_k$ with rank e→p
- PS contains the temporal predicate MEETS with rank t^2 and the meta-predicate HOLDS with rank $p \times t$

The structure of SCRTL is illustrated as figure 3.7, which is simple and easy for practical applications.

Informally speaking, SCRTL only considers n propositions $P_1, \ldots, P_n$ whose truth value vary with time, and the time structure is described by MEETS relation and DUR function.

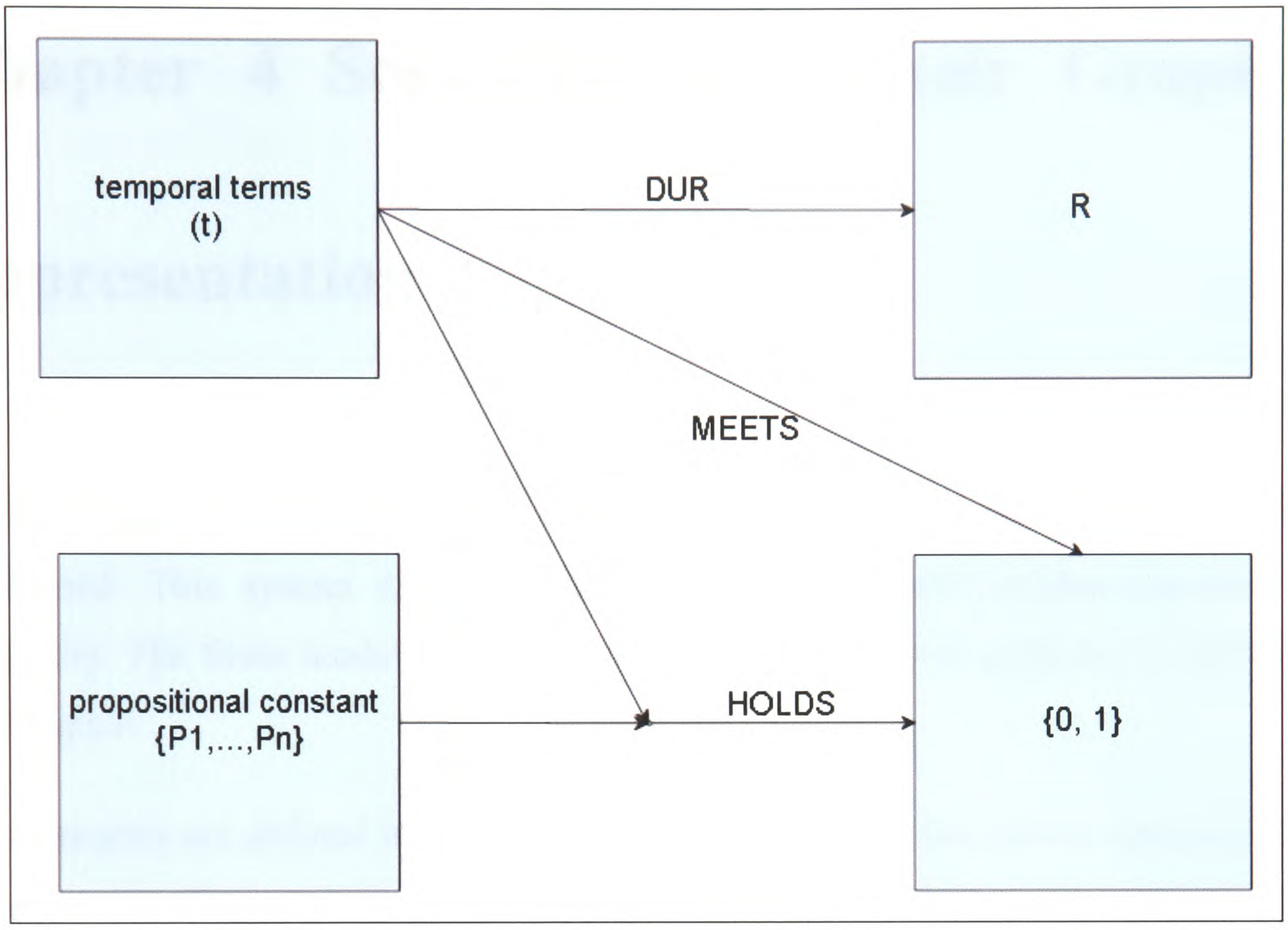


Figure 3.7 Structure of SCRTL

The application of SCRTL will be discussed in detail in next chapter.

Chapter 4 Scenarios and Their Graphical Representation

In chapter 3, the simplified complete reified temporal logic system SCRTL was introduced. This system can be expediently used for real problem because of its simplicity. The finite model of SCRTL, which will be termed scenario, is discussed in this chapter.

Scenarios are defined in section 4.1 and the graphical and matrix representation of scenarios are discussed in section 4.2 and section 4.3. Section 4.4 introduces some extensions for representing incomplete knowledge and general time relations and section 4.5 discusses the matching of scenarios.

Section 4.1 Definition of Scenarios

Let us Consider a finite model of SCRTL $M=<T, P, R, I>$, where T is a finite set of time elements, P is a finite set of primitive propositions and I is the interpreting function that:

1) Every propositional constant P_j is interpreted as a primitive proposition in P.

2) $HOLD_M \subseteq P \times T$

3) $MEETS_M \subseteq T \times T$

4) $DUR_M: T \rightarrow R$

We will study this model in details for representing temporal knowledge.

Section 4.1.1 Time

For the reason of general treatments, we shall take the time theory proposed previously by Ma and Knight [MK1994] as the temporal basis. This time theory addressed both points and intervals as temporal primitives on an equal footing: neither points have to be defined as limits of intervals, nor intervals have to be constructed out of points. The distinction between time intervals and time points is characterized by means of a duration assignment function, DUR, from the set of time elements to non-negative real numbers, i.e., R^{+0}. A time element t is called an (time) interval if DUR(t) > 0; otherwise, t is called a (time) point. Such a temporal theory is indeed an extension to the interval-based axiomatization of Allen and Hayes [AH1989]. As shown in [MK1994], analogous to the 13 relations introduced by Allen for intervals [All1983], there are 30 distinct temporal relations over time elements including both intervals and points, which can be derived from the single immediate predecessor relation, "Meets". These 30 derived temporal relations can be classified into the following 4 groups:

- Relations that relate points to points:

  {Equal, Before, After}

- Relations that relate points to intervals:

  {Before, After, Meets, Met-by, Starts, During. Finishes}

- Relations that relate intervals to points:

  {Before, After, Meets, Met-by, Started-by, Contains, Finished-by}

- Relations that relate intervals to intervals:

  {Equal, Before, After, Meets, Met-by, Overlaps, Overlapped-by, Starts, Started-by, During, Contains, Finishes, Finished-by}

Section 4.1.2 State

For each time element t (point or interval), there are some (possibly zero) propositions holds true on/over it. The state associate with the time element t is defined as the set of all holding propositions, namely $s_t = \{P_k \in P : HOLDS(P_k, t)\}$.

Informally speaking, the state and duration describe the properties of the time elements separately while the MEETS predicate describe the temporal relations between these time elements.

Section 4.1.3 Scenario

Definition 4.1 A *scenario* is a 5-tuple S=<T, P, HOLDS, MEETS, DUR>, where

1) T is a finite set of time elements.

2) P is a finite set of propositions.

3) HOLDS={HOLDS(s_i, t_i): $s_i \subseteq P$, $t_i \in T$}

4) MEETS={MEETS(t_i, t_j): for some t_i, $t_j \in T$}

5) DUR: $T \rightarrow R^{+0}$ indicates the duration of the time elements.

Example 4.1 let S_1=<T, P, HOLDS, MEETS, DUR>, where

T={Mon, Tue, Wed, Thur, Fri, Sat}

P={“Blocked nose”, “Cough”, “Fever”, “Headache”, “Sore throat”}

HOLDS={ HOLDS({“Blocked nose”}, Mon),

HOLDS({“Blocked nose”, “Sore throat”}, Tue),

HOLDS({“Sore throat”, “Cough”}, Wed),

HOLDS({“Sore throat”, “Cough”, “Fever”}, Thur), HOLDS({“Fever”, “Cough”, “Headache”}, Fri),

HOLDS({“Fever”, “Headache”}, Sat)

}

MEETS={ MEETS(Mon, Tue), MEETS(Tue, Wed), MEETS(Wed, Thur),
MEETS(Thur, Fri), MEETS(Fri, Sat)

}.

DUR: (day)=1 for all day$\in$T.

S_1 is a simple scenario of the flu symptoms.

Section 4.1.4 Graphical representation

Although the above representation of scenario is formal and adequate, it is not intuitive.

In [KM1992], a graphical representation for expressing temporal knowledge in terms of MEETS relations and duration knowledge has been introduced by means of a directed and partially weighted graph, where time elements are denoted as edges of the graph, relation MEETS(t_i, t_j) is represented by t_i being in-edge and t_j being out-edge to a common node, and for time elements with known duration, the corresponding edges are weighted by their durations respectively.

Such a graphical representation can be directly extended to express temporal scenarios.

In fact, a given scenario S=<T, P, HOLDS, MEETS, DUR> can be represented in terms of a temporal network, defined as a directed, attributed simple graph G_s, called scenario Graph, where:

1) Each time element t in T is denoted as a directed edge of the graph labelled by t that is bounded by a pair of nodes, which are called the *head-node*, and the

tail-node, of the edge, respectively.

2) Each relation MEETS(t_i, t_j) in MEETS is represented by means of merging the *head-node* of t_i and the *tail-node* of t_j as a common node, of which t_i is an in-edge and t_j is an out-edge, respectively (see figure 4.1).

3) Each formula HOLDS(s_i, t_i) in HOLDS is represented by means of simply adding s_i as an additional label to the edge labelled by the corresponding t_i. For any time element t in T, if there is no HOLDS knowledge, it will be labelled by the empty state {}.

4) Each piece of duration knowledge DUR(t) = r in DUR is expressed as a real number, *r*, alongside the corresponding edge t.

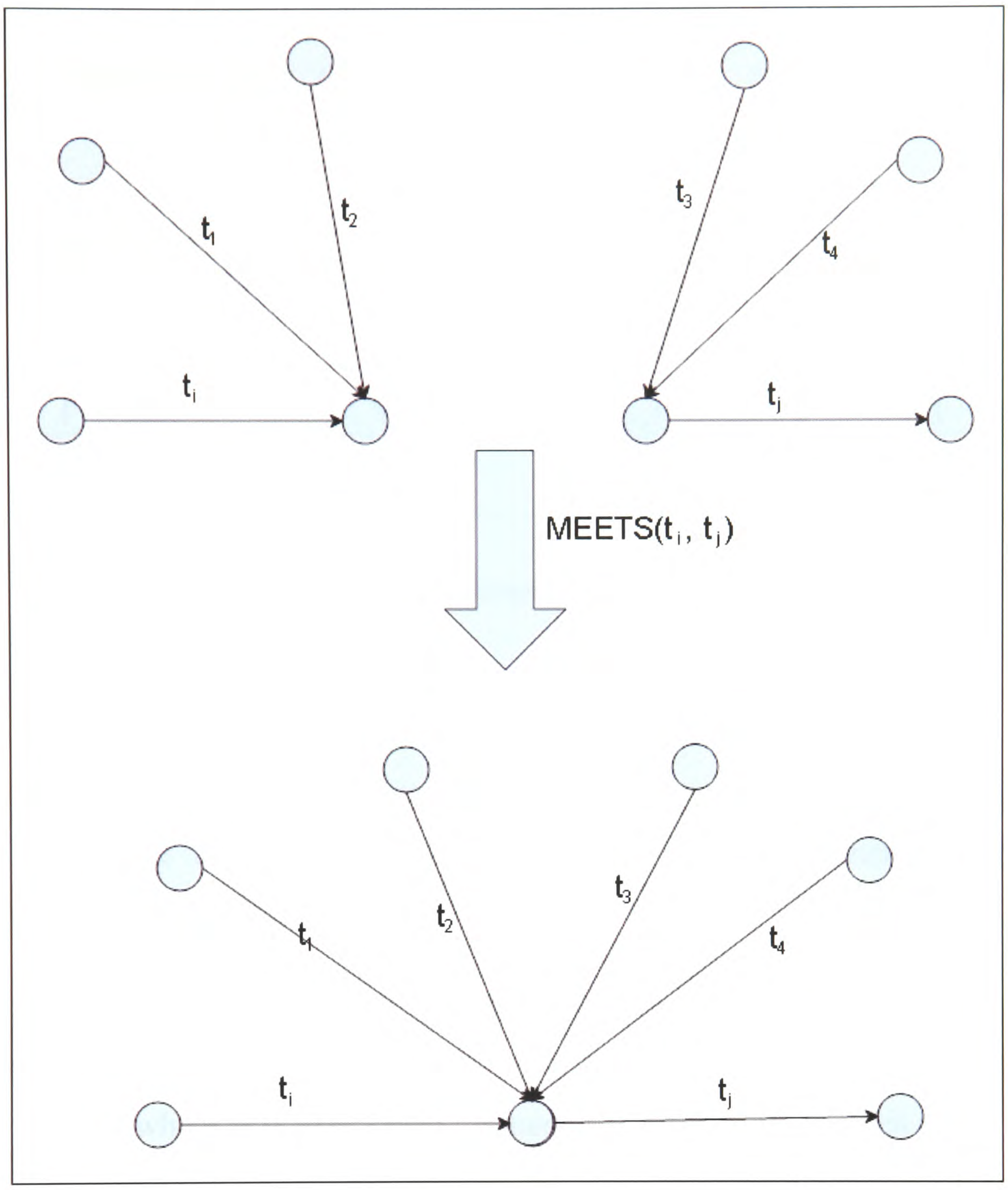


Figure 4.1 Merging the begin-node of t_i and the end-node of t_j as a common node if MEETS(t_i, t_j)

For example the scenario graph of above example S_1 is constructed as figure 4.2

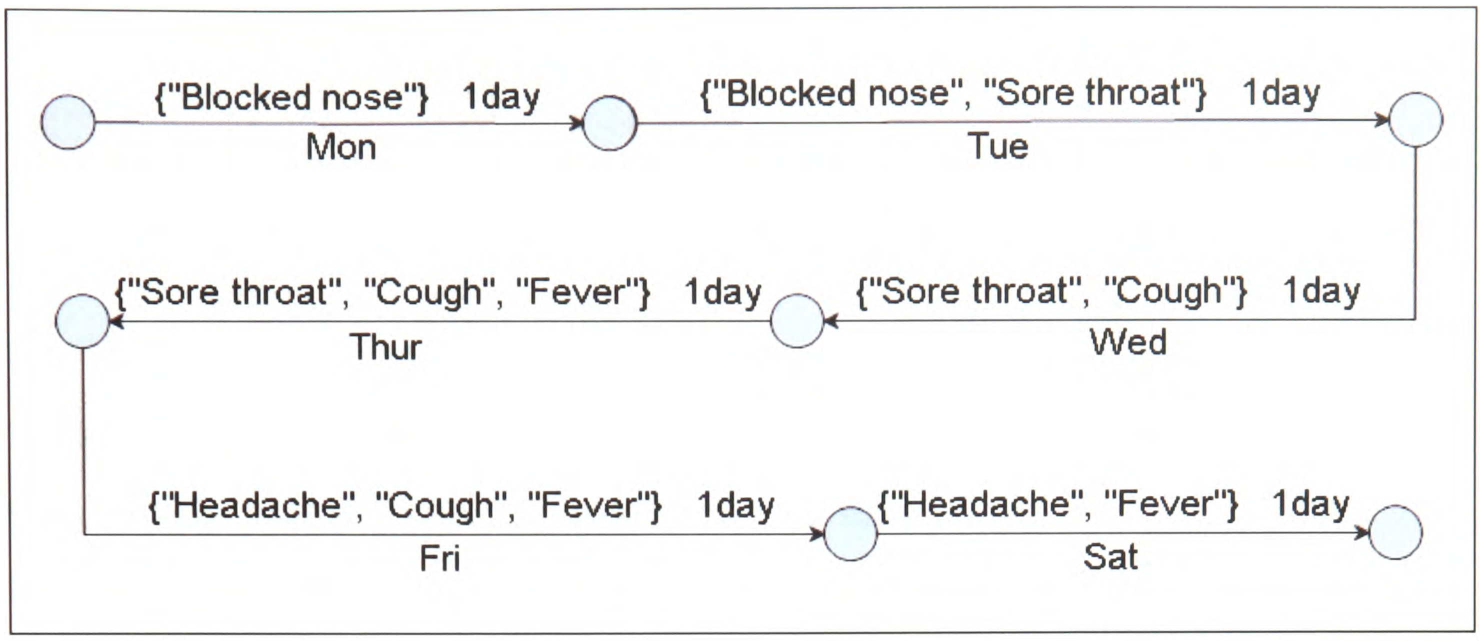

{"Blocked nose"} 1day
Mon
{"Blocked nose", "Sore throat"} 1day
Tue
{"Sore throat", "Cough", "Fever"} 1day
Thur
{"Sore throat", "Cough"} 1day
Wed
{"Headache", "Cough", "Fever"} 1day
Fri
{"Headache", "Fever"} 1day
Sat

Figure 4.2 Scenario graph of S_1

Section 4.1.5 Matrix representation of scenario

Although the graphical representation is very comprehensive for human; it is not suitable for storage and manipulation in computers. So the adjacency matrices are widely used for representing graphs in computers

In what follows, we shall simply assume $|P| = n$. Corresponding to scenario graph G_s with m nodes, we define a m-by-m-matrix N_s, named the adjacency matrix (or edge attribute matrix), where $N_s(i, j)$ is a (n+1)-dimension vector $l_{i,j} \in R^{n+1}$, such that:

- For any adjacent pair of nodes i and j in G, if (i, j) is an edge representing time element t, then $l_{i,j}(k)= 1$ if $Holds(P_k, t)$, otherwise $l_{i,j}(k) = 0$, $1 \leq k \leq n$; and $l_{i,j}(n+1) = DUR(t)$
- For any non-adjacent pair of nodes i and j in G, $l_{i,j} = <w, w, \ldots, w>$ (denoted as $l_{\varnothing}$), where w is a negative real number (usually w is set as -1), which will be use to adjust the edit-distance of deleting operations in graph matching process

Also we shall use $N_{s,k}$ to denote the matrix whose i-j-entry is the k-th element of i-j-entry in N_s.

For example, the adjacency matrix of scenario graph figure 4.2 is:

$$\begin{bmatrix} 1_{\emptyset} & <1,0,0,0,0,1> & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} \\ 1_{\emptyset} & 1_{\emptyset} & <1,0,0,0,1,1> & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} \\ 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & <0,1,0,0,1,1> & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} \\ 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & <0,1,1,0,1,1> & 1_{\emptyset} & 1_{\emptyset} \\ 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & <0,1,1,1,0,1> & 1_{\emptyset} \\ 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & <0,0,1,1,0,1> \\ 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} & 1_{\emptyset} \end{bmatrix}$$

It can be easily seen that this is not an economic representation. Since the scenario graph is usually sparse graph, the sparse matrix representation is more suitable, for example the above matrix can be sparsely represented as:

N_{s1}={(1,2,<1,0,0,0,0,1>), (2,3,<1,0,0,0,1,1>), (3,4, <0,1,0,0,1,1>), (4,5, <0,1,1,0,1,1>), (5,6,<0,1,1,1,0,1>), (6,7,<0,0,1,1,0,1>), (others, $1_{\emptyset}$)}

Section 4.2 Scenario with Incomplete Knowledge

Scenario of above definition is said to be complete since it contains whole information of HOLDS, MEETS and DUR predicates. To construct such a scenario, one has to know all the information of the time elements and propositions, which is usually unrealizable. So we shall define a model for partial information scenarios.

Section 4.2.1 Definition of partial scenario

Definition 4.2: A *partial scenario* is a 5-tuple S=<T, P, HOLDS, MEETS, DUR>, where

1) T, P and MEETS are defined as complete scenario.

2) HOLDS:P×T→{0,+1,-1}is a function indicating if a proposition holds true on a time t (+1), or holds false (0), or holds unknown (-1).

3) DUR:T→R^+∪{0, -1} is a function mapping time elements to its duration, or -1 if the duration is unknown.

Example 4.2 For the example4.1, obviously the patient got a fever on Thursday, Friday and Saturday, and the patient had a normal temperature from Monday to Wednesday.

Suppose that maybe the patient did not notice if he got fever or not from Monday to Wednesday, then the scenario will be S_2=<T, P, HOLDS', MEETS, DUR>, where

1) T, P, MEETS and DUR are the same as S_1.

2) HOLDS' is a function from P×T to {0,+1,-1} which can be arranged as table 4.1

Table 4.1 HOLDS function of a scenario

	Mon	Tue	Wed	Thr	Fri	Sat
"Blocked nose"	1	1	0	0	0	0
"Cough"	0	0	1	1	1	0
"Fever"	-1	-1	-1	1	1	1
"Headache"	0	0	0	0	1	1
"Sore throat"	0	1	1	1	0	0

The -1's indicate the unknown information.

Section 4.2.2 Graphical representation

The graphical representation of partial scenarios is similar to the representation of complete scenarios except that each edge of a complete scenario is labeled by state and duration while each edge t of a partial scenario is labeled by an n+1 dimensional vector $l_t \in R^{n+1}$, where $l_t(k)$=HOLDS(P_k, t) for k≤n and $l_t(n+1)$=DUR(t).

For example the partial scenario of example 4.2 can be graphically represented as:

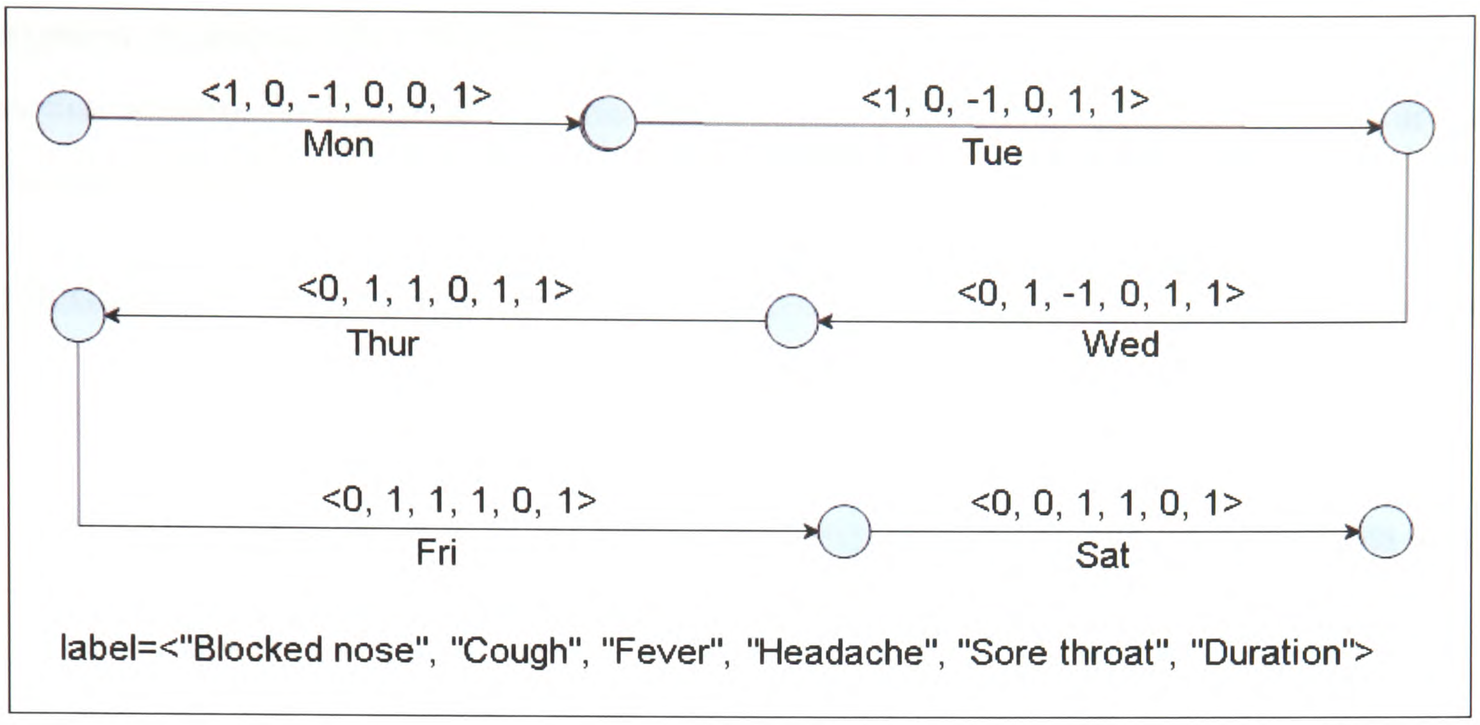


Figure 4.3 Scenario graph of S_2

Section 4.2.3 Matrix representation

The matrix representation of a partial scenario is the same as that for a complete scenario.

For example the matrix representation of scenario S_2 is

$$\begin{bmatrix}
l_\varnothing & <1,0,-1,0,0,1> & l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing \\
l_\varnothing & l_\varnothing & <1,0,-1,0,1,1> & l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing \\
l_\varnothing & l_\varnothing & l_\varnothing & <0,1,-1,0,1,1> & l_\varnothing & l_\varnothing & l_\varnothing \\
l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing & <0,1,1,0,1,1> & l_\varnothing & l_\varnothing \\
l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing & <0,1,1,1,0,1> & l_\varnothing \\
l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing & <0,0,1,1,0,1> \\
l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing & l_\varnothing
\end{bmatrix}$$

As the same, the above matrix can be sparsely written as

N_{s2}={(1,2,<1,0,-1,0,0,1>), (2,3,<1,0,-1,0,1,1>), (3,4, <0,1,-1,0,1,1>), (4,5, <0,1,1,0,1,1>), (5,6,<0,1,1,1,0,1>), (6,7,<0,0,1,1,0,1>), (others, $l_\varnothing$)}

Section 4.3 Scenario with General Temporal Relations

In our daily life, we seldom say "MEETS" to express time relations, instead, the temporal relations like BEFORE, AFTER are more frequently used. So we define another schema for temporal scenario where all the 13 possible temporal relations are all allowed for describing time structures.

Definition 4.3: An *extended scenario* is a 5-tuple S=<T, P, HOLDS, TRS, DUR>, where

1) T, P, HOLDS and DUR are defined as complete scenario.

2) TRS is the set of time relations. TRS={r(t_i, t_j): t_i, t_j are two time elements and r is one of the 13 possible temporal relations}

Example 4.3 For example S_1, if the patient cured on the next Saturday after some treatments, then the new scenario could be S_3=<T', P', HOLDS', TRS, DUR'>, where

1) T'=T$\cup${nSat}

2) P'=P and HOLDS'=HOLDS.

3) TRS=MEETS$\cup${AFTER(nSat, Sat)}

4) DUR'=DUR$\cup$DUR'(nSat)=1

An extended scenario can be easily transferred to a partial scenario by

- Before(t_1, t_2) $\leftrightarrow$ $\exists t \in$**T**(Meets(t_1, t) $\wedge$ Meets(t, t_2))

- Starts(t_1, t_2) $\leftrightarrow$ $\exists t_3, t, t_4 \in$**T**(Meets(t_3, t_1) $\wedge$ Meets(t_3, t_2) $\wedge$ Meets(t_1, t)$\wedge$ Meets(t, t_4) $\wedge$ Meets(t_2, t_4))

- Finishes(t_1, t_2) $\leftrightarrow$ $\exists t_3, t, t_4 \in$**T**(Meets(t_3, t) $\wedge$ Meets(t_3, t_2) $\wedge$ Meets(t, t_1)$\wedge$ Meets(t_1, t_4) $\wedge$ Meets(t_2, t_4))

- During(t_1, t_2) $\leftrightarrow \exists t_3, t_4 \in$**T**(Meets($t_3$, t_1) $\wedge$Meets(t_1, t_4) $\wedge$Starts(t_3, t_2) $\wedge$Finishes(t_4, t_2))

- Overlaps(t_1, t_2) $\leftrightarrow \exists t \in$**T**(Finishes(t, t_1) $\wedge$ Starts(t, t_2))

- After(t_1, t_2) ↔ Before((t_2, t_1)
- Met-by(t_1, t_2) ↔Meets(t_2, t_1)
- Overlapped-by(t_1, t_2) ↔ Overlaps(t_2, t_1)
- Started-by(t_1, t_2) ↔ Starts(t_2, t_1)
- Contains(t_1, t_2) ↔ During(t_2, t_1)
- Finished-by(t_1, t_2) ↔ Finishes(t_2, t_1)

Based on this transformation, the graphical and matrix representations can be directly used for the extended scenarios. Take the drill-sticking case in figure 1.1 as an example. There are eight propositions associated with the drill-sticking, which means P={"hook load increasing", "erratic flow out", "erratic torque", "increasing drag", "erratic drag", "erratic increasing torque", "increasing torque", "increasing pressure"}. For the time elements Ix2, since "erratic flow out" holds on it and its duration is unknown, so Ix2 can be expressed by an edge with label <0, 1, 0, 0, 0, 0, 0, 0, -1>. Similarly Ix3 can be expressed by an edge with label <0, 1, 1, 0, 0, 0, 0, 0, -1> indicating that "erratic flow out", "erratic torque" are holding on Ix3 and its duration is unknown. Because Ix2 is before Ix3, there exists a time elements $t_{2,3}$ that MEETS(Ix2, $t_{2,3}$) and MEETS($t_{2,3}$, Ix3). In the same way, the drill-sticking case in figure 1.1 can be reconstructed as a scenario graph as figure 4.4:

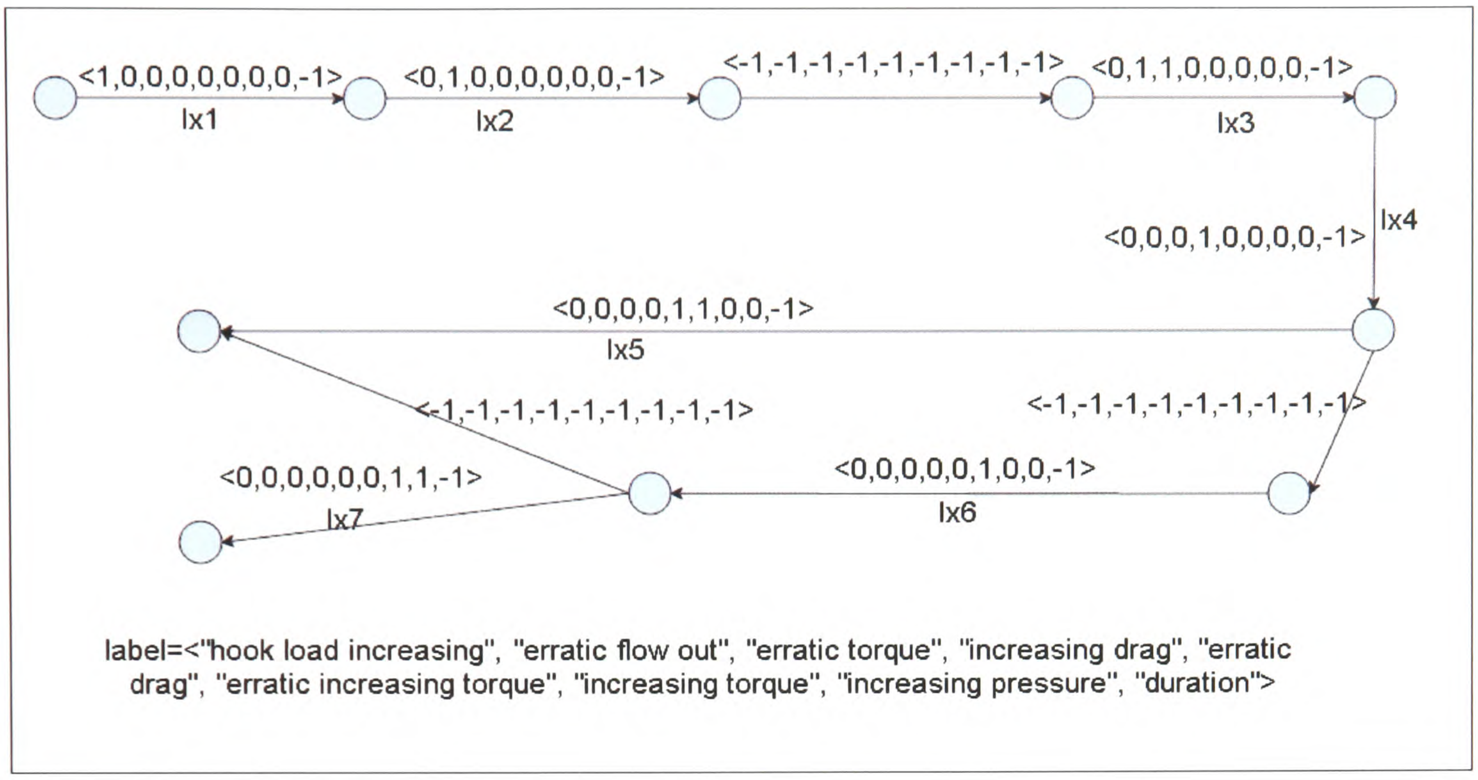


Figure 4.4 Scenario graph of drill-sticking

Section 4.4 Matching Temporal Scenarios

As introduced in chapter 1, object similarity is very important for case based reasoning, pattern recognition and cluster analysis. For our cases, we have to explore the object similarity between scenarios or corresponding scenario graphs.

In this thesis, two different similarity measurements are introduces based on embedded mapping and graph matching.

Section 4.4.1 Similarity based on embedded mapping

We [MZH2007] propose a similarity measurement of scenarios formulated by the embedded mapping as following:

Let S_1=<T_1, P_1, $HOLDS_1$, $MEETS_1$, DUR_1> and S_2=<T_2, P_2, $HOLDS_2$, $MEETS_2$, DUR_2> be two scenarios such that $|T_1| \leq |T_2|$, then a embedded mapping φ is a one to one function from {1, 2, ..., $|T_1|$} to {1, 2, ..., $|T_2|$} and the scenario similarity is defined as a linear combinations of the following similarity:

- Similarity of scenarios size:

$$\mathrm{sim}_{\mathrm{size}}(S_1, S_2) = \frac{|T_1|}{|T_2|}$$

- Similarity of HOLDS relations

$$\mathrm{Sim}_{\mathrm{HOLDS}}(S_1, S_2, \varphi) = \frac{1}{|T_1|} \sum_{k=1}^{|T_1|} \frac{\mathrm{Intersection}(s_k^1, s_{\varphi(k)}^2)}{\mathrm{Union}(s_k^1, s_{\varphi(k)}^2)}$$

where intersection and union are the set-intersection and set-union operators.

- Similarity of MEETS relations

$$\mathrm{Sim}_{\mathrm{MEETS}}(S_1, S_2, \varphi) = 2 \frac{\sum_{i=1}^{|T_1|} \sum_{j=1}^{|T_1|} N_1(i,j) N_2(\varphi(i), \varphi(j))}{\sum_{i=1}^{|T_1|} \sum_{j=1}^{|T_1|} N_1(i,j)^2 + N_2(\varphi(i), \varphi(j))^2}$$

where N_1 and N_2 are the MEETS-adjacency matrices of scenarios S_1 and S_2 respectively.

- Similarity of durations

$$\mathrm{Sim}_{\mathrm{MEETS}}(S_1, S_2, \varphi) = 2 \frac{\sum_{i=1}^{|T_1|} \mathrm{DUR}(t_i^1) \mathrm{DUR}(t_i^2)}{\sum_{i=1}^{|T_1|} \mathrm{DUR}(t_i^1)^2 + \mathrm{DUR}(t_i^2)^2}$$

- The overall similarity with respect to embedded mapping φ is:

$$\mathrm{Sim}(S_1, S_2, \varphi) = \mathrm{Sim}_{\mathrm{size}}(S_1, S_2) \frac{w_1 \mathrm{Sim}_{\mathrm{HOLDS}}(S_1, S_2, \varphi) + w_2 \mathrm{Sim}_{\mathrm{MEETS}}(S_1, S_2, \varphi) + w_3 \mathrm{Sim}_{\mathrm{DUR}}(S_1, S_2, \varphi)}{(w_1 + w_2 + w_3)}$$

where the w_i is the importance coefficient of the similarity of HOLDS, MEETS, DUR, which can be set originally or obtained by learning process.

- Finally, the similarity between scenario S_1 and S_2 are defined as:

$$\mathrm{Sim}(S_1, S_2) = \max_{\varphi} \mathrm{sim}(S_1, S_2, \varphi)$$

A navigation-based algorithm has been provide to calculate the scenario similarity defined above and several tests have been done in [MZH2007] which showed that the above similarity reflects the conventional idea of edit distance, where the closer two scenarios are to each other, the more similar they are. However, due to embedded checking of all mappings, the computational complexity of the associated navigation-based algorithm is exponential. So a quicker algorithm or a better definition of similarity is needed.

Section 4.4.2 Similarity based on graph matching

Given two scenarios S_1 and S_2, let G_{s1} and G_{s2} be their graphical representations, then the distance of these two scenarios are defined by the distance of the two graphs G_{s1} and G_{s2}.

Assume the adjacency matrices are N_{s1} and N_{s2}, with size $m_1 \times m_1$ and $m_2 \times m_2$, respectively. Without losing the generality, we assume $m_1 = m_2 = m$. In fact, if $m_1 < m_2$, we can simply add $m_2 - m_1$ isolated dummy nodes to graph G_{s1} to get an extended graph, whose characteristic matrix will have the same size as that of N_{s2}, i.e., $m_2 \times m_2$. Similar treatment can be applied to the case where $m_2 < m_1$.

The *similarity degree* between st_1 and st_2 is then defined by:

$$\mathrm{sim}(s_1, s_2) = 1 - \frac{\min\limits_{Q \in \mathrm{perm}(m)} \sum\limits_{k=1}^{n+1} \left\| N_{s1,k} - Q N_{s2,k} Q^T \right\|_F}{\sum\limits_{k=1}^{n+1} \left(\left\| N_{s1,k} \right\|_1 + \left\| N_{s2,k} \right\|_F \right)} \tag{4.1}$$

where perm(m) denotes the set of all m-by-m permutation matrices and $\|\bullet\|_F$ denotes the Frobenius-norm which will be discussed in the next chapter.

Since $\left\|N_{s1,k} - QN_{s2,k}Q^{T}\right\|_{F} \le \left\|N_{s1,k}\right\|_{F} + \left\|QN_{s2,k}Q^{T}\right\|$ and $\left\|QN_{s2,k}Q^{T}\right\|_{F} = \left\|N_{s2,k}\right\|_{F}$

Then $0 \le \dfrac{\sum_{k=1}^{n+1}\left\|N_{s1,k} - QN_{s2,k}Q^{T}\right\|_{F}}{\sum_{k=1}^{n+1}(\left\|N_{s1,k}\right\|_{1} + \left\|N_{s2,k}\right\|_{F})} \le 1$

So it is easy to see that sim(s_1, s_2) falls within the range of [0, 1].

The algorithm for the formula (4.1) will be fully discussed in next chapter.

Chapter 5 Graph Matching Algorithms for Matching Scenarios

In chapter 4, the scenario pattern matching problem was formulated as the attributed graph matching problem. So in this chapter, we shall propose effective and efficient algorithms for matching scenario graphs.

Section 5.1 introduces the graph matching problem selected for this thesis. The eigen-decomposition graph matching algorithm and the symmetric polynomial transformation graph matching algorithm are discussed and improved in section 5.2 and section 5.3. Section 5.4 proposes the similarity graph matching framework to analyze node similarity based graph matching algorithm uniformly.

Section 5.1 Graph Matching Problems in This Thesis

Firstly, the graph matching problems selected for this thesis have to be clearly defined.

Section 5.1.1 Definition of graph matching problems

Before bringing forward the formulation of graph matching problem, two important notions, the permutation and permutation matrices, have to be introduced firstly.

Definition 5.1: A *permutation* p is an one-to-one function from {1, 2,..., n} to itself.

For example, $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 1 & 5 & 2 \end{pmatrix}$ denotes the permutation that maps each number in first row to its corresponding number of second row, such as, 1 to 4, 2 to 3. A permutation p can be simply denoted as a vector $(p(1), p(2), \ldots, p(n))^T$. So the permutation $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 1 & 5 & 2 \end{pmatrix}$ can be simply written as vector $(4, 3, 1, 5, 2)^T$.

The set of all permutations of n elements is denoted as SG(n).

Let p and q be two permutation vectors of n elements and M be a n-by-n matrix then M(p, q), M(p, :) and M(:, q) denote the matrix whose (i, j)-th entry is the (p(i), q(j))-th, (p(i), j)-th and (i, q(j)) entry of matrix M respectively.

Definition 5.2: A *permutation matrix* P is a n-by-n matrix such that

1) $\sum_{i=1}^{n} P(i,j) = 1$ for all j=1, 2, ..., n

2) $\sum_{j=1}^{n} P(i,j) = 1$ for all i=1, 2, ..., n

3) $P(i,j) \in \{0, 1\}$ for all i,j=1,2,...,n

And the set of all n-by-n matrices are denoted as Perm(n);

Definition 5.3: There is a natural one-to-one correspondence between the permutation set SG(n) and the permutation matrices set Perm(n) defined as:

s2p: SG(n)→Perm(n) such that $s2p(p)=I_n(p,:)$,

p2s: Perm(n) →SG(n) such that $p2s(P)=P\times(1,2,\ldots,n)^T$,

for every permutation $p \in SG(n)$ and permutation matrix $P \in Perm(n)$ (see glossary).

So in the rest of the thesis, we will use the lower case p or q to denote a permutation and the upper case P or Q to denote its corresponding permutation matrix.

For example the corresponding permutation matrix of permutation vector $(4,3,5,2)^T$

is $I_n((4,3,1,5,2)^T,:) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$.

Proposition 5.1: Let p be a permutation. P is the corresponding permutation matrix, then for every n-by-n matrix M

$$M(p, :)=P\times M \text{ and } M(:, p)=M\times P^T$$

Now considering two graphs G and H with same size n, the graph matching problem is finding the optimal permutation from the nodes of G to nodes of H to minimize their "difference".

For any permutation p, these "differences" can be described by

$$\|G - H(p,p)\|$$

Where $\|\bullet\|$ can be any matrix norm. In this thesis, we choose the Frobenius norm, which is a popular choice used in [Ume1988, Alm1991, Wyk2002]. An alternative choice is L_1-norm that can be found in [AD1993].

Definition 5.4: $\|\bullet\|_F$ denotes the Frobenius norm. For every complex matrix M,

$$\|M\|_F = \sqrt{\sum_{i,j} |M_{i,j}|^2}$$

Proposition 5.2: Let A, B, M be complex matrices, U and V be unitary matrices then:

$$\|A + B\|_F \leq \|A\|_F + \|B\|_F$$

$$\|AB\|_F \leq \|A\|_F \|B\|_F$$

$$\|UMV\|_F = \|M\|_F$$

$$\|M^*\|_F = \|M\|_F$$

$$\|M\|_F^2 = \|r\,e\,al(M)\|_F^2 + \|i\,m\,ag(M)\|_F^2$$

$$\|M\|_F^2 = \left\|\frac{M+M^*}{2}\right\|_F^2 + \left\|\frac{M-M^*}{2}\right\|_F^2$$

Proof is trivial.

So the "difference" of two graphs under permutation p can be rewritten as:

$$\|G - PHP^T\|_F$$

And the *attributed graph matching problem* can be formulated as:

$$\underset{P\in Perm(n)}{\arg s\ \min} \sum_{k=1}^{m} \|G_k - PH_kP^T\|_F \tag{5.1}$$

where G_k, $H_k \in R^{n\times n}$ are the k-th adjacency attributed matrices of attributed graphs G and H respectively.

This definition is widely used in [Wyk2002, Alm1991, and Ume1988].

As a special case for m=1, the formula (5.1) reduces to the so called *weighted graph matching problem*:

$$\underset{P\in Perm(n)}{\arg s\ \min} \|G - PHP^T\|_F \tag{5.2}$$

Where G, $H \in R^{n\times n}$ are the adjacency weighted matrices of weighted graphs G and H

Obviously, algorithms for attributed graph matching can be directly applied for weighted graph matching problems, but not vice versa. However, all the matching algorithms discussed in this thesis are all applicable for the attributed graph matching problems. So in thesis, we pay more attention to the weighted graph matching problems (5.2) because of its simplicity, and point out how the algorithms are revised for general attributed graph matching problems (5.1)

Section 5.1.2 Random generation of matrices

Since the graph matching problem is NP-complete [GJ1979, Abd1998, MB1998, GR1996], all the existing polynomial algorithms fail to work for some cases. Therefore, it is important to provide statistical evaluations showing the performances and limitations of the matching algorithms. But, first of all, we need an algorithm to generate random matrices as samples for graph matching algorithms testing.

The probability distribution of the samples directly influences the statistical evaluations of these graph matching algorithms. So every evaluation should be provided together with the probability distribution of its samples or with the random matrices generating algorithms which actually give the probability distribution in some implicit form.

Although there are many algorithms for generating orthogonal matrices [Hei1978, Ste1980, TT1982], unitary matrices [Zyc1994, PZK1998], correlation matrices [DH2000, DHST2005] and general matrices [CN1997, Mez2007], in this thesis, we shall choose a simpler way to generate certain kind of random matrices based on the following two reasons:

- Reduce the computational complexity in order to test enough samples.
- It's hard for one to judge which distribution is better without knowing the real problem to be solved.

For our experiments, it is important to randomly generate general real and symmetric matrices, which is implemented in Matlab as following:

Table 5.1 Generating real matrices

```
function y=Greal(n,type)
//type=0 for sparse matrix.
   y=rand(n);
   if type==0
      y=y.*(rand(n)<1/n);
   end
```

Table 5.2 Generating symmetric matrices

```
function y=Gsym(n,type)
//type=0 for sparse matrix.
   A=rand(n);
   if type==0
      A=A.*(rand(n)<1/n);
   end
   y=(A+A')/2;
```

Although these programs are very simple, it is important for repeating all the following experiments in this thesis.

Section 5.1.3 Evaluation criterions

To evaluate a given graph matching algorithm, we use two most important criterions, namely mean error and computational time.

Definition 5.5: Let A be a matching algorithm, for a given graph pairs G and H, the *computational error* of the algorithm A is defined by:

$$\text{Error(A, G, H)} = \left\| G - P_A H P_A^T \right\|_F - \min_{P \in \text{perm}(n)} \left\| G - PHP^T \right\|_F \qquad (5.3)$$

where P_A is the optimal matching calculated by algorithm A.

And the *mean error* is defined as the expected value of certain graph classes, that is

Mean-error(A)=Expectation(Error(A, G, H)),

where G and H are taken as random variables.

In further experiments, the mean error of a matching algorithm is not gained by the theoretical deduction but via numerical simulation.

Section 5.1.4 Comparison of several graph matching algorithms

Firstly, we compared several standard graph matching algorithms including the eigen-decomposition method (EDGM), the symmetric polynomial-transformation method (SPGM), the hubs and authorities method (HAGM) and the least square Kronecker product-successive projection method (LSKPGM).

Test 1: matching isomorphic graphs

500 isomorphic graph pairs are generated with size from 5 to 20 and the result is illustrated as figure 5.1 for dense graphs and figure 5.2 for sparse graphs.

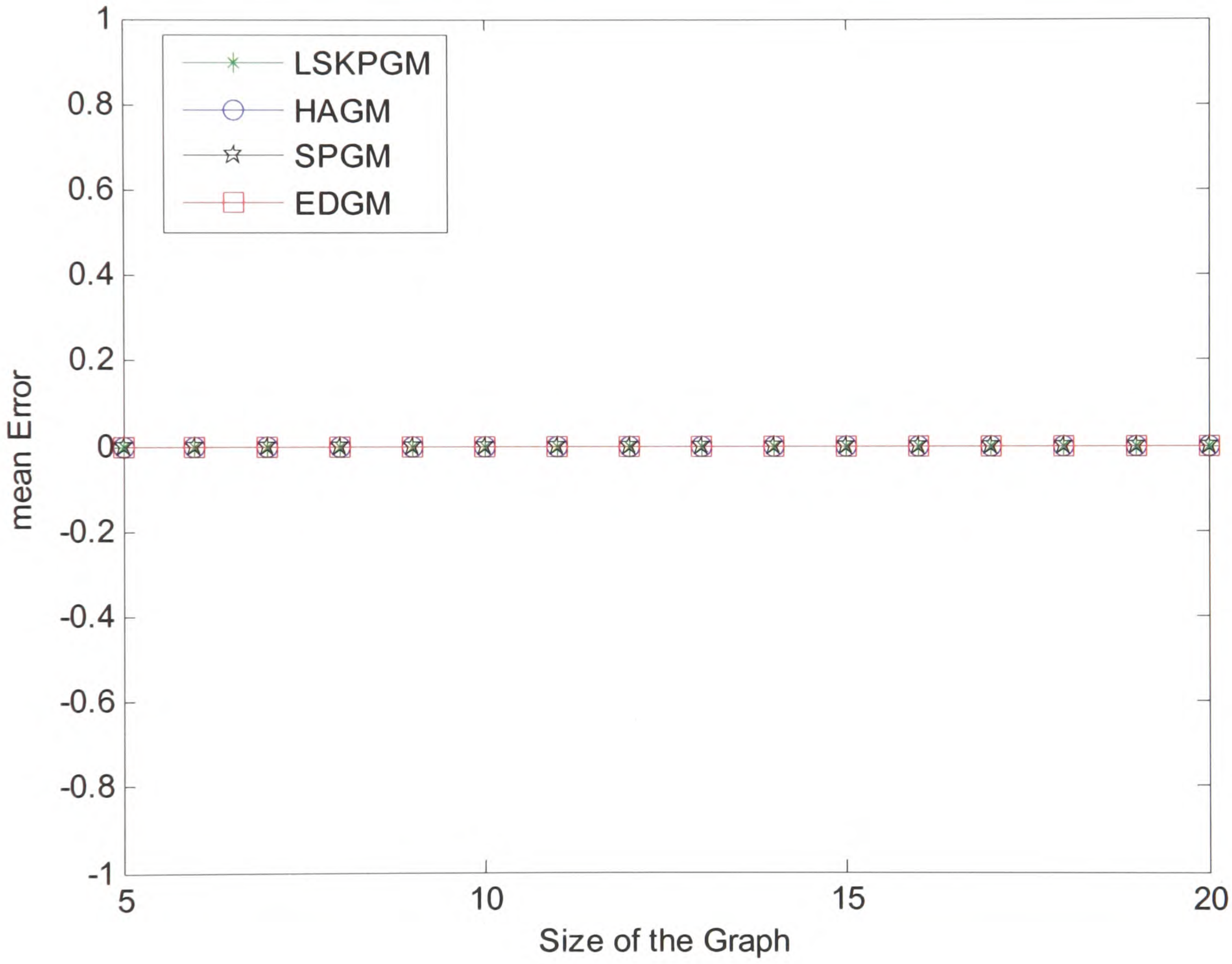


Figure 5.1 LSKPGM, HAGM, SPGM and EDGM for matching dense isomorphic graph pairs

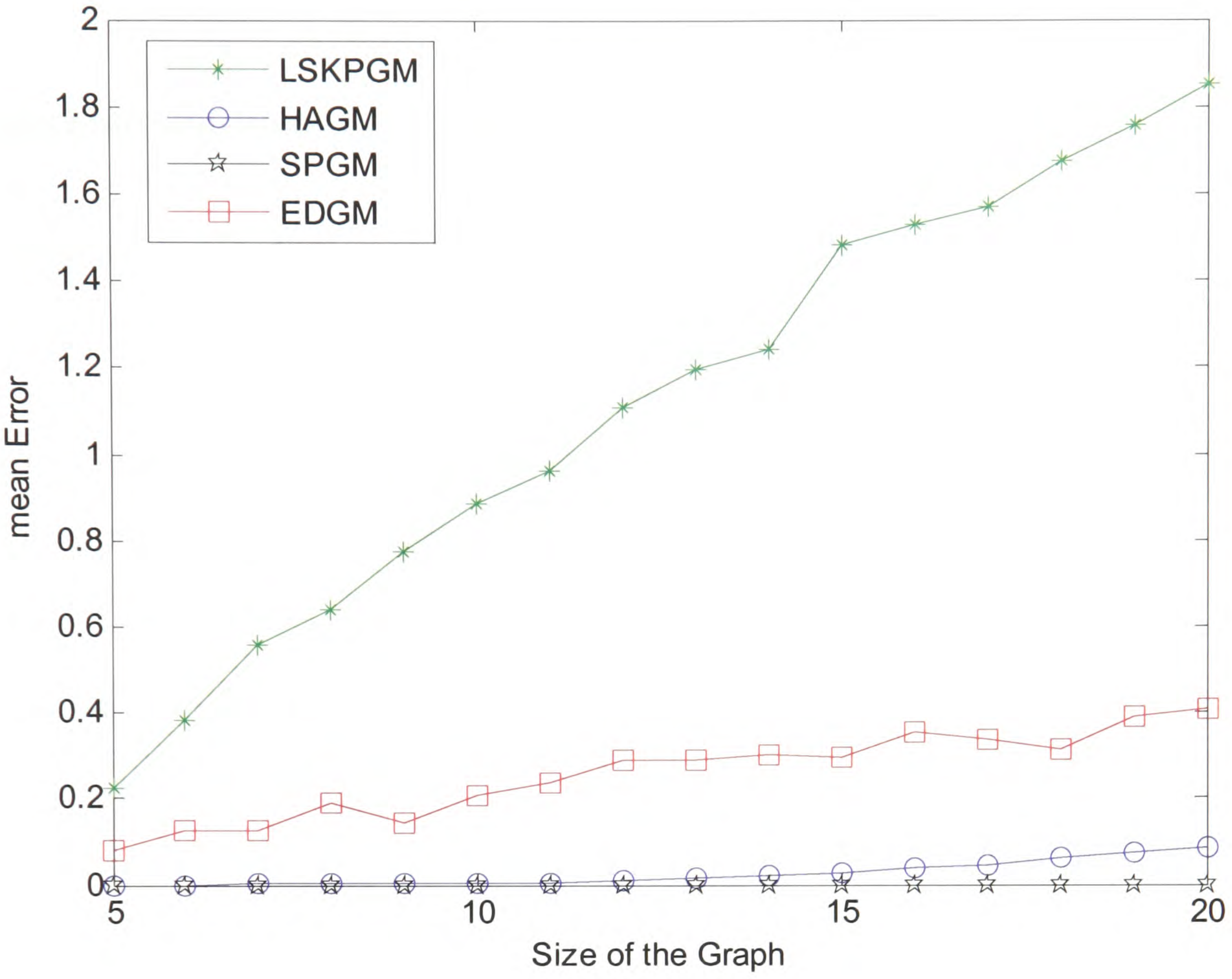


Figure 5.2 LSKPGM, HAGM, SPGM and EDGM for matching sparse isomorphic graph pairs

Figure 5.1 shows that all these algorithms can match dense isomorphic graph pairs perfectly. They almost all get zero error matching result. In statistical sense, isomorphism between dense graph pairs is easily found. But the case is different for sparse graph pairs. As shown in figure 5.2, only the SPGM algorithm matches sparse isomorphic graph pairs effectively, while others get ascending matching errors when the size of graph increases.

So the SPGM can be a candidate for our scenario graph matching.

Test 2: matching non-isomorphic graph pairs with perturbations

The same as test 1, 500 isomorphic graph pairs G and H are generated with size from 5 to 20, for each pair, H is disturbed by adding a perturbation matrix E whose entries are uniformly random real numbers in the range from 0 to +ε. ε is called *perturbation coefficient*, which is set to be 0.10 in this test. The matching results are shown as figure 5.3 for dense graphs and figure 5.4 for sparse graphs.

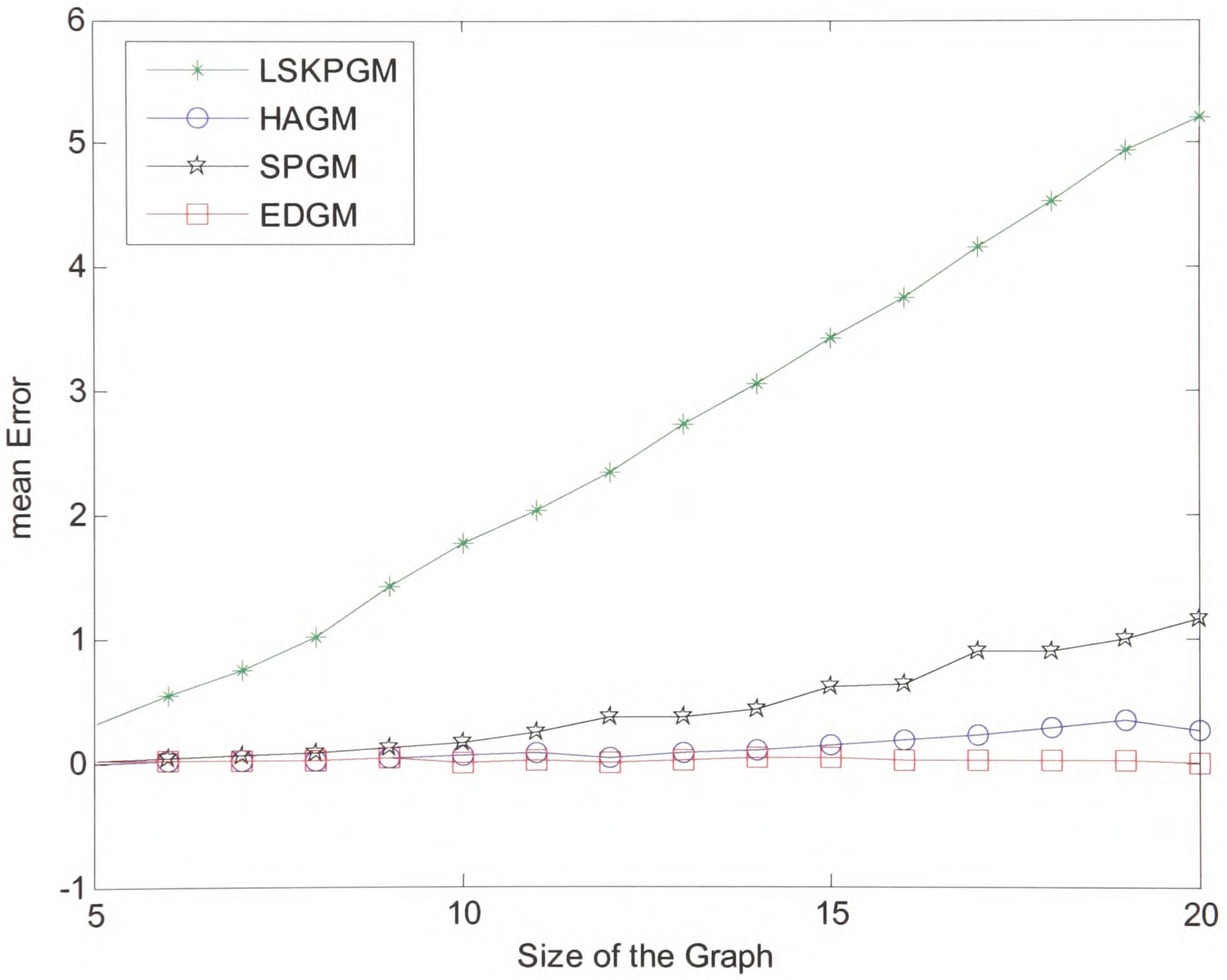


Figure 5.3 LSKPGM, HAGM, SPGM and EDGM for matching dense graph pairs with perturbation coefficient ε=0.10.

Figure 5.3 shows that for graph pairs with small distance, only the EDGM algorithm get a satisfied matching result, while others all get large matching errors.

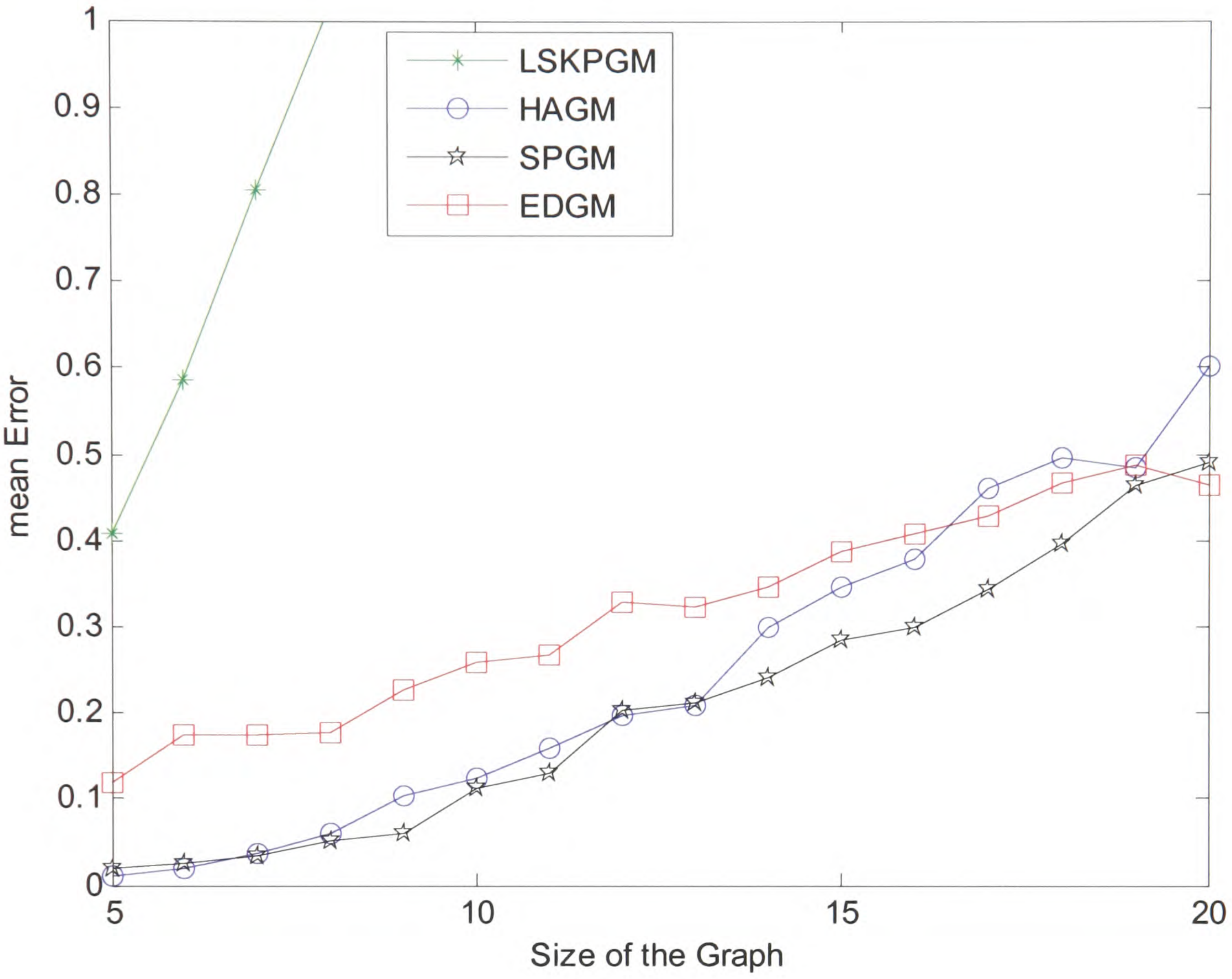


Figure 5.4 LSKPGM, HAGM, SPGM and EDGM for matching sparse graph pairs with perturbation coefficient ε=0.10.

Figure 5.4 shows that all these four algorithms fail to work for sparse graph pairs with small distance. Unfortunately, we do have to solve such kind of graph matching problems since the scenario graph pairs are usually sparse with small distance.

So the EDGM algorithm will be another candidate for our scenario graph matching, where it has to be revised for sparse graph pairs.

Test 3: the computational time of these algorithms

Graphs are generated as test 1, and the result is illustrated as figure 5.5.

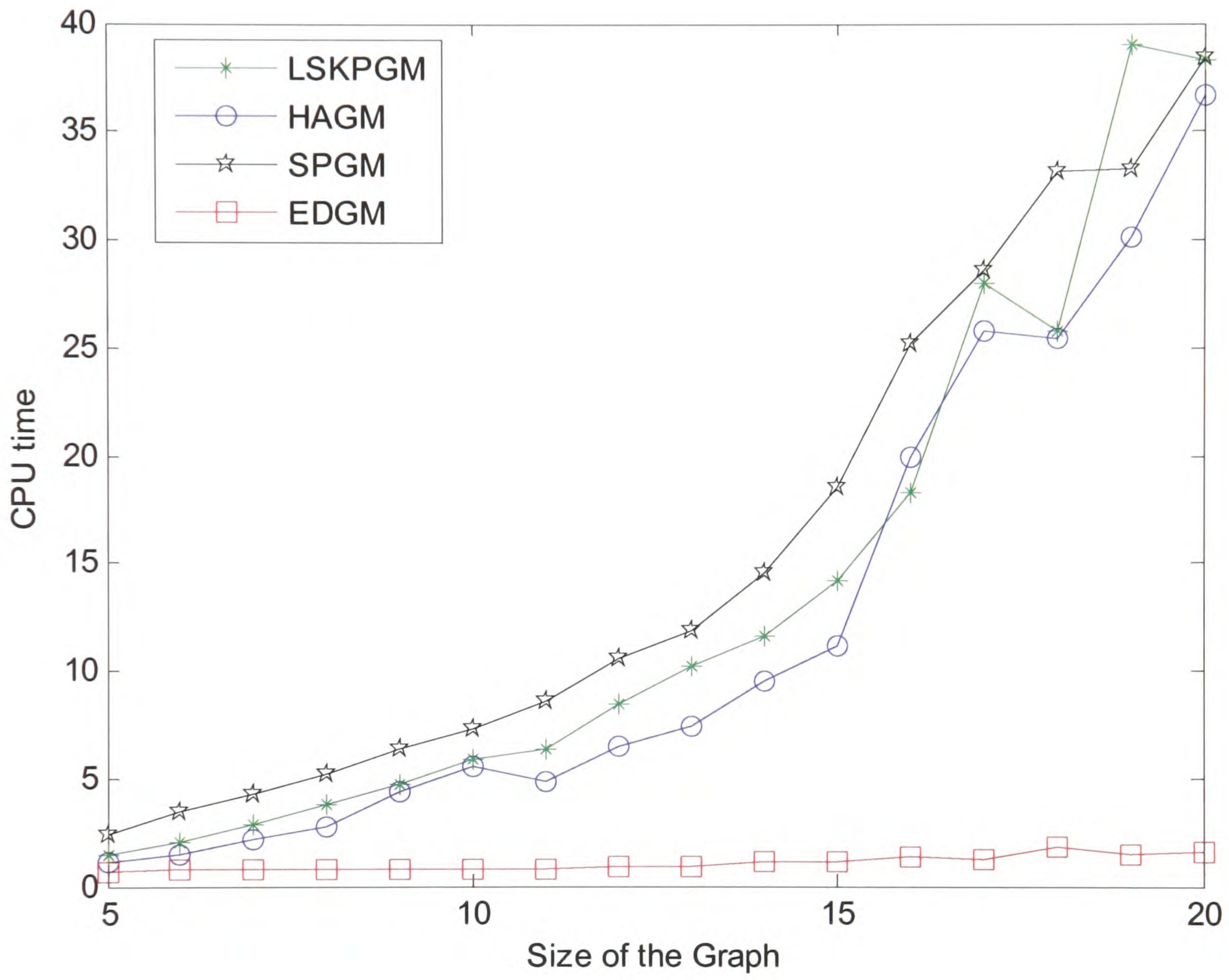


Figure 5.5 CPU time consuming comparison of LSKPGM, HAGM, SPGM and EDGM

Figure 5.5 shows that EDGM algorithm costs the least computational time.

Conclusion

Based on these preliminary tests, the SPGM and EDGM algorithms are selected for our scenario graph matching problems. But, SPGM algorithm only works well for isomorphic graph pairs while EDGM only works well for dense graphs, so both of them need to be improved for sparse non-isomorphic graph pairs.

Section 5.2 Eigen-decomposition Graph Matching Algorithm and Its Improvement

Umeyama [Ume1988] presented an Eigen-decomposition based graph matching algorithm for matching both undirected and directed graphs.

Section 5.2.1 Eigen-decomposition graph matching algorithm

Matching undirected graphs

Let G and H be two weighted undirected graphs. The EDGM algorithm has three dominated steps:

1) Calculating the Eigen-decomposition of adjacency matrix.

Theorem 5.1: (diagonalization of symmetric matrices) for every symmetric real matrix M there exists a real orthogonal matrix O such that D= O^TMO is a diagonal matrix. [HJ1985]

From above well known that symmetric matrices G and H can be decomposed as $G=VD_GV^T$ and $H=WD_HW^T$, where D_G and D_H are the diagonal matrices of the eigenvalues (in ascending order) of G and H, respectively, and V and W are two orthogonal matrices.

2) Construct the node similarity matrix

The similarity matrix is constructed by:

$$S=|V|\times|W|^T \tag{5.4}$$

where $|V|$, $|W|$ denote the matrices whose (j,k)-entry is the absolute value of corresponding entry of matrices V and W, and S(j,k) means the similarity of the j-th node of G and the k-th node of H.

3) Calculate the maximum similarity match.

The optimal matching is calculated by:

$$\underset{P \in \mathrm{Perm}(n)}{\mathrm{args\ max}} \sum_{i,j=1}^{n} S(i,j) \times P(i,j)$$

which is called the maximum linear assignment problem.

Given a real matrix M, the *maximum linear assignment problem* is defined as

$$\mathrm{max\_assign}(M) = \underset{P \in \mathrm{Perm}(n)}{\mathrm{args\ max}} \sum_{i,j=1}^{n} M(i,j) \times P(i,j)$$

And the *minimum linear assignment* problem is defined as

$$\mathrm{min\_assign}(M) = \underset{P \in \mathrm{Perm}(n)}{\mathrm{args\ min}} \sum_{i,j=1}^{n} M(i,j) \times P(i,j)$$

Obviously that max_assign(M)=min_assign(-M).

The (maximum or minimum) linear assignment problem can be efficiently solved by Hungarian algorithm [Kuh1955, mun1957, AMO1993].

Matching directed graphs

The process of applying the EDGM algorithm for directed graphs is almost the same as that of undirected graph except the adjacency matrix is replaced by its Hermitian matrix.

Let G be a directed weighted graph. An important point is that diagonalization theorem 5.1 doesn't holds for general real matrix. In other words, directed graph G may not be diagnoalizable in general cases. So in [Ume1988] a *Hermitian matrix of a graph* G is proposed:

$$\mathrm{Ht}(G)=\frac{G+G^T}{2}+\sqrt{-1}\frac{G-G^T}{2}$$

Obviously that Ht(G) is a Hermitian matrix for every real matrix G.

Theorem 5.2: (diagonalization of Hermitian matrices) for every Hermitian complex matrix M there exists a real Unitary matrix U such that D= U^*MU is a real diagonal matrix. [HJ1985]

So Hermitian matrices Ht(G) and Ht(H) can be decomposed as Ht(G)=VD_GV^* and Ht(H)=WD_HW^*, where D_G and D_H are the diagonal matrices of the eigenvalues (in ascending order) of G and H, respectively, and V and W are two unitary matrices.

The second and third step is the same as undirected graphs.

Matching of attributed graphs

The EDGM algorithm is presented only for weighted graph matching. In fact it can be easily extended for attributed graph matching.

Let $\{G_k:1\leq k\leq m\}$be a attributed graph, where G_k is the k-th adjacency matrix. Similarly as directed graphs, assume that k-th eigenvector matrix of Ht(G_k) is V_k, then Similarity of attributed graph G and H can be constructed as

$$S=|V|\times|W|^T\text{, where}|V|=\begin{bmatrix}|V_1| & |V_2| & \cdots & V_m\end{bmatrix}\text{, and }|W|=\begin{bmatrix}|W_1| & |W_2| & \cdots & W_m\end{bmatrix}$$

And the optimal matching is calculated by Hungarian algorithm [Kuh1955, mun1957, AMO1993].

Section 5.2.2 Computational complexity

The computational complexities of the three dominate steps are

1) Calculating Eigen-decomposition of n-by-n matrices: O(n^3) [GV1996]

2) Calculating matrices multiplication: O(n^3) [Knu1998]

3) Using Hungarian algorithm: O(n^3) [AMO1993]

So the total computational complexity is O(n^3).

Section 5.2.3 Three limitations

The EDGM algorithm only work well for graphs satisfying all the three constraints:

1) Nearly Isomorphic

2) Isolating eigenvalues.

3) Dissimilar rows of absolute eigenvectors.

We use some examples to show how the EDGM algorithm fails to work if any of the constraint is not satisfied.

The need of the "Nearly Isomorphic" constraint

The same as above tests, 500 pairs of isomorphic graphs G and H are generate. For each pair G and H, they are made no longer isomorphic to each other by means of perturbing H with a noise E, ranging from 0 to 0.15. The result is illustrated in figure 5.6.

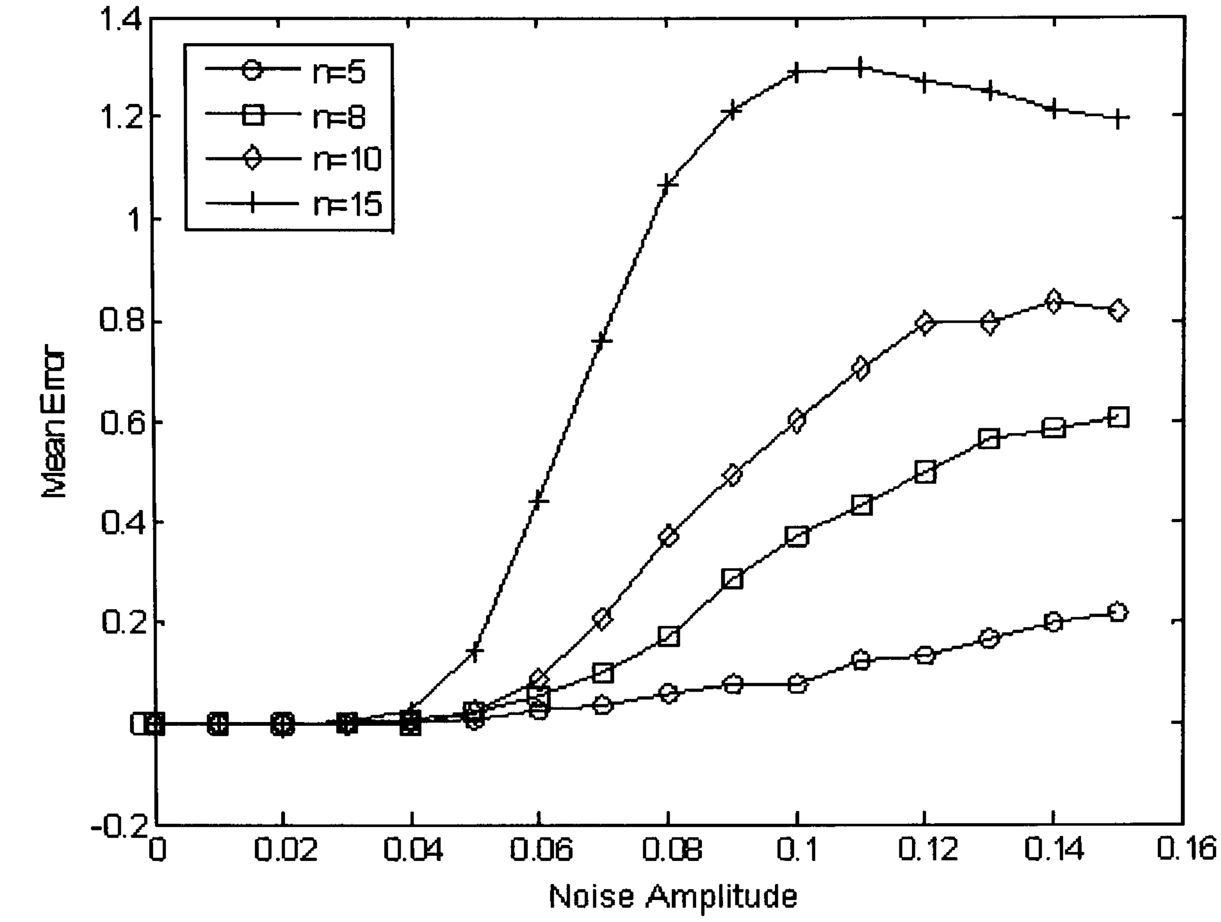


Figure 5.6 Mean error of EDGM algorithm relative to noise

From figure 5.6, we can see that the calculating error of the EDGM algorithm grows quickly when the noise amplitude or the size of graph increases, which confirms our claim that the "nearly isomorphic" property is needed for EDGM algorithm.

The need of the "Isolating Eigenvalues" constraint

Here, it shall be demonstrate by example that without the "Isolating eigenvalues" condition, the EDGM method may fail to work. Consider the following graph pair:

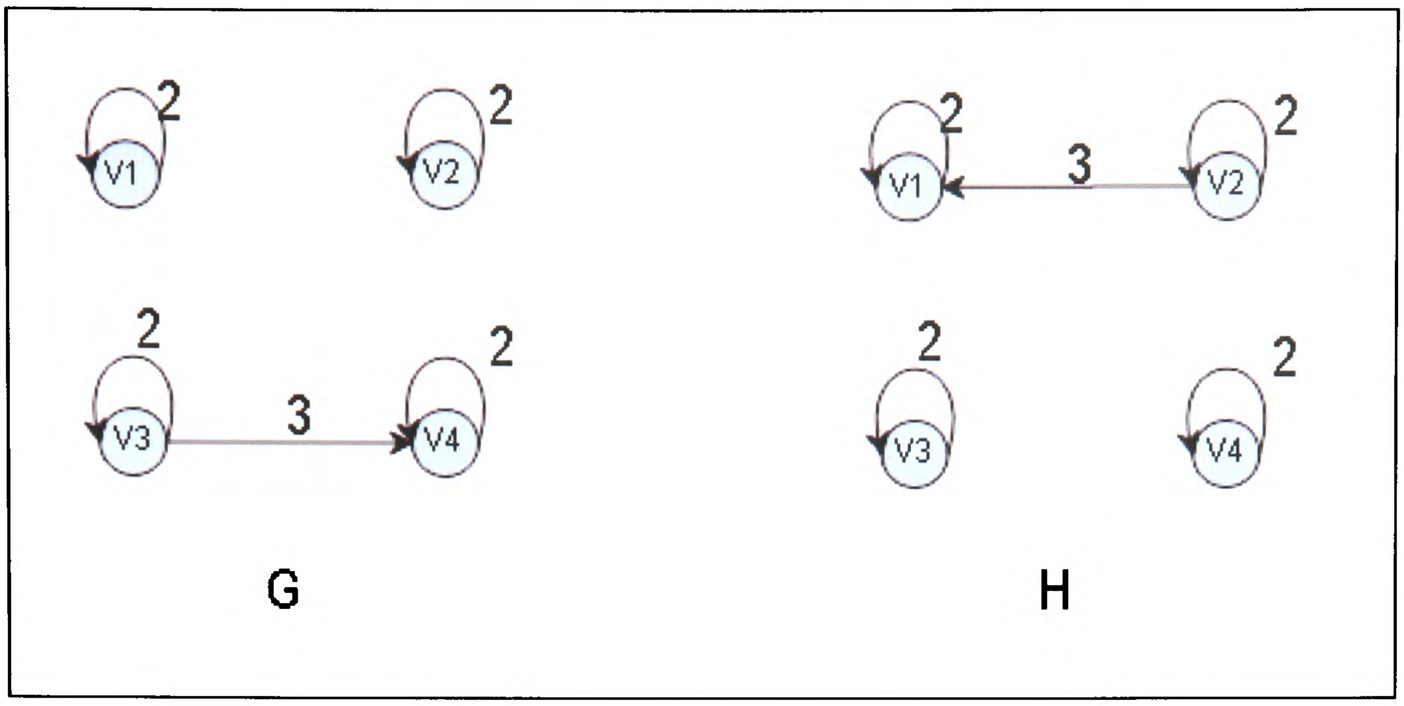


Figure 5.7 Graph pairs with multiple eigenvalues.

The adjacency matrices of G and H are:

$$G=\begin{bmatrix}2&0&0&0\\0&2&0&0\\0&0&2&3\\0&0&0&2\end{bmatrix}, H=\begin{bmatrix}2&0&0&0\\3&2&0&0\\0&0&2&0\\0&0&0&2\end{bmatrix}$$

G and H are isomorphic since $G = PHP^T$ for $P=\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}$

Let A=Ht(G) and B=Ht(H), the eigenvalues of A and B are

$\lambda(A) = \lambda(B)$ =[-0.1213, 2, 2, 4.1213]

We get the approximate solution:

$$P_E = \text{Hungarian}(|V||W|^T) = \begin{bmatrix}0&0&0&1\\0&0&1&0\\1&0&0&0\\0&1&0&0\end{bmatrix}$$

$\left\| G - P_E H P_E^T \right\|_F = 4.2426$, the EDGM algorithm fails to find the best solution, that is, an isomorphic correspondence between G and H which gives a distance of 0 instead.

The need of the "Dissimilar Rows" constraint

Now, we show that "dissimilar rows constraint" is also needed. For instance

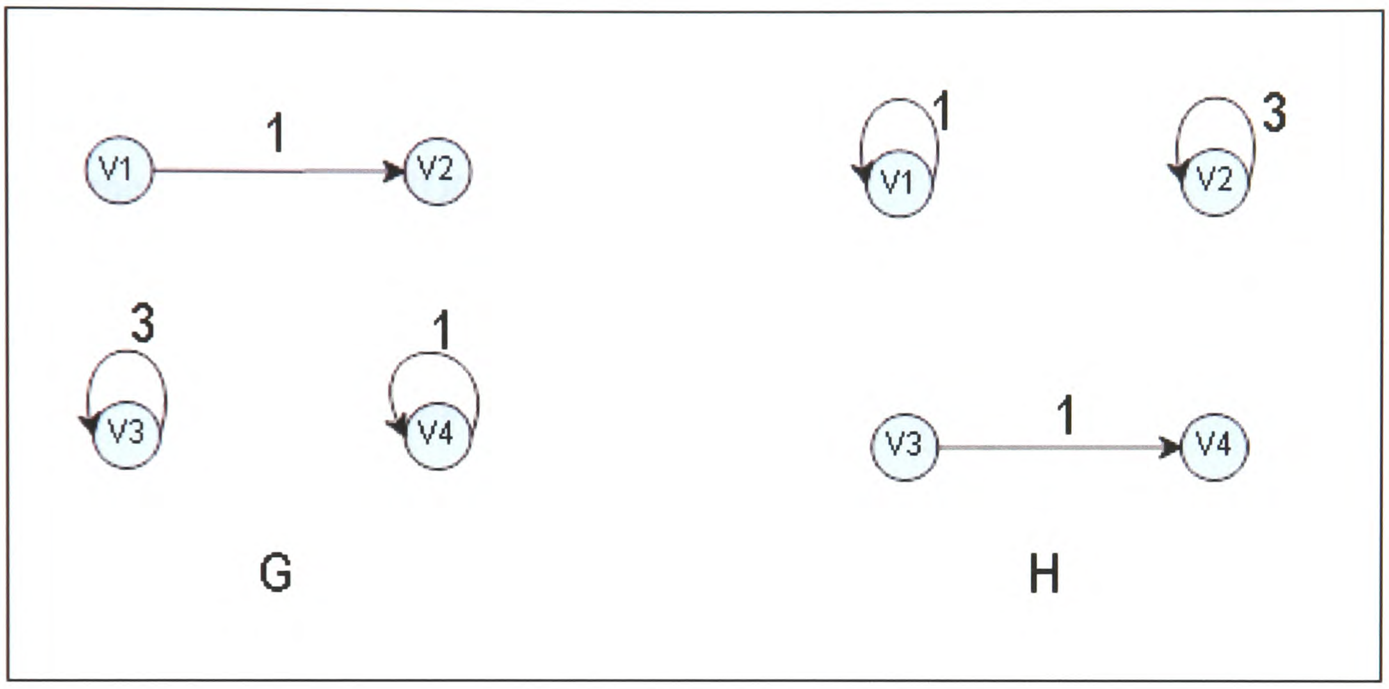


Figure 5.8 Graphs with similar rows of absolute eigenvector

The adjacency matrices of G and H are:

$$G = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

G and H are isomorphic since $G = PHP^T$ for $P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

Let A=Ht(G) and B=Ht(H). The eigenvalues of A and B are

$\lambda(A) = \lambda(B)$=[-1.2153, 2.6386, 3.6255, 5.9512], which are all single and well isolated.

The absolute eigenvectors of Ht(G) and Ht(H) are:

$$|V| = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 \\ \sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad |W| = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 \\ \sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 \end{bmatrix}$$

And the solution from EDGM algorithm is:

$$P_E = \text{Hungarian}(|V||W|^T) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$\|G - PHP^T\|_F = \sqrt{2}$, the algorithm still fails to find the best permutation because the matrices $|V|$ and $|W|$ both have two same rows.

Section 5.2.4 Improvement

In order to theoretically explain why the three constraints are necessary and extend Umeyama's algorithm for general cases where some of the constraints are not satisfied, we introduce here a new approximate formula to graph matching problems.

The approximate formula

Given a Hermitian matrix A with $\lambda(A) = [\lambda_1 = \cdots = \lambda_{n1} < \lambda_{n1+1} = \cdots = \lambda_{n1+n2} < \cdots \le \lambda_n]$ as its eigenvalues, that is, matrix A has k distinct eigenvalues with repeating times $n_1,\ldots,n_k$, respectively, where $\sum_{i=1}^{k} n_i = n$, we can decompose matrix $A=VD_AV^*$, where $V=[V_1,\ldots,V_k]$, and V_j is the eigen-space of the j-th distinct eigenvalue of matrix A.

A simple but important property for the eigen-decomposition is that A=(VX)D$_A$(VX)* is also an spectral decomposition of matrix A, for every unitary matrix X∈U(n$_1$...,n$_k$), where U(n$_1$,...,n$_k$) denotes the set of all block matrices whose j-th diagonal matrix is an n$_j$-by-n$_j$ unitary matrix.

Proposition 5.3: Let Hermitian matrix B=WD$_B$W*, we have the following theorem:

$$\left\|A-PBP^T\right\| \le \left\|PW-VX\right\|_F\left(\left\|D_A\right\|_F+\left\|D_B\right\|_F\right)+\left\|D_A-D_B\right\|_F \tag{5.5}$$

Proof:

$$\left\|A-PBP^T\right\|_F=\left\|(VX)D_A(VX)^*-PWD_BW^*P^T\right\|_F$$

$$=\Big\|\left((VX)D_A(VX)^*-(PW)D_A(VX)^*\right)+\left((PW)D_A(VX)^*-(PW)D_B(VX)^*\right)$$
$$+\left((PW)D_B(VX)^*-(PW)D_B(PW)^*\right)\Big\|_F$$

$$\le\left\|(VX)D_A(VX)^*-(PW)D_A(VX)^*\right\|_F+\left\|(PW)D_A(VX)^*-(PW)D_B(VX)^*\right\|_F$$
$$+\left\|(PW)D_B(VX)^*-(PW)D_B(PW)^*\right\|_F$$

$$=\left\|(VX-PW)D_A(VX)^*\right\|_F+\left\|(PW)(D_A-D_B)(VX)^*\right\|_F+\left\|(PW)D_B(VX-PW)^*\right\|_F$$

$$\le\left\|(VX-PW)\right\|_F\left\|D_A\right\|_F+\left\|D_A-D_B\right\|_F+\left\|(VX-PW)^*\right\|_F\left\|D_B\right\|_F$$

$$\le\left\|PW-VX\right\|_F\left(\left\|D_A\right\|_F+\left\|D_B\right\|_F\right)+\left\|D_A-D_B\right\|_F$$

So it is reasonable to use the following approximate formula to solve graph matching problems:

$$\underset{\substack{P\in Perm(n)\\ X\in U(n_1,\dots,n_k)}}{\arg s}\ \min\left\|PW-VX\right\|_F \tag{5.6}$$

An error estimation theorem for the approximate formula

We shall theoretically prove the accuracy of solving graph matching problem (5.2) by approximate formula (5.6).

Theorem 5.3: Let A and B be two Hermitian matrices with eigenvalues $\lambda(A)=\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ and $\lambda(B)=\beta_1 \le \beta_2 \le \cdots \le \beta_n$, then

(i) (Weyl-Lidskii [Lid1950]) $|\lambda_i - \beta_i| \le \|A-B\|_2$

(ii) Davis-Kahn $\sin\theta$ theorem [DK1970] $d_F(V_1, W_1) = \|\sin\theta(V_1, W_1)\|_F \le \frac{\|R\|_F}{\delta}$

(iii) $d_p(V_1, W_1) = \min_U \|V_1 - W_1 U\|_F \le \sqrt{2} d_F(V_1, W_1)$

Where $V=[V_1, V_2]$ and $W=[W_1, W_2]$ be two unitary matrices. Such that $V^H A V = \begin{bmatrix} D_{A,1} & 0 \\ 0 & D_{A,2} \end{bmatrix}$, $W^H B W = \begin{bmatrix} D_{B,1} & 0 \\ 0 & D_{B,2} \end{bmatrix}$, $R = BV_1 - V_1 D_{A,1}$, and $\delta = \min\{|\lambda - \beta| : \lambda \in \lambda(D_{A,1}), \beta \in \lambda(D_{B,2})\}$, $\lambda(D_{A,1}) \subset [a,b], \lambda(D_{B,2}) \subset \Re \backslash [a,b]$, $a,b \in \Re$ is real field.

The theorem 5.3 only discusses the perturbation of an eigenvalue or its eigenspace. In our case, we have to prove the total perturbation of all eigenvalues or eigenspaces.

Theorem 5.4: Let A and B be two Hermitian matrices with eigenvalues $\lambda(A)=\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ and $\lambda(B)=\beta_1 \le \beta_2 \le \cdots \le \beta_n$, then

(i) (Hoffman-Wielandt [HW1953]) $e(A,B) = \sqrt{\sum_{i=1}^{n} (\lambda_i - \beta_i)^2} \le \|A-B\|_F$ (5.7)

(ii) $\sqrt{\sum_{j=1}^{k} d_F(V_j, W_j)^2} = \sqrt{\sum_{j=1}^{k} \|\sin\theta(V_j, W_j)\|_F^2} \le \frac{\|A-B\|_F}{\delta}$

(iii) $\min_{U \in U(n_1,\ldots,n_k)} \|V - WU\|_F = \sqrt{\sum_{j=1}^{k} d_p(V_k, W_k)^2} \le \sqrt{2}\frac{\|A-B\|_F}{\delta}$

Where the eigenvalues and eigenspaces of matrices A and B are Δ_A, Δ_B, V, W respectively, which are split into k groups as:

$\Delta_A = [\{\lambda_1, \cdots \lambda_{n1}\}, \{\lambda_{n1+1}, \cdots, \lambda_{n1+n2}\}, \cdots \{\cdots, \lambda_n\}]$, $\Delta_B = [\{\eta_1, \cdots \eta_{n1}\}, \{\eta_{n1+1}, \cdots, \eta_{n1+n2}\}, \cdots \{\cdots, \eta_n\}]$

$V = [V_1, V_2, \cdots, V_j]$, $W = [W_1, W_2, \cdots, W_j]$, such that $\max(\Delta_{A,j}) < \min(\Delta_{B,j+1})$ and $\delta = \min_{k \ne j}\{|\lambda_k - \eta_j| : \lambda_k \in \Delta_{A,k}, \eta_j \in \Delta_{B,j}\}$

Proof: (Since only (ii) and (iii) are our theorem, so here we just prove (ii) and (iii))

(ii) Let $R_j = BV_j - V_j D_{A,j}$ then from theorem 5.3 (ii), we get

$\|\sin\theta(V_j, W_j)\|_F^2 \le \frac{\|BV_j - V_j D_{A,j}\|_F^2}{\delta^2}$, so

$$\sqrt{\sum_{j=1}^{k} d_F(V_j, W_j)^2} = \sqrt{\sum_{j=1}^{k} \|\sin\theta(V_j, W_j)\|_F^2} \le \sqrt{\sum_{j=1}^{k} \frac{\|BV_j - V_j D_{A,j}\|_F^2}{\delta^2}}$$

$$= \frac{\|B[V_1, V_2, \ldots, V_k] - [V_1, V_2, \ldots, V_k] D_A\|_F}{\delta} = \frac{\|A-B\|_F}{\delta}$$

(iii) $\min_{U \in U(n_1,\ldots,n_k)} \|V - WU\|_F = \sqrt{\sum_{j=1}^{k} \min_{U_k} \|V_k - W_k U_k\|_F^2} = \sqrt{\sum_{j=1}^{k} d_p(V_k, W_k)^2}$

From theorem 5.3 (iii) and theorem 5.4 (ii), one gets:

$$\sqrt{\sum_{j=1}^{k} d_p(V_k, W_k)^2} \le \sqrt{2\sum_{j=1}^{k} d_F(V_k, W_k)^2} \le \sqrt{2}\frac{\|A-B\|_F}{\delta}$$

Symmetrically, one can get

$$\min_{U\in U(n_1,\dots,n_k)} \|W-VU\|_F \le \sqrt{2}\frac{\|A-B\|_F}{\delta} \tag{5.8}$$

Based on formula (5.5), (5.7) and (5.8), one can easily prove the following important error estimation theorem.

Theorem 5.5: Let G and H be two weighted matrices. If there exists a permutation P_0 such that $G = P_0HP_0^T + E$. Let and $(\bar{P},\bar{X})$ be the optimal solution of (5.6), then

$$\left\|G-\bar{P}H\bar{P}^T\right\|_F \le \left(\sqrt{2}\frac{\|D_A\|_F+\|D_B\|_F}{\delta}+1\right)\|E\|_F$$

Proof: Let A=Ht(G) and B=Ht(H) then from proposition5.2

$$\left\|A-P_0BP_0^T\right\|_F = \left\|\frac{G+G^T}{2}+\sqrt{-1}\frac{G-G^T}{2}-\frac{P_0HP_0^T+(P_0HP_0^T)^T}{2}+\sqrt{-1}\frac{P_0HP_0^T-(P_0HP_0^T)^T}{2}\right\|_F$$

$$=\left\|G-P_0HP_0^T\right\|_F = \|E\|_F$$

So from formula (5.8) we get

$$\min_{X\in U(n_1,\dots,n_k)} \|P_0W-VX\|_F \le \sqrt{2}\frac{\|E\|_F}{\delta},$$

Since $(\bar{P},\bar{X})$ minimize formula (5.6),

$$\|\bar{P}W-V\bar{X}\|_F \le \sqrt{2}\frac{\|E\|_F}{\delta}$$

So from formula (5.5)

$$\left\|A-\bar{P}B\bar{P}^T\right\| \le \|\bar{P}W-V\bar{X}\|_F\left(\|D_A\|_F+\|D_B\|_F\right)+\|D_A-D_B\|_F$$

$$\leq \sqrt{2}\frac{\|E\|_F}{\delta}\left(\|D_A\|_F + \|D_B\|_F\right) + \|E\|_F = \left(\sqrt{2}\frac{\|D_A\|_F + \|D_B\|_F}{\delta} + 1\right)\|E\|_F$$

Theorem 5.5 shows that then the error of the formula (5.6) solving the weighted graph matching problems (5.2) is linearly dependent on the distance of the two graphs G and H. Especially, when G and H are isomorphic, the solution of formula (5.6) is the isomorphism of G and H.

For the attributed graph matching problems (5.1), we have the similar theorem.

Theorem 5.6: Let $\{G_j\}, \{H_j\}, 1 \leq j \leq m$ be two attributed graphs and there exist a permutation matrix P_0 and a real number $\varepsilon > 0$, such that $\left\|G_j - P_0 H_j P_0^T\right\|_F \leq \varepsilon$. If $(\overline{P}, \overline{X})$ be the argument minimizing

$$\min_{\substack{P \in Perm(n) \\ X \in U(n_1, \cdots, n_t)}} \left\|P[t_1 W_1, t_2 W_2, \cdots, t_m W_m] - [t_1 V_1, t_2 V_2, \cdots, t_m V_m] X\right\|_F \qquad (5.9)$$

Where $t_k = \|\lambda(A_k)\|_F + \|\lambda(B_k)\|_F$, then

$$\sum_{i=k}^{m} \left\|G_k - \overline{P} H_k \overline{P}^T\right\|_F \leq mK\varepsilon$$

Where $K = \max_k \{\sqrt{2}\frac{\|\lambda(A_k)\|_F + \|\lambda(B_k)\|_F}{\delta_k} + 1\}$

It can be easily seen from the theorem 5.6 that if the distance between graph G and H is small enough, then the solution gained from formula (5.6) will be satisfactory. In other word, theorem 5.5 guarantees the accuracy of the approximate formula (5.6).

Deducing Umeyama's formula

In fact, formula (5.6) is an optimization on the space of permutation matrices and unitary matrices, which is a mixed 0-1 non-linear programming. Thus, even for the case that all the eigenvalues of matrix A and B are single, it is still not easy to reach the optimization for all graph matching problems. For the case where all the eigenvalues of matrix A and B are single, formula (5.6) can be specified as:

$$\underset{\substack{P\in\mathrm{Perm}(n)\\ x_1,\ldots,x_n\in U(1)}}{\arg s\ \min}\left\|P[w_1,\ldots,w_n]-[v_1x_1,\ldots,v_nx_n]\right\|_F \tag{5.10}$$

where U(1) is the set of all unit complex numbers.

To reach (5.10), we can minimize the distance of the absolute values as an approximation:

$$\underset{P\in\mathrm{Perm}(n)}{\arg s\ \min}\left\|P[|w_1|,\ldots,|w_n|]-[|v_1|,\ldots,|v_n|]\right\|_F \tag{5.11}$$

In this way, we get Umeyama's EDGM algorithm.

The above induction shows the relationship between Umeyama's method and the approximate formula (5.6), and therefore provides a theoretical support to the claims of three constraints of EDGM algorithm. In fact, on one hand, formula (5.6) provide a approximate solution to nearly-isomorphic graph matching with a guaranteed accuracy as specified by theorem 5.3; on the other hand, with the additional "isolating eigenvalues" constraint, formula (5.6) turns out to be formula (5.10), which, with the additional constraint "Dissimilar rows", leads to formula (5.11) that is equivalent to Umeyama's EDGM algorithm.

Unitary invariant meta-basis for Euclid space

In formula (5.6), the optimization on both permutation matrices and unitary matrices makes the problem hard to be solved. However, if the unitary matrix X can be determined somehow beforehand, the problem will become much easier.

The requirement of the unitary matrix X for formula (5.6) is due to fact that there are infinite orthonormal basis for a given Euclid space, rather than a unique one. We shall use an n-by-m matrix V to denote the orthonormal basis of m-dimensional Euclid space in n-dimensional complex space C^n, where each column of V is a vector of the basis. Obviously, each matrix VX, $X \in U(m)$ is also an orthonormal basis of the Euclid space. If we can define a meta-basis which is unique for each Euclid space, then X could be eliminated from formula (5.6).

Proposition 5.4: Let $\vec{1}_n = [1, 1,.., 1]^T$ be the n-dimensional vector with all its elements as 1, then the vector $v=VV^*\vec{1}_n$ is a *unitary invariant vector* which is unique for each Euclid space V.

If the vector v is not a zero vector, then we get a unitary invariant orthonormal vector of Euclid space V. Using the Gram Schmidt Ortho Normalization, the Euclid space V can be orthogonally decomposed as $V = v \oplus Z$. The same as Euclid space V, we can define a unitary invariant vector for space Z. In this way, we define the *unitary invariant meta-basis* for a Euclid space V as shown in table 5.3.

We call the matrix V' defined here a meta-basis of the given Euclid space V. It is important to note that, in some cases, V' may be a real orthonormal basis of the given Euclid space, while in other cases, V' is just a group of orthonormal vectors of the given Euclid space (not necessarily to be a basis – it even can be empty). Obviously, for each Euclid space V, the meta-basis defined in this way is unique.

Table 5.3 Algorithmic definition 5.6 of the meta-basis

```
Function V'=meta-basis(V)
  [n, m]=size(V);       // V is a n-by-m matrix.
  $v = VV^{*}\vec{1}_n$ ;
  if  norm(v)==0
   V'=[];      // V' is empty. fail to find.
   return;
  else if m ==1
      V' =v/norm(v);
      return;
   else
      v=v/norm(v);
      $V = v \oplus Z$ ;//orthogonal decomposition.
      V' =[v, meta-basis(Z)];  //recursively
here.
      end;
   end;
```

Unitary invariant meta-basis graph matching algorithm

Formula 5.6 can be rewritten as

$$\underset{\substack{P\in Perm(n)\\ U_j\in U(n_j)}}{\arg s\ \min}\left\|P[W_1,\dots,W_k]-[V_1U_1,\dots,V_kU_k]\right\|_F \tag{5.12}$$

where V$_j$ is the eigen-space of the j-th eigenvalue of matrix A=Ht(G), and W$_j$ is the corresponding block matrix formed in the same manner as that of V$_j$, rather than the eigen-space of the j-th eigenvallue of B=Ht(H).

To eliminated U$_j$ in formula (5.12), we use the meta-basis V$_j$' of V$_j$ and meta-base W$_j$'of W$_j$, rather than V$_j$ and W$_j$ themselves. In this way, since the meta-base is not dependent on unitary transformation, therefore, formula (5.12) can be simplified as:

$$\underset{P\in Perm(n)}{\arg s\ \min}\left\|P[W_1',\dots,W_k']-[V_1',\dots,V_k']\right\|_F \tag{5.13}$$

N.B. In the case where the meta-basis of V$_j$ and W$_j$ have different numbers of columns, columns from the bigger one will be deleted to make them same.

The formula (5.13) can be simply solved by Hungarian algorithm. Such graph matching algorithm is named unitary invariant meta-basis graph matching algorithm (referred as MBGM algorithm).

Section 5.2.5 Comparison

It has been illustrated that the EDGM algorithm can not work well for sparse graph matching problems. In this section, we compare the MBGM algorithm with EDGM algorithms for sparse graph matching.

Test 1: matching sparse isomorphic graphs

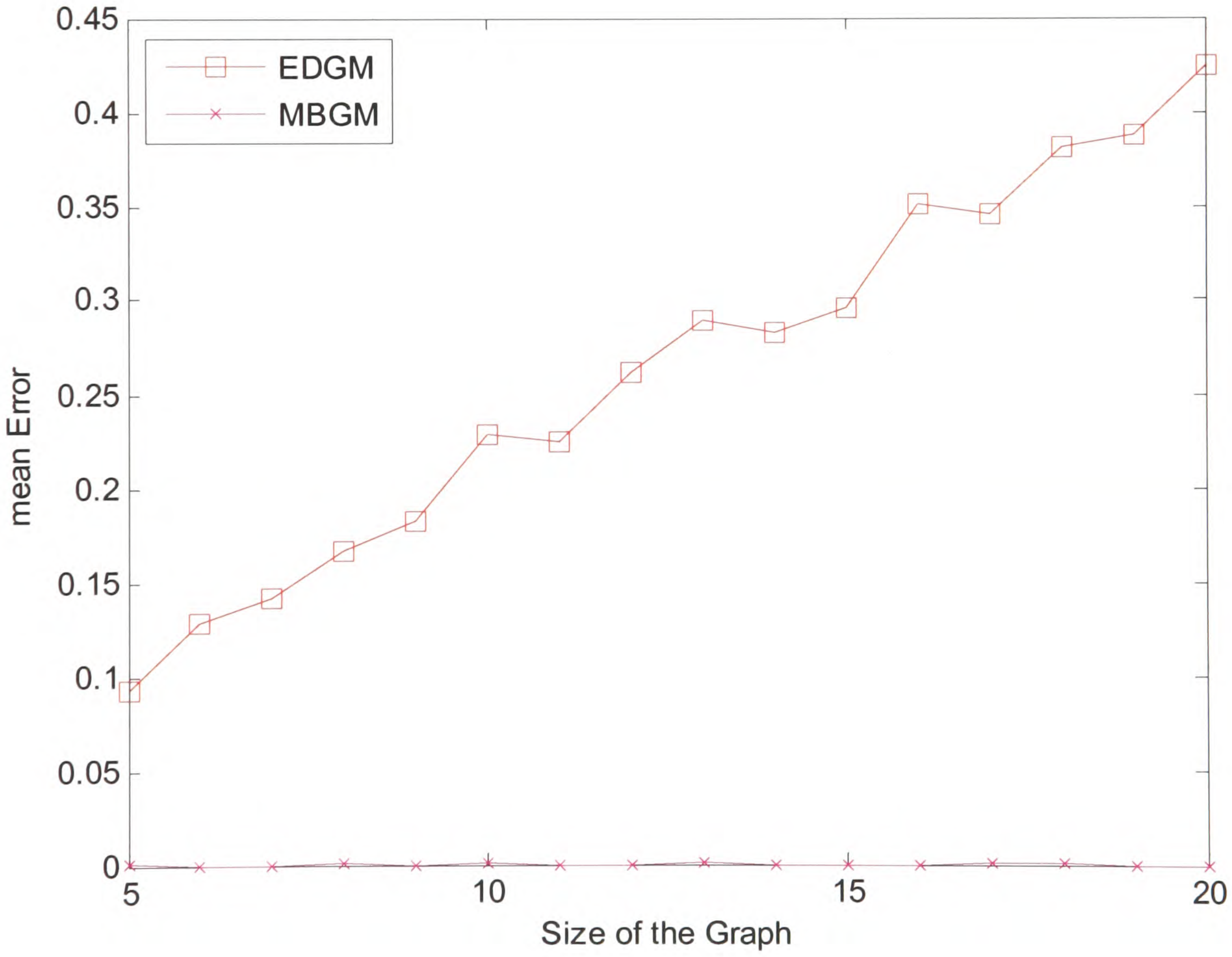


Figure 5.9 EDGM, MBGM for matching isomorphic sparse graph pairs

Test 2: matching sparse graph pairs with perturbations

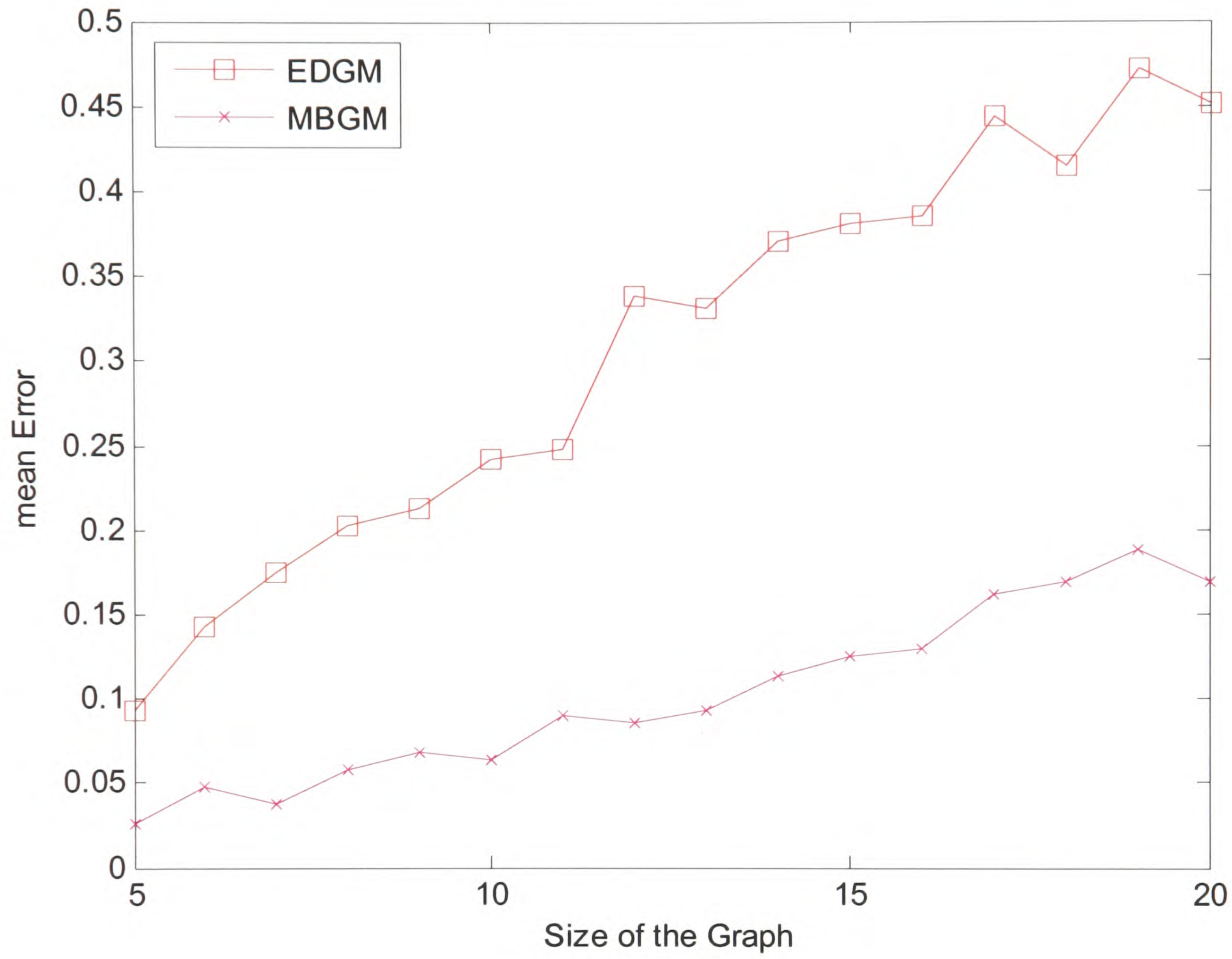


Figure 5.10 EDGM, MBGM for matching sparse graph pairs with perturbation coefficient ε=0.10

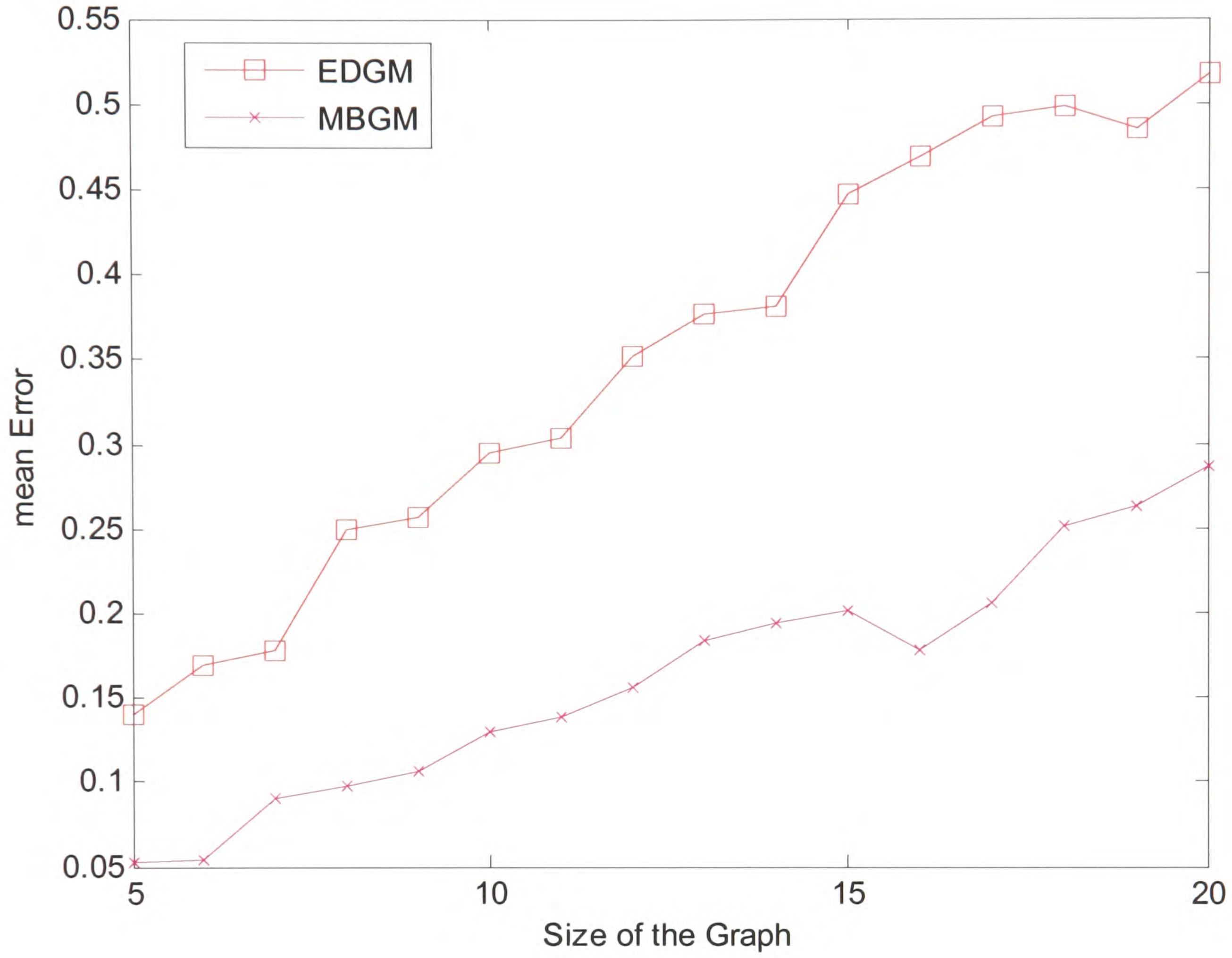


Figure 5.11 EDGM, MBGM for matching sparse graph pairs with perturbation coefficient ε=0.15

Test 3: CPU time comparison

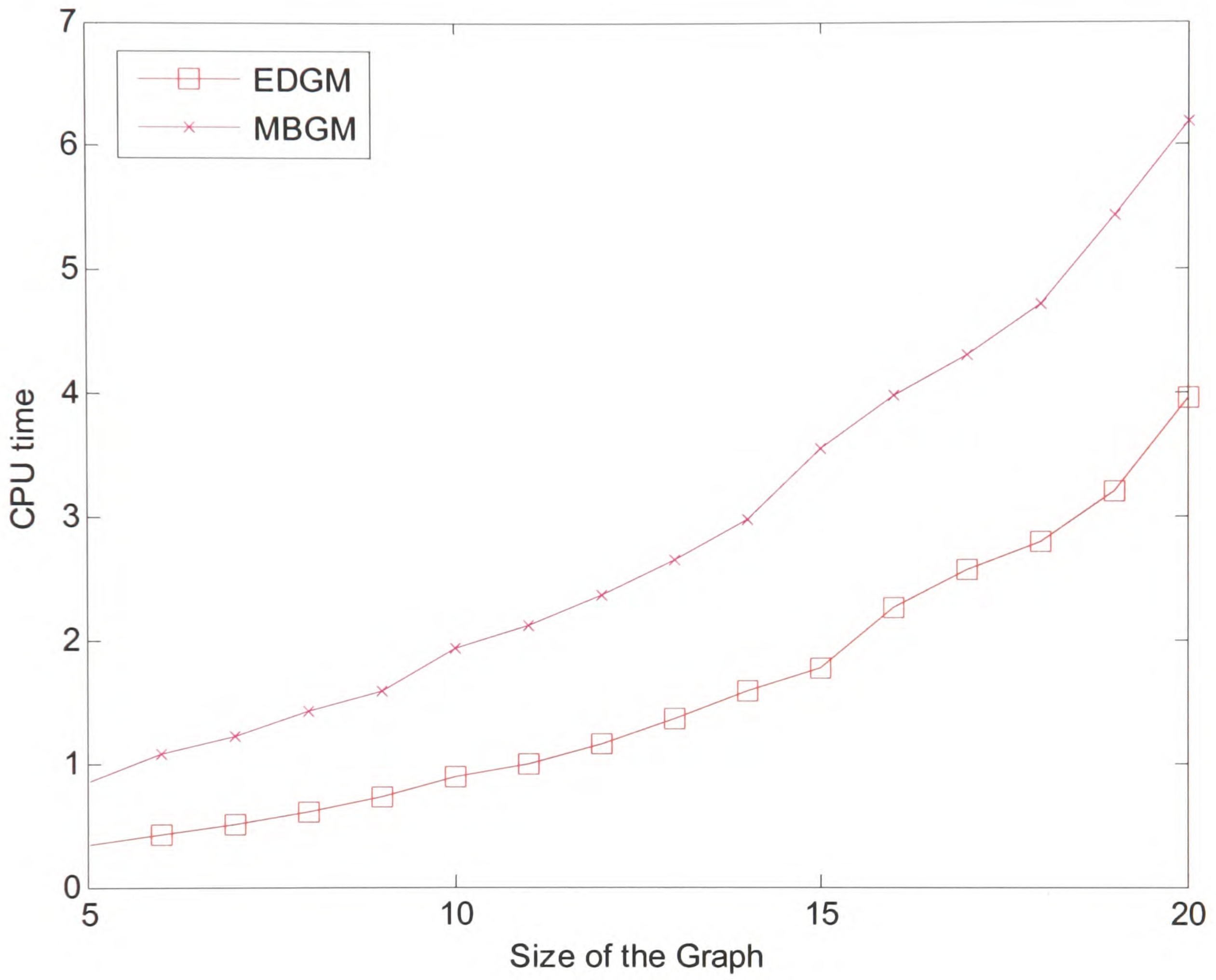


Figure 5.12 CPU time Consuming of EDGM and MBGM

Section 5.2.6 Conclusion

The tests in section 5.3.5 show that the MBGM algorithm improves the matching accuracy to a certain extent. Especially for matching sparse isomorphic graph pairs, the MBGM algorithm almost gets zero error matching result.

But for matching graph pairs with some perturbations, the MBGM algorithm still has large matching error. Besides, the MBGM algorithm takes more CPU consuming to get such an improvement of accuracy.

Section 5.3 Symmetric Polynomial Transform Graph Matching Algorithm and Its Improvement

In [Alm1991], Almohamod presented a method based on fundamental symmetric polynomials for weighted graph matching problems.

A symmetric polynomial is a transformation that maps a set of input data (roots of the polynomial) into a set of coefficients that are invariant under permutation of the input data set. In [Alm1991], a symmetric polynomial is derived for each node in a graph by considering the weights of the node as the roots of the polynomial and these nodes are compared one-to-one through their polynomial coefficients.

Section 5.3.1 Symmetric polynomial transform graph matching algorithm

Matching of weighted undirected graphs

Let G and H be two weighted undirected graphs, which means $G=G^T$ and $H=H^T$. The SPGM algorithm has three dominated steps:

1) Constructing node-attribute matrices.

Considering the j-th node of G, it has the weights

$$(G(j,1), G(j,2), \ldots, G(j,n))$$

The symmetric polynomial $Q_j(G(j,1), G(j,2), \ldots, G(j,n))$ is defined by

$$Q_j(G(j,1), G(j,2), \ldots, G(j,n))=Q_j(x)=\prod_{k=1}^{n}(x-G(j,k))$$

which has the following coefficients:

$$Q_j(x)=\{C_G(j,1), C_G(j, 2),\ldots, C_G(j, n)\}$$

The resulting coefficients matrix C_G, of size nxn, is called a node attribute matrix of

G. For graph H, the same node-attribute matrix can be constructed as C_H.

2) Constructing the node distance matrix

At this stage, it is not known which node in graph H matches the j-th node of graph G. But a node distance matrix D can be constructed by

$$D(j,k)=\sqrt{\sum_{l=1}^{n}(C_G(j,l)-C_H(k,l))^2} \qquad (5.14)$$

Where D(j,k) means the distance of the j-th node of G and the k-th node of H.

3) Calculating the minimum distance match.

Take the matrix D as the cost matrix of assigning n workers to n tasks, and find the cost minimizing assignment. This can be solved by Hungarian algorithm efficiently.

Matching of weighted directed graphs

The process for applying the SPGM algorithm for directed graphs is almost the same as undirected graph except the construction of node attribute matrices.

Let G be a directed weighted graph. Then the weight of the out-edge of j-th node is

$$(G(j,1), G(j,2), \ldots, G(j,n))$$

which differs from the in-edge of j-th node

$$(G(1,j), G(2,j), \ldots, G(n,j))$$

So the j-th node of graph G, two polynomials can be constructed as

$$Q_j(x)=\prod_{k=1}^{n}(x-G(j,k)) \quad q_j(x)=\prod_{k=1}^{n}(x-G(k,j))$$

Therefore, two coefficients matrix CR_G and CC_G is constructed.

The final node-attribute matrix is

$$C_G=[CR_G, CC_G].$$

The second and third steps are the same as that for undirected graphs.

Matching of attributed graphs

The SPGM algorithm is presented only for weighted graph matching. In fact it can be easily extended for attributed graph matching.

Let $\{G_k:1\leq k\leq m\}$be a attributed graph, where G_k is the k-th adjacency matrix. Similarly as directed graphs, the k-th attribute matrix of graph G is

$$C_{G,k}=[CR_{G,k}, CC_{G,k}]$$

And the node attribute matrix of graph G is

$$C_G=[CR_{G,1}, CC_{G,1}, CR_{G,2}, CC_{G,2}, \ldots, CR_{G,m}, CC_{G,m}]$$

Then the node distance matrix can be constructed and the optimal matching can be calculated by Hungarian algorithm.

Section 5.3.2 Computational complexity

The computational complexities of the three dominate steps are

1) Calculating coefficients of 2n polynomials: $O(n^3)$ [Alm1997]

2) Calculating the distance matrix (5.14): $O(n^3)$

3) Using Hungarian algorithm: $O(n^3)$ [AMO1993]

So the total computational complexity is $O(n^3)$.

Section 5.3.3 Analysis

The spirit of SPGM algorithm is exploring some kinds of node similarity or distance based on the node attribute.

The node attribute of a graph must be constructed independent on the order of its nodes. In other words, the nodes of a graph G can be numbered in some different way, then one can get different adjacency matrix, but the attribute of a certain node should not effected by such re-numbering.

Take the j-th node of G as example, although the weights G(j,1), G(j,2) may change if other nodes are re-ordered, the set {(G(j,1), G(j,2), …, G(j,n))}remains the same, that is the out-edge weights of j-th node, so does in-edge weight set { (G(1,j), G(2,j), …, G(n,j)) }, so these set can be select as the as the node attribute of j-th node. The SPGM algorithm uses the coefficients of the polynomial $Q_j(x)$ and $q_j(x)$ as the attribute of j-th node, which is equivalent to the set.

For example, figure 5.13 shows that a weighted graph G1 with four nodes has been re-ordered to G2. The out-edge weights of node v_1 in graph G_1 is (0, 1, 2, 3), which has change to (0, 3, 2, 1) in graph G_2. But they are still the same elements just been re-ordered.

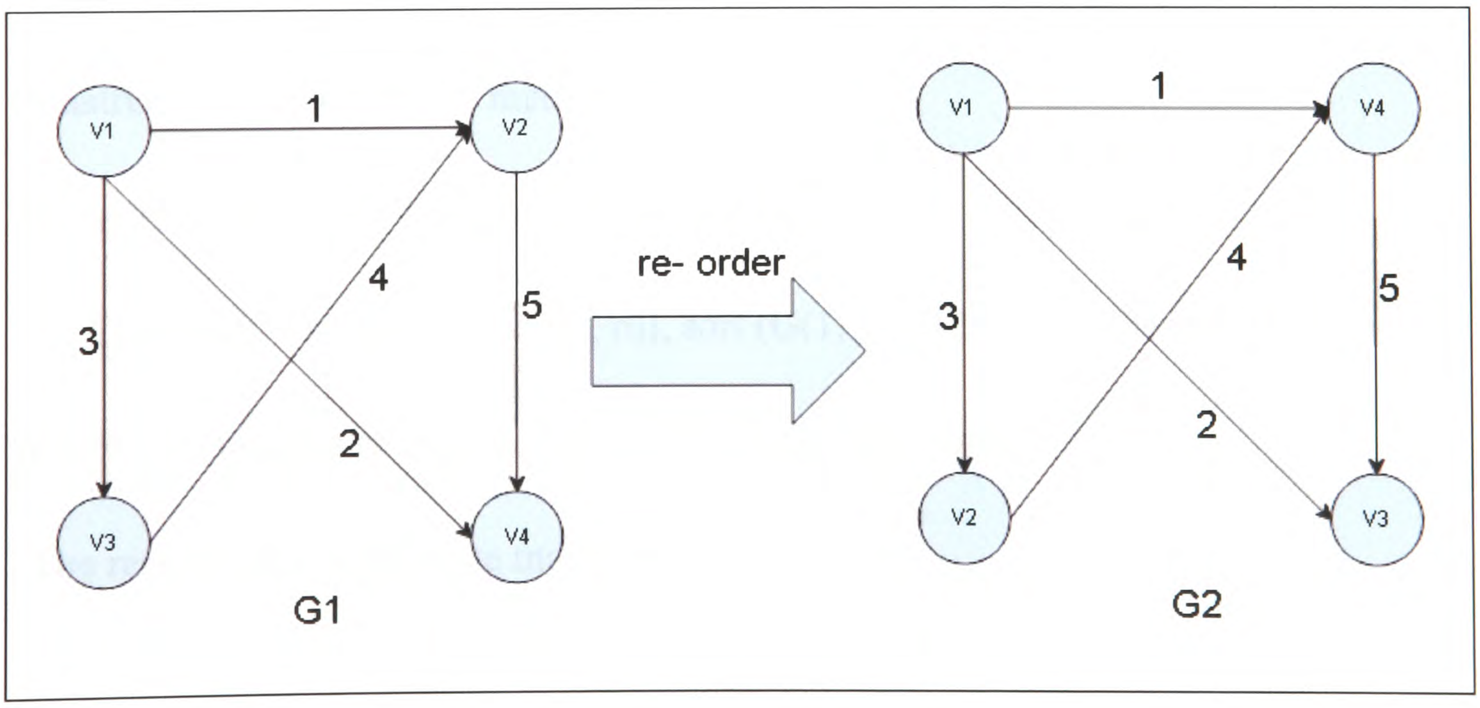


Figure 5.13 Re-order of the nodes of a graph

For the undirected graphs, since the out-edge weights are the same as the in-edge weights, so only the out-edge weights are selected to construct the node attribute matrices.

Section 5.3.4 Constraint

From the tests in section 5.1.4, it is easily can be seen that the SPGM algorithm works well only for isomorphic graph pairs. The matching error grows with the distance between two graphs becomes large.

Section 5.3.5 Improvement

It has been pointed out that the SPGM algorithm tries to construct node similarity based on some kind of node attribute. Since the sets {(G(j, 1), G(j, 2), …, G(j, n))} and { (G(1, j), G(2, j), …, G(n, j)) } doesn't effected by the order of the nodes of graph G. So the SPGM algorithm chooses the coefficients of the polynomials transform of the two sets.

One may ask is it possible just use the two sets {(G(j, 1), G(j, 2), …, G(j, n))} and { (G(1, j), G(2, j), …, G(n, j)) } as the node attribute of node j. The answer is positive, and we present a new sort based graph matching algorithm (STGM), which has three dominated steps:

1) Constructing node-attribute matrices.

The node attribute of j-th node of G, is constructed as:

[sort (G(j, 1), G(j, 2), …, G(j, n)), sort (G(1, j), G(2, j), …, G(n, j))]

Where sort() means re-arrange the vector in ascending order

The resulting node attribute matrix

$$C_G = \begin{bmatrix} \mathrm{sort}(G(1,1),G(1,2),...,G(1,n)) & \mathrm{sort}(G(1,1),G(2,1),...,G(n,1)) \\ \vdots & \vdots \\ \mathrm{sort}(G(n,1),G(n,2),...,G(n,n)) & \mathrm{sort}(G(1,n),G(2,n),...,G(n,n)) \end{bmatrix},$$

2) Constructing the node similarity matrix

The node similarity matrix S can be constructed by

$$S=C_G \times C_H^T \tag{5.15}$$

Where S(j,k) means the similarity of the j-th node of G and the k-th node of H.

3) Calculating the maximum similarity match.

This is the maximum linear assignment problem which can be solved by Hungarian algorithm efficiently.

The STGM simply takes the in and out edges weights as the node attribute, constructs node similarity matrix by formula (5.15) and calculates optimal match by Hungarian algorithm.

Section 5.3.6 Comparison

It has been illustrated that the SPGM algorithm can not work well for matching graph pairs with perturbations. In this section, we compare the STGM algorithm with SPGM algorithms for matching graph pairs with perturbations.

Test 1: matching dense graphs with perturbation

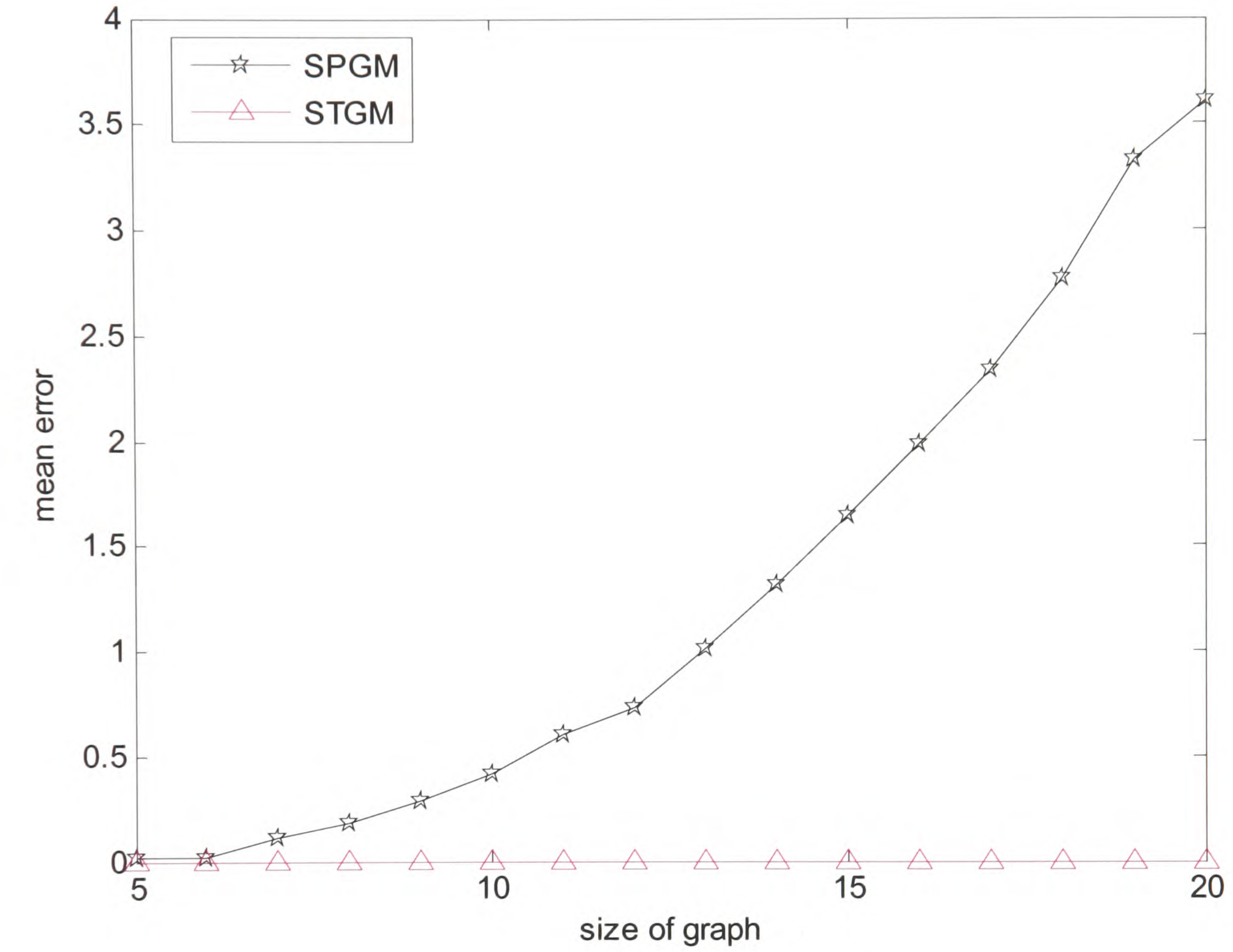


Figure 5.14 STGM, SPGM for matching dense graph pairs with perturbation coefficient ε=0.10

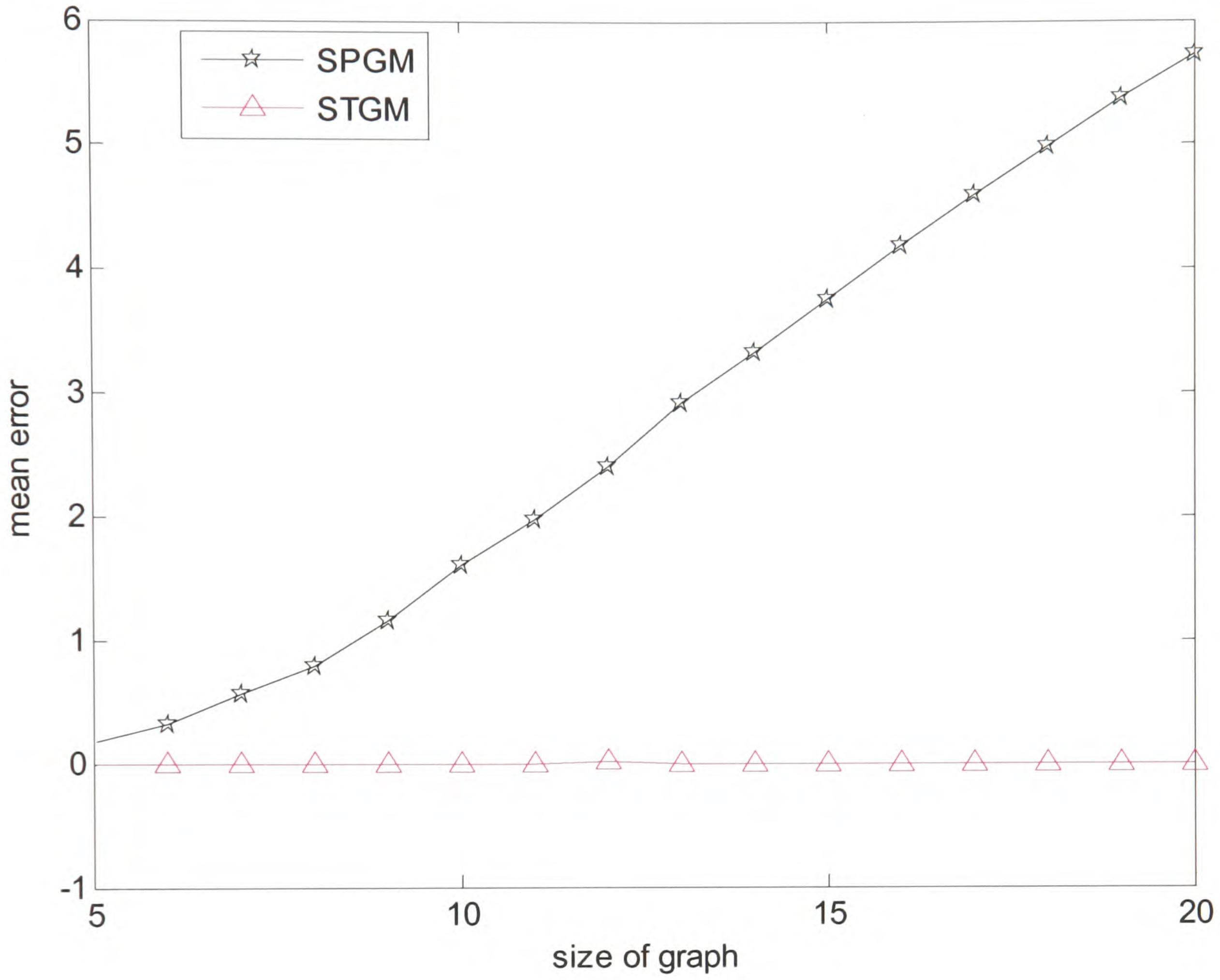


Figure 5.15 STGM, SPGM for matching dense graph pairs with perturbation coefficient ε=0.20

Test 2: matching sparse graphs with perturbation

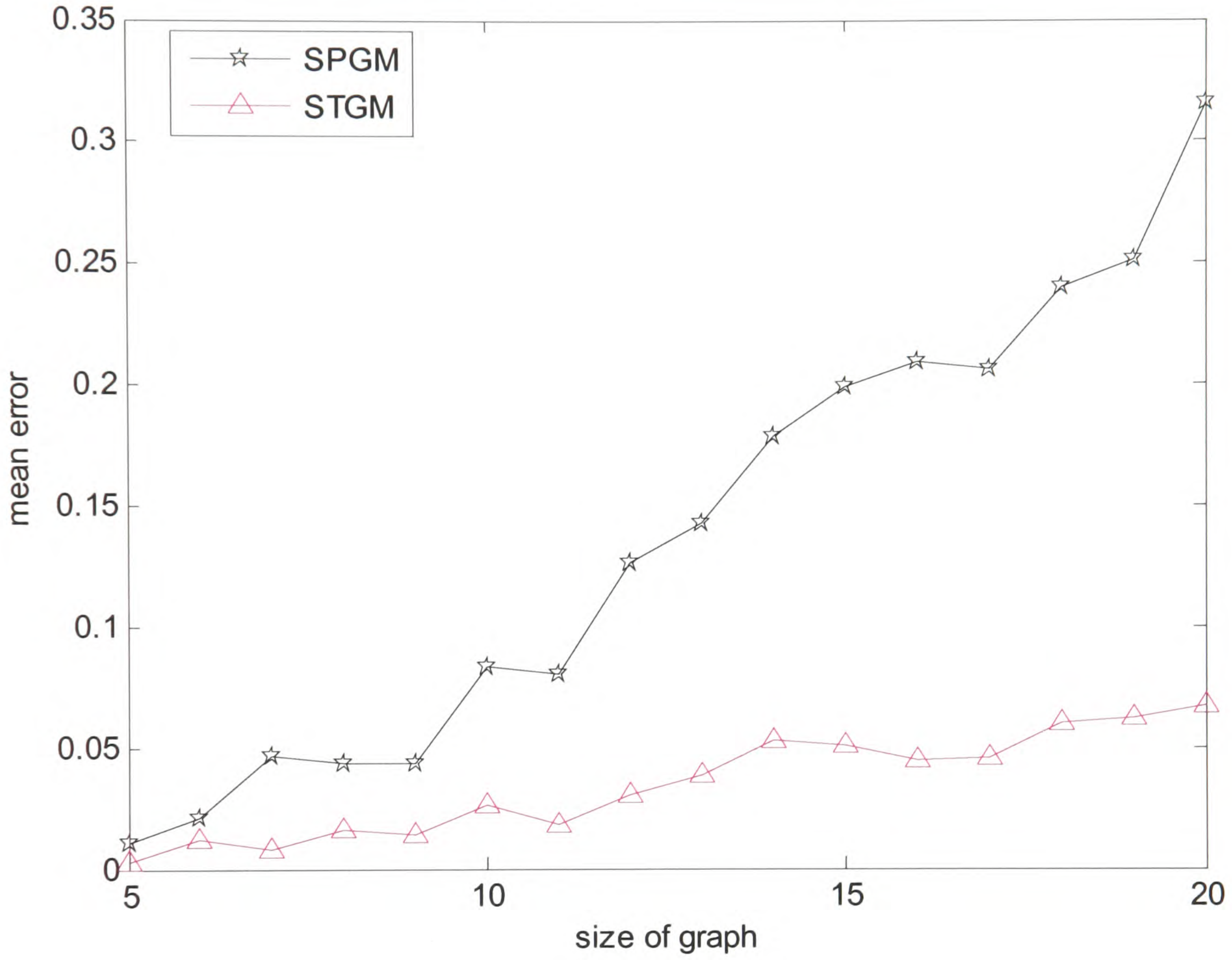


Figure 5.16 STGM, SPGM for matching sparse graph pairs with perturbation coefficient ε=0.10

Test3: CPU time comparison

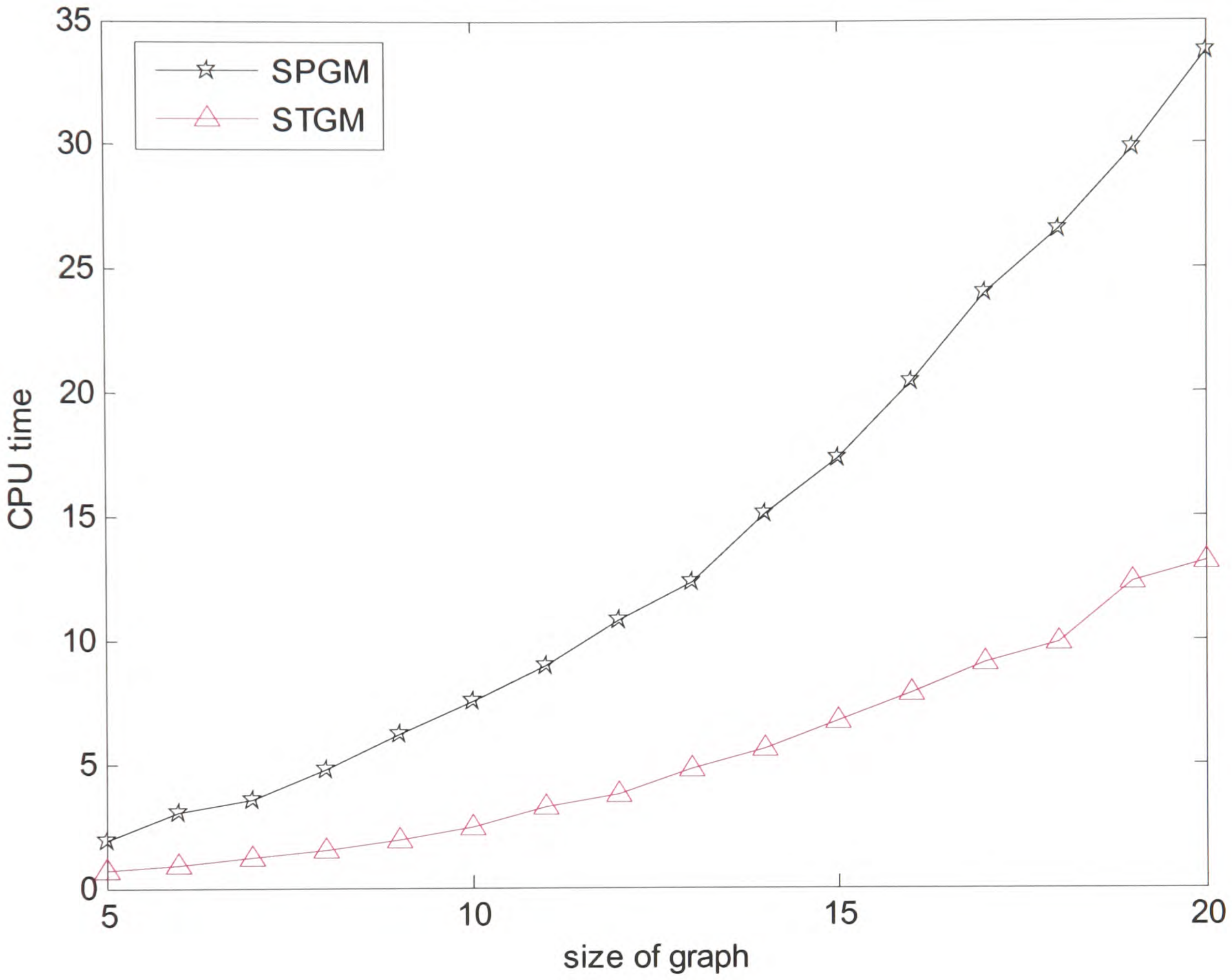


Figure 5.17 CPU time Consuming of SPGM and STGM

Section 5.3.7 Conclusion

The tests in last section show that the STGM algorithm greatly improves the accuracy of matching the non-isomorphic dense graph pairs and, to some extent, improve the accuracy of matching non-isomorphic sparse graph pairs. Besides, the CPU time consuming of the STGM algorithm decreases instead of increasing compared with SPGM algorithm.

Section 5.4 Node Similarity Graph Matching Algorithms

We have noticed that the STGM algorithm gets a little matching error for matching non-isomorphic sparse graph pairs.

In this section, we shall not improve the STGM or MBGM algorithms even better, but discuss the difficulties in solving graph matching problems in such a way.

Instead of discussing constraints for each algorithm, we shall deal with them together by the notion of algorithm framework.

Section 5.4.1 Node similarity graph matching framework

The *node similarity graph matching framework* (NSGM) is a general and abstract method for solving graph matching problem.

Definition 5.7: Given two graphs G and H, the NSGM framework has the following two significant steps:

1) Constructing node-similarity matrix S(G, H).

The entry S(G, H)(i,j) denotes the similarity of i-th node of G and j-th node of H. In some cases, if a node-distance matrix D is provided instead of node similarity matrix, then similarity matrix S can be simply set as –D or m-D, where m is the maximum element of D.

2) Calculating the maximum similarity match.

$$\underset{P\in \mathrm{Perm}(n)}{\mathrm{args\ max}} \sum_{i,j=1}^{n} S(i,j)\times P(i,j) \qquad (5.16)$$

Requirement

One may ask whether the selection of node similarity function S(G, H) is arbitrary or not. The answer is negative. Since the entry of S(G, H) denotes the similarity between the nodes of graph G and H, the similarity must be independent on the order of the nodes as discussed in section 5.4.3. So the node similarity function S must satisfy the following important constraint:

Independence of Order I:

$$S(PGP^T, QHQ^T) = P \times S(G,H) \times Q^T, \text{ for all } P, Q \in \mathrm{Perm}(n) \qquad (5.17)$$

The independence of order I constraint is the most important and basic principal for the node similarity graph matching framework and directly determine the applying scope of the node similarity graph matching framework.

Section 5.4.2 Examples of node similarity based graph matching algorithm

In this section, we shall introduce some examples of node similarity based graph matching algorithms.

Theorem 5.7: the following functions are all node similarity functions.

1) $S_1(G,H)(i,j) = -|G(i,i) - H(j,j)|$

2) $S_2(G,H)(i,j) = \sum_{k=1}^{n}\sum_{l=1}^{n} G(k,i) \times H(l,j)$

3) $S_3(G,H) = \mathrm{unvec}((H \otimes G + H^T \otimes G^T)^{\infty} \times \mathrm{vec}(1_{n\times n}))$,

where m can be any natural number and $1_{n\times n}$ is a n-by-n matrix with all entries 1, $\otimes$ denotes the Kronecker product operator and vec() and unvec() denote the vectorization operator and its reverse..

The following proposition is important to prove the above theorem.

Proposition 5.5: Let G be a n-by-n matrix, P and Q be two permutation matrices with p, q as their corresponding permutation vectors, then

$$PGQ(i, j)=G(p(i), q(j)) \qquad (5.18)$$

Where PQG(i, j) stands for the (i, j)-th entry if the matrix M=PQG.

Proof to Theorem 5.7:

1)

$$\begin{aligned} S_1(PGP^T, QHQ^T)(i,j) &= -\left|PGP^T(i,i) - QHQ^T(j,j)\right| \\ &= -\left|G(p(i),p(i)) - H(q(j),q(j))\right| \\ &= S_1(G,H)(p(i),q(j)) \\ &= PS_1(G,H)Q(i,j) \end{aligned}$$

2)

$$\begin{aligned} S_2(PGP^T, QHQ^T)(i,j) &= \sum_{k=1}^{n}\sum_{l=1}^{n} PGP^T(k,i) \times QHQ^T(l,j) \\ &= \sum_{k=1}^{n}\sum_{l=1}^{n} G(p(k),p(i)) \times H(q(l),q(j)) \\ &= \sum_{k=1}^{n}\sum_{l=1}^{n} G(k,p(i)) \times H(l,q(j)) \\ &= S_2(G,H)(p(i),q(j)) \\ &= PS_2(G,H)Q^T(i,j) \end{aligned}$$

3)

It is well known [Wil1997] that

$$\mathrm{vec}(AXB) = B^T \otimes A\mathrm{vec}(X)$$

Let $X_m(G, H)=\mathrm{unvec}((H \otimes G + H^T \otimes G^T)^m \times \mathrm{vec}(1_{n\times n}))$, then

$X_0(G, H)=1_{n\times n}$, $X_{m+1}(G, H)=GX_m(G, H)H+G^TX_m(G, H)H^T$ and $S_3(G, H)=\lim_{m\to\infty} X_m$

We prove that

$$X_m(PGP^T, QHQ^T)=PX_m(G, H)Q^T \tag{5.19}$$

by induction on m.

Obviously, formula (5.19) holds for m=0

Assume (5.19) is true for m=k then

$$\begin{aligned}
&X_{m+1}(PGP^T, QHQ^T) \\
&= PGP^T X_m(PGP^T, QHQ^T)QHQ^T + PG^T P^T X_m(PGP^T, QHQ^T)QH^T Q^T \\
&= PGP^T PX_m(G,H)Q^T QHQ^T + PG^T P^T PX_m(G,H)Q^T QH^T Q^T \\
&= PGX_m(G,H)HQ^T + PG^T X_m(G,H)H^T Q^T \\
&= P\left(GX_m(G,H)H + G^T X_m(G,H)H^T\right)Q^T \\
&= PX_{m+1}(G,H)Q^T
\end{aligned}$$

Which means (5.19) is true for m=k+1.

From the induction principal (5.19) holds for all integer m.

$$S_3(PGP^T, QHQ^T) = \lim_{m\to\infty} X_m(PGP^T, QHQ^T)$$

$$= \lim_{m\to\infty} PX_m(G,H)Q^T = P \lim_{m\to\infty} X_m(G,H)Q^T = PS_3(G,H)Q^T$$

The function S_1, S_2 and S_3 have been proved to be node similarity functions, so we can get three corresponding node similarity based graph matching algorithms. It is easily can be seen that the S_3 node similarity function based graph matching algorithm is exactly the hubs and authorities graph matching algorithm proposed by Kleinberg [Kle1999]. We also get the following theorems showing that the S_2 node similarity function based graph matching algorithm is exactly the least square Kronecker product-successive projection graph matching algorithm proposed in [Wyk2002].

Theorem 5.8: The S_2 node similarity function based graph matching algorithm is the least square Kronecker product-successive projection graph matching algorithm.

Proof: for the LSKPGM algorithm, the node similarity S(i, j) is calculated by

$$S(i,j)=\frac{\sum_{k=(i-1)n+1}^{i*n}\sum_{l=(j-1)n+1}^{j*n}\Phi_{k,l}}{n}$$

$$\text{Where } \Phi^T=\Theta^{+}\Pi,\ \Theta=\begin{bmatrix}\mathrm{vec}(H_1)^T\\ \vdots\\ \mathrm{vec}(H_m)^T\end{bmatrix} \text{ and } \Pi=\begin{bmatrix}\mathrm{vec}(G_1)^T\\ \vdots\\ \mathrm{vec}(G_m)^T\end{bmatrix}$$

For the weighted graphs, where m=1, above formulae degrade to

$$\Phi=\Pi^T(\Theta^{+})^T=\frac{\mathrm{vec}(G)\mathrm{vec}(H)^T}{\|\mathrm{vec}(H)\|_2^2}$$

$$\text{So the node similarity } S(i,j)=\frac{\sum_{k=(i-1)n+1}^{i*n}\sum_{l=(j-1)n+1}^{j*n}\Phi_{k,l}}{n}=\frac{\sum_{k=1}^{n}\sum_{l=1}^{n}G(k,i)\times H(l,j)}{n\|\mathrm{vec}(H)\|_2^2}$$

Since the solution of maximum linear assignment doesn't effect by a positive scalar, so the LSKPGM algorithm coincide with the S_2-NSGM algorithm.

One may notice that some graph matching algorithms derived from different fields can be unified by the node similarity graph matching framework. In next section, we shall introduce more node similarity graph matching algorithms.

Section 5.4.3 Node attribute functions

Obviously, to define node similarity S(G, H), one has to consider both graphs G and H at the same time. It will be much easier if one can define this similarity by dealing with two graphs separately. So a new important notion node attribute function will be introduced.

Definition 5.8: A *node-attribute function* is a function $f : R^{n\times n} \to R^{n\times m}$ which satisfies

Independence of Order II:

$$f(PGP^T) = Pf(G)\text{, for all } G \in R^{n\times n} \text{ and } P \in \text{perm}(n) \qquad (5.18)$$

Intuitively, the weight matrix of a graph G describes the edge attributes and the node attribute function f maps these edge-attributes to node attributes of graph G.

The independence of order II constraint has to be satisfied to make sure the node attributes of a graph are independent on the order of its nodes.

Theorem 5.9: The following functions are all node-attribute functions:

4) $f_4(G)(k,:)$=[sum(G(k,:)), sum(G(:,k))]

5) $f_5(G)(k,:)$=[poly(G(k,:)), poly(G(:,k))]

6) $f_6(G)(k,:)$=[sort(G(k,:)), sort(G(:,k))]

7) $f_7(G)(:,k)=\text{diag}\left(\frac{G^k}{n^k}\right)$, k=1,2,...,n.

Proof:

4)

$$\begin{aligned} f_4(PGP^T)(k,:) &= [\text{sum}(PGP^T(k,:)), \text{sum}(PGP^T(k,:))] \\ &= [\text{sum}(G(p(k),p)), \text{sum}(G(p(k),p))] \\ &= [\text{sum}(G(p(k),:)), \text{sum}(G(p(k),:))] \\ &= f_4(G)(p(k),:) \\ &= Pf_4(G)(k,:) \end{aligned}$$

Proof of 5) 6) is the same as 4).

7)

$$\begin{aligned}
f_7(PGP^T)(:,k) &= \mathrm{diag}\left(\frac{(PGP^T)^k}{n^k}\right) \\
&= \mathrm{diag}\left(\frac{PG^kP^T}{n^k}\right) \\
&= P\mathrm{diag}\left(\frac{G^k}{n^k}\right) \\
&= Pf_7(G)(:,k)
\end{aligned}$$

The next example is more difficult.

Theorem 5.10: Let f_8 be a function calculated by the following three steps, then f_8 is a node attributed function satisfying independence of order II.

1) Calculating the Hermitian matrix $Ht(G) := \frac{G+G^T}{2} + \frac{G-G^T}{2}\sqrt{-1}$

2) Calculating eigen-decomposition of the matrix $Ht(G)=VDV^*$, where D is the diagonal matrix of eigenvalues in descending order.

Suppose that Ht(G) has k distinct eigenvalue $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_k$ with repeat times n_1, n_2, …, n_k and eigenspace V_1, V_2, …, V_k..

3) Let $f_8(G)=[V_1^{'}, V_2^{'}, \cdots, V_k^{'}]$, where $V_i^{'}$ is the meta-basis of eigenspace V_i.

Proof:

We calculate $f_8(PGP^T)$ step by step.

Obviously, $Ht(PGP^T)=PHt(G)P^T$

Since Ht(G)=UDU*,

$$Ht(PGP^T) = [PV_1U_1,\cdots,PV_kU_k]D([PV_1U_1,\cdots,PV_kU_k])^T, \text{ where } U_i \in U(n_i)$$

Calculate meta-basis of PV_iU_i. The meta-basis is defined by the unitary invariant vector defined in proposition5.4, so we only have to verify that unitary invariant vector satisfies the independence of order II.

The unitary invariant vector of eigenspace V_i is $V_iV_i^* \vec{1}_n$

The unitary invariant vector of eigenspace PV_iU_i is

$$(PV_iU_i)(PV_iU_i)^*\vec{1}_n = PV_iU_iU_i^*V_i^*P^T\vec{1}_n = PV_iV_i^*\vec{1}_n$$

So $f_8(PGP^T)=Pf_8(G)$

Theorem 5.11: Let f_9 be a function calculated by the following three steps:

1) Calculating Hermitian matrix $Ht(G) := \frac{G+G^T}{2} + \frac{G-G^T}{2}\sqrt{-1}$

2) Calculating eigen-decomposition of the matrix Ht(G)=VDV*, where D is the diagonal matrix of eigenvalues in descending order.

3) Let $f_8(G)=|V|$

If all the eigenvalues of Ht(G) are single, then f_9 is a node attributed function satisfying independence of order II.

Proof:

The same as theorem5.10, We calculate $f_9(PGP^T)$ step by step.

$Ht(PGP^T)=PHt(G)P^T$

Ht(G)=UDU*, Since all the eigenvalues of Ht(G) is single, then

$$\mathrm{Ht}(PGP^T) = [PV_1x_1,\cdots,PV_kx_k]D([PV_1x_1,\cdots,PV_kx_k])^T\text{, where } x_i \in U(1)$$

$$\text{Calculate } f_9(PGP^T)=\left|[PV_1x_1,\cdots,PV_kx_k]\right| = \left|[PV_1,\cdots,PV_k]\right| = P\left|V\right| = Pf_9(G)$$

Using the node-attribute function, one can easily construct the corresponding node-similarity function by the following theorem.

Theorem 5.12: Let f be a node-attribute function, and S_f is defined as:

$$S_f(G,H) = f(G)\times f(H)^T \tag{5.20}$$

Then S_f is a node-similarity function.

The Proof is trivial.

Therefore, each node-attribute function in theorem 5.9, theorem 5.10 and theorem 5.11 defines a corresponding node-similarity function denoted as S_4, to S_9. The matching algorithms by node-similarity functions S_5 and S_6 are exactly the symmetric polynomials transformation graph matching algorithm SPGM and the improved sort based graph matching algorithm STGM; The matching algorithms by node-similarity functions S_9 and S_8 are exactly the eigen-decomposition graph matching algorithm EDGM and the improved meta-basis based graph matching algorithm MBGM.

The node similarity graph matching algorithm based on the similarity function S_1 and S_4 and S_7 is simply named as S_1-NSGM and S_4-NSGM and S_7-NSGM.

Section 5.4.4 Comparisons

In this section, these node similarity based graph matching algorithms associated with node-similarity function S_1 to S_9 are numerically compared.

Test 1: matching isomorphic sparse graph pairs

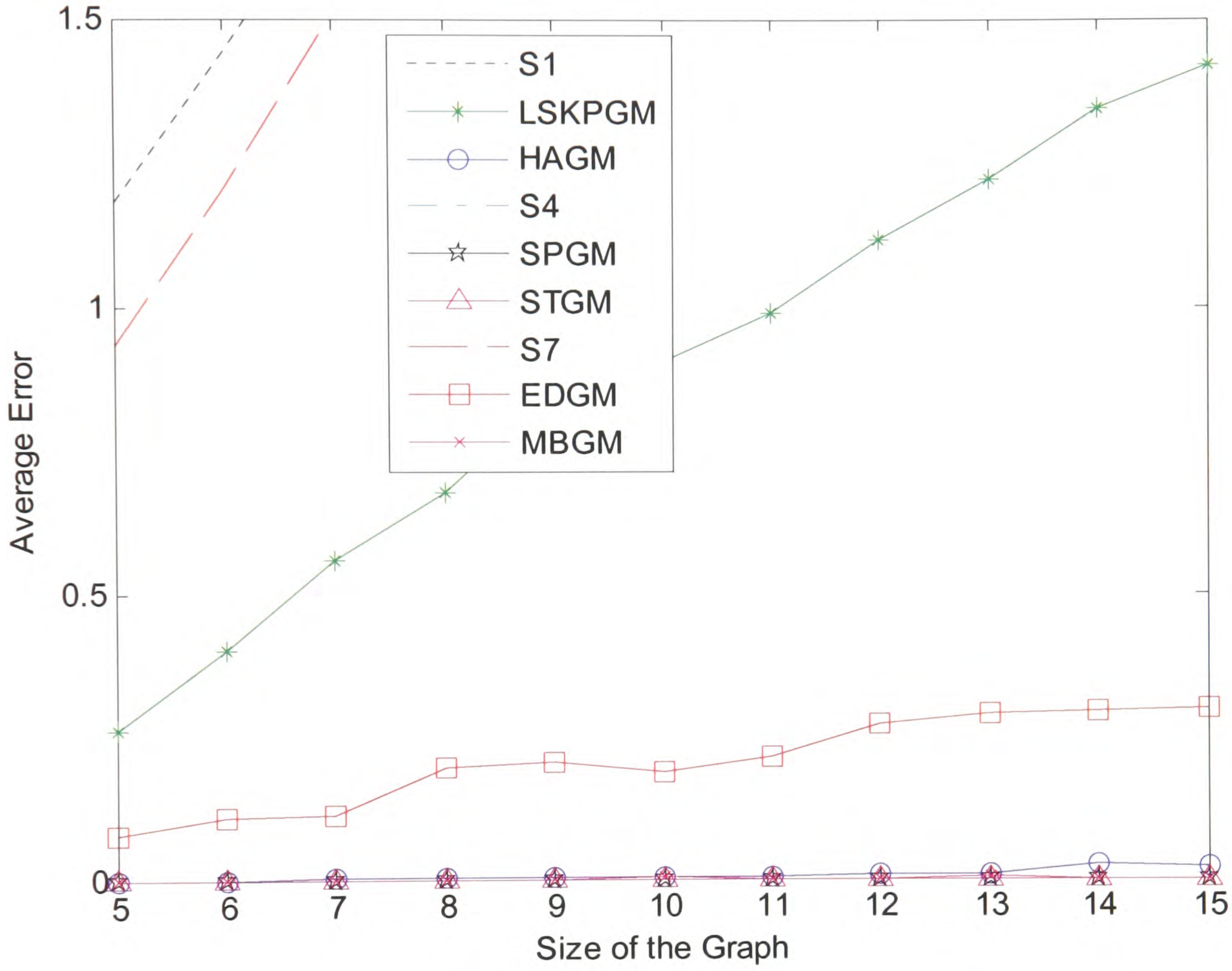


Figure 5.18 Nine algorithms for matching isomorphic sparse graph pairs

Test 2: matching non-isomorphic dense graph pairs

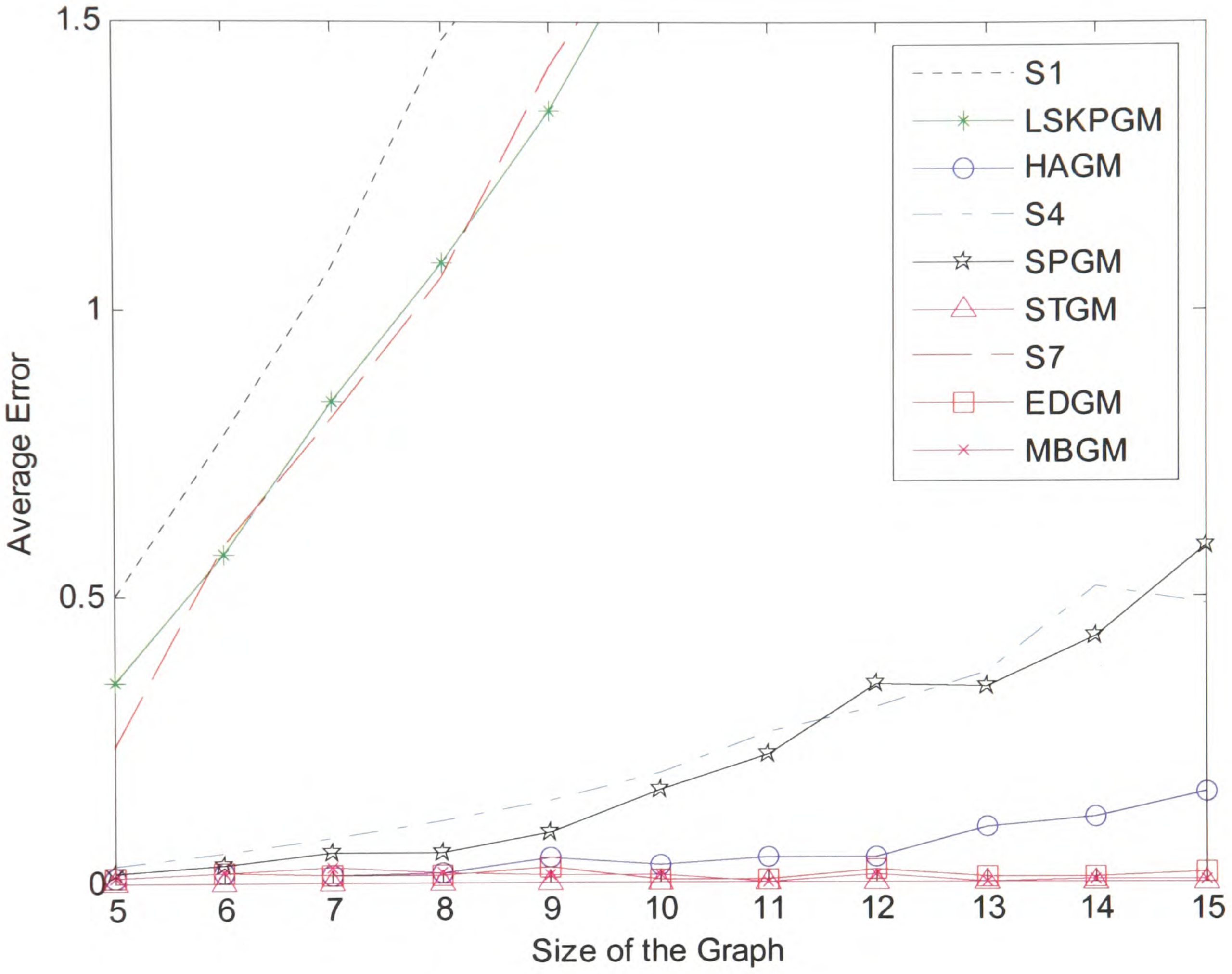


Figure 5.19 Nine algorithms for matching dense graph pairs with perturbation coefficient ε=0.10

Test 3: matching non-isomorphic sparse graph pairs

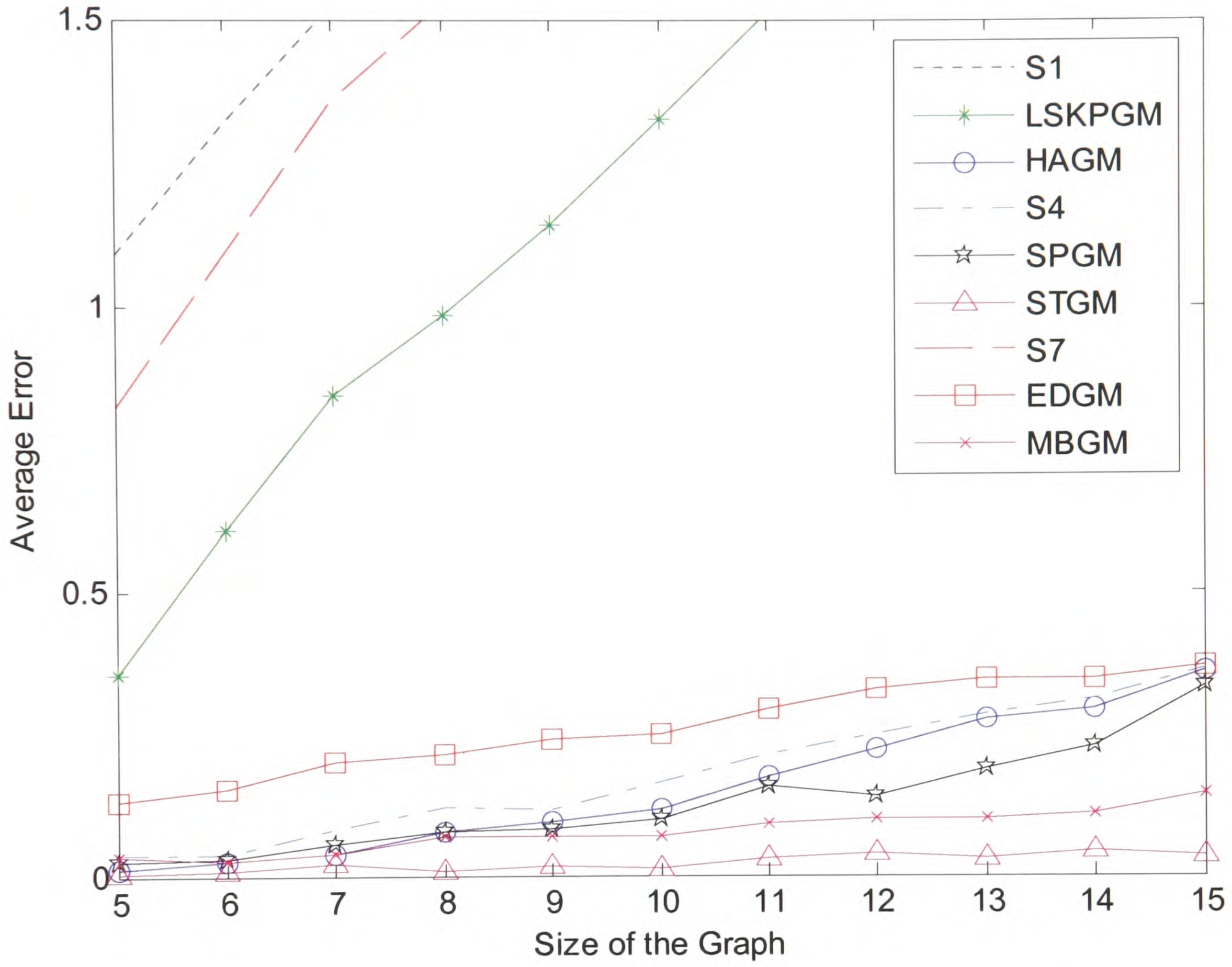


Figure 5.20 Nine algorithms for matching sparse graph pairs with perturbation coefficient ε=0.10

Test 4: CPU time

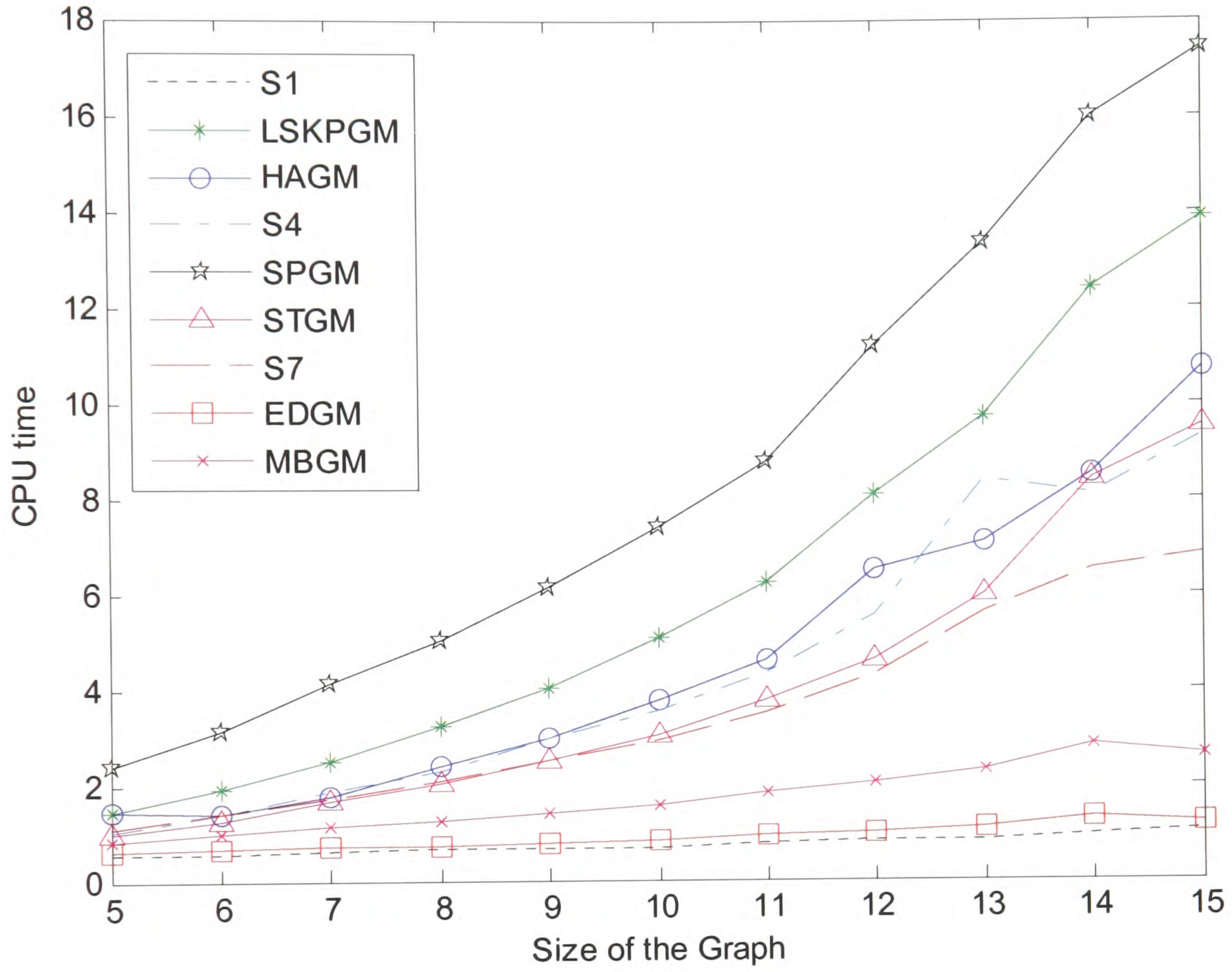


Figure 5.21 CPU time Consuming of nine algorithms

Conclusion

In all the tests above, the STGM algorithm works best. For matching isomorphic graph pairs (either dense or sparse) and non-isomorphic dense graph pairs, the STGM algorithm gets almost zero error. For matching non-isomorphic sparse graph pairs, the STGM still keeps the error very low.

Section 5.4.5 Computational Complexity

All the node similarity based algorithms have two main steps, the calculation of node-similarity and the calculation of maximum similarity match. The computational of the NSGM algorithm based on S_1 to S_9 is list as table 5.4

Table 5.4 Computational complexity of NSGM algorithms

	S1	LKPGM	HAGM	S4	SPGM	STGM	S7	MBGM	EDGM
Step1	$O(n^2)$	$O(n^4)$	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O(n^2 \log n)$	$O(n^3)$	$O(n^4)$	$O(n^3)$
Step2	$O(n^3)$								
In all	$O(n^3)$	$O(n^4)$	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O(n^4)$	$O(n^3)$

Obviously, all these algorithm works efficiently with complexity $O(n^3)$ or $O(n^4)$.

Section 5.4.6 Extensions

Weighted average node similarity

In last section, nine different node similarity functions are constructed, where each of them expresses certain kind of node similarity of graphs. And these similarities can be combined into some more complicated similarity functions.

Theorem 5.13: Given m similarity functions $S_1, \ldots, S_m$, the weighted average function

$$S=\sum_{i=1}^{m} w_i S_i \tag{5.21}$$

Then S is also a node similarity function.

Based on this weighted average node similarity function, new matching algorithm can be developed.

Matching attributed graphs

In above discussion, only matching weighted graphs is considered. However, the node similarity based graph matching algorithm can be easily extended to match attribute graphs.

Given two attributed graphs $G=\{G_k\}$, $H=\{H_k\}$, $1\le k\le m$, and Let S be a node similarity function, then the node similarity of G and H can be defined by:

$$S(G, H)=\sum_{k=1}^{m} w_k S(G_k, H_k) \qquad (5.22)$$

Where w_k is the importance coefficient of k-th attribute

Again, we transfer the attribute graph matching problem (5.1) to maximum linear assignment problem (5.16).

Section 5.4.7 Limitation

In this section, one of the important conclusions of this thesis, which is that all the node similarity graph matching algorithms fail to work for circles, will be proposed.

Theorem 5.14: All node similarity graph matching algorithms fail to work for circles.

Proof:

Let G and H be two circles as show in figure 5.22, then the adjacency matrices are

$$G=\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \ H=\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Let S be any node similarity function, then S satisfies Independence of Order I, so

$$S(PGP^T, H) = P \times S(G, H), \text{ for all } P \in \mathrm{Perm}(n) \qquad (5.23)$$

Specially, Let

$$P_0 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Then it can be easily verified that $P_0GP_0^T = G$

So $S(G,H) = S(P_0GP_0^T, H) = P_0 \times S(G,H)$

Which means all rows of S(G, H) are equal

$$S(G,H) = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,1} & a_{1,2} & \cdots & a_{1,n} \end{bmatrix}$$

Then any permutation $P \in \text{Perm}(n)$ is the solution of maximum linear assignment problem:

$$\underset{P \in \text{Perm}(n)}{\text{args max}} \sum_{i,j=1}^{n} S(i,j) \times P(i,j)$$

Specially the identity matrix I is a optimal solution for the above formula, where

$$\left\|G - IHI^T\right\|_F = \left\|G - H\right\|_F = 2$$

But in fact graph G and H are isomorphic by the permutation

$$\overline{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

So the node similarity based graph matching algorithm fails to find the best match.

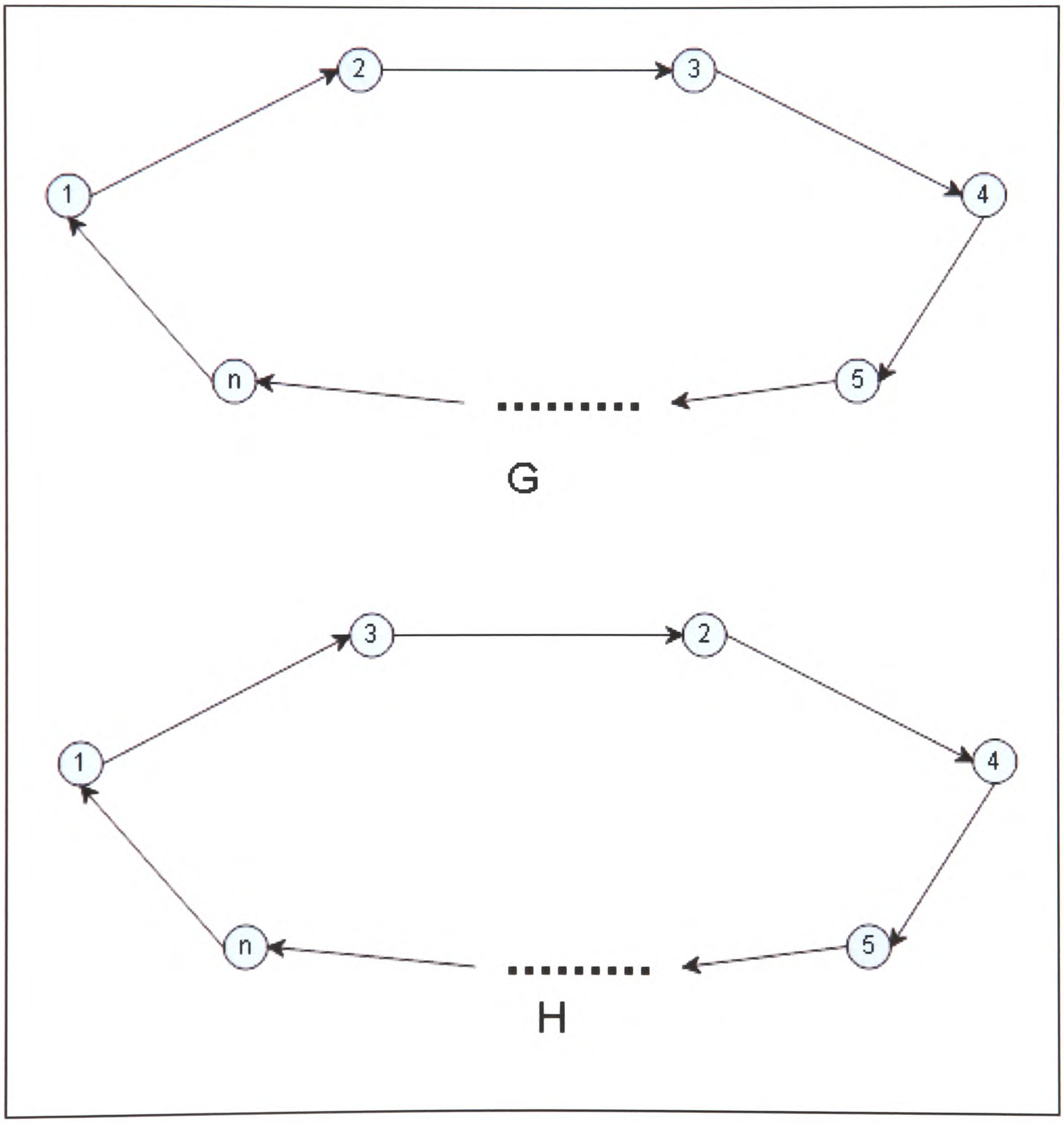


Figure 5.22 Two Circles

Theorem 5.14 only claims that all the node similarity based algorithms are not applicable for circles, how about others?

In fact, the essence of this failure is that if graph G is self-similar, which means there is a non-trivial automorphism of graph G, the node similarity based graph matching algorithm will fail to distinguish those similar nodes of G. Figure 5.23 is a simple example which has non-trivial automorphism, such that nodes 1, 2 and 3 are not distinguishable and nodes 4, 5 and 6 are not distinguishable.

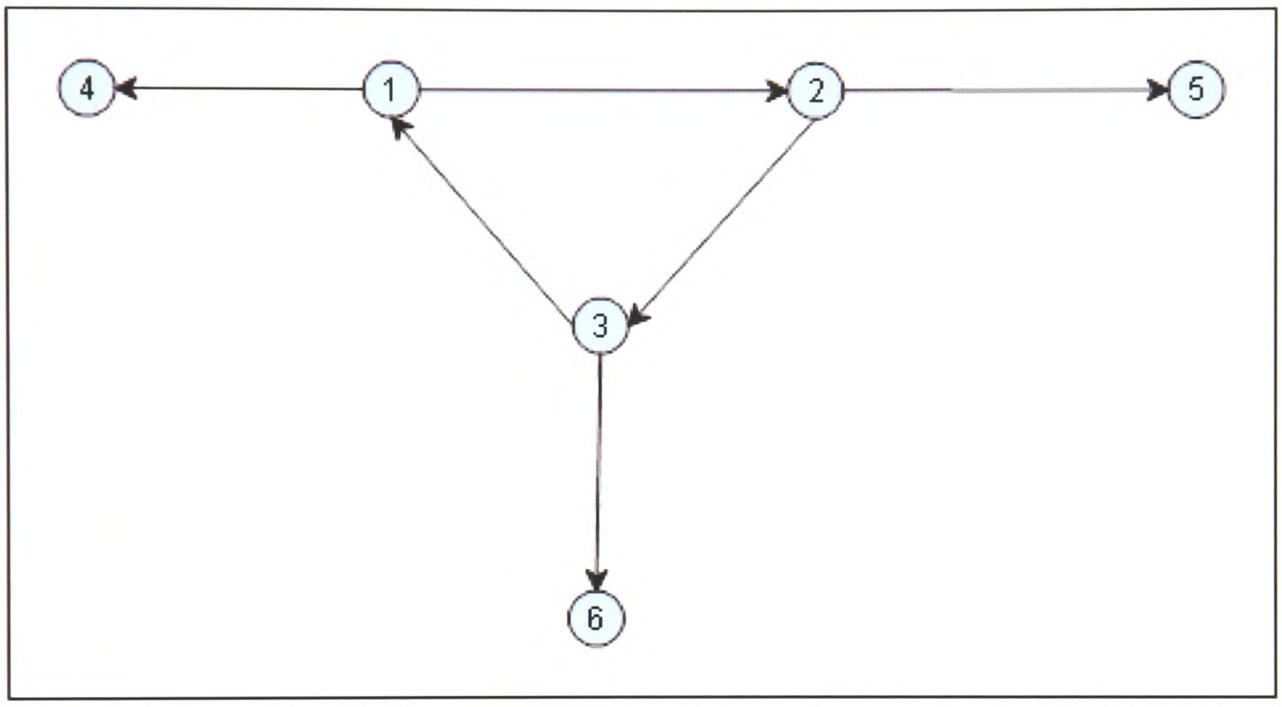


Figure 5.23 Self-similar graph

Chapter 6 Applying Matching Algorithms for Scenarios

The graph matching algorithms compared and improved in last chapter will be applied for matching scenario graphs in this chapter.

Section 6.1 Introduction of Experiments

In chapter 4, scenario is defined as a 5-tuple <T, P, HOLDS, MEETS, DUR> and graphically represented by an attribute graph. The similarity of scenario graphs is defined as by formula 4.1.

$$\mathrm{sim}(s_1, s_2) = 1 - \frac{\min\limits_{Q \in \mathrm{perm}(m)} \sum\limits_{k=1}^{n+1} \left\| N_{s1,k} - Q N_{s2,k} Q^T \right\|_F}{\sum\limits_{k=1}^{n+1} \left(\left\| N_{s1,k} \right\|_1 + \left\| N_{s2,k} \right\|_F \right)}$$

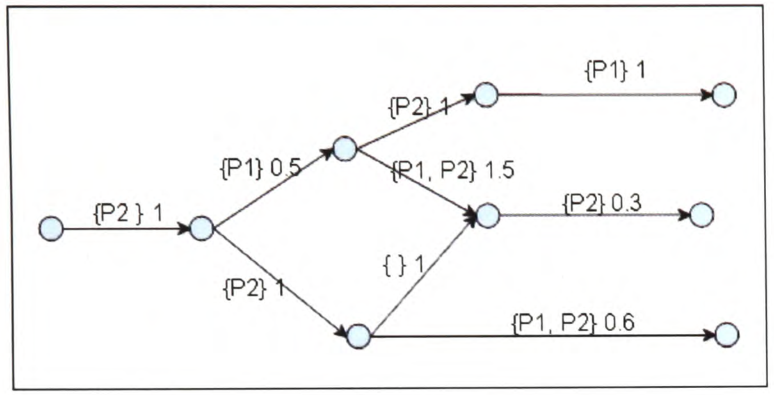
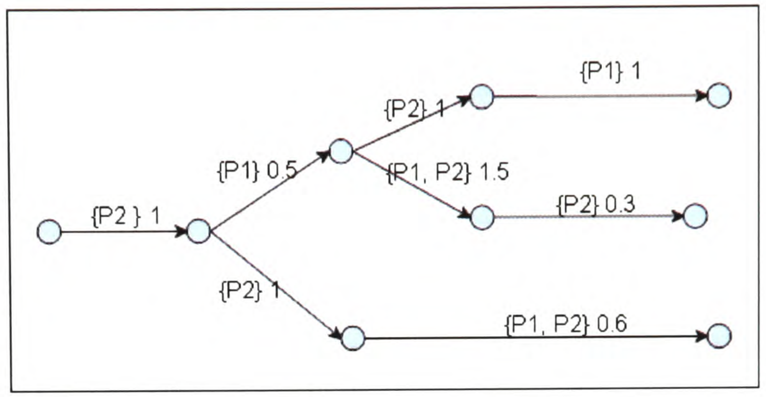
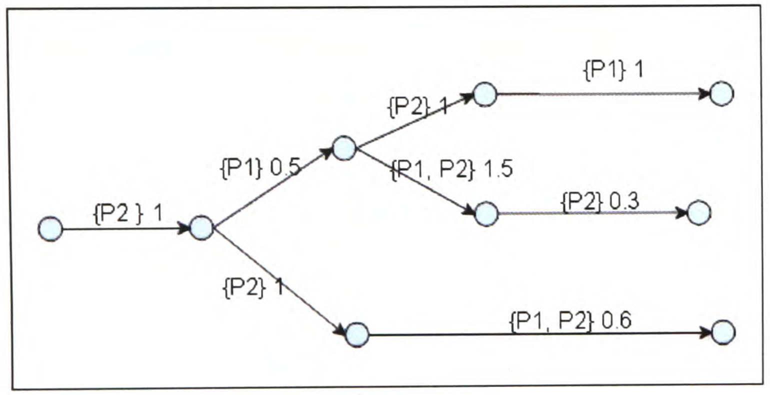
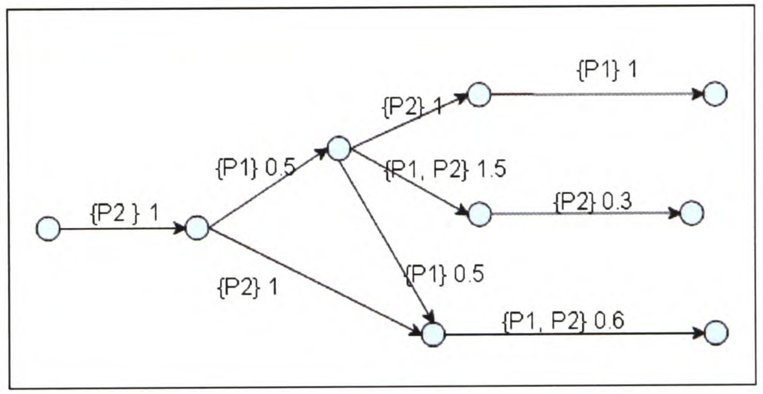
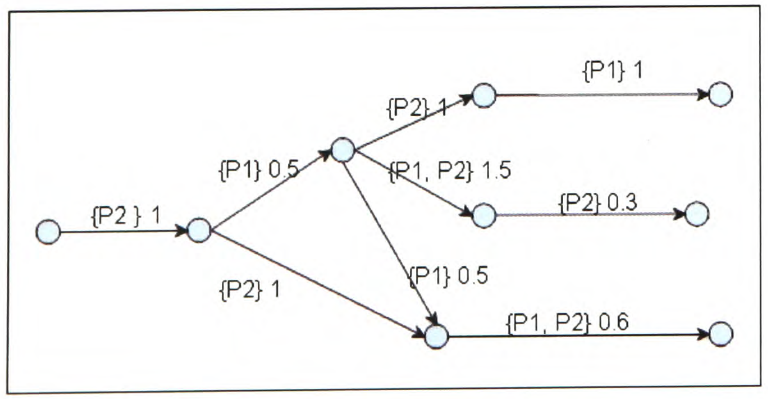
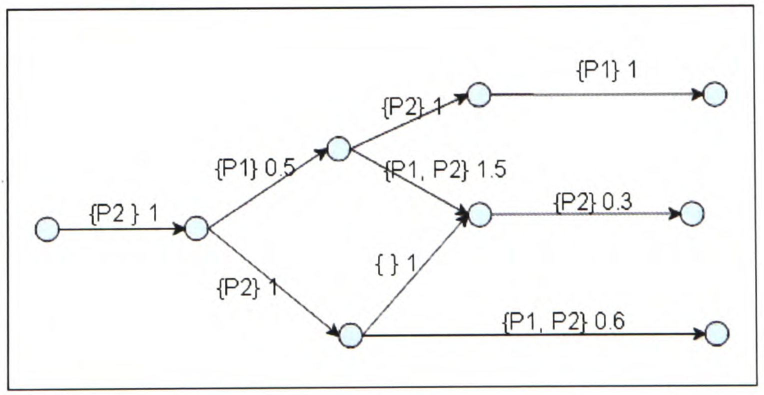
To evaluate the graph matching algorithms for matching scenario graphs, we shall randomly generate 30 pairs of scenario graphs (isomorphic or non-isomorphic, dense or sparse). The meta-basis based graph matching algorithm MBGM, the sort based graph matching algorithm STGM, and the hubs and authorities graph matching algorithm HAGM are applied on these scenario graphs to calculate their similarities.

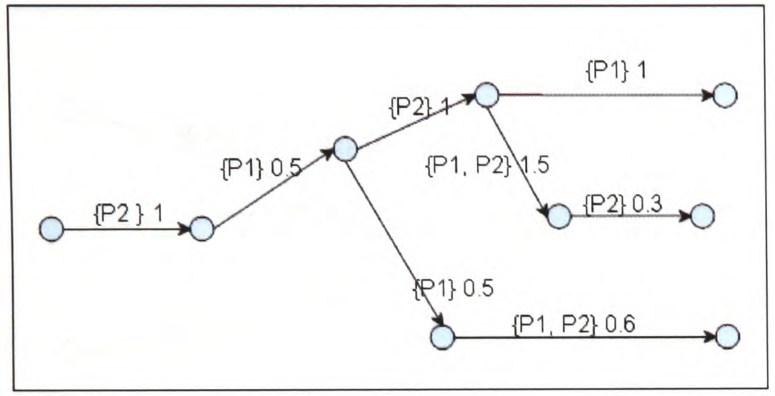
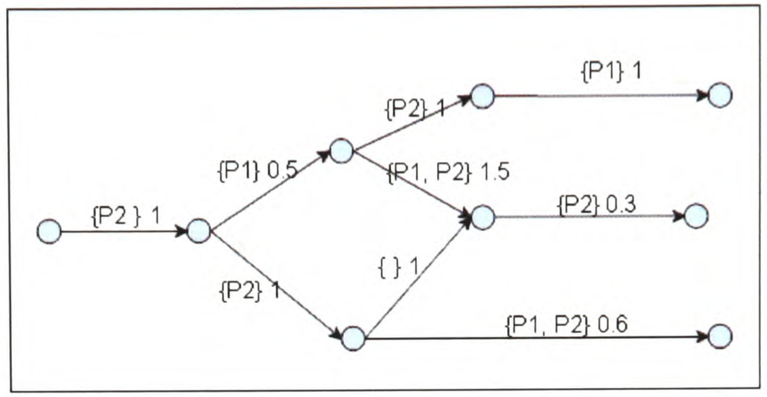
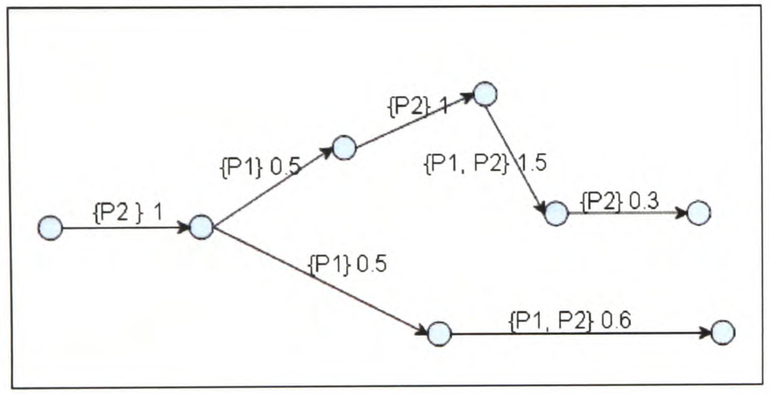
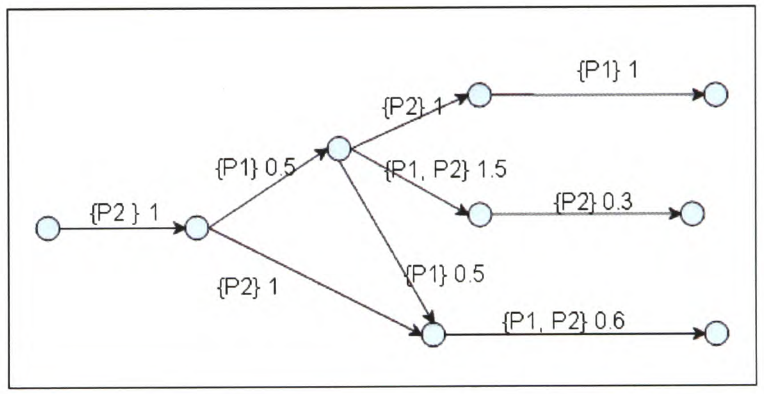
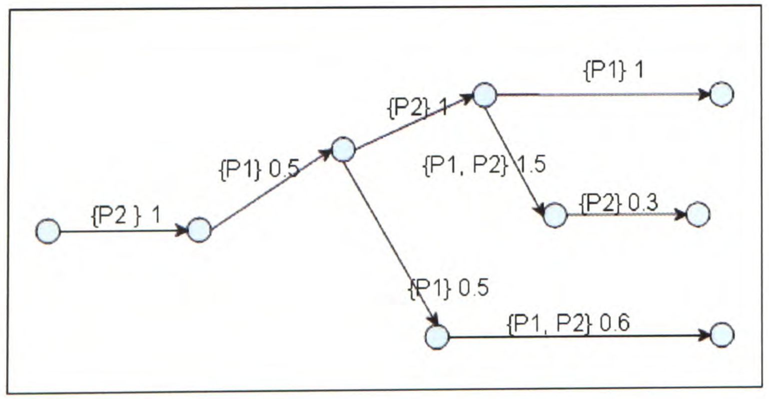
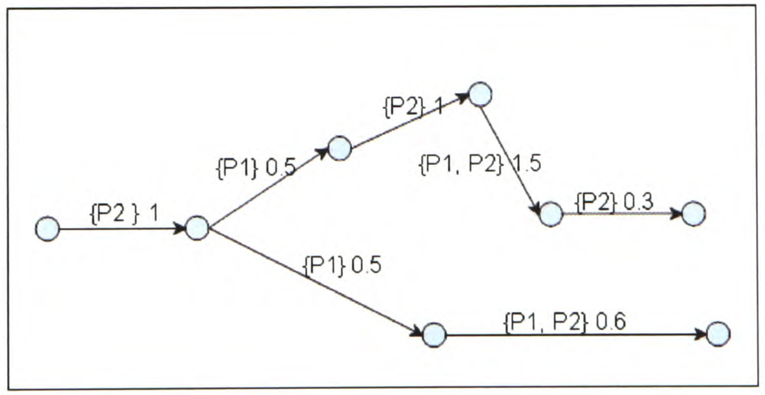
We also use a brutal search method (referred as APGM) to find best matching for each graph pairs as a standard to evaluate other algorithm. However, the brutal search method are not applicable for large size graphs (n>10).

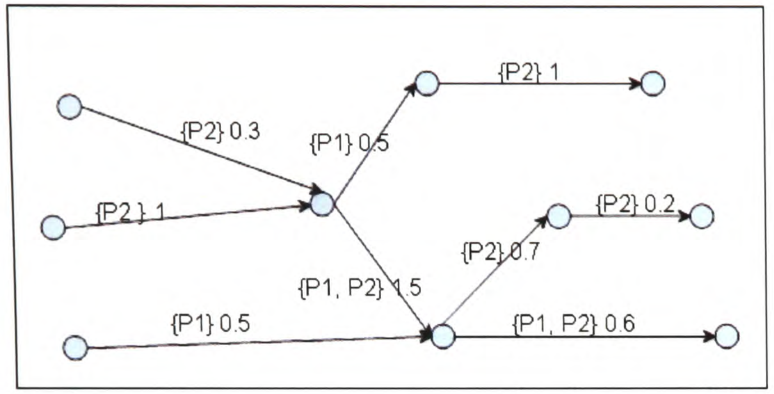
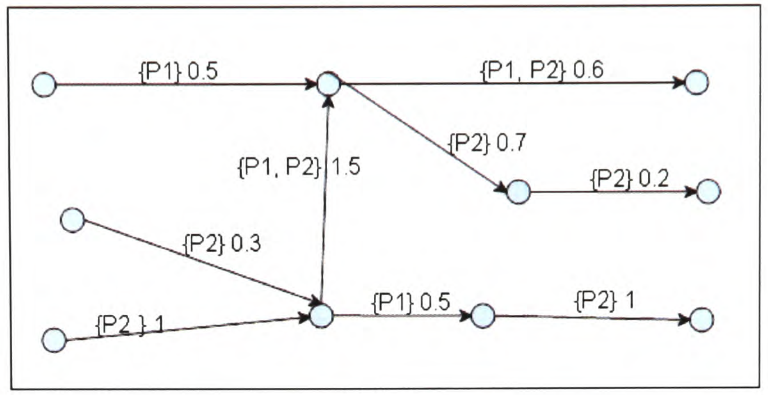
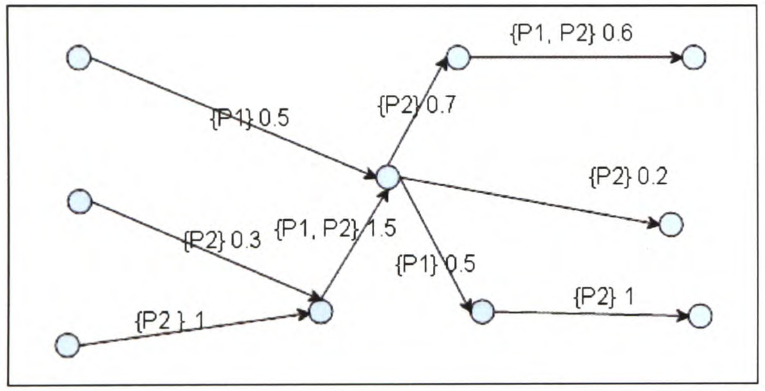
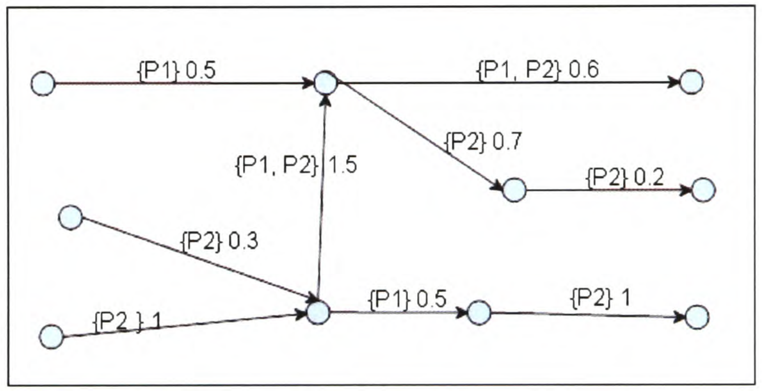
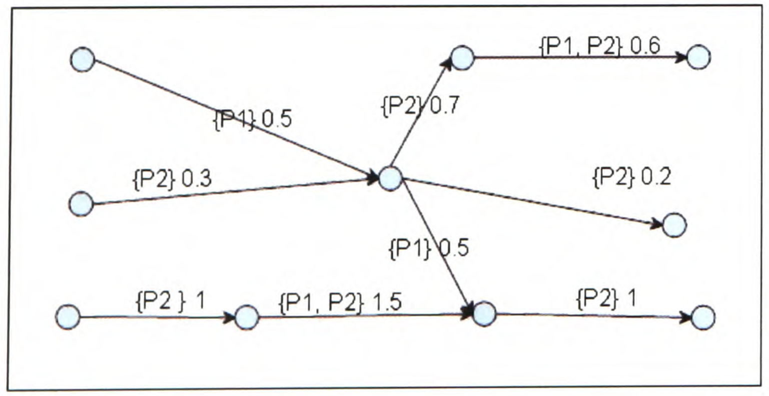
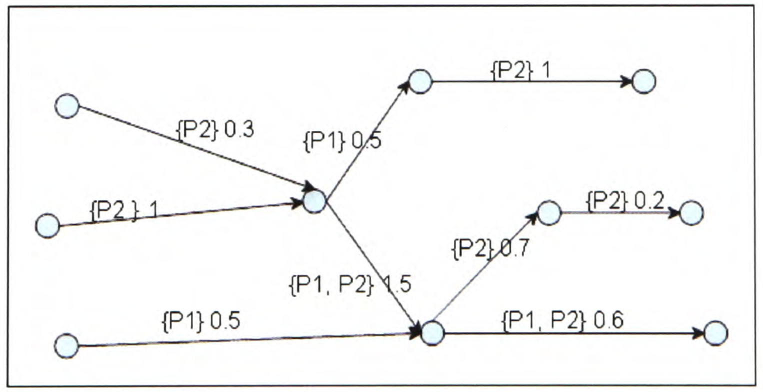
We assume the P={P_1, P_2}, which means only two propositions are considered, and w_1=w_2=w_3 which means that P_1, P_2 and duration are equivalent important to determine the scenario similarity in our case.

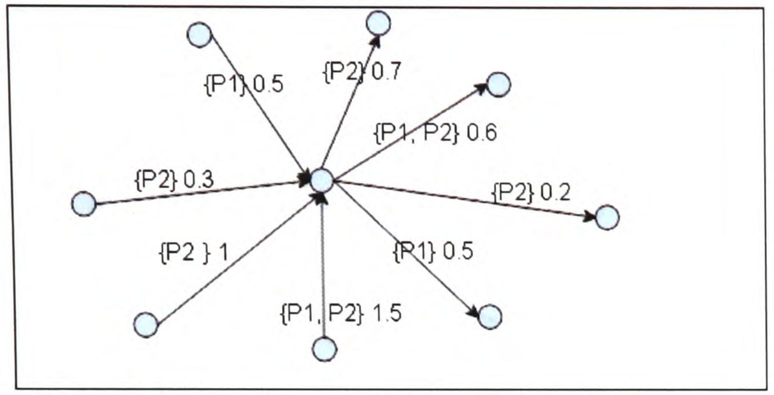
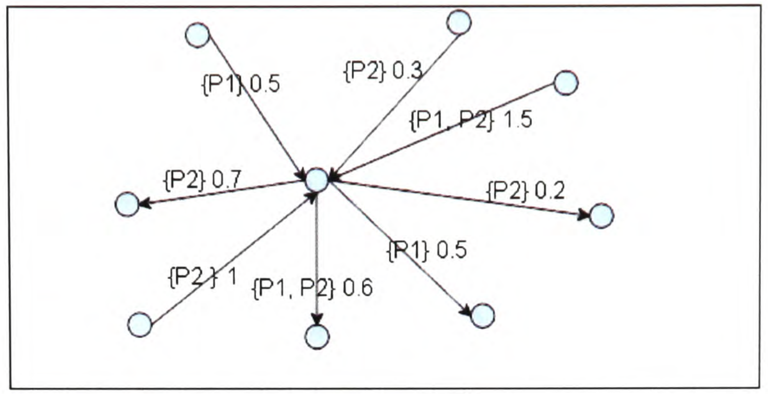
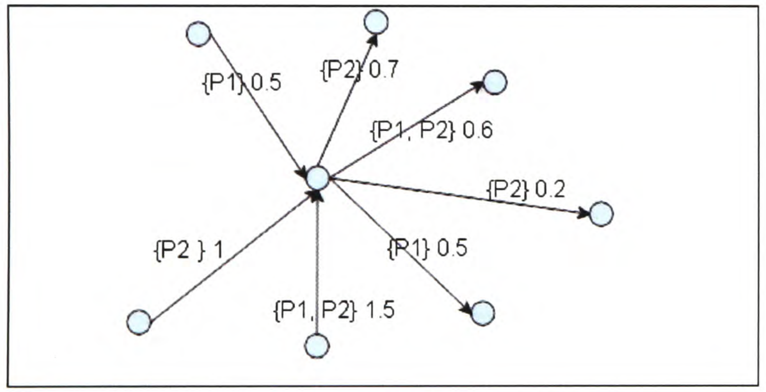
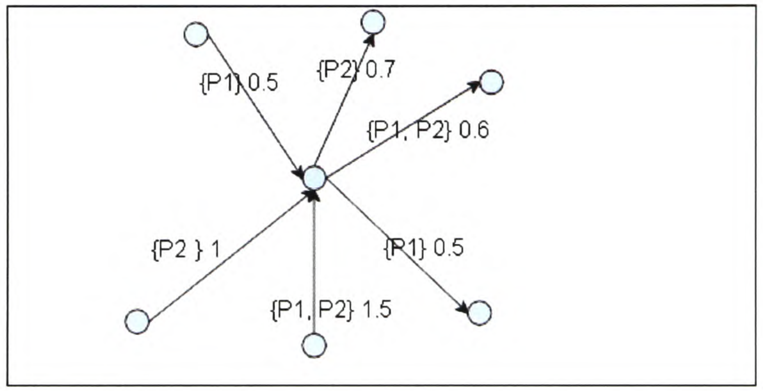
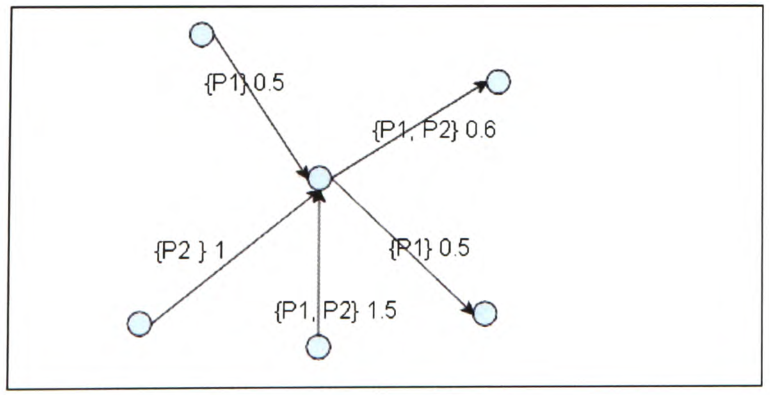
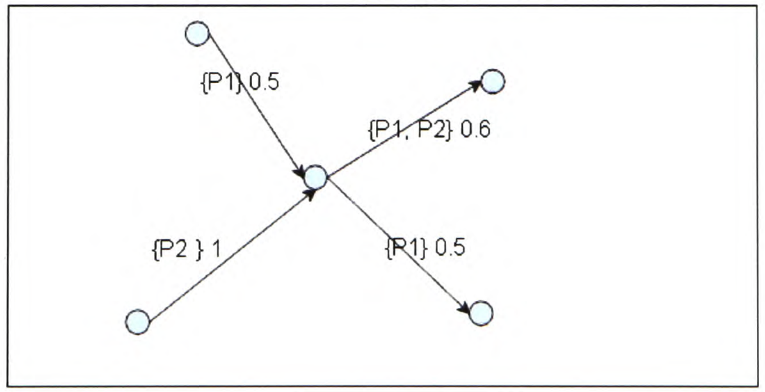
Section 6.2 Experiments

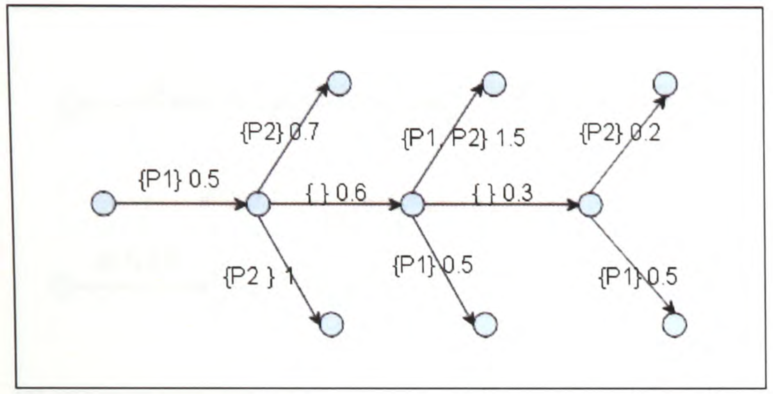
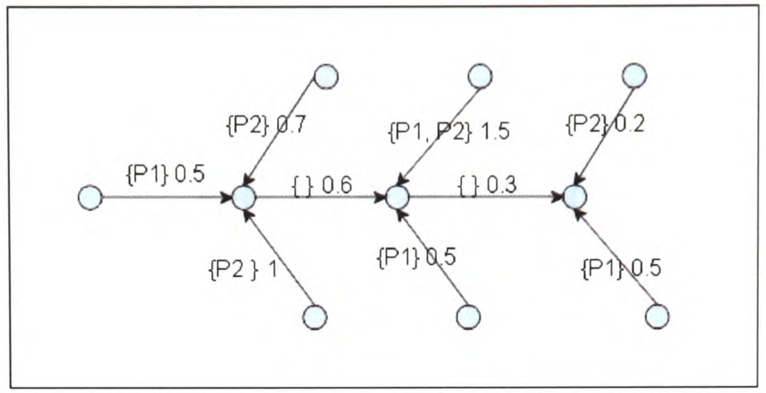
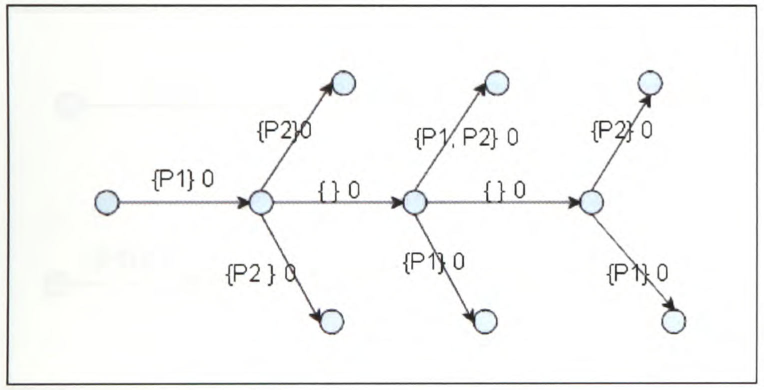
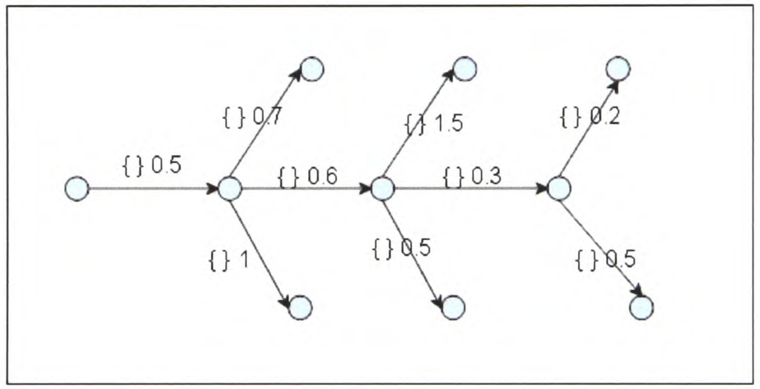
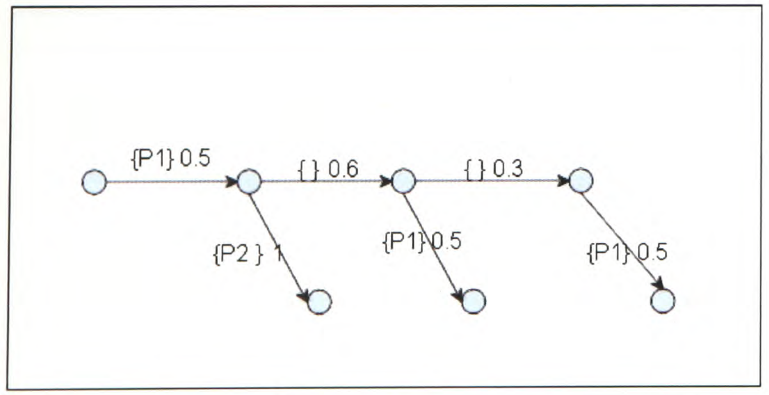
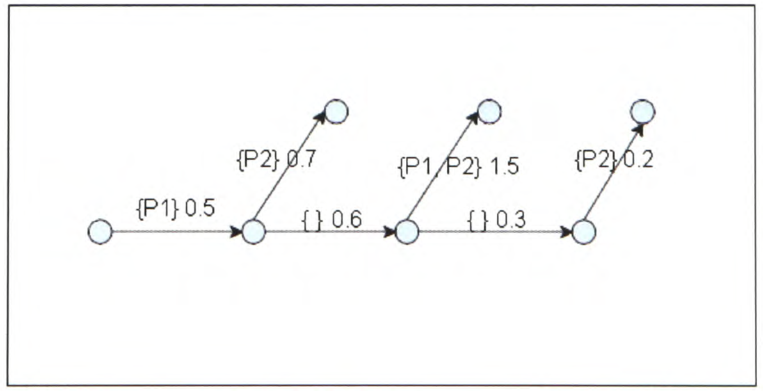
The test results are shown as following tables, where the first and second columns are two scenario graphs and the 3th-6th columns are similarities calculated by HAGM, MBGM, STGM and APGM algorithms respectively.

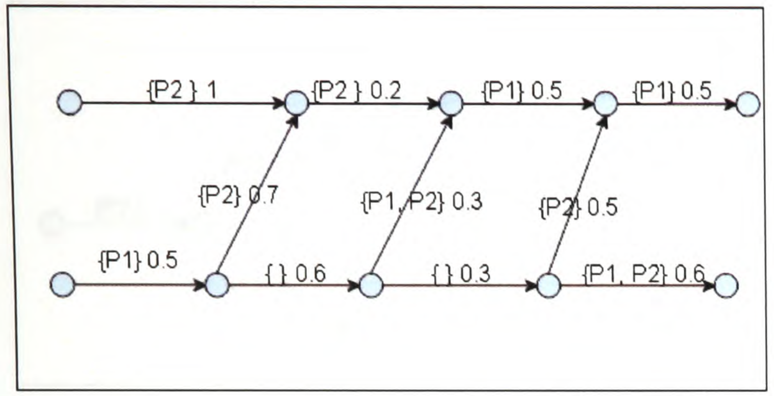
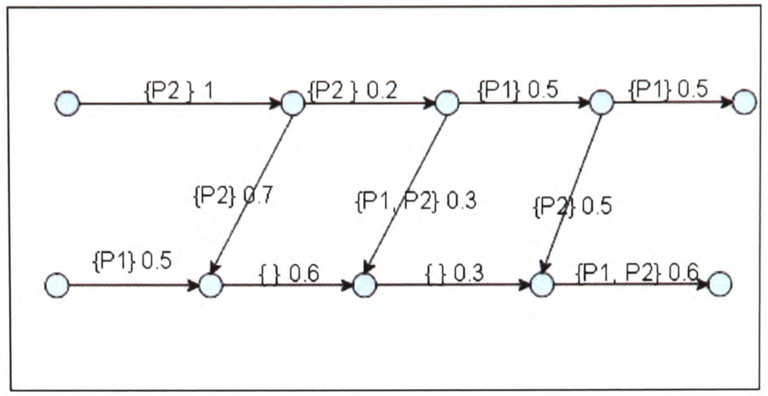
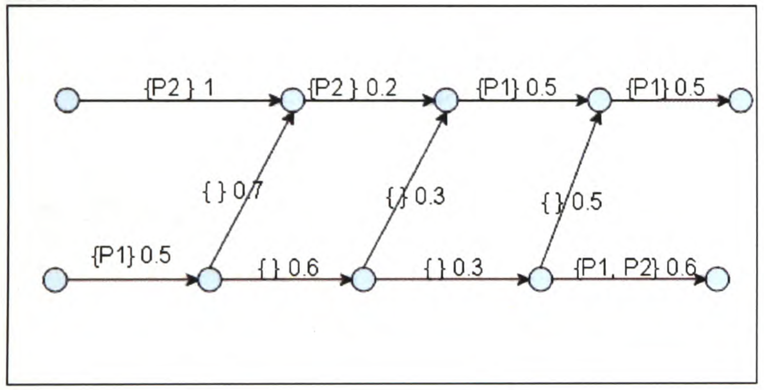
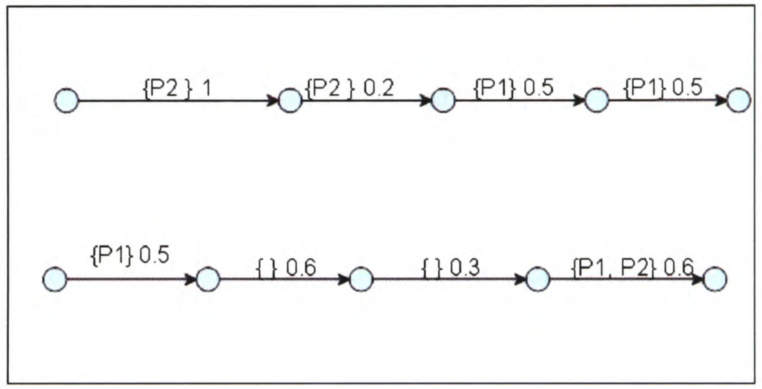
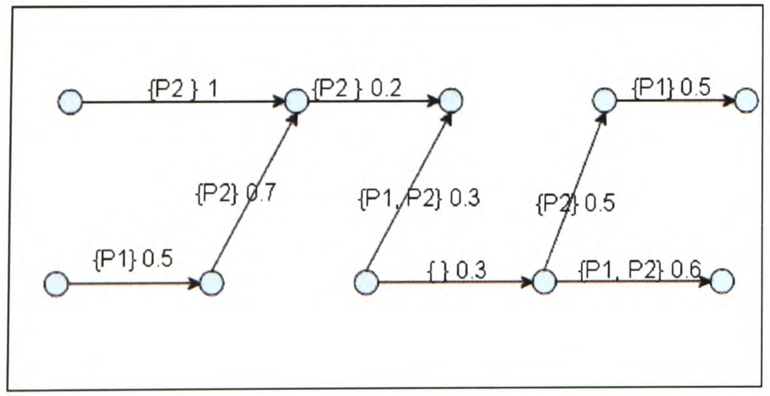
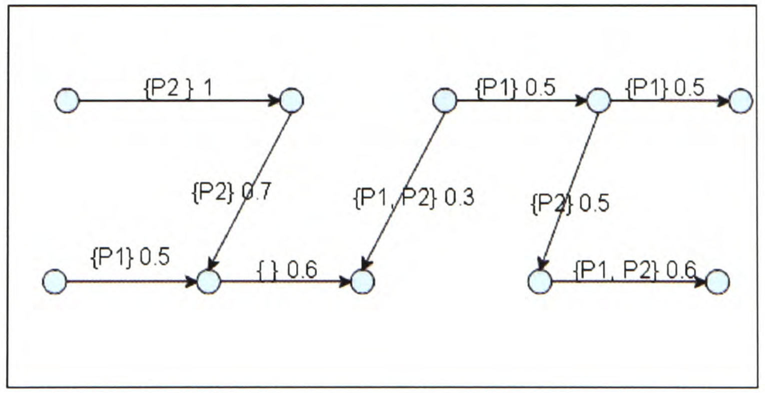
S$_1$	S$_2$	HAGM	EDGM	STGM	APGM
 {P2 } 1 {P1} 0.5 {P2} 1 {P2} 1 {P1, P2} 1.5 { } 1 {P1} 1 {P2} 0.3 {P1, P2} 0.6	 {P2 } 1 {P1} 0.5 {P2} 1 {P2} 1 {P1, P2} 1.5 {P1} 1 {P2} 0.3 {P1, P2} 0.6	0.9247	0.9247	0.7133	0.9247
 {P2 } 1 {P1} 0.5 {P2} 1 {P2} 1 {P1, P2} 1.5 {P1} 1 {P2} 0.3 {P1, P2} 0.6	 {P2 } 1 {P1} 0.5 {P2} 1 {P2} 1 {P1, P2} 1.5 {P1} 0.5 {P1} 1 {P2} 0.3 {P1, P2} 0.6	0.9153	0.9153	0.9153	0.9153
 {P2 } 1 {P1} 0.5 {P2} 1 {P2} 1 {P1, P2} 1.5 {P1} 0.5 {P1} 1 {P2} 0.3 {P1, P2} 0.6	 {P2 } 1 {P1} 0.5 {P2} 1 {P2} 1 {P1, P2} 1.5 { } 1 {P1} 1 {P2} 0.3 {P1, P2} 0.6	0.6328	0.8840	0.6853	0.8840

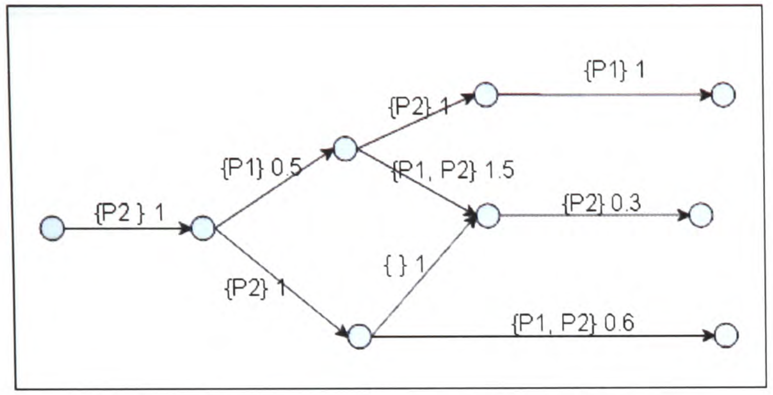
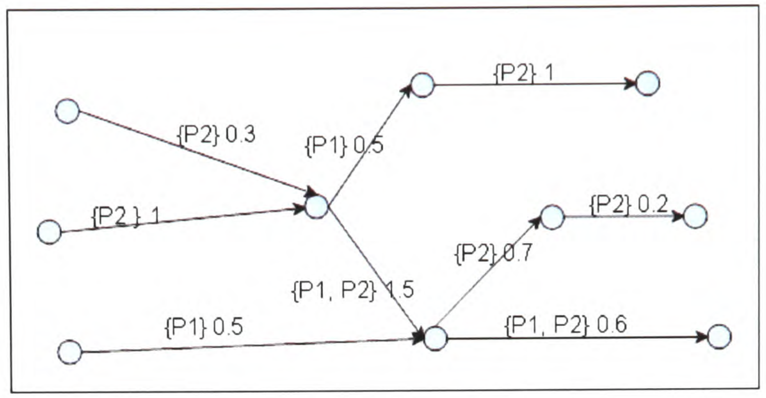
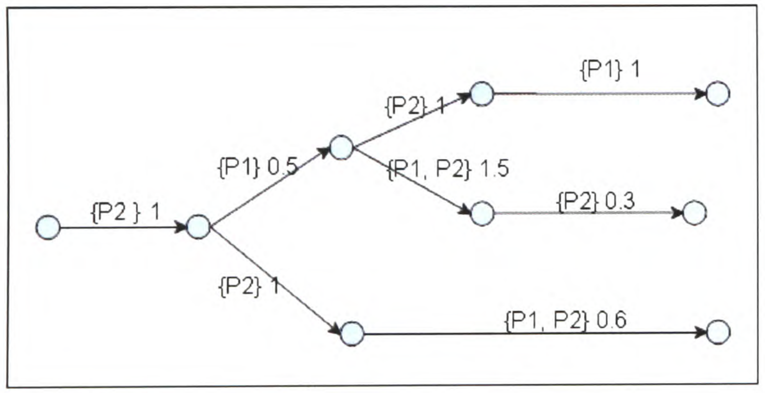
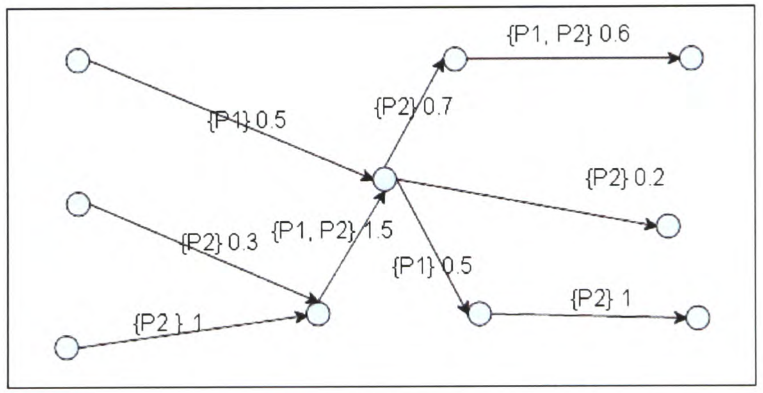
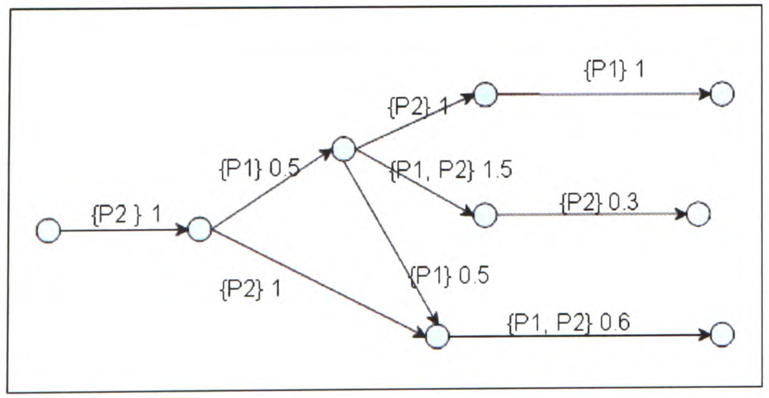
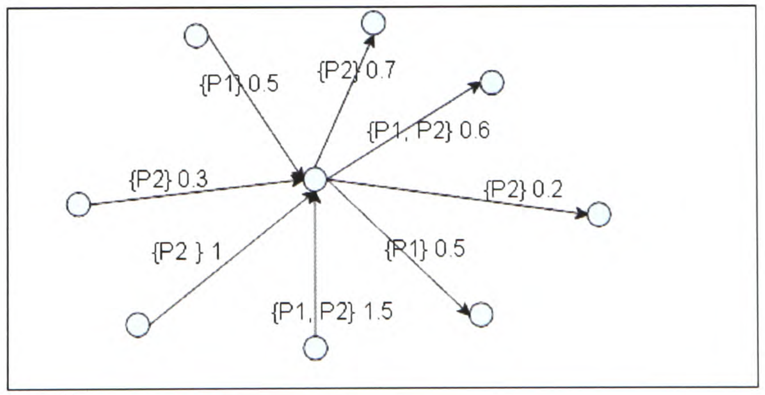
S_1	S_2	HAGM	EDGM	STGM	APGM
 {P2 } 1 {P1} 0.5 {P2} 1 {P1} 1 {P1, P2} 1.5 {P2} 0.3 {P1} 0.5 {P1, P2} 0.6	 {P2 } 1 {P1} 0.5 {P2} 1 {P1} 1 {P1, P2} 1.5 {P2} 0.3 { } 1 {P2} 1 {P1, P2} 0.6	0.6338	0.7025	0.6393	0.8135
 {P2 } 1 {P1} 0.5 {P2} 1 {P1, P2} 1.5 {P2} 0.3 {P1} 0.5 {P1, P2} 0.6	 {P2 } 1 {P1} 0.5 {P2} 1 {P1} 1 {P1, P2} 1.5 {P2} 0.3 {P1} 0.5 {P2} 1 {P1, P2} 0.6	0.5841	0.5791	0.6007	0.7268
 {P2 } 1 {P1} 0.5 {P2} 1 {P1} 1 {P1, P2} 1.5 {P2} 0.3 {P1} 0.5 {P1, P2} 0.6	 {P2 } 1 {P1} 0.5 {P2} 1 {P1, P2} 1.5 {P2} 0.3 {P1} 0.5 {P1, P2} 0.6	0.6093	0.6793	0.5925	0.7307

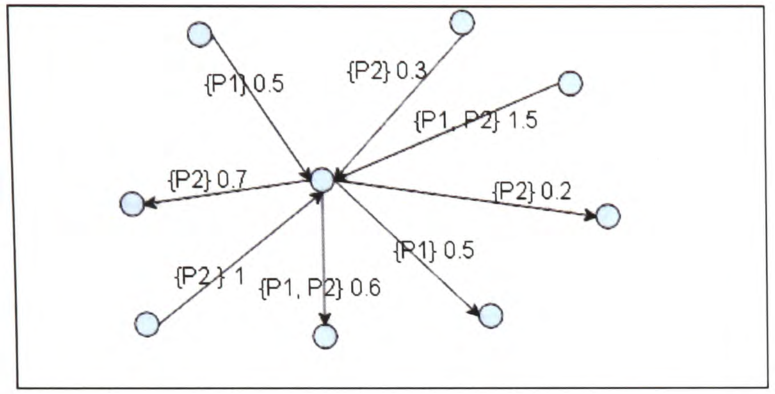
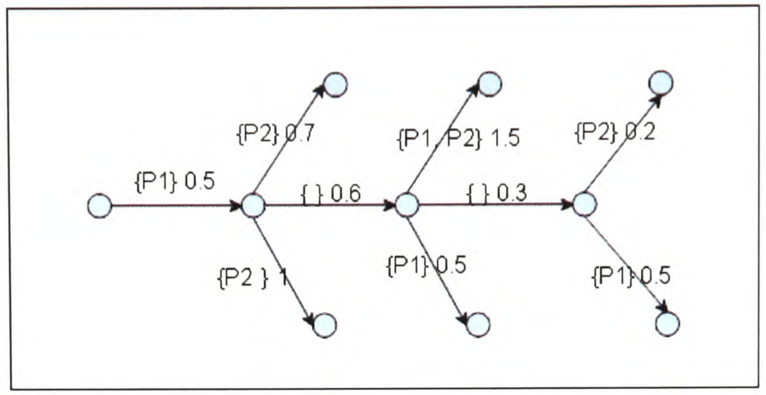
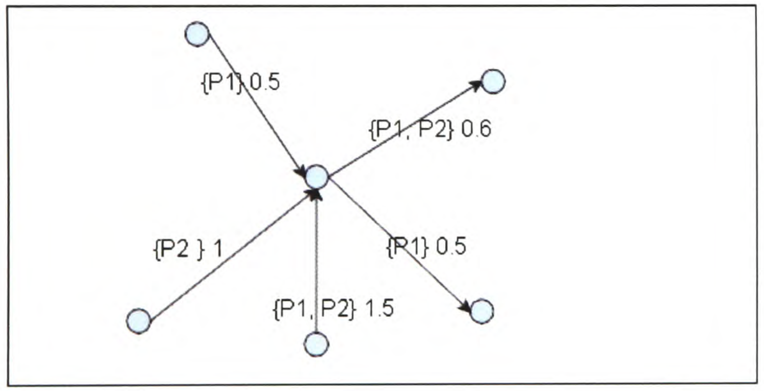
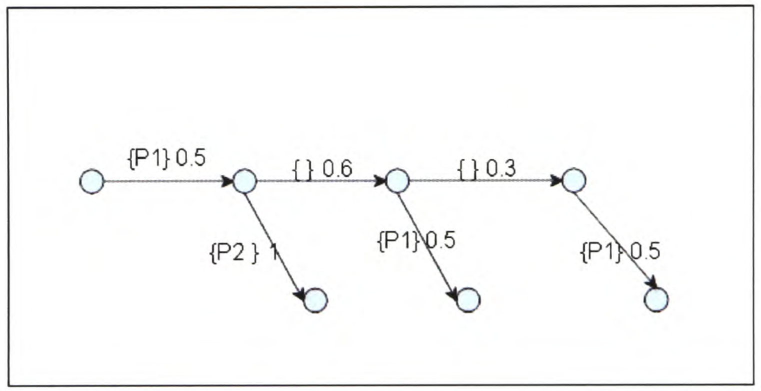
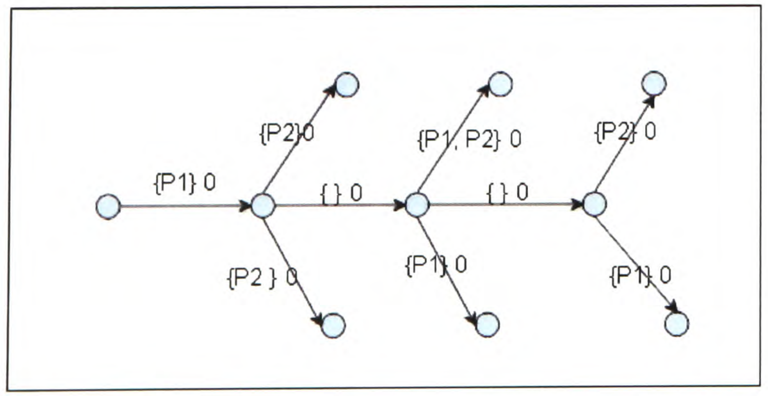
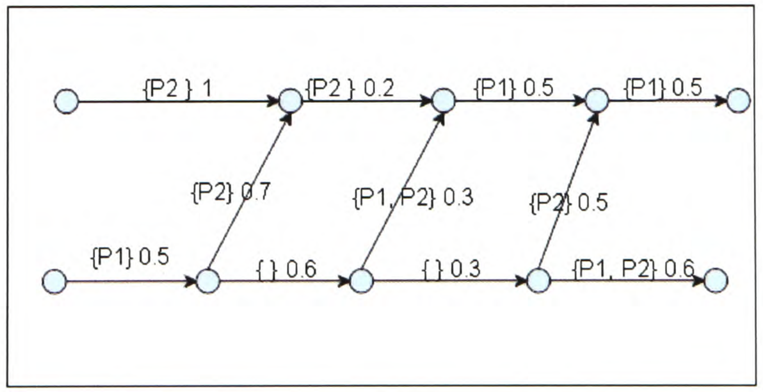
S$_1$	S$_2$	HAGM	EDGM	STGM	APGM
		0.6523	1.0000	1.0000	1.0000
		0.7972	0.8921	0.8921	0.8921
		0.6792	0.8155	0.6650	0.8220

S_1	S_2	HAGM	EDGM	STGM	APGM
 {P1} 0.5 {P2} 0.7 {P1, P2} 0.6 {P2} 0.3 {P2} 0.2 {P2 } 1 {P1} 0.5 {P1, P2} 1.5	 {P1} 0.5 {P2} 0.3 {P1, P2} 1.5 {P2} 0.7 {P2} 0.2 {P1} 0.5 {P2 } 1 {P1, P2} 0.6	0.6431	1.0000	1.0000	1.0000
 {P1} 0.5 {P2} 0.7 {P1, P2} 0.6 {P2} 0.2 {P2 } 1 {P1} 0.5 {P1, P2} 1.5	 {P1} 0.5 {P2} 0.7 {P1, P2} 0.6 {P2 } 1 {P1} 0.5 {P1, P2} 1.5	0.6284	0.7530	0.7530	0.7530
 {P1} 0.5 {P1, P2} 0.6 {P2 } 1 {P1} 0.5 {P1, P2} 1.5		0.5864	0.7065	0.7065	0.7065

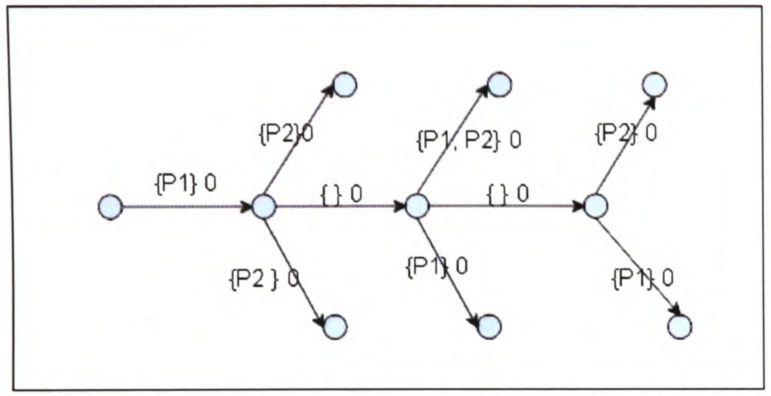
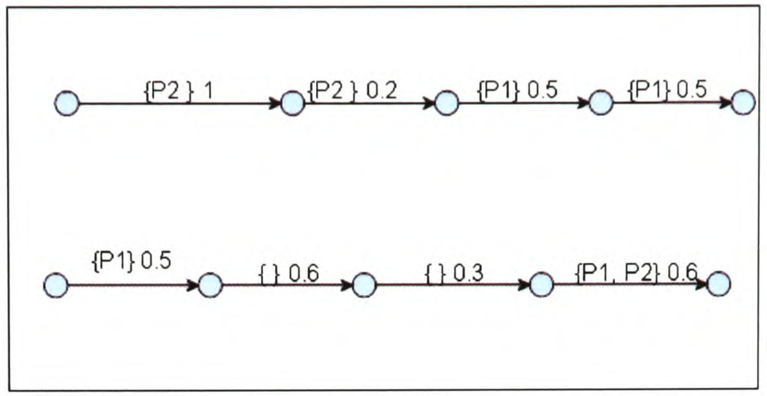
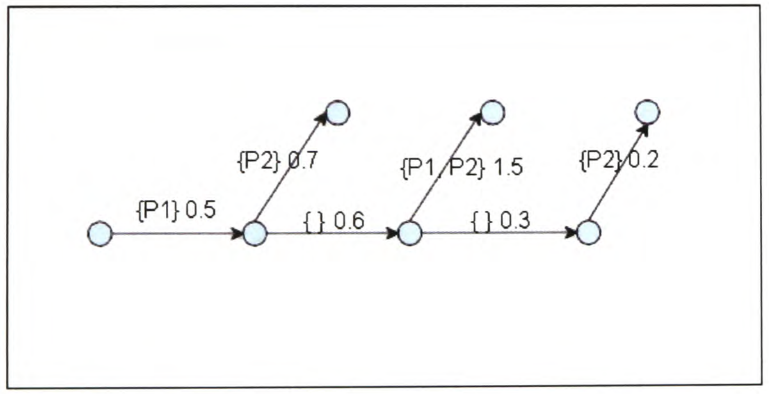
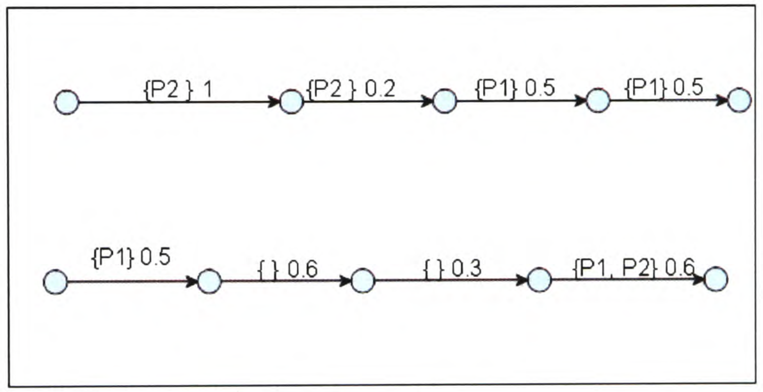
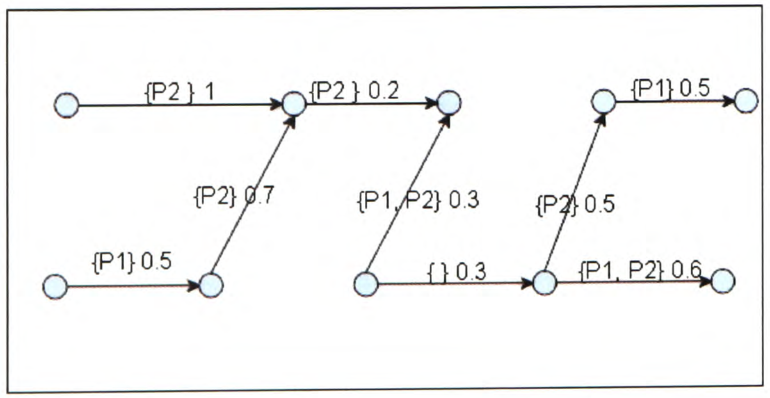
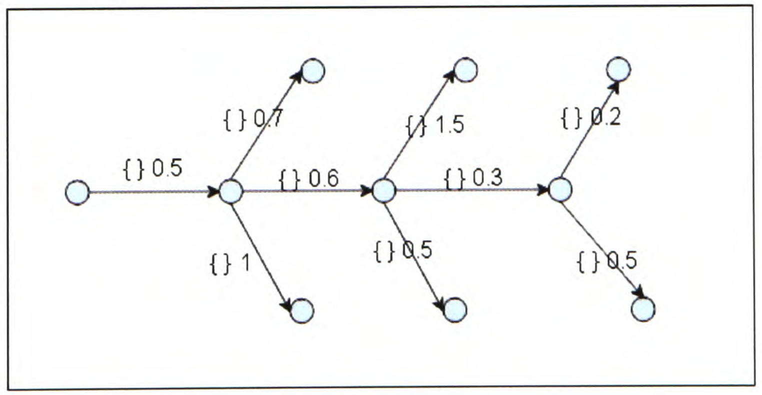
S$_1$	S$_2$	HAGM	EDGM	STGM	APGM
 {P1} 0.5 {P2} 0.7 {P2 } 1 { } 0.6 {P1, P2} 1.5 {P1} 0.5 { } 0.3 {P2} 0.2 {P1} 0.5	 {P1} 0.5 {P2} 0.7 {P2 } 1 { } 0.6 {P1, P2} 1.5 {P1} 0.5 { } 0.3 {P2} 0.2 {P1} 0.5	0.6718	0.7143	0.7219	0.7454
 {P1} 0 {P2} 0 {P2 } 0 { } 0 {P1, P2} 0 {P1} 0 { } 0 {P2} 0 {P1} 0	 { } 0.5 { } 0.7 { } 1 { } 0.6 { } 1.5 { } 0.5 { } 0.3 { } 0.2 { } 0.5	0.8234	0.8924	0.8043	0.8924
 {P1} 0.5 {P2 } 1 { } 0.6 {P1} 0.5 { } 0.3 {P1} 0.5	 {P1} 0.5 {P2} 0.7 { } 0.6 {P1, P2} 1.5 { } 0.3 {P2} 0.2	0.7460	0.9135	0.7460	0.9135

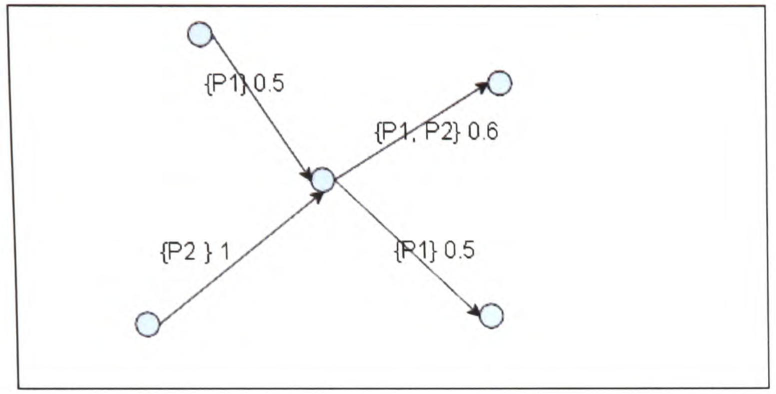
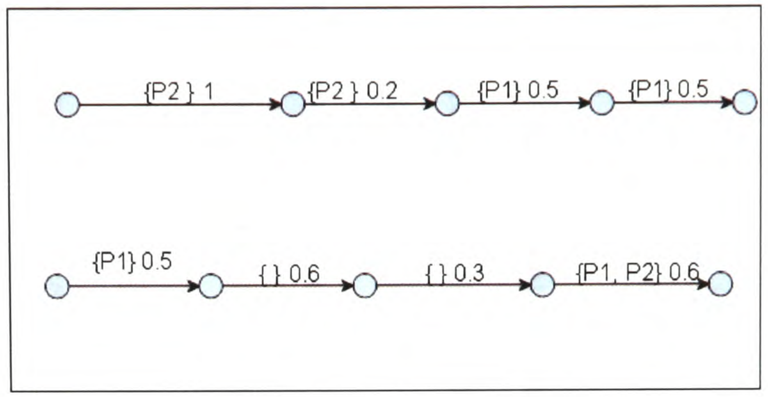
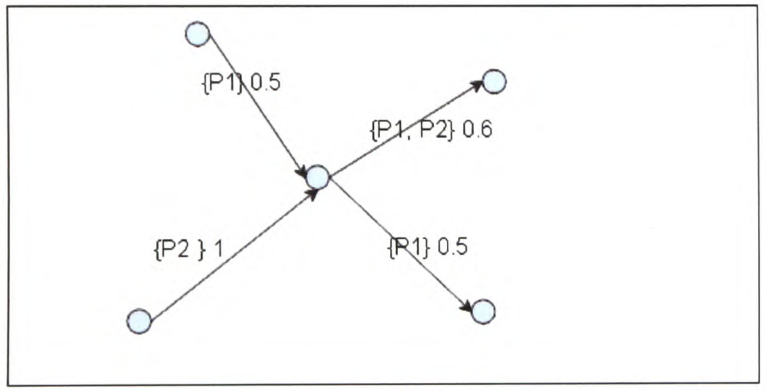
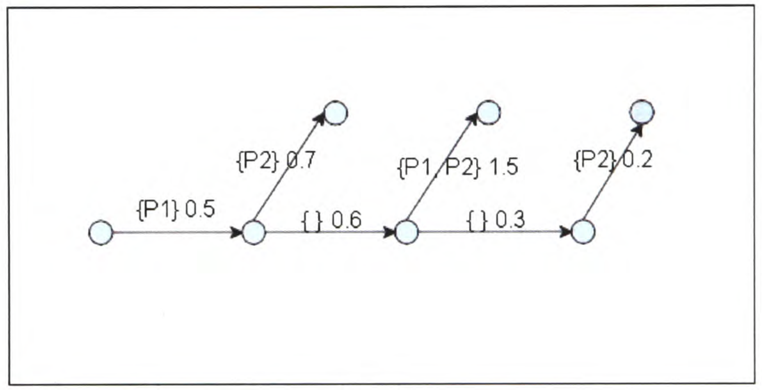
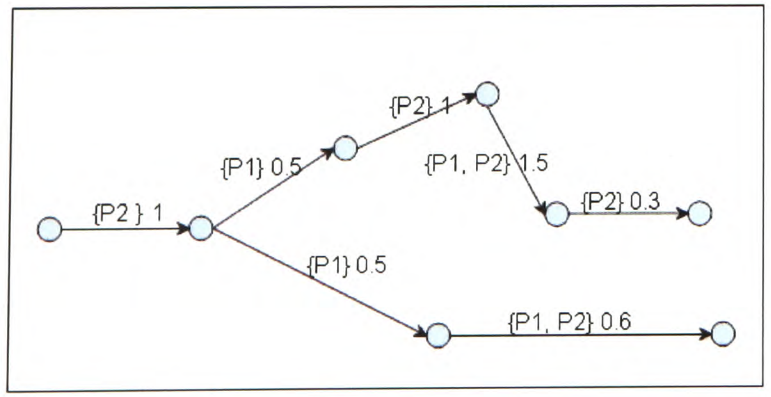
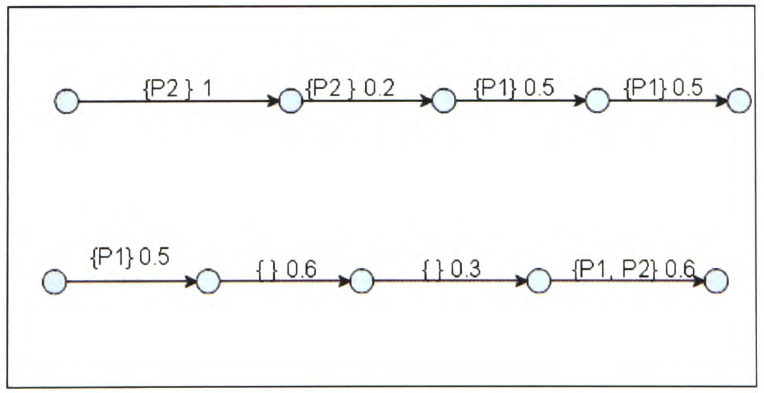
S_1	S_2	HAGM	EDGM	STGM	APGM
 {P2} 1 {P2} 0.2 {P1} 0.5 {P1} 0.5 {P2} 0.7 {P1, P2} 0.3 {P2} 0.5 {P1} 0.5 { } 0.6 { } 0.3 {P1, P2} 0.6	 {P2} 1 {P2} 0.2 {P1} 0.5 {P1} 0.5 {P2} 0.7 {P1, P2} 0.3 {P2} 0.5 {P1} 0.5 { } 0.6 { } 0.3 {P1, P2} 0.6	0.6702	0.9069	0.6846	0.9069
 {P2} 1 {P2} 0.2 {P1} 0.5 {P1} 0.5 { } 0.7 { } 0.3 { } 0.5 {P1} 0.5 { } 0.6 { } 0.3 {P1, P2} 0.6	 {P2} 1 {P2} 0.2 {P1} 0.5 {P1} 0.5 {P1} 0.5 { } 0.6 { } 0.3 {P1, P2} 0.6	0.8951	0.8951	0.8951	0.8951
 {P2} 1 {P2} 0.2 {P1} 0.5 {P2} 0.7 {P1, P2} 0.3 {P2} 0.5 {P1} 0.5 { } 0.3 {P1, P2} 0.6	 {P2} 1 {P1} 0.5 {P1} 0.5 {P2} 0.7 {P1, P2} 0.3 {P2} 0.5 {P1} 0.5 { } 0.6 {P1, P2} 0.6	0.7525	0.9178	0.7525	0.9178

S_1	S_2	HAGM	EDGM	STGM	APGM
 {P2 } 1 {P1} 0.5 {P2} 1 {P2} 1 {P1, P2} 1.5 { } 1 {P1} 1 {P2} 0.3 {P1, P2} 0.6	 {P2} 0.3 {P2 } 1 {P1} 0.5 {P1} 0.5 {P1, P2} 1.5 {P2} 1 {P2} 0.7 {P2} 0.2 {P1, P2} 0.6	0.5710	0.7233	0.5727	0.7233
 {P2 } 1 {P1} 0.5 {P2} 1 {P2} 1 {P1, P2} 1.5 {P1} 1 {P2} 0.3 {P1, P2} 0.6	 {P1} 0.5 {P2} 0.3 {P2 } 1 {P1, P2} 1.5 {P2} 0.7 {P2} 0.2 {P1} 0.5 {P1, P2} 0.6 {P2} 1	0.5784	0.6479	0.5675	0.7247
 {P2 } 1 {P1} 0.5 {P2} 1 {P2} 1 {P1, P2} 1.5 {P1} 0.5 {P1} 1 {P2} 0.3 {P1, P2} 0.6	 {P1} 0.5 {P2} 0.7 {P1, P2} 0.6 {P2} 0.3 {P2} 0.2 {P2 } 1 {P1} 0.5 {P1, P2} 1.5	0.6160	0.7120	0.6558	0.7121

S₁	S₂	HAGM	EDGM	STGM	APGM
		0.5818	0.6540	0.6181	0.6857
		0.6622	0.6015	0.6622	0.6622
		0.6890	0.8004	0.6978	0.8004

S					

S_1	S_2	HAGM	EDGM	STGM	APGM
 {P1} 0 {P2} 0 {P2 } 0 { } 0 {P1, P2} 0 {P1} 0 { } 0 {P2} 0 {P1} 0	 {P2 } 1 {P2 } 0.2 {P1} 0.5 {P1} 0.5 {P1} 0.5 { } 0.6 { } 0.3 {P1, P2} 0.6	0.7125	0.7292	0.7103	0.7971
 {P1} 0.5 {P2} 0.7 { } 0.6 {P1, P2} 1.5 { } 0.3 {P2} 0.2	 {P2 } 1 {P2 } 0.2 {P1} 0.5 {P1} 0.5 {P1} 0.5 { } 0.6 { } 0.3 {P1, P2} 0.6	0.5513	0.5555	0.5498	0.6163
 {P2 } 1 {P2 } 0.2 {P1} 0.5 {P2} 0.7 {P1, P2} 0.3 {P2} 0.5 {P1} 0.5 { } 0.3 {P1, P2} 0.6	 { } 0.5 { } 0.7 { } 1 { } 0.6 { } 1.5 { } 0.5 { } 0.3 { } 0.2 { } 0.5	0.6853	0.6999	0.7482	0.8007

S_1	S_2	HAGM	EDGM	STGM	APGM
	 {P2 } 1 {P2 } 0.2 {P1} 0.5 {P1} 0.5 {P1} 0.5 { } 0.6 { } 0.3 {P1, P2} 0.6	0.5383	0.5395	0.5304	0.5717
	 {P2} 0.7 {P1, P2} 1.5 {P2} 0.2 {P1} 0.5 { } 0.6 { } 0.3	0.4923	0.5745	0.5278	0.6049
 {P2} 1 {P1} 0.5 {P1, P2} 1.5 {P2} 0.3 {P2 } 1 {P1} 0.5 {P1, P2} 0.6	 {P2 } 1 {P2 } 0.2 {P1} 0.5 {P1} 0.5 {P1} 0.5 { } 0.6 { } 0.3 {P1, P2} 0.6	0.5793	0.7337	0.5964	0.7337

Section 6.3 Conclusion

The tests in last section show that the MBGM algorithm finds the best match in half (17 out of 30) of the cases. For the other 13 cases, the MBGM still get a satisfactory matching result with little matching error.

On the other hand, the STGM algorithm only finds the best match for 1/4 (7 out of 30) of the cases. For the other 23 cases, the STGM get large matching error.

One may ask that STGM algorithm seems work very well for the test in chapter 5, why does it fail to work in the cases of last section?

In fact, the attribute matrix of a scenario graph contains several 0-1 matrices to denote the holds relation of propositions and a real matrix to denote the duration of time elements. So in chapter 5 we only deal with the general case, where the elements of the adjacency matrix are all continuous real numbers. In such cases, the STGM algorithm gets almost zero matching result.

But for the special cases, where the adjacency matrix is 0-1 matrix, the matching accuracy decrease since the node attribute degrade as the in-degree and out-degree of the corresponding node.

In a word, STGM algorithm is suitable for matching continuous real matrices, while MBGM is more applicable for matching 0-1 matrices.

Chapter 7 Conclusions

In a word, this thesis presented a complete reified temporal logic system and a graphical representation and matching framework for representing and reasoning about temporal knowledge.

The complete reified temporal logic, CRTL, proposed in chapter 3 is actually a formulism and extension of Ma and Knight's reified temporal logic system MK within first order logic framework. So it simply inherits syntax, semantics, axioms and inference rules from normal standard many sorted first order logic. Also the well-known Herbrand's resolution method is applicable for automatic reasoning and proving for CRTL theories.

On the other side, the CRTL system also can be seen as a reification of the temporal argument method BTK and a simplification of Reichelt's temporal reified temporal logic TR. So it retains all the expressive power of the approach of temporal reification.

In all, CRTL tries to balance the expressiveness and complexity without losing logical clarity and completeness.

The simplified subsystem SCRTL is aiming at the practical application. SCRTL is still within the first order framework to retain its clear definition and sound and complete axiomatization; besides, based on its simplicity, the temporal models of SCRTL are intuitively represented as attribute graphs. For general treatment, these graphically representations have been extended for scenarios with incomplete knowledge and general temporal relations.

Based on this graphically representation, the matching of scenarios is transferred into the graph matching problem.

For the scenario graph matching, the traditional graph matching algorithms are elaborately classified and selected for matching scenario graphs. Graph matching algorithms are no longer classified as search-based methods and optimization-based methods; instead, there are grouped as: explicit-search methods, implicit-search methods, and node similarity based methods. Explicit-search methods are not suitable for our cases since they usually require exponential calculating time. Implicit-research methods are not selected for our scenario graph matching either, due to their complexities of implementation and improvement. So the node similarity based methods are the best choice for our scenario graph matching problems.

After initially testing, two important node similarity graph matching algorithms EDGM and SPGM are selected as candidates for our scenario graph matching.

EDGM algorithm is critically examined and three important constraints have been pointed out to indicate the applicable fields of this algorithm. To conquer such constraints, an approximate formula (5.6) for graph matching problem is presented together with an error estimation theorem to guarantee its accuracy. Based on such an approximate formula, the meta-basis based graph matching algorithm is proposed. Numerical tests show that the MBGM algorithm significantly improves matching accuracy for sparse isomorphic graph pairs, and partially decreases the matching error for sparse non-isomorphic graph pairs.

The SPGM algorithm is also critically examined, analyzed, and improved as a new vector sort based graph matching algorithm STGM. Numerical tests illustrate that the STGM algorithm greatly improve the matching accuracy for dense and sparse non-isomorphic graph pairs. In addition, the STGM algorithm costs less CPU time in finding these better matching.

Although random graph tests have shown great improvements of calculating accuracy for matching graph pairs by SPTM and MBGM algorithms, random scenario graph tests just provide a satisfying matching result, where the MBGM algorithm is more suitable and gets a better matching result. These tests on one hand indicate that the evaluation of graph matching algorithms directly effected by the probability distribution of random graphs; on the other hand, leave us some future work to develop special graph matching algorithms for certain kinds of graphs.

The node similarity graph matching framework is a novel feature of this thesis. Graph matching algorithms are no longer treated separately; instead, they are unified by a general algorithm framework, although they may derive from many different theories. This unification makes it possible to define, analyze and extend the node similarity graph matching algorithms together, instead of dealing with them one by one.

A negative conclusion claims that all these node similarity graph matching algorithms do not work for matching circles, which on one hand puts forward the essential limitation of these approaches; and on the other hand, leaves us an important future work to overcome such constraint to make these algorithm applicable for even more general cases.

Chapter 8 Future Work

There is room for the improvement and generalization of the methods used in this study. In this chapter, further work to temporal reification and graph matching algorithms are discussed.

Section 8.1 Reification of General Logic Systems

In chapter 3, a complete reified temporal logic system CRTL is presented within the many sorted first order logic framework. The CRTL system can be seen as a reification of syntax of temporal argument methods or modal temporal logic methods. It also has been pointed out in chapter 3 that Reichgelt's TR system reifies both syntax and semantics of an object logic system. So TR system is more complicated and expressive.

These approaches can be extended for reification of general logic systems, not only temporal logic systems. An important work has been done by Gabby, Hodkinson and Reynolds in [GHR1994], where a more general reification has been proposed, which also reifies the proof system of the object logic system.

However, general reification needs both logical foundation and computational algorithms. On one hand, the reification itself has to be clearly defined with sound and complete axiomatic system. One the other hand, the process from an object logic system to its corresponding reified logic should be computable or even polynomial-computable.

The relationship between the object logic system and its corresponding reified logic is also an interesting subject.

Section 8.2 Graphical Representation of Temporal Models

The complete reified temporal logic system CRTL defined in chapter3 allows general functions and predicates. But for the graphical representation of its temporal model, only the simplest case, temporal model of SCRTL, has been represented as attributed simple graph and matched by graph matching algorithms.

So a graphical representation of general temporal model is necessary for intuitive representation of general temporal knowledge. The notion hyper-graph would be important for such expansion. Formally a hyper-graph is a pair (N, E), where N is the set of nodes and E is the set of edges, such that each edge $e \in E$ is a subset of N.

Meanwhile the normal matching algorithms also has to be extended for matching such general graphical representation of temporal information. Again, the hyper-graph matching theory [BDK2005] will be important for such temporal information matching problems.

Section 8.3 Testing Graph Matching Algorithms for Certain Probability Distributions

It has been pointed out in chapter 5 and exemplified in chapter 6 that the probability distribution of random graphs directly effects the evaluation of graph matching algorithms. So it is important to make it even clearer that how the matching result is influenced by probability distribution of the graphs.

This work is also important to provide a standard for generally evaluating different graph matching algorithms.

Section 8.4 General Eigen-decomposition Graph Matching Algorithm

In chapter 5, an important approximate formula 5.6 for graph matching problems has been proposed together with an error estimation theorem 5.5 to guarantee its accuracy. We also bring forward a meta-basis based graph matching algorithm based on the approximate formula. But the meta-basis based graph matching algorithm is within the node similarity graph matching framework, which means it may not work for circles and some other self similar graphs.

However, the approximate formula 5.6, itself is beyond the node similarity graph matching framework, which means this approximate formula is suitable for matching self-similar graphs because of its accuracy guarantee theorem 5.5. So the algorithm directly solving the approximate formula, which is an optimization on unitary matrices and permutation space, will be more accurate and applicable for self similar graphs.

Non-linear programming or iterative methods could be good tries for solving the approximate formula 5.6.

Section 8.5 Extending Node Similarity Based Graph Matching Algorithms

In chapter 5, one of the important conclusions is that node similarity based graph matching algorithm doesn't work for circles. Although this theorem claims that there is somewhere the node similarity graph matching framework can not reach, it doesn't make it clear enough how far the node similarity graph matching framework can go. In other words, the boundary of the node similarity graph matching framework is not clear.

On the hand, there are surely some cases (including scenario graphs) that one has to deal with circles and other self-similar graphs. So it is important to extend these algorithms to overcome such constraints without lose much efficiency.

Section 8.6 Testing Real Life Examples

This thesis pays more attention to the theoretical work such as the completeness of the reified temporal logics and the accuracy guarantee theorems of some graph matching algorithms. Most of the experiments are artificial or randomly generated by algorithms. A real life test is needed in future work to verify the effectiveness of this theory.

Our theory could be applied to the area of medical information systems, where a patient's medical history is obviously very important for diagnosis. In fact, to prescribe the right treatment, the doctor needs to know not only the patient's current status, but also his/her previous health situations, including: How long has the patient been ill? Did the patient have the same problem or relevant disease previously? Has the patient had some treatment already before seeing the doctor? Has the patient been allergic to any drugs in the past? These heath histories could be constructed as a temporal scenario and matched by the algorithms discussed in chapter 5.

The weather forecast could be another possible application for our theory. Since without a good understanding of climate phenomena based on past observations the weather expert cannot make good predictions of the future. In fact, to provide correct and accurate forecast, the weather expert needs to know not only the current weather parameters summarized as temperature, air pressure, precipitation amount, wind speed and residual snow/ice amount, etc., but also the weather scenarios over some certain prior periods such as, how long did the heat wave last, was there lightning before or during the rain, did snow melt then refreeze, and so on. Such information can also be constructed as a temporal scenario and again the graph matching algorithms can be directly applied.

References

[Abd1998] A. M. Abdulkader: Parallel Algorithms for Labelled Graph Matching. PhD thesis, Colorado School of Mines, 1998

[ABM1999] C. Areces, P. Blackburn and M. Marx: Hybrid logic is the bounded fragment of first order logic. in Proceedings of 6th Workshop on Logic, Language, Information and Computation, pp. 33—50, 1999.

[ABM2000] C. Areces, P. Blackburn and M. Marx: The computational complexity of hybrid temporal logics. Logic Journal of IGPL 8(5), pp.653-679, 2000

[ACT1997] R. Allen, L. Cinque, S. Tanimoto, L. Shapiro and D. Yasuda: A Parallel Algorithm for Graph Matching and Its MarPlas Implementation, IEEE Transactions on Parallel and Distributed Systems, Vol. 8, No. 5, pp.490-501, 1997

[AD1993] H.A. Almohamad and S.O. Duffuaa: A Linear Programming Approach for the Weighted Graph Matching Problem. IEEE Trans. PAMI, 15(5), pp. 522-525, 1993

[AH1985] J. Allen and P. Hayes: A Common-Sense Theory of Time. Proceedings of the IJCAI, 9, 528-531, 1985

[AH1989] J. Allen and P. Hayes: Moments and Points in an Interval-based Temporal-based Logic. Computational Intelligence (Canada), 5, 225-238, 1989

[All1983] J. Allen.: Maintaining knowledge about temporal intervals. Communications of the ACM 26 (11), pp.832-843, 1983

[Alm1991] H.A. Almohamad: A Polynomial Transform for Matching Pairs of Weighted Graphs. Applied Mathematical Modelling, Vol. 15, pp. 216-222, 1991

[Alt2006] H. Altaf, "A Minimal Hybrid Logic for Intervals", Logic Journal of the

IGPL, 14(1), Oxford University Press, 2006, pp. 35-62.

[AMO1993] R. Ahuja, T. Magnanti and J. Orlin: Network Flows. Prentice Hall, 1993

[Auw2007] S. Auwatanamongkol: Inexact graph matching using a genetic algorithm for image recognition. Pattern Recognition Letters Volume 28, Issue 12, pp.1428-1437, 2007

[BDK2005] H. Bunke, P. Dickinson and M. Kraetzl: Theoretical and Algorithmic Framework for Hypergraph Matching. Lecture Notes in Computer Science Volume 3617, pp. 463-470, 2005

[BFG2002] H. Bunke, P. Foggia, C. Guidobaldi, C. Sansone and M. Vento: AComparison of Algorithms for Maximum Common Subgraph on RandomlyConnected Graphs: Lecture Notes in Computer Science Vol. 2396, pp.123-132, 2002

[BH1994] L. Branting and J. Hastings: An empirical evaluation of model-based case matching and adaptation. In Proceedings of the Workshop on Case-Based Reasoning, AAAI-94, Seattle, Washington pp.72-78, 1994

[BMJ1999] H. Bunke, A. Munger and X. Jiang: Combinatorial Search versus Genetic Algorithms: A Case Study Based on the Generalized Median Graph Problem. Pattern Recognition Letters, Vol. 20, pp. 1271.1277, 1999

[BS1998] H. Bunke and K. Shearer: A Graph Distance Metric Based on the Maximal Common Subgraph, Pattern Recognition Letters, Vol. 19, pp.255-259, 1998

[BT1999] P. Blackburn and M. Tzakova: Hybrid languages and temporal logic. Logic Journal of IGPL 7(1), pp.27-54, 1999

[BTK1991] F. Bacchus, J. Tenenberg and J.A. Koomen: A non-reified temporal logic. Artificial Intelligence, 52 pp. 87-108, 1991

[Bur1982] J.P. Burgess: Axioms for tense logic. I. ``Since" and ``until". Notre Dame J. Formal Logic Volume 23, Number 4 pp.367-374, 1982

[CDF1995] B. Charron-Bost, C. Delporte-Gallet and H. Fauconnier: Local and temporal predicates in distributed systems: ACM Transactions on Programming Languages and Systems (TOPLAS) Volume 17, Issue 1, pp.157 - 179 , 1995

[CL1994] T.W. Chen and W.C. Lin: A Neural Network Approach to CSG-Based 3-D Object Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No. 7, pp. 719-726, 1994

[CM1998] S. Cerrito and M.C. Mayer: Using linear temporal logic to model and solve planning problems. Lecture Notes in Computer Science Volume 1480, pp. 141-152, 1998

[CN1997] M.C. Cario and B. L. Nelson: Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. Tech. Rep., Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL U.S.A, 1997

[CR1992] P. Cucka and A. Rosenfeld : Linear Feature Compatibility for Pattern-Matching Relaxation, Pattern Recognition, Vol. 25, No. 2, pp.189-196, 1992

[CWH1997] A.D.J. Cross, C. Wilson and E.R. Hancock: Inexact Matching Using Genetic Search. Pattern Recognition, Vol. 30, No. 6, pp. 953-970, 1997

[CYS1996] L.Cinque, D. Yashuda, L. Shapiro, S. Tanimoto and B. Allen: An Improved Algorithm for Relational Distance Graph Matching, Pattern Recognition, Vol. 29, No. 2, pp.349-359, 1996

[DDL2002] C.J. Date, H. Darwen and N. Lorentzos: Temporal Data & the Relational Model, First Edition (The Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann, 1st edition, pp.422, 2002

[DH2000] Philip I. Davies and Nicholas J. Higham: Numerically stable generation of correlation matrices and their factors. BIT Numerical Mathematics, 40(4). pp. 640-651, 2000

[DHST2005] I.S. DHILLON, R.W. HEATH, M.A. SUSTIK and J.A. TROPP: Generalized finite algorithms for constructing Hermitian matrices with prescribed diagonal and spectrum. SIAM Journal on Matrix Analysis and Applications Volume 27, Issue 1, pp. 61 – 71, 2005

[DK1970] C. Davis and W.M. Kahan: The Rotation of Eigenvectors by a Perturbation. III, SIAM J. Numer. Anal. 7, pp.1-46, 1970

[DLH1998] F. Dietrich, X. Logean and J.P. Hubaux: Testing Temporal Logic Properties in Distributed Systems. IFIP Conference Proceedings, Vol. 131 Proceedings of the IFIP TC6 11th International Workshop on Testing Communicating Systems, pp.247- 258, 1998

[Dow1979] D. Dowty: Word Meaning and Montague Grammar. Dordrecht: D. Reidel, 1979

[EGR1994] M. Enciso, I.P. de Guzmân and C. Rossi: A Temporal Logic for Program Specification. GULP-PRODE (2) pp.309-323, 1994

[FLD1994] J. Feng, M. Laumy and M. Dhome: Inexact matching using neural networks. In E.S. Gelsema and L.N. Kanal (eds): Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems, pp. 177-184, 1994

[Gal1990] A.P. Galton: A Critical Examination of Allen's Theory of Action and Time. Artificial Intelligence 42, pp.159–188, 1990

[GB2002] S. Günter and H. Bunke Self-organizing Map for Clustering in the Graph Domain, Pattern Recognition Letters, Vol. 23, pp. 405.417, 2002

[GHR1994] D.M. Gabbay, I. Hodkinson and M. Reynolds: Temporal Logic Mathematical Foundations and Computational Aspects, Volume 1. Oxford, 1994

[Gib2004] W. Gibson: Pattern Recognition. Penguin Books Ltd, 2004

[Gil1956] P. C. Gilmore: An Addition to "Logic of Many-Sorted Theories". Compositio Mathematica (13), pp.277-281, 1956

[GJ1979] M.R. Garey, D.S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman & co., New York, 1979

[GR1996] S. Gold and A. Rangarajan: A Graduated Assignment Algorithm for Graph Matching. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 4, pp. 377-388, 1996

[GV1996] G.H. Golub and C.F. Van Loan: Matrix Computations, 3rd ed., Johns Hopkins University Press, Baltimore, 1996

[Han2000] B. Hansen: Weather reasoning predication using case-based reasoning and Fuzzy Set Theory. MSc Thesis, Technical University of Nova Scotia, Halifax, Nova Scotia, Canada, 2000

[Hei1978] Richard M. Heiberger: Generation of Random Orthogonal Matrices. Applied Statistics, Vol. 27, No. 2, pp. 199-206, 1978

[HJ1985] R.A. Horn and C.R. Johnson: Matrix Analysis. Cambridge University Press, 1985

[HS1986] J.Y. Halpern and Y. Shoham: A Propositional Modal Logic of Time Intervals. LICS1986, pp. 279-292, 1986

[HW1953] A.J. Hoffman and H.W. Wielandt: The Variation of the Spectrum of a Normal Matrix. Duke Math. J. 20, pp.37-39, 1953

[IZ1986] D. K. Isenor and S. G. Zaky: Fingerprint Identification Using Graph Matching. Pattern Recognition, vol. 19, No. 2, pp.113-122, 1986

[Jac1997] M. Jaczynski: A Framework for the Management of Past Experiences with Time-Extended Situations. In Proceedings of the 6th International Conference on Information and Knowledge Management (CIKM'97), Las Vegas, Nevada, USA, November 10-14, pp. 32-39, 1997

[JAS2002] M.D. Jaere, A. Aamodt and P. Skalle: Representing Temporal Knowledge for Case-Based Prediction. LNCS, Vol. 2416, Proceedings of the 6th European Conference on Advances in Case-Based Reasoning, pp.174 – 188, 2002

[JH2006] D. Justice, A.O. Hero: A Binary Linear Programming Formulation of the Graph Edit Distance. IEEE Trans. Pattern Anal. Mach. Intell. 28(8), pp.1200-1214, 2006

[Kam1968] J. A. W. Kamp: Tense Logic and the Theory of Linear Order. Ph.D. Diss., University of California, Los Angeles, 1968

[Kle1999] J. Kleinberg, J: Authoritative sources in a hyperlinked environment. Journal of the ACM, Vol. 46, No. 5, pp. 604-632, 1999

[KM1992] B. Knight and J. Ma: A General Temporal Model Supporting Duration Reasoning. Artificial Intelligence Communication, Vol. 5(2), pp.75-84, 1992

[KMN1998] B. Knight, J. Ma and E. Nissan: Representing Temporal Knowledge In Legal Discourse. Law, Computers, and Artificial Intelligence / Information and Communications Technology Law, Vol.7 (3), pp. 199-211, 1998

[Knu1998] D.E. Knuth: The Art of Computer Programming Volume 2: Semi-numerical Algorithms. Third Edition, 1998

[Kol1996] J. Kolodner: Case-Based Reasoning. Morgan Kaufmann, San Francisco, California, 1993

[Kuh1955] H.W. Kuhn: The Hungarian Method for the assignment problem. Naval Research Logistics Quarterly, 2:83-97, 1955

[Lea1996] D. Leake: Case-Based Reasoning: Experiences, Lessons, and Future Directions. AAAI Press, Menlo Park, California, 1996

[Lev1972] G. Levi: A note on the derivation of maximal common subgraphs of two directed or undirected graphs. Calcolo, Vol. 9, pp. 341–354, 1972

[Lev2006] M. Levene: An Introduction to Search Engines and Web Navigation. Addison Wesley, 2006

[LH1998] H.C. Liu and J.S. Huang J-S: Pattern Recognition Using Evolution Algorithms with Fast Simulated Annealing. Pattern Recognition Letters, Vol. 19, pp.403-413, 1998

[Lid1950] V.B. Lidskii: The proper values of sum and product of symmetric matrices. Doklady Akad. Nauk SSSR, 75, pp.769-772, 1950

[Lif1987] V. Lifschitz: A Theory of Action. Proceedings of 10th IJCAI, pp.966–972, 1987

[LRS1991] S.W. Lu, Y. Ren, and C.Y. Suen: Hierarchical attributed graph representation and recognition of handwritten Chinese characters. Pattern Recognition 24, pp.617–632, 1991

[LT2002] G. Liang, Z. Tang: Modeling Real-Time Systems with Continuous-Time Temporal Logic: Lecture Notes in Computer Science Volume 2495, pp. 231-236, 2002

[Lu1989] Z. Lu: Mathematical Logic for Computer Science. World Scientific, 1989

[May2006] M.C. Mayer, C. Limongelli, A. Orlandini and V. Poggioni: Linear temporal logic as an executable semantics for planning languages Journal of Logic. Language and Information Volume 16, Number 1, pp. 63-89, 2006

[MB1996] B.T. Messmer and H. Bunke: Automatic learning and recognition of graphical symboles in engineering drawings. In K. Tombre and R. Kasturi (eds): Graphics Recognition, Lecture Notes in Computer Science 1072, pp.123-134, Springer Verlag, 1996

[MB1998] B.T. Messmer and H. Bunke H: A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 5, pp. 493-503, 1998

[McD1982] D.V. McDermott: A Temporal Logic for Reasoning about Processes and Plans. Cognitive Science 6, pp.101–155, 1982

[McG1982] J. McGregor: "Backtrack search algorithms and the maximal common subgraph problem", Software-Practice and Experience, Vol. 12, pp. 23–34, 1982

[McT1908] J.E. McTaggart: The Unreality of Time. In "Mind", 1908

[McT1968] J.E. McTaggart: The Nature of Existence, vol. 1-2, Cambridge University Press, Cambridge, 1968

[Mez2007] F. Mezzadri: How to generate random matrices from the classical compact groups. AMS, Vol. 54, 2007

[MGA1989] E. Mjolsness, G. Gindi and P. Anandan: Optimization in Model Matching and Perceptual Organization, Neural Computation, Vol. 1, pp. 218-229, 1998

[MH2000] R. Myers and E.W. Hancock: election Strategies for Ambiguous Graph Matching by Evolutionary Optimisation, in Lecture Notes in Computer Science Vol. 1876: pp.397-406, 2000

[MH2001] R. Myers and E.W. Hancock: Least-commitment Graph Matching with Genetic Algorithms. Pattern Recognition, Vol. 34, pp. 375-394, 2001

[MH2006] J. Ma and P. Hayes: Primitive Intervals Vs Point-Based Intervals: Rivals Or Allies. The Computer Journal, Vol.49 (1), 32-41, 2006

[MK1994] J. Ma and B. Knight B: A General Temporal Theory. The Computer Journal, 37(2), 114-123, 1994

[MK1996] J. Ma and B. Knight: A Reified Temporal Logic. The Computer Journal 39(9), pp. 800-807, 1996

[MK2003] J. Ma and B. Knight: A Framework for Historical Case-Based Reasoning. Lecture Notes in Computer Science, Volume 2689, pp.1067, 2003

[ML2005] J. Ma and B. Luo: Representing and Recognizing Scenario Patterns. Lecture Notes in Computer Science, Vol.3614, 140-149, 2005

[MT1993] K. Meinke and J.V. Tucker: Many-sorted logic and its applications. John Wiley & Sons, Inc. New York, NY, USA, 1993

[Mun1957] J. Munkres: Algorithms for the Assignment and Transportation Problems. Journal of the Society of Industrial and Applied Mathematics, 5(1):pp.32-38, 1957

[MZH2007] J. Ma, G. Zhao and E. Hancock: A Navigation-based Algorithm for Matching Scenario Patterns. Proceedings of International Conference on Artificial Intelligence and Pattern Recognition (AIPR-07), pp.151-158, Orlando, FL, USA, ISRST 2007

[MZH2007] J. Ma, G. Zhao, E. Hancock: A Navigation-based Algorithm for Matching Scenario Patterns. Proceeding of AIPR07 pp.151-157, 2007

[Nak1994] G. Nakhaeizadeh: Learning Prediction of Time Series: A Theoretical and Empirical Comparison of CBR with Some Other Approaches. In Proceedings of the Workshop on Case-Based Reasoning, AAAI-94, Seattle, Washington pp. 67-71, 1994

[NB2007] M. Neuhaus and H. Bunke: A Quadratic Programming Approach to the Graph Edit Distance Problem. Lecture Notes in Computer Science Graph-Based Representations in Pattern Recognition, Volume 4538 pp. 92-102, 2007

[Ost1989] J.S. Ostroff: Temporal logic for real time systems. Wiley Advanced Software Development Series pp.209, 1989

[Pei1969] A.N. Prior: Papers on Time and Tense, Oxford: Clarendon Press,1969

[PG1995] M. Peng and N. Gupta: Invariant and Occluded Object Recognition Based on Graph Matching. International Journal on Electrical Engineering Education, Vol. 32, pp.31-38, 1995

[Poo1993] J. Poole: Similarity in legal case based reasoning as degree of matching in conceptual graphs. In M.M. Richter, S. Wess, K.D. Althoff, and F. Maurer (eds.): proceedings of First European Workshop on Case-Based Reasoning, pp.54–58, 1993

[Pri1957] A.N. Prior: Time and Modality, Oxford: Clarendon Press, 1957

[Pri1967] A.N. Prior: Past, Present and Future, Oxford: Clarendon Press, 1967

[Pri1969] A.N. Prior, A. N: Papers on Time and Tense, Oxford: Clarendon Press, 1969

[PZK1998] M. Pozniak, K. Zyczkowski, M. Kus: Composed ensembles of random unitary matrices. Journal of Physics, Mathematical and General, Volume 31, Number 3, pp. 1059-1071(13), 1998

[RB1979] D.H. Rouvray and A.T. Balaban: Chemical applications of graph theory. In R.J.Wilson and L.W.Beineke (eds.): Applications of Graph Theory, pp.177–221, Academic Press, 1979

[Rei1989] H. Reichgelt: A Comparison of First-order and Modal Logics of Time. In Jackson, P. and van Harmelen, H.R.F. (eds.). Logic-based Knowledge Representation, pp. 143-176, 1989

[Ric1989] B. Richards, I. Bethke, J. van der Does and J. Oberlander: Temporal Representation and Inference. London: Academic Press, 1989

[Rom1989] H.C. Romesburg: Cluster Analysis for Researchers. Lifetime Learning Pub, 1984

[SBV2001] K. Shearer, H. Bunke and S. Venkatesh: Video Indexing and Similarity Retrieval by Largest Common Subgraph Detection Using Decision Trees, Pattern Recognition, Vol. 34, pp.1075-1091, 2001

[SC1985] A. Sistla and E. Clarke. Complexity of propositional linear temporal logics. J.ACM, 33, pp.733-749, 1985

[SH1981] L.G. Shapiro R.M. Haralick: Structural Descriptions and Inexact Matching. IEEE Transactions on Pattern Analysis and Machine Intelligence,Vol. 3, No. 5, pp.504-519, 1981

[Sha1995] S. Shams: Multiple Elastic Modules for Visual Pattern Recognition. Neural Networks, Vol. 8, No. 9, pp. 1439-1456, 1995

[Sha1999] Y. Shahar: Timing is everything: temporal reasoning and temporal data maintenance medicine. Eds. W Horn et al., AIMDM'99, LNAI 1620, Springer Verlag, pp. 30-46, 1999

[Sho1987] Y. Shoham: Temporal Logics in AI: Semantical and Ontological

Considerations. Artificial Intelligence, 33, pp. 89-104, 1987

[SJ1999] R.T. Snodgrass, C.S. Jensen: Developing Time-Oriented Database Applications in SQL (Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann; pp.504 1999

[SSG2005] O. Sammoud, C. Solnon, K. Ghèdira: Ant Algorithm for the Graph Matching Problem. EvoCOP 2005, pp.213-223, 2005

[Sta1979] J. Stavi: Functional completeness over the rationals. Unpublished, Bar-Ilan University, Ramat-Gan, Israel, 1979

[Ste1980] G.W. Stewart: The efficient generation of random orthogonal matrices with an application to condition estimators. SIAM J. Num. Anal., 17, pp. 403-409, 1980

[SY1998] P.N. Suganthan and H. Yan: Recognition of Handprinted Chinese Characters by Constrained Graph Matching, Image and Vision Computing, Vol. 16, pp. 191-201, 1998

[TA1997] M. Turner and J. Austin: A neural relaxation technique for chemical graph matching. Proceedings of 5th International Conference of Artificial Neural Networks, Cambridge, 1997

[Tay1985] B. Taylor: Modes of Occurrence. Aristotelian Society Series, Volume 2, Oxford: Basil Blackwell, 1985

[TF1983] W.H. Tsai and K.S. Fu: Subgraph Error-Correcting Isomorphisms for Syntactic Pattern Recognition. IEEE Transactions on Systems, Man and Cybernetics, Vol. 13, No. 1, pp.48-62, 1983

[TH2001] A. Torsello and E.R. Hancock: Efficiently Computing Weighted Tree Edit Distance Using Relaxation Labeling. Lecture Notes in Computer Science Energy Minimization Methods in Computer Vision and Pattern Recognition Volume 2134, pp. 438-453, 2001

[TH2003] A. Torsello and E.R. Hancock: Computing approximate tree edit distance using relaxation labeling. Pattern Recognition Letters, Volume 24, Issue 8, pp.1089-1097, 2003

[TK2006] S. Theodoridis, K. Koutroumbas: Pattern Recognition (3rd edition). Elsevier, 2006

[TT1982] M.A. Tanner and R.A. Thisted: Generation of random orthogonal matrices. Applied Statistics 31, pp. 190-192, 1982

[Ull1976] J.R. Ullmann: An Algorithm for Subgraph Isomorphism. Journal of the ACM, 23(1):31–42, 1976

[Ume1988] S. Umeyama: An Eigen-decomposition Approach to Weighted Graph Matching Problems. IEEE Trans. PAMI, 10(5), pp. 695-703, 1988

[Ven1990] Y. Venema: Expressiveness and completeness of an interval tense logic. Notre Dame J. Formal Logic Volume 31, Number 4 pp.529-547, 1990

[Vil1994] L. Villa:A survey on temporal Reasoning in Artificial Intelligence. AI Commun. 7, pp. 4-28, 1994

[Wan1952] H. Wang: Logic of Many-Sorted Theories. The Journal of Symbolic Logic, Vol. 17, No. 2, pp. 105-116, 1952

[Wat1997] I. Watson: Applying Case-Based Reasoning: Techniques for Enterprise Systems. Morgan Kaufmann, San Mateo, California, 1997

[WFK1997] L. Wiskott, J. Fellous, N. Krüger, and C. von der Malsburg: Face Recognition by Elastic Bunch Graph Matching. IEEE Trans. on Pattern Analysis and Machine Intelligence 19(7), pp.775-779, 1997

[WI2006] Y. Wang1 and N. Ishii: A genetic algorithm and its parallelization for graph matching with similarity measures. Artificial Life and Robotics Volume 2, Number 2, pp.68-73, 2006

[Wil1997] S. Willi-Hans: Matrix calculus & the Kronecker product with applications & C++ programs. World Scientific Publishing Company, September 1997

[Won1992] E.K. Wong: Model matching in robot vision by subgraph isomorphism. Pattern Recognition, Vol. 25, pp.287–304, 1992

[Wyk2002] B.J. van, Wyk.: Kronecker Product Successive Projection and Related Graph Matching Algorithms. Ph.D. diss., University of the Witwatersrand, Johannesburg, 2002

[XO1990] L. Xu and E. Oja: Improved simulated annealing, Boltzmann machine, and attributed graph matching: In L.Almeida (ed): LNCS 412, pp. 151–161, 1990

[Zag2003] L. Zager: Graph similarity and matching. Master's thesis, Massachusetts

Institute of Technology, 2003

[ZLM2006] G. Zhao, B. Luo and J. Ma: Matching Case History Patterns in Case-Based Reasoning. Lecture Notes in Computer Science - ICIC(2), Vol.345, 312-321, 2006

[ZLM2007] G. Zhao, B. Luo and J. Ma: Matching Scenario Patterns by Using Linear Programming. Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery, Vol.3, pp.346-350, IEEE CS Press, 2007

[ZLT2007] G. Zhao, B. Luo, J. Tang and J. Ma: Using Eigen-Decomposition Method for Weighted Graph Matching. Lecture Notes in Computer Science - Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues: Volume 4681, pp.1283-1294. 2007

[ZMS2008] G. Zhao, J. Ma, F. Shen and M. Petridis: A Sound and Complete Reified Temporal Logic. In proceedings of the Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD2008)

[ZV2007] L.A. Zager and G.C. Verghese: Graph similarity scoring and matching, Applied Mathematics Letters, v. 21, n.1, 86-94, 2007.

[Zyc1994] K. Zyczkowski et al: Random unitary matrices. J. Phys. A: Math. Gen. 27 pp.4235-4245, 1994

A Sound and Complete Reified Temporal Logic

Guoxing Zhao[1], Jixin Ma[1], Fuxing Shen[2] and Miltos Petridis[1]
[1]*School of Computing and Mathematical Sciences, University of Greenwich, U.K*
[2]*College of Information Science and Technology, Beijing Normal University, China*
{g.zhao, j.ma, m.petridis}@gre.ac.uk

Abstract

There are mainly two known approaches to the representation of temporal information in Computer Science: modal logic approaches (including tense logics and hybrid temporal logics) and predicate logic approaches (including temporal argument methods and reified temporal logics). On one hand, while tense logics, hybrid temporal logics and temporal argument methods enjoy formal theoretical foundations, their expressiveness has been criticised as not power enough for representing general temporal knowledge; on the other hand, although current reified temporal logics provide greater expressive power, most of them lack of complete and sound axiomatic theories. In this paper, we propose a new reified temporal logic with a clear syntax and semantics in terms of a sound and complete axiomatic formalism which retains all the expressive power of the approach of temporal reification.

1. Introduction

Temporal representation and reasoning plays a fundamental and increasingly important role in most areas of Computer Science. A natural approach to represent and reason about time-dependent knowledge is to associate them with instantaneous time points and/or durative time intervals. In particular, there are various ways to use logic formalisms for temporal knowledge representation and reasoning. Based on the chosen logic frameworks, temporal theories can be classified into *modal logic approaches* and *predicate logic approaches* (see Table 1).

Modal logic approaches semantically re-interpret the classical possible-worlds by making each possible world represent a different time. It accommodates the concepts of time by means of extending the propositional or predicate calculus to include modal temporal operators such as Fφ, Pφ, Hφ and Gφ, representing that formula φ "will be true", "was true", "will be always true" and "was always true", respectively. For instance, time transitivity can expressed by $(FF\varphi \rightarrow F\varphi) \wedge (PP\varphi \rightarrow P\varphi)$ and "Jack will love Rose forever" can expressed as GLove(Jack, Rose).

Generally speaking, each of the current temporal modal logic (US [9], HS [8], BT [5] and HLI [3]) has its own clear syntax, semantics, fully axiomatized sound and complete formal deduce system, together with an algorithm to determine the formula validity or satisfiability. However, the expressiveness of the approach of temporal modal logic has been noted to be limited. For instance, tense logics such as H_t and US cannot even characterize time irreflexivity, where in Hybrid logics, such a property has to be characterized by some non-intuitive formula like $\downarrow\varphi P\neg\varphi$ [5]. In addition, modal logic approaches cannot express propositions which associate with more than one time elements. Therefore, it is difficult to use modal temporal logics to represent and reason about action, change, causality, and so on.

Table 1. Some temporal logic approaches

Modal-logic approaches		Predicate-logic approaches	
Tense logic	Hybrid logic	Temporal argument	Reified logic
H_t: Prior (1957) [12]	BT: Blackburn and Tzakova (1999) [5]	TM: Reichgelt (1989) [13]	TR: Reichgelt (1989) [13]
US: Kamp (1968) [9]		STL: Shoham (1987) [14]	ITL: Allen (1983) [1]
HS: Halpern and Shoham (1991) [8]	HLI: Altaf (2006) [3]	BTK: Bacchus et al (1991) [4]	RTL: Ma and Knight (1996) [10]

Predicate temporal logic approaches are normally many-sorted languages including a sort of temporal elements and a sort of non-temporal elements. There are three kinds of functions and predications: (i) temporal predicates that only take temporal terms as arguments to describe temporal relationships; (ii) non-temporal predicates that take only non-temporal terms as arguments to describe non-temporal relationships; (iii) comprehensive predicates that take both temporal and non-temporal terms as arguments to describe global relationships between temporal and non-temporal terms.

Temporal argument approaches try to devise temporal logics by means of simply including time elements as additional arguments to functions and predicates in first order language. In this way, the method of temporal argument directly employs the syntax, semantics and the axiom system of the standard first order logic, and therefore, the completeness and soundness of temporal argument theories remain as default. Compared with modal temporal logics, on one hand, the method of temporal argument has more expressive power in representing properties of the time itself. For instance, using the method of temporal argument, the irreflexivity of time can be simply characterized by: $\forall t\neg(t<t)$. However, on the other hand, to express statements involving "tense" knowledge, the method of temporal argument needs formulas which are more complicated than those would be in modal temporal approaches. E.g., using the method of temporal argument, statement "Jack will love Rose forever" needs to be expressed as something like: $\forall t(t > now) \rightarrow Love(Jack, Rose, t))$. In addition, since time is represented just as an additional argument(s) to functions and/or predicates, neither conceptual nor notational special status to time is accorded in temporal argument approaches. Therefore, it is not expressive enough to talk about the generalities of the temporal aspect of assertions. For example, using the method of temporal argument, one cannot express common-sense knowledge such as "effects cannot precede their causes" [14].

It has been point out that [10], in general, reified temporal logics that reifies some initial logics are more expressive in:

- According the special status to time;
- Classifying different types of temporal occurrence;
- Representing incompatibility and negation;
- Reasoning about event, change and causality;
- Representing relationships between events and their effects.

However, each of the current reified temporal logics is somehow quite complicated, making it difficult to have a clear simple description of the syntax, semantics, and complete and sound axiomitization. For instance, the syntax and semantics are actually missing from Allen's interval-based temporal logic [1], while in Ma and Knight's reified temporal logic [10] which does have a clear syntax and semantics, it is difficult (if not impossible) to prove its completeness and soundness. The only exception is Reichgelt's TR [13] which is actually developed totally within the framework of first order language, and therefore inherits all the properties of the first order theory. However, TR is too complicated to achieve acceptable efficiency in applications.

The expressive power of these temporal logic approaches can be illustrated as Fig.1, where an arrow from system X to system Y denotes that X can be expressed/subsumed from Y.

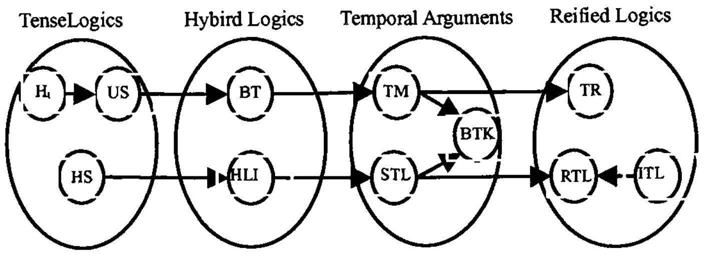


Fig.1. Expressiveness of the existing temporal logic approaches

In this paper, we shall introduce a reified temporal logic (CRTL) with a clear syntax, and semantics, where the completeness and soundness of the theory is guaranteed by the fundamental initial first order language. On one hand, it is simpler and more efficient than Reichgelt's reified first order logic [13] which is the only known reified temporal system with a complete and sound theory; on the other hand, it retains all the expressive power of those temporal predicate approaches including that of Allen [1] and that of Ma and Knight [10]. The structure of CRTL, including its syntax and semantics, as well as the formalism and inference rules are presented in section 2. Section 3 illustrates the expressive power of CRTL. Finally, section 4 concludes the paper.

2. The language

In CRTL, terms have three types: temporal terms, non-temporal terms and propositional terms. Functions can be either logic or non-logic: logic functions and non-logic functions. While logic functions map propositional terms to propositional terms, non-logic function are partitioned into three types: temporal functions that map temporal and/or non-temporal terms to temporal terms; non-temporal functions that map temporal and/or non-temporal terms to non-temporal terms; and propositional functions that map non-temporal terms to propositional terms. In addition, relations in CRTL are also partitioned into three types: temporal relations, non-temporal relations and meta-relations, where temporal relations take temporal terms alone as its arguments, non-temporal relations take non-temporal terms alone as its arguments and meta-relations take both propositional terms and temporal terms as its arguments.

The structure of CRTL can be represented as Fig.2:

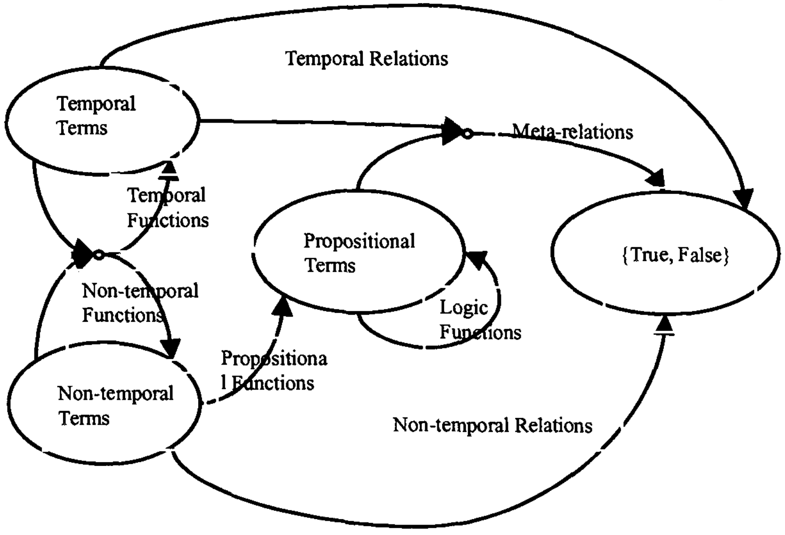


Fig. 2. Structure of CRTL

2.1. Syntax of CRTL

CRTL is a many-sorted first order logic with equality, including the sort of temporal terms T, the sort of non-temporal terms U, and the sort of propositional terms P. Variables of T, U and P are denoted by $(t,t_1,t_2,\ldots)$, $(u,u_1,u_2,\ldots)$ and $(p,p_1,p_2,\ldots)$, etc. The signature of CRTL is 3 tuple L=<*C*, *F*, *R*>, where

- *C*=*UC* ∪ *TC*:
  1. Each uc∈*UC* is a non-temporal constant symbol;
  2. Each tc∈*TC* is a temporal constant symbol;
- *F*=*UF* ∪ *TF* ∪ *PF* ∪ *LF*
  1. Each uf∈*UF* is a non-temporal function symbol with sort $U^n \times T^m \rightarrow U$;
  2. Each tf∈*TF* is a temporal function symbol with sort $U^n \times T^m \rightarrow T$;
  3. Each pf∈*PF* is a propositional function symbol with sort $U^n \rightarrow P$;
  4. *LF*={NOT, AND} ∪ {FORALL-u_j : j∈N}
     - 4.1. "NOT" is a function symbol with sort P→P,
     - 4.2. "AND" is a function symbol with sortTL $P^2 \rightarrow P$,
     - 4.3. Each "FORALL-u_j" is a u_j-bounded function symbol with sort P→P
- *R*=*UR* ∪ *TR* ∪ *MR*
  1. Each ur∈*UR* is a non-temporal relation symbol with sort U^n;
  2. Each tr∈*TR* is a temporal relation symbol with sort T^m;
  3. Each mr∈*MR* is a meta relation symbol with sort $P^n \times T^m$

Note1: In this paper, the negation, conjunction and universal quantifier of propositional terms are denoted as "NOT", "AND" and "FORALL", distinct from that as for conventional well-formed-formulas symbolized by "¬", "∧" and ∀, respectively.

Note2: Here, connectives and quantifiers of propositional terms are taken as functions from propositional terms to propositional terms. It is important to note that, connectives "NOT" and "AND" are treated as variable-free functions while quantifiers FORALL-u_j are treated as variable-bounded functions[1].

Definitions of "term", "atomic formula" and "well-formed-formula", etc. in CRTL are given in the conventional way as in standard many-sorted first order logic.

2.2. Semantics of CRTL

A Modal of CRTL is formally defined as a 4-tuple M = <*U*, *T*, *P*, σ>, where

- *U*, *T* and *P* are non-empty universes, representing the set of non-temporal objects, temporal time elements and propositional terms, respectively.
- σ is an interpretation function interpreting every symbol defined in CRTL's signature as below:
  1. Each symbol uc∈*UC* is interpreted as a non-temporal element σ(uc)∈*U*;
  2. Each symbol tc∈*TC* is interpreted as a temporal element σ(tc)∈*T*;
  3. Each symbol uf∈*UF* is interpreted as a non-temporal function from $U^n \times T^m$ to *U*;
  4. Each symbol tf∈*TF* is interpreted as a temporal function from $U^n \times T^m$ to *T*;
  5. Each symbol pf∈*PF* is a interpreted as a propositional function from U^n to *P*;
  6. Symbols NOT, AND and "FORALL-u_j" are interpreted as functions from propositional terms to propositional terms.

[1] A variable-binding function f_x is a normal function with special property that every occurrence x in $f_x(\ )$ is bounded. In this case, f_x is called a x-bounded function.
For example, Let Q[x, y] be the set of all the 2-variable polynomials with variables x, y and rational coefficients. Let Sqr and $\frac{\partial}{\partial y}$ denote the two functions, square and partial derivation, from Q[x,y] to Q[x,y], respectively. Then Sqr is a variable-free function, and $\frac{\partial}{\partial y}$ is a y-bounded function. Obviously, x in equation $Sqr(x+y)=x^2+2xy+y^2$ can be substituted by y^2, which can not be done in $\frac{\partial}{\partial y}(x+y)=1$.
That is one of the main differences between variable-free functions and variable-bounded functions.

7. Each symbol ur$\in$*UR* is interpreted as a non-temporal relation on U^n;
8. Each symbol tr$\in$*TR* is interpreted as a temporal relation on T^m;
9. Each symbol mr$\in$*MR* is interpreted as a meta relation on $P^n \times T^m$.

All the other interpretations to variables, formulas, connectives and quantifiers are the same as in standard many-sorted first order logic.

3. The advantages of CRTL

In the above section, we present the syntax and semantics for a new reified temporal logic, CRTL. Unlike those existing reified temporal theories most of which lack of a sound and complete proof theory, the completeness and soundness of this new logic can be inherited directly from the initial object language, that is, the standard first order logic.

CRTL is a true reified temporal logic since a sort of meta-relations are formally defined which allow one to predicate and quantify over propositional terms and temporal terms together, and therefore provide one the expressive power to represent and reason about both temporal and non-temporal relationships between the propositional terms. Its expressive power compared with the other temporal logic systems can be illustrated as Fig.3, where, as in Fig.1, an arrow from system X to system Y denotes that X can be expressed/deduced from Y.

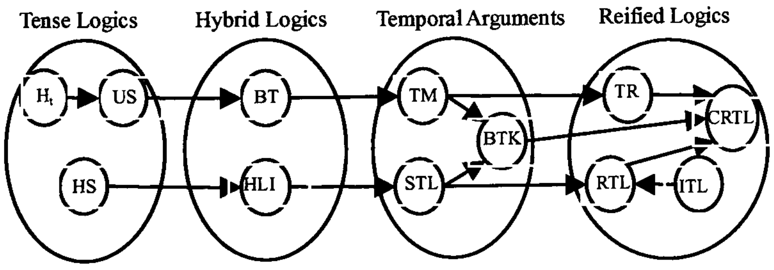


Fig. 3. Expressive power of CRTL

Specially, CRTL can be seen as an expansion of RTL or a simplification of TR. Therefore, it is straightforward to see that, using CRTL, one can either directly subsume or equivalently express the other existing reified temporal theories.

3.1. Expressing time structures

Since CRTL follows the predicate logic approach, it is handy to express various time structures/models by means of specializing the set of time elements and the set of temporal relations. In fact:

3.1.1. Point-based time structure. A typical point-based time structure can be defined by means of simply taking the set of time elements as a collection of points *Point*, and specifying a primitive temporal relation ≤ which (partially or totally) orders *Point*. As derived time objects, point-based interval may be defined in terms of the following four forms:

$$(p_1, p_2) = \{p \mid p \in Point \wedge p_1 < p < p_2\}$$
$$[p_1, p_2) = \{p \mid p \in Point \wedge p_1 \le p < p_2\}$$
$$(p_1, p_2] = \{p \mid p \in Point \wedge p_1 < p \le p_2\}$$
$$[p_1, p_2] = \{p \mid p \in Point \wedge p_1 \le p \le p_2\}$$

Following Allen's terminology [1], the immediate predecessor order relation over intervals, Meets, can be defined as in terms of the primitive relation ≤ over primitive time points:

$$\begin{aligned} &\text{Meets}(t_1, t_2) \Leftrightarrow \\ &\quad \exists p_1,p,p_2 \in R(t_1 = (p_1, p) \wedge t_2 = [p, p_2) \\ &\qquad\qquad \vee\, t_1 = [p_1, p) \wedge t_2 = [p, p_2)) \\ &\qquad \vee\, t_1 = (p_1, p) \wedge t_2 = [p, p_2] \\ &\qquad\qquad \vee\, t_1 = [p_1, p) \wedge t_2 = [p, p_2] \\ &\qquad \vee\, t_1 = (p_1, p] \wedge t_2 = (p, p_2) \\ &\qquad\qquad \vee\, t_1 = [p_1, p] \wedge t_2 = (p, p_2) \\ &\qquad \vee\, t_1 = (p_1, p] \wedge t_2 = (p, p_2] \\ &\qquad\qquad \vee\, t_1 = [p_1, p] \wedge t_2 = (p, p_2]) \end{aligned}$$

It is easy to see that the intuitive meaning of Meets(t_1,t_2) is that, on the one hand, intervals t_1 and t_2 don't overlap each other (i.e., they don't have any part in common, not even a point); on the other hand, there is not any other time standing between them.

Other order relations analogous to those introduced by Allen for primitive intervals [1], that is, Equal, Before, After, Met_by, Overlaps, Overlapped_by, Starts, Started_by, During, Contains, Finishes, Finished_by can also be defined in terms of Meets. E.g.: Before(t_1, t_2) ↔ ∃t(Meets(t_1, t) ∧ Meets(t, t_2))

When applied to points and/or point-based intervals, these 13 can be classified into the following 4 groups:

- Relations that relate points to points:
  {Equal, Before, After}
- Relations that relate points to intervals:
  {Before, After, Meets, Met_by, Starts, During. Finishes}
- Relations that relate intervals to points:
  {Before, After, Meets, Met_by, Started_by, Contains, Finished_by}
- Relations that relate intervals to intervals:
  {Equal, Before, After, Meets, Met_by, Overlaps, Overlapped_by, Starts, Started_by, During, Contains, Finishes, Finished_by}

3.1.2. Interval-based time structure. Allen's temporal system [1] is a typical interval-based time theory. One may deduce such a time structure from

CRTL by means of simply taking intervals alone as primitive and completely excluding time points from the sort of time elements. Unlike the Meet relation as defined in section 3.1.1 as derived relation from ≤, the Meets relation here can be defined as primitive over intervals in terms of the following axioms:

T1. $\forall t_1,t_2,t_3,t_4(\mathrm{Meets}(t_1, t_2) \wedge \mathrm{Meets}(t_1, t_3) \wedge \mathrm{Meets}(t_4, t_2) \rightarrow \mathrm{Meets}(t_4, t_3))$

That is, if an interval meets two other intervals, then any interval that meets one of these two must also meets the other.

T2. $\forall t \exists t_1,t_2(\mathrm{Meets}(t_1, t) \wedge \mathrm{Meets}(t, t_2))$

That is, each interval has at least one immediate predecessor, as well as at least one immediate successor.

T3. $\forall t_1,t_2,t_3,t_4(\mathrm{Meets}(t_1, t_2) \wedge \mathrm{Meets}(t_3, t_4) \rightarrow$

$$\mathrm{Meets}(t_1, t_4)$$
$$\nabla\, \exists t'(\mathrm{Meets}(t_1, t') \wedge \mathrm{Meets}(t', t_4))$$
$$\nabla\, \exists t''(\mathrm{Meets}(t_3, t'') \wedge \mathrm{Meets}(t'', t_2)))$$

where ∇ stands for "exclusive or". That is, any two meeting places are either identical or there is at least an interval standing between the two meeting places if they are not identical.

T4. $\forall t_1,t_2,t_3,t_4(\quad \mathrm{Meets}(t_3, t_1) \wedge \mathrm{Meets}(t_1, t_4) \wedge \mathrm{Meets}(t_3, t_2) \wedge \mathrm{Meets}(t_2, t_4)) \rightarrow t_1 = t_2)$

That is, the interval between any two meeting places is unique.

N.B. For any two adjacent time elements t_1 and t_2, that is $\mathrm{Meets}(t_1, t_2)$, $t_1 \oplus t_2$ will be used to denote their ordered union. The existence of such an ordered union is guaranteed by axioms T2 and T3, while its uniqueness is guaranteed by axiom T4.

Allen's other order relations for primitive intervals [1] including Equal, Before, After, Met_by, Overlaps, Overlapped_by, Starts, Started_by, During, Contains, Finishes, Finished_by can be defined in terms of Meets.

3.1.3. Point&Interval-based time structure. On the one hand, it has been argued that defining intervals as derived temporal objects out of points may lead to the so-called Dividing Instant Problem [1, 6, 16], that is the puzzle encountered when attempting to represent what happens at the boundary instant (point) which divides two successive intervals. On the other hand, it has been pointed out both points and intervals are needed for making temporal reference to instantaneous phenomena with zero duration, and periodic phenomena which last for some positive duration, respectively. For general treatments, one may take the set of time elements to include both points and intervals as primitive on an equal footing: points do not have to be defined as limits of intervals and intervals do not have to be constructed out of points. An example of such a theory is that of Ma and Knight [10]. To support a point&interval-based time structure, one can define a new sort D as the set of non-negative real numbers, and a duration assigning function DUR from T to D. If Dur(t) > 0 then t is called an interval, otherwise, t is called a point. In addition to axioms T1. – T4. in section 3.1.2., one needs the following two additional axioms:

T5. $\forall t_1,t_2(\mathrm{Meets}(t_1, t_2) \rightarrow \mathrm{Dur}(t_1) > 0 \vee \mathrm{Dur}(t_2) > 0)$

That is, time elements with zero duration cannot meet each other.

T6. $\forall t_1,t_2(\mathrm{Meets}(t_1, t_2) \rightarrow$

$$\mathrm{Dur}(t_1 \oplus t_2) = \mathrm{Dur}(t_1) + \mathrm{Dur}(t_2))$$

That is, the ordered union operation ⊕ over time elements is consistent with the conventional "addition" operation over the duration assignment function Dur.

3.2. Expressing temporal causal relationships

To talk about temporal incidence, we define a meta-relation True∈*MR* over sort P×T by the following axioms:

<C1> $\forall p \forall t(\mathrm{True}(\mathrm{NOT}(p), t) \leftrightarrow \neg\mathrm{TRUE}(p, t))$

<C2> $\forall p_1,p_2 \forall t(\mathrm{True}(\mathrm{AND}(p_1, p_2), t) \leftrightarrow \mathrm{True}(p_1, t) \wedge \mathrm{True}(p_2, t))$

<C3> $\forall p \forall t(\mathrm{True}(\mathrm{FORALL\text{-}u}(p), \quad t') \quad \leftrightarrow \forall u(\mathrm{True}(p, t)))$

We shall use formula $\mathrm{Causes}(t_1, t, t_2, p_1, p, p_2)$ to denote a causal law, which intuitively states that, under the precondition that proposition p_1 hold true over time t_1, the truth holding of proposition p over time t will cause the truth holding of proposition p_2 over time t_2. This can be formally characterized by the the following axiom:

<C4> $\mathrm{Causes}(t_1, t, t_2, p_1, p, p_2) \wedge \mathrm{True}(p_1, t_1) \wedge \mathrm{True}(p, t) \Rightarrow \mathrm{True}(p_2, t_2)$

In order to characterize temporal relationships between events and their effects, we impose the following temporal constraints:

(C5) $\mathrm{Causes}(t_1, t, t_2, p_1, p, p_2) \Rightarrow \mathrm{Meets}(t_1, t) \wedge (\mathrm{Meets}(t_1, t_2) \vee \mathrm{Before}(t_1, t_2))$

It is important to note that axiom (C5) presented above actually specifies the so-called (most) general temporal constraint (GTC) [14]. Such a GTC guarantees the common-sense assertion that "the beginning of the effect cannot precede the beginning of the cause".

Actually, there are 8 possible temporal order relations between times t_1, t and t_2 which satisfy (C5). These include:

(1) The effect becomes true immediately after the end of the event and remains true for some time after the event.

(2) The effect holds only over the same time over which the event is in progress.

(3) The beginning of the effect coincides with the beginning of the event, and the effect ends before the event completes.

(4) The beginning of the effect coincides with the beginning of the event, and the effect remains true for some time after the event.

(5) The effect only holds over some time during the progress of the event.

(6) The effect becomes true during the progress of the event and remains true until the event completes.

(7) The effect becomes true during the progress of the event and remains true for some time after the event.

(8) There is a time delay between the event and its effect.

4. Conclusion

In this paper, we have introduced a new reified temporal logic, CRTL, with a clear syntax and semantics, which, by inheriting from the initial first order language, enjoys a sound and complete axiomatic system. This is the main improvement made to the reification approach to temporal representation and reasoning. It is a true reified logic since a sort of meta-relations is formally defined that allow one to predicate and quantify over propositional terms, and therefore provides the expressive power to represent and reason about both temporal and non-temporal relationships between propositional terms. It is demonstrated that the new logic proposed here retains the appealing characteristics of some most representatively existing temporal systems that utilize the technique of reification.

5. Reference

1. J. F. Allen, "Maintaining Knowledge about Temporal Intervals", *Communications of the ACM* 26(11), 1983, pp. 832–843.
2. J. F. Allen, "Towards a General Theory of Action and Time", Artificial *Intelligence*, 23, 1984, pp. 123-154.
3. H. Altaf, "A Minimal Hybrid Logic for Intervals", *Logic Journal of the IGPL*, 14(1), Oxford University Press, 2006, pp. 35-62.
4. F. Bacchus, J. Tenenberg and J.A. Koomen, "A non-reified temporal logic", Artificial *Intelligence*, 52, 1991, pp. 87-108.
5. P. Blackburn, M. Tzakova, "Hybrid languages and temporal logic", *Logic Journal of IGPL* 7(1), 1999, pp. 27-54.
6. A. P. Galton, "A Critical Examination of Allen's Theory of Action and Time", Artificial *Intelligence*, 42 , 1990, pp. 159-188.
7. A.P. Galton, "An Investigation of Non-intermingling Principles in Temporal Logic", *Journal of Logic and Computation*, 6(2), 1996, pp. 267-290.
8. J.Y. Halpern and Y. Shoham, "A Propositional Model Logic of Time Intervals", *Journal of the Association for Computing Machinery*, 38(4), 1991, pp. 935-962.
9. J. A. W. Kamp, "Tense Logic and the Theory of Linear Order". Ph.D. Diss., University of California, Los Angeles, 1968.
10. J. Ma and B. Knight, "A Reified Temporal Logic", *the Computer Journal* 39(9), 1996, pp. 800-807.
11. J. Ma and B. Knight, "A General Temporal Theory", *the Computer Journal* 37(2), 1994, pp. 114-123.
12. A.N. Prior, "Time and Modality", Clarendon Press, Oxford, 1957.
13. H. Reichgelt, "A Comparison of First-order and Modal Logics of Time", *In Jackson, P. and van Harmelen, H.R.F. (eds.). Logic-based Knowledge Representation*, 1989, pp. 143-176.
14. Y. Shoham, "Temporal Logics in AI: Semantical and Ontological Considerations", Artificial *Intelligence*, 33, 1987, pp. 89-104.
15. H. Wang, "Logic of many-sorted theories", *Journal of Symbolic. Logic* 17, 1952, pp. 105–116.
16. L. Villa, "A survey on temporal Reasoning in Artificial Intelligence", *AI Commun.* 7, 1994, pp. 4-28.

A Navigation-based Algorithm for Matching Scenario Patterns

Jixin Ma[1], Guoxing Zhao[1], Edwin Hancock[2]
[1]School of Computing and Mathematical Sciences, University of Greenwich, U.K.
[2]Department of Computer Science, University of York, U.K.
j.ma@gre.ac.uk, g.zhao@gre.ac.uk, erh@cs.york.ac.uk

Abstract

This paper presents a unified scheme for formalizing scenario patterns, which are defined as vectors of states with the corresponding temporal constraints, whereas states are represented as collections of Boolean-valued propositions whose truth-values are dependent on the time. A temporal network, called scenario graph, is introduced to graphically represent scenario patterns formalized in terms of the unified scheme. A navigation-based algorithm is proposed, and the simulation experiments demonstrate that the method can be used to match scenario graphs.

1. Introduction

The notion of state is fundamental for many real-time applications. In conventional state-based systems, various states of the world in the discourse are usually represented in terms of isolated snapshots, while the state histories (which will be formally characterized as scenario patterns) with rich internal temporal aspects are neglected in most approaches.

Pattern recognition aims at the operation and design of technologies to pick up meaningful patterns in data [10]. While pattern classification is about putting a particular instance of a pattern in a category, the goal of pattern matching is to determine how similar a pair of patterns are [9].

Graphs have been noted as a powerful and versatile tool used in pattern recognition. As sampled in [4], applications of graph matching include document processing, image analysis, biometric identification and video analysis, etc. However, conventional graph based approaches in pattern recognition mainly concern geometric and/or spatial relationships or correspondences, where complicate temporal relationships between the modeled objects/elements have not been explicitly dealt with in most graph-based representation and matching methods in pattern recognition.

Generally speaking, temporal representation and reasoning is essential for many areas in computer science, where one is interested not only in the representation of distinct episodes of an enterprise, but also in the temporal relations among the episodes. In particular, an appropriate representation and recognition of scenario patterns is necessary for many state-based systems, where the history of states, rather than isolate states, plays an important role in solving problems including explanation, diagnosis, prediction, planning, process management, and history reconstruction, etc. For instance, in the area of medical information systems, we may use HasCold, HasFever, HasCough and HasHeadache to represent that a patient "has a cold", "has a high fever", "has a dry cough" and "has a headache", respectively. However, without the corresponding temporal relations, a pattern in terms of the collection of these 4 isolate statements (data) is in general not meaningful/helpful enough for a doctor to prescribe the right treatments. In fact, the doctor needs to know not only the patient's individual symptoms, but also the temporal relations between these symptoms, and probably, the patient's previous health records, etc.

A natural approach to represent the temporal constraints on certain states is to associate the states with time elements. For example, For example, by means of using the so-called method of temporal arguments [3], we may use *Has*(*Cold*, T_{Cold}), *Has*(*Fever*, T_{Fever}), *Has*(*Cough*, T_{Cough}) and *Has*(*Headache*, $T_{Headache}$) to represent that a patient "has a cold", "has a high fever", "has a dry cough" and "has a headache", over the corresponding times , T_{Cold}, T_{Fever}, T_{Cough} and $T_{Headache}$, respectively. Then, various temporal relations between the involved time elements will characterize different scenarios patterns. For instance, as illustrated in Figure 1 and Figure 2 respectively, the following different two sets of temporal relations, i.e.,

- *During*(T_{Fever}, T_{Cold}), *Overlaps*(T_{Cold}, T_{Cough}), *Overlaps*(T_{Cold}, $T_{Headache}$), *After*(T_{Cough}, T_{Fever}), *During*($T_{Headache}$, T_{Cough}).

- $During(T_{Headache}, T_{Cold})$, $During(T_{Fever}, T_{Cold})$, $After(T_{Fever}, T_{Headache})$, $Overlaps(T_{Fever}, T_{Cough})$.

will characterize two distinct medical scenarios pattern samples with respects to the same collection of symptoms.

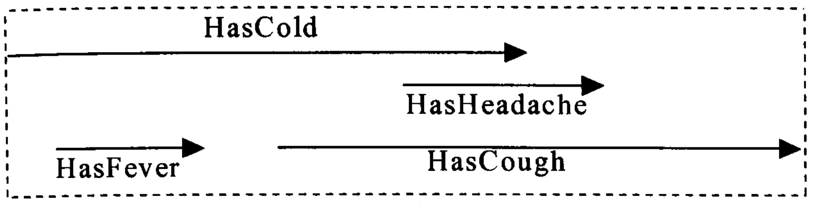


Figure 1. Sample 1

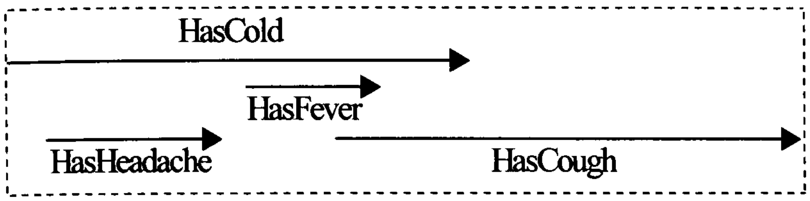


Figure 2. Sample 2

The objective of this paper is to introduce a formal framework in terms of a unified scheme for formalizing scenario patterns and the corresponding graphical representation. The formalism is presented in section 2. In section 3, a temporal network, called Scenario Graph, is introduced for graphical representation of scenario patterns. Illustrated by the experimental results, it is demonstrated in section 4 that graph-matching algorithms can be directly adopted for recognizing scenario patterns. Finally, section 5 provides a brief summary and concludes the paper.

2. The formalism

We shall use the time theory proposed in [6] as the temporal basis of the formalism. The theory takes a nonempty sort, *T*, of *primitive time elements*, with a primitive order relation Meets over time elements, and a function *Dur* from time elements to non-negative real numbers. Time element t is called a *point* if $Dur(t) = 0$; otherwise, i.e., if $Dur(t) > 0$, t is called an *interval*.

The basic set of axioms concerning the triad (*T*, *Meets*, *Dur*) is given as below:

(T1) $Meets(t_1, t_2) \wedge Meets(t_1, t_3) \wedge Meets(t_4, t_2)$
$\Rightarrow Meets(t_4, t_3)$

That is, if a time element meets two other time elements, then any time element that meets one of these two must also meets the other. This axiom is actually based on the intuition that the "place" where two time elements meet is unique and closely associated with the time elements [2].

(T2) $\forall t \exists t_1,t_2(Meets(t_1, t) \wedge Meets(t, t_2))$

That is, each time element has at least one immediate predecessor, as well as at least one immediate successor.

(T3) $Meets(t_1, t_2) \wedge Meets(t_3, t_4)$
$\Rightarrow Meets(t_1, t_4)$
$\nabla \exists t'(Meets(t_1, t') \wedge Meets(t', t_4))$
$\nabla \exists t''(Meets(t_3, t'') \wedge Meets(t'', t_2))$

Here, ∇ stands for "exclusive or". That is, any two meeting places are either identical or there is at least a time element standing between the two meeting places if they are not identical.

(T4) $Meets(t_3, t_1) \wedge Meets(t_1, t_4)$
$\wedge Meets(t_3, t_2) \wedge Meets(t_2, t_4)$
$\Rightarrow t_1 = t_2$

That is, the time element between any two meeting places is unique.

N.B. In this paper, for any two adjacent time elements, that is time elements t_1 and t_2 such that $Meets(t_1, t_2)$, we shall simply use $t_1 \oplus t_2$ to denote their ordered union. The existence of such an ordered union of any two adjacent time elements is guaranteed by axioms (T2) and (T3), while its uniqueness is guaranteed by axiom (T4).

(T5) $Meets(t_1, t_2) \Rightarrow Dur(t_1) > 0 \vee Dur(t_2) > 0$

That is, time elements with zero duration cannot meet each other.

(T6) $Meets(t_1, t_2) \Rightarrow Dur(t_1 \oplus t_2) = Dur(t_1) + Dur(t_2)$

That is, the "ordered union" operation over time elements is consistent with the conventional "addition" operation over the duration assignment function, i.e., *Dur*.

Analogous to those introduced by Allen [1], other order relations between time elements can be derived in terms of the primitive relation *Meets*, as below:

$Equal(t_1, t_2) \Leftrightarrow \exists t',t''(\quad Meets(t', t_1) \wedge Meets(t', t_2) \wedge Meets(t_1, t'') \wedge Meets(t_2, t''))$
$Before(t_1, t_2) \Leftrightarrow \exists t(Meets(t_1, t) \wedge Meets(t, t_2))$
$Overlaps(t_1, t_2) \Leftrightarrow \exists t,t_3,t_4(t_1 = t_3 \oplus t \wedge t_2 = t \oplus t_4)$
$Starts(t_1, t_2) \Leftrightarrow \exists t(t_2 = t_1 \oplus t)$
$During(t_1, t_2) \Leftrightarrow \exists t_3,t_4(t_2 = t_3 \oplus t_1 \oplus t_4)$
$Finishes(t_1, t_2) \Leftrightarrow \exists t(t_2 = t \oplus t_1)$
$After(t_1, t_2) \Leftrightarrow Before(t_2, t_1)$
$Overlapped\text{-}by(t_1,t_2) \Leftrightarrow Overlaps(t_2, t_1)$
$Started\text{-}by(t_1, t_2) \Leftrightarrow Starts(t_2, t_1)$
$Contains(t_1, t_2) \Leftrightarrow During(t_2, t_1)$
$Finished\text{-}by(t_1,t_2) \Leftrightarrow Finishes(t_2, t_1)$,
$Met\text{-}by(t_1,t_2) \Leftrightarrow Meets(t_2, t_1)$

In this paper, we shall use the term *fluenst* to represent Boolean-valued propositions whose truth-values are dependent on time. We shall denote the set of all the fluents as *F*, and use a "global predicate" [8]: *Holds(f, t)*, to state that fluent *f* holds true with respect to time t, provided that:

(H1) $Holds(f, t) \Leftrightarrow \forall t'(Part(t', t) \Rightarrow Holds(f, t'))$

Here, $Part(t_1, t_2)$ denotes that time t_1 is a part (not necessarily proper) of time t_2 [1, 6]. That is:

$Part(t_1, t_2) \Leftrightarrow \quad Equal(t_1, t_2) \vee Starts(t_1, t_2)$
$\vee During(t_1, t_2) \vee Finishes(t_1, t_2)$

Therefore, (H1) states that, if a fluent holds true throughout an interval if and only if it holds true over/at every part of the interval.

(H2) $Holds(f, t_1) \wedge Holds(f, t_2) \wedge Meets(t_1, t_2)$
$\Rightarrow Holds(f, t_1 \oplus t_2)$

That is, if a fluent holds true over two adjacent time elements respectively, then it also holds true over their ordered union.

A *state* of the world in the discourse is defined as a collection of fluents. The set of states is denoted as S. We shall use $Belongs(f, s)$ to represent fluent f belongs to state s [7]:

(B1) $\exists s\, \forall f(\neg Belongs(f, s))$

That is, there exists a state which is an empty collection of fluents.

(B2) $\neg Belongs(f, s) \vee \neg Belongs(not(f), s)$

That is, any state cannot contain both a fluent and its negation.

(B3) $\neg Belongs(not(f_1), s_1)$
$\Rightarrow \exists s_2 \forall f_2(Belongs(f_2, s_2) \Leftrightarrow (Belongs(f_2, s_1) \vee f_1 = f_2)))$

That is, any fluent can be added to an existing state to form a new state, as long as the state does not contain the negation of the fluent.

(B4) $s_1 = s_2 \Leftrightarrow \forall f(Belongs(f, s_1) \Leftrightarrow Belongs(f, s_2))$

Without confusion, we also use $Holds(s, t)$ to denote that state s holds for time t, provided that every fluent f belongs to state s holds for time t:

(H3) $Holds(s, t) \Leftrightarrow \forall f(Belongs(f, s) \Rightarrow Holds(f, t))$

In addition, we introduce two binary operators, *Union* and *Intersection*, so that $Union(s_1, s_2)$ and $Intersection(s_1, s_2)$ denote the *union*, and the *intersection*, of states s_1 and s_2, respectively:

(B5) $Belongs(f, Union(s_1, s_2))$
$\Leftrightarrow Belongs(f, s_1) \vee Belongs(f, s_2)$

(B6) $Belongs(f, Intersection(s_1, s_2))$
$\Leftrightarrow Belongs(f, s_1) \wedge Belongs(f, s_2)$

Any given scenario, *st*, is formalized in terms of a unified scheme, represented as a quadruple:

$st = <State^{st}, Holds^{st}, Meets^{st}, Dur^{st}>$, such that
$State^{st} = \{s^{st}{}_i \mid s^{st}{}_i \in S, \mathrm{i} = 1, \ldots, \mathrm{m}\}$,
$Holds^{st} = \{Holds(s^{st}{}_i, t^{st}{}_i) \mid \quad t^{st}{}_i \in T, 1 \le \mathrm{i} \le \mathrm{m}\}$
$Meets^{st} = \{Meets(t^{st}{}', t^{st}{}'') \mid \text{for some } t^{st}{}', t^{st}{}'' \in T^{st}\}$
$Dur^{st} = \{Dur(t) = r \mid \text{for some } t \in T^{st}, r \in R\}$

where T^{st} is the minimal subset of T closed under the following rules:

$t^{st}{}_i \in T^{st}, \mathrm{i} = 1, \ldots, \mathrm{m}.$
$t \in T^{st} \Leftrightarrow \exists t' \in T^{st}(Meets(t, t') \vee Meets(t', t))$

3. Scenario Graphs

In [5], a graphical representation for expressing temporal knowledge has been introduced in terms of a directed and partially weighted graph. It can be extended to express scenarios presented in the unified structure as introduced in section 2. In fact, a given scenario *st* can be represented in terms of a temporal network, defined as a directed, partially weighted/labeled simple graph G^{st}, called *Scenario Graph*, where:

- Each time element t in T^{st} is denoted as a directed arc of the graph labeled by t that is bounded by a pair of nodes, which are called the *tail-node*, and the *head-node*, of the arc, respectively.
- Each relation $Meets(t_i, t_j)$ in $Meets^{st}$ is represented by means of merging the *head-node* of t_i and the *tail-node* of t_j as a common node, of which t_i is an in-arc and t_j is an out-arc, respectively (see Figure 3). In this case, arc t_i is said to be adjacent to arc t_j.
- Each formula $Holds(s^{st}{}_i, t^{st}{}_i)$ in $Holds^{st}$ is represented by means of simply adding $s^{st}{}_i$ as an additional label to the arc labeled by the corresponding $t^{st}{}_i$. For any time element t in T^{st}, if there is no *Holds* knowledge, it will be labeled by the empty state {}.
- Each piece of duration knowledge $Dur(t) = r$ in Dur^{st} is expressed as a real number, r, alongside the corresponding arc t.

For scenario graph G^{st}, we define a $|T^{st}|$-by-$|T^{st}|$ Meets-adjacency matrix M^{st}, where

$$M(i,j) = \begin{cases} 1, & if\, Meets(t_i, t_j) \\ 0, & otherwise \end{cases}$$

4. A navigation-based algorithm

In what follows, by assuming the set of fluents F is finite, i.e., $F = \{f_1, \ldots, f_n\}$, we shall propose a navigation-based algorithm for matching scenarios graphs.

4.1 The algorithm

Given two scenarios, st_1 and st_2, we assume $|T_1^{st}| \le |T_2^{st}|$. We use ϕ to denote a one-to-one function from $\{1, \ldots, |T_1^{st}|\}$ to $\{1, \ldots, |T_2^{st}|\}$, and compute the following similarity degrees:

- Similarity of graph size:

$$sim_{size}(st_1, st_2) = \frac{|T_1^{st}|}{|T_2^{st}|}$$

- Similarity of *Holds* relations:

$$sim_{Holds}(st_1, st_2, \phi) = \frac{1}{|T_1^{st}|} \sum_{i=1}^{|T_1^{st}|} \frac{|Intersection(s_1^i, s_2^{\phi(i)})|}{|Union(s_1^i, s_2^{\phi(i)})|}$$

- Similarity of duration assignments:

$$sim_{Dur}(st_1,st_2,\phi)=2\frac{\sum_{i}^{|T_1^{st}|}\left|Dur(t_1^i)*Dur(t_2^{\phi(i)})\right|}{\sum_{i=1}^{|T_1^{st}|}\left|Dur(t_1^i)\right|^2+\left|Dur(t_2^{\phi(i)})\right|^2}$$

- Similarity of *Meets* relations:

For scenarios st_1 and st_2, we use M_1 and M_2 to denote their corresponding Meets-adjacency matrices respectively, and, with respect to ϕ, we derive a $|T_1^{st}|*|T_1^{st}|$ matrix M_2^{ϕ} from M_1 and M_2, such that:

$$M_2^{\phi}(i,j)=M_2(\phi(i),\phi(j))$$

Then, the similarity of Meets relations between scenarios st_1 and st_2, with respect to function ϕ, is defined as:

$$sim_{Meets}(st_1,st_2,\phi)=2\frac{\left|<M_1,M_2^{\phi}>\right|}{\left\|M_1\right\|^2+\left\|M_2^{\phi}\right\|^2}$$

where

$$<M_1,M_2^{\phi}>=\sum_{i=1}^{|T_1^{st}|}\sum_{j=1}^{|T_1^{st}|}M_1(i,j)*M_2^{\phi}(i,j)$$

- The overall similarity between scenarios st_1 and st_2, with respect to function ϕ, is defined as:

$$\begin{aligned}sim(st_1,st_2,\phi)&=sim_{size}(st_1,st_2)\\&*(w_1*sim_{Holds}(st_1,st_2,\phi)+w_2*sim_{Dur}(st_1,st_2,\phi)\\&+w_3*sim_{Meets}(st_1,st_2,\phi))/(w_1+w_2+w_3)\end{aligned}$$

- Finally, the similarity between scenarios st_1 and st_2 is defined as:

$$sim(st_1,st_2)=\max_{\phi} sim(st_1,st_2,\phi)$$

4.2 Experimental results

The algorithm has been implemented in MatLab. What follows describes some experiments conducted, where the corresponding weights taken in the algorithm are: $w_1 = w_2 = 0.25$ and $w_3 = 0.5$.

4.2.1. Test 1. As shown in Figure 3 – Figure 9, for the 7 pairs of scenario graphs (ST_1 and ST_2), the corresponding results computed from the algorithm provide a well stable similarity measurement as expected.

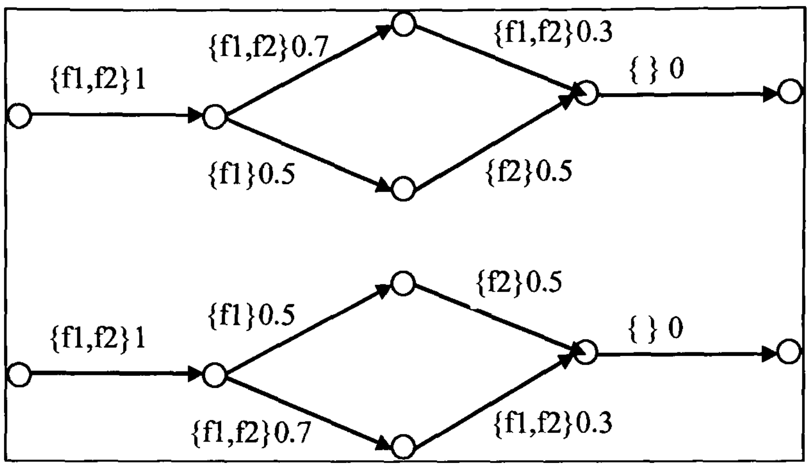


Figure 3. Sim(st$_{3,1}$, st$_{3,2}$) = 1

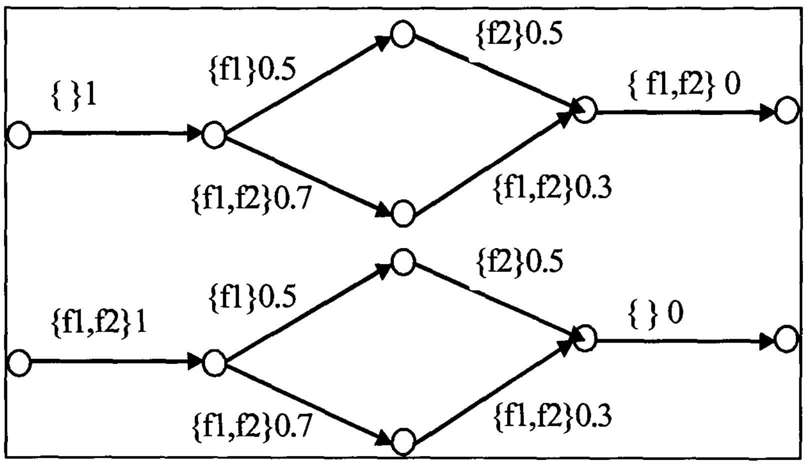


Figure 4. Sim(st$_{4,1}$, st$_{4,2}$) =0.8333

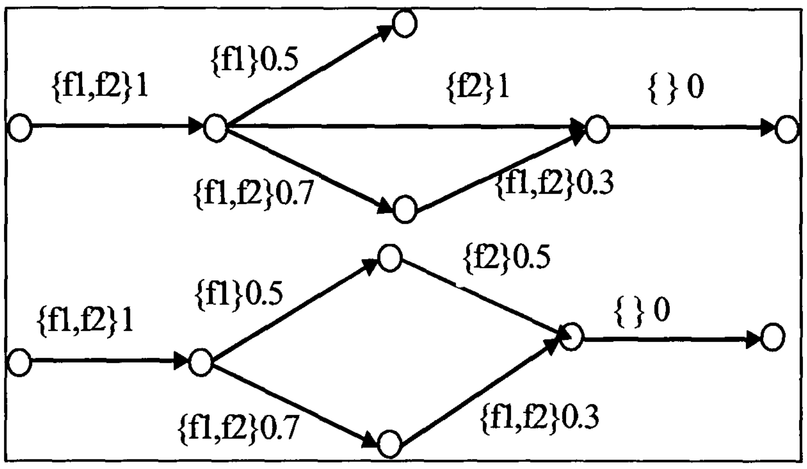


Figure 5. Sim(st$_{5,1}$, st$_{5,2}$) =0.9093

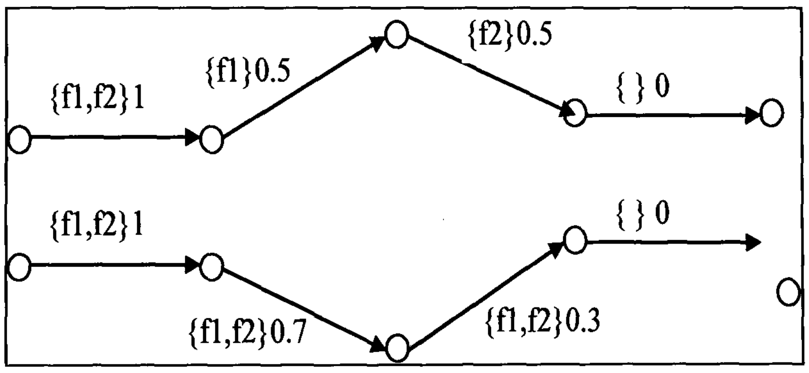


Figure 6. Sim(st$_{6,1}$, st$_{6,2}$) =0.9310

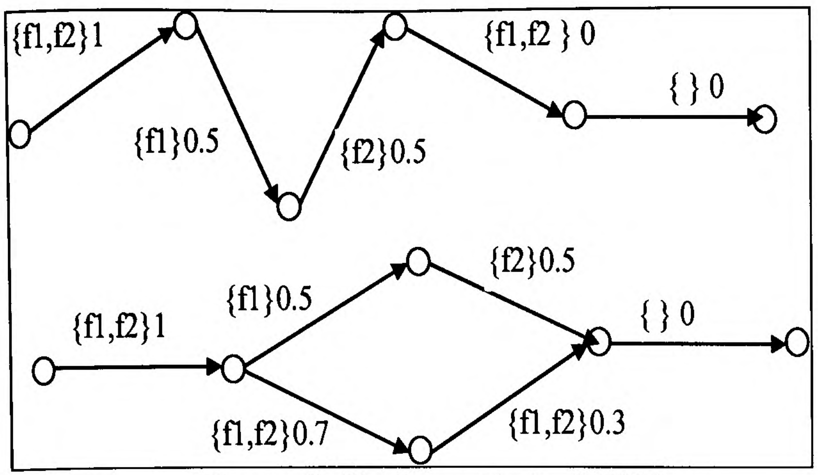


Figure 7. Sim($st_{7,1}$, $st_{7,2}$) =0.7231

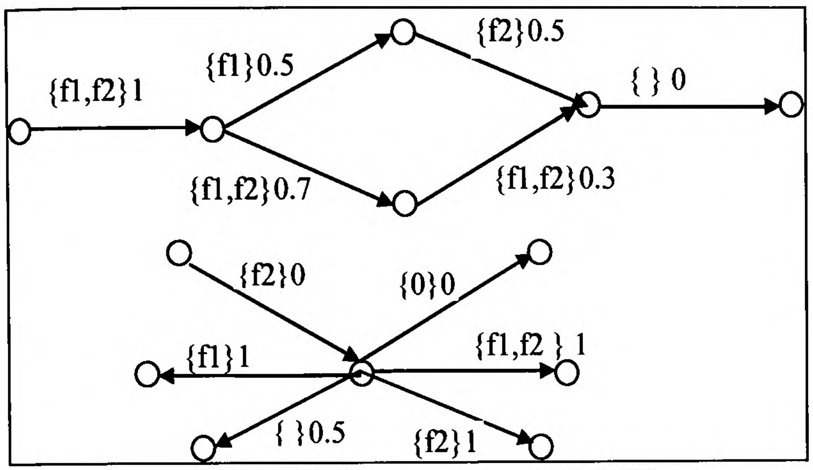


Figure 8. Sim($st_{8,1}$, $st_{8,2}$) =0.5429

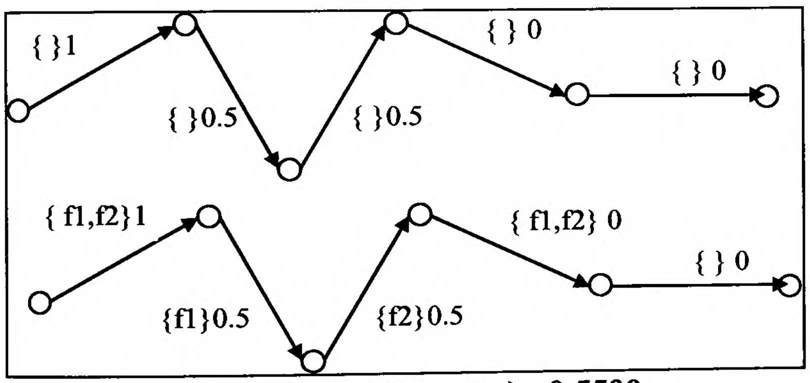


Figure 9. Sim($st_{9,1}$, $st_{9,2}$) =0.5500

4.2.2. Test 2. Figures 10 – Figure 12 represent three graphs corresponding to real-world scenarios in medical records, respectively:

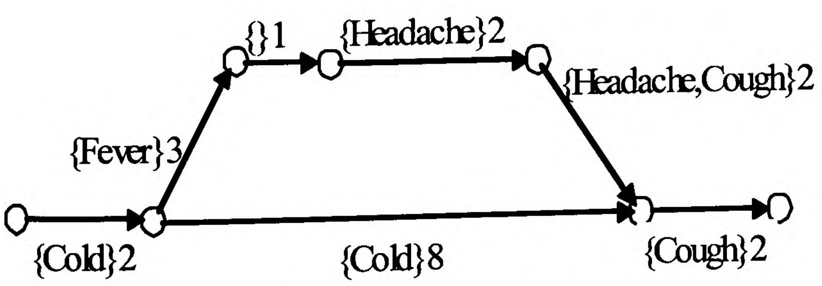


Figure 10. Scenario 1

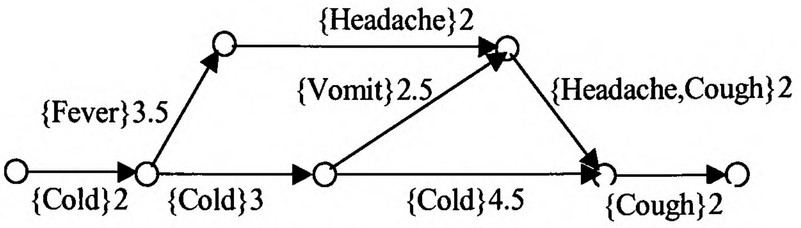


Figure 11. Scenario 2

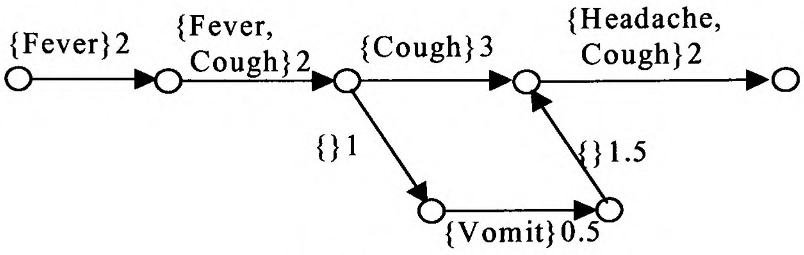


Figure 12. Scenario 3

The computed similarities among these three scenarios are listed in Table 1.

Table 1. Similarity list

	Scenario 1	Scenario 2	Scenario 3
Scenario 1	1	0.7211	0.6878
Scenario 2	0.7211	1	0.6042
Scenario 3	0.6878	0.6042	1

Taking the above given three scenario graphs as the model graphs, we modified each of them in various ways to obtain a series of modified scenario graphs. Figure 13 – Figure 15 show the relationships between the similarity and the corresponding noise/modification, where each line represent a collection of scenario graphs including one of the 3 model scenarios and the series of the corresponding modified ones.

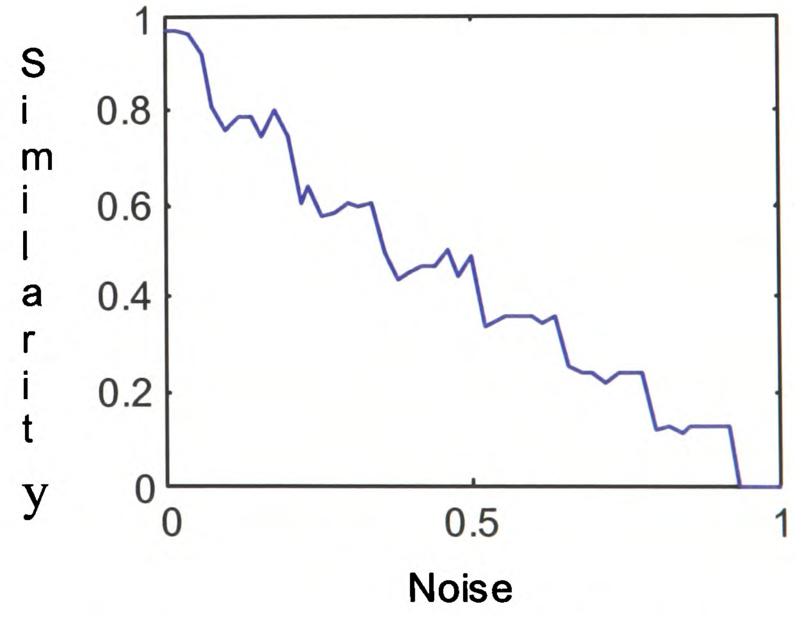


Figure 13. Scenario 1 and its modifications

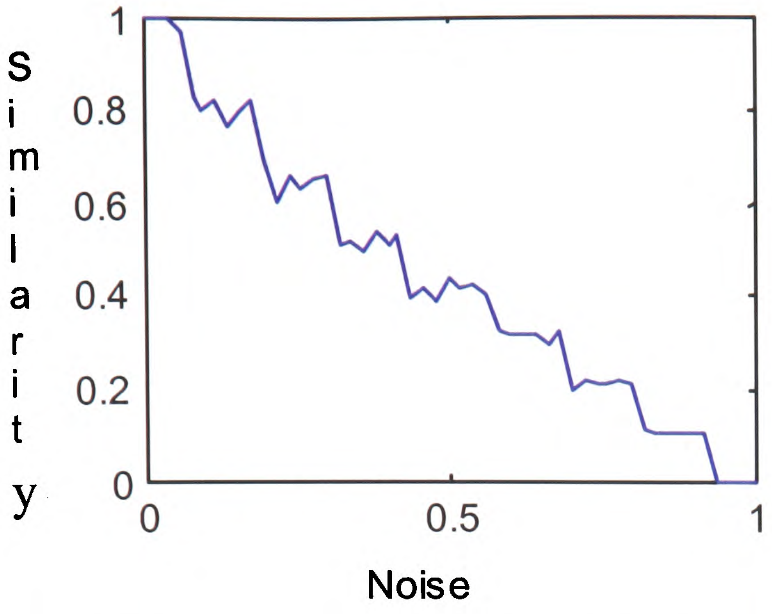


Figure 14. Scenario 2 and its modifications

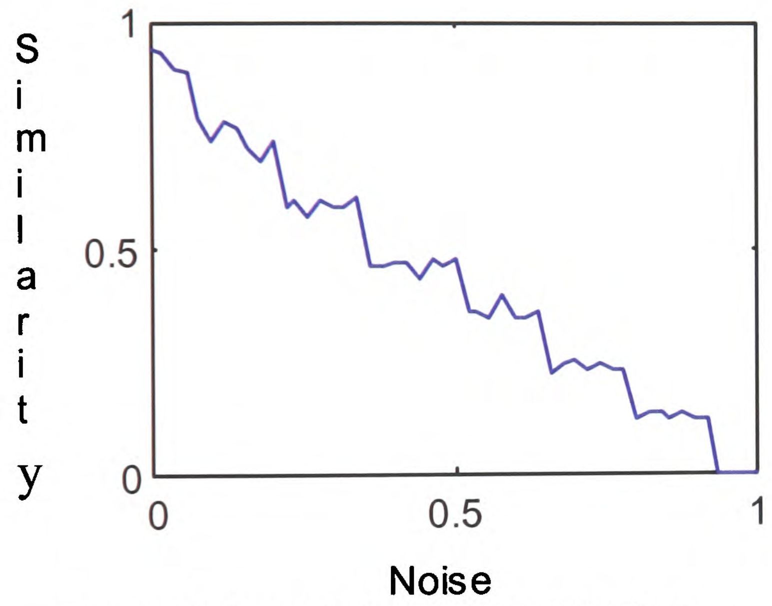


Figure 15. Scenario 3 and its modifications

Finally, Figure 16 as shown below provides an overview of the similarity/dissimilarity among the three collections of scenario graphs generated in Test 2.

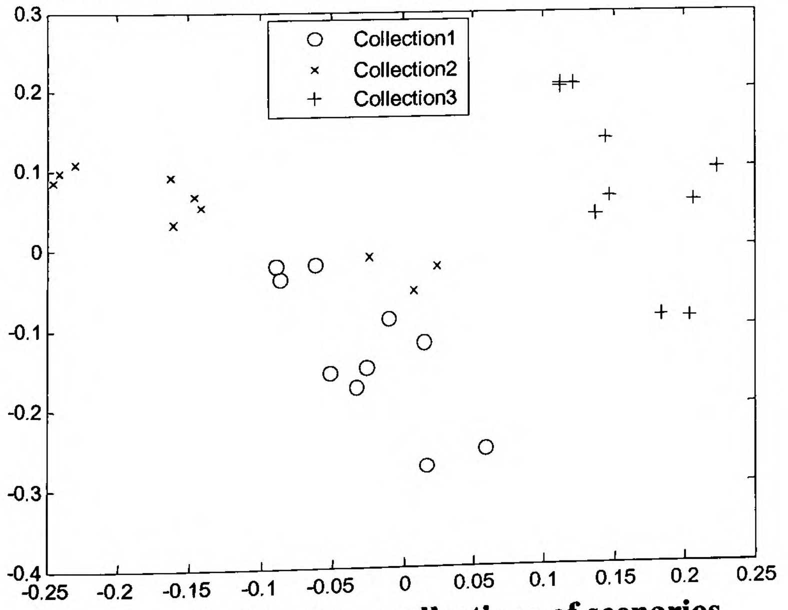


Figure 16. The three collections of scenarios

It is easy to see that the similarity defined here reflects the conventional idea of edit distance in graph matching. In other words, the more similar a pair of two scenario graphs is, the closer they are to each other. Therefore, given some certain criteria in terms of the similarity/dissimilarity, it will be straightforward to use the proposed algorithm to cluster scenario graphs into various groups.

5. Conclusions

In this paper, we have introduced a framework for representing and recognizing scenario patterns with rich internal temporal aspects. The framework consists of a unified scheme for scenario formalization and a temporal network for graphical representation. It is shown that scenario pattern recognition and matching can be simply transformed into graph matching. However, due to the embedded checking of all permutations, the computational complexity associated with the proposed navigation-based algorithm is actually NP-hard. On the other hand, it is easy to see that, by means of re-indexing the arcs of any given scenario graph, the corresponding *Meets*-adjacent matrix can be turned out to be strictly upper-diagonal. In other words, scenario graphs in general are quite regularly sparse. Therefore, it is believed that exploiting such kind sparsity can lead to more efficient algorithms/methods to improve the corresponding computational complexity. This remains the main target as for the future work. The future work will also concern real world applications such as medical treatments and weather forecasting.

6. References

[1] J. Allen, "Maintaining knowledge about temporal intervals", *Communications of the ACM* 26 (11), (1983) pp.832-843.

[2] J. Allen, and P. Hayes, "Moments and Points in an Interval-based Temporal-based Logic", *Computational Intelligence* 5, (1989) pp.225-238.

[3] B. Haugh, "Non-Standard Semantics for the Method of Temporal Arguments", *Proceedings of 10th IJCAI* (1987) pp.449-455.

[4] C. Irniger, and H. Bunke, "Theoretical Analysis and Experimental Comparison of Graph Matching Algorithms for Database Filtering", *Proceedings of* 4^{th} *IAPR International Workshop on Graph Based Representations in Pattern Recognition*, York, U.K. (2003) pp.118-129.

[5] B. Knight, and J. Ma, "A General Temporal Model Supporting Duration Reasoning", *Artificial Intelligence Communication*, Vol.5(2), (1992) pp.75-84.

[6] J. Ma, and B. Knight, "A General Temporal Theory", *the Computer Journal* 37(2), (1994) pp.114-123.

[7] M. Shanahan, "A Circumscriptive Calculus of Events", *Artificial Intelligence* 77, (1995) pp.29-384.

[8] Y. Shoham, "Temporal Logics in AI: Semantical and Ontological Considerations", *Artificial Intelligence* 33, (1987) pp.89-104.

[9] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Second Edition, Academic Press (2003).

[10] D. Tveter, *The Pattern Recognition Basis of Artificial Intelligence*, Wiley-IEEE Computer Society Press (1998).

APPENDIX C MATCHING CASE HISTORY PATTERNS IN CASE-BASED REASONING

Matching Case History Patterns in Case-Based Reasoning

Guoxing Zhao[1], Bin Luo[2] and Jixin Ma[1]

[1]School of Computing and Mathematical Sciences, University of Greenwich, U.K.
{g.Zhao,j.ma}@gre.ac.uk
[2]School of Computer Science and Technology, AnHui University, China
luobin@ahu.edu.cn

Abstract. This paper introduces a mechanism for representing and recognizing case history patterns with rich internal temporal aspects. A case history is characterized as a collection of elemental cases as in conventional case-based reasoning systems, together with the corresponding temporal constraints that can be relative and/or with absolute values. A graphical representation for case histories is proposed as a directed, partially weighted and labeled simple graph. In terms of such a graphical representation, an eigen-decomposition graph matching algorithm is proposed for recognizing case history patterns.

1 Introduction

The notion of *case* is fundamental for many real life applications. In conventional case-based systems, various cases in the world under consideration are usually represented as isolated episodes. Generally speaking, temporal representation and reasoning is essential for many areas in computer science, where one is interested not only in the representation of distinct episodes of an enterprise, but also in the temporal relations among the episodes. In particular, appropriate temporal representation and reasoning is fundamental for many case-based systems, where the history of cases, rather than isolated cases, plays an important role in solving problems including explanation, diagnosis, prediction, planning, process management, and history reconstruction, etc. For instance, in the area of medical information systems, the patients' medical histories are obviously very important. In fact, to prescribe the right treatments, the doctor needs to know not only the patients' current status, but also their previous health records. Similarly, in weather forecasting, without a good understanding of climate phenomena based on past observations, the weather expert cannot make good predictions of the future.

This research is supported in part by National Nature Science Foundation of China (No.60375010)

APPENDIX C MATCHING CASE HISTORY PATTERNS IN CASE-BASED REASONING

Despite the fact that temporal representation and temporal reasoning have been neglected in most conventional case-based reasoning systems which only address snapshot episodes, a few interesting approaches have been proposed to incorporate the temporal concepts into isolated elemental cases. Examples of these are that of Nakhaeizadeh [1], of Branting and Hasting [2], of Jaczynski and Trousse [3], of Hansen [4], and of Jare, Adamodt and Skalle [5]. The underlying time models employed in most of these systems are point-based, and therefore, it is required that absolute time points [1-4], or intervals delimited by a pair of points [5], must be associated with the time-dependent statement being addressed. However, there are many applications in which there may be just some relative temporal knowledge about the time-depended statements to hand, where their precise time characters such as the exact starting and finishing time are not available (e.g., “John ran 3 miles yesterday morning”, “John arrived at the office before Mary went to home”, etc.).

Pattern recognition aims at the operation and design of technologies to pick up meaningful patterns in data [6]. While pattern classification is about putting a particular instance of a pattern in a category, the goal of pattern matching is to determine how similar a pair of patterns are [7]. The objective of this paper is to introduce a mechanism for case history representation and recognition. Section 2 presents the formalism, including: the temporal basis which allows expression of both absolute time values and relative temporal relations; a formal characterization of fluents and elemental cases; and two equivalent schemas for case history representation. A network, called *Case History Graph*, given in terms of a directed, partially weighted and labeled simple graph, is introduced in section 3 for graphical representation of case histories. In section 4, an eigen-decomposition algorithm is proposed for matching case history graphs, where some illustrated experimental results. Finally, section 5 provides the conclusions.

2 The formalism

We shall describe the formalism in terms of a many-sorted reified logic with equality [8], consisting of four disjoint sorts objects *T*, *F*, *C* and *H*, called *time elements*, *fluents*, *elemental cases* and *case histories*, respectively.

Firstly, each time element is defined to be in one of the following four forms:

$$(p_1, p_2) = \{p \mid p_1, p_2, p \in R \wedge p_1 < p < p_2\}$$

$$[p_1, p_2) = \{p \mid p_1, p_2, p \in R \wedge p_1 \leq p < p_2\}$$

$$(p_1, p_2] = \{p \mid p_1, p_2, p \in R \wedge p_1 < p \leq p_2\}$$

$$[p_1, p_2] = \{p \mid p_1, p_2, p \in R \wedge p_1 \leq p \leq p_2\}$$

where R is the set of real numbers, and ≤, < and = are the conventional order relations over real numbers.

In this paper, p_1 and p_2 in the above shall be called the left-bound and right-bound of time element t, respectively. The absolute values as for the left and/or right bounds of some time elements might be unknown. In this case, real number variables are used for expressing relative relations to other time elements.

If the left-bound and right-bound of time element t are the same, we shall call t a time point, otherwise t is called a time interval. Without confusion, we shall take time element $[p, p]$ as identical to p. Also, if a time element is not specified as open or closed at its left (right) bound, we shall use "<" instead of "(" and "[" as for its left bracket; similarly, we shall use ">" instead of ")" and "]" as for its right bracket. In addition, we define the duration of a time element *t*, *Dur*(*t*), as the distance between its left bound and right bound:

$$t = \langle p_1, p_2 \rangle \Leftrightarrow Dur(t) = p_2 - p_1$$

Following Allen's terminology [9], we shall use *Meets* to denote the immediate predecessor order relation over time elements, defined by:

$$\begin{aligned} Meets(t_1, t_2) \Leftrightarrow \exists p_1, p, p_2 \in R(\quad & t_1 = (p_1, p) \wedge t_2 = [p, p_2) \\ & \vee\, t_1 = [p_1, p) \wedge t_2 = [p, p_2) \\ & \vee\, t_1 = (p_1, p) \wedge t_2 = [p, p_2] \\ & \vee\, t_1 = [p_1, p) \wedge t_2 = [p, p_2] \\ & \vee\, t_1 = (p_1, p] \wedge t_2 = (p, p_2) \\ & \vee\, t_1 = [p_1, p] \wedge t_2 = (p, p_2) \\ & \vee\, t_1 = (p_1, p] \wedge t_2 = (p, p_2] \\ & \vee\, t_1 = [p_1, p] \wedge t_2 = (p, p_2]) \end{aligned}$$

It is easy to see that the intuitive meaning of *Meets*(t_1, t_2) is that, on the one hand, while t_1 is an "earlier" time element compared with t_2, there are no other time elements standing between them; on the other hand, time elements t_1 and t_2 don't overlap each other (i.e., they don't have any part in common, not even a point).

Analogous to those introduced by Allen [9], other order relations between time elements can be derived in terms of the primitive relation *Meets*, including *Equals*, *Before*/*After*, *Meets*/*Met-by*, *Overlaps*/*Overlapped-by*, *Starts*/*Started-by*, *During*/*Contains* and *Finishes*/*Finished-by* [10]. As shown in [11], such a time model as adapted describe here has all the expressive power and convenience of the approach that treats intervals as primitive [9, 10, 12]. Specially, since the open/closed nature of a time element may be unspecified, it can overcome the disadvantage of conventional point-based approaches in representing possibly incomplete temporal knowledge, and bypass some historical puzzles such as the so-called *Dividing Instant Problem* [13].

In what follows in this paper, we shall use *TR* to denote the set of these 13 exclusive temporal order relations.

Secondly, a *fluent* is defined a statement (or proposition) whose truth-value is dependent on times. The sort of fluents *F* is characterized as the minimal set closed under the following rules:

$$f_1, f_2 \in F \Rightarrow f_1 \vee f_2 \in F$$

$$f \in F \Rightarrow \mathrm{not}(f) \in F$$

APPENDIX C MATCHING CASE HISTORY PATTERNS IN CASE-BASED REASONING

In order to associate a fluent with a time element, we shall use a global-predicate [8, 14], *Holds*(f, t), to state that fluent f holds true over time t:

$$Holds(f, <p_1, p_2>) \Leftrightarrow \forall p_3,p_4 \in R(p_1 \leq p_3 \wedge p_4 \leq p_2 \quad \Rightarrow Holds(f, <p_3, p_4>))$$

$$Holds(f, <p_1, p_2>) \wedge Holds(f, <p_2, p_3>) \wedge Meets(<p_1, p_2>, <p_2, p_3>) \\ \Rightarrow Holds(f, <p_1, p_3>)$$

Thirdly, an *elemental case* is defined as a collection of fluents. We shall use *Belongs*(f, c) to denote that fluent f belongs to case c [15].

Without confusion, we also use *Holds*(c, t) to state that case c holds over time t, provided that every fluent f belongs to case c holds true over time t:

$$Holds(s, t) \Leftrightarrow \forall f(Belongs(f, s) \Rightarrow Holds(f, t))$$

In addition, we introduce two binary operators, *Union* and *Intersection*, over the sort of elemental cases C, so that *Union*(c_1, c_2) and *Intersection*(c_1, c_2) denote the *union*, and the *intersection*, of case c_1 and case c_2, respectively:

$$Belongs(f, Union(c_1, c_2)) \Leftrightarrow Belongs(f, c_1) \vee Belongs(f, c_2)$$

$$Belongs(f, Intersection(c_1, c_2)) \Leftrightarrow Belongs(f, c_1) \wedge Belongs(f, c_2)$$

Finally, a *case history h*, can be formalized in terms of one of the following two equivalent schemas:

In the first schema, *Schema I*, a case history is represented as a quadruple <$Case^h$, $Holds^h$, $Relation^h$, Dur^h>, where $Case^h$ is a collection of elemental cases, $Holds^h$ is a collection of *Holds* formulae, $Relation^h$ is a collection of temporal order relations, and Dur^h is a collection of duration knowledge. That is:

Schema I

$$\begin{aligned}
&h = <Case^h, Holds^h, Relation^h, Dur^h> \\
&Case^h = \{c^h{}_i \mid c^h{}_i \in C, i = 1, \ldots, m\} \\
&Holds^h = \{Holds(c^h{}_i, t^h{}_i) \mid t^h{}_i \in T, 1 \leq i \leq m\} \\
&Relatio^h = \{Relation_{1,2}(t^h{}_1, t^h{}_2) \mid \text{for some } t^h{}_1, t^h{}_2 \in T^h, Relation_{1,2} \in TR\} \\
&Dur^h = \{Dur(t) = r \mid \text{for some } t \in T^h, r \in R\}
\end{aligned}$$

where T^h is the minimal subset of T closed under the following rules:

$$\begin{aligned}
&t^h{}_i \in T^h, i = 1, \ldots, m; \\
&t \in T^h \Leftrightarrow \exists t' \in T^h(Meets(t, t') \vee Meets(t', t))
\end{aligned}$$

It is for the reason of general treatment that the temporal relationships presented in the above *Schema I* are given in the form of a collection of order relations each of which can be any one of those 13 in TR, that is, *Equal*, *Before*, *After*, *Meets*, *Overlaps*, *Overlapped-by*, *Met-by*, *Starts*, *Started-by*, *During*, *Contains*, *Finishes* and *Finished-by*. However, since all these order relations can be derived from the single *Meets* relation, we shall have another schema, *Schema II*, which is equivalent to the *Schema I*:

Schema II

$$h = <Case^h, Holds^h, Meets^h, Dur^h>$$
$$Case^h = \{c^h{}_i \mid c^h{}_i \in C, i = 1, \ldots, m\}$$
$$Holds^h = \{Holds(c^h{}_i, t^h{}_i) \mid t^h{}_i \in T, 1 \leq i \leq m\}$$
$$Meets^h = \{Meets(t^h{}_1, t^h{}_2) \mid \text{for some } t^h{}_1, t^h{}_2 \in T^h\}$$
$$Dur^h = \{Dur(t) = r \mid \text{for some } t \in T^h, r \in R\}$$

3 Graphical Representation of Scenarios

In [16], a graphical representation for expressing temporal knowledge has been introduced in terms of a directed and partially weighted graph. It can be extended to express case histories presented in the *Schema II* as introduced in section 2. In fact, a given case history h can be represented in terms of a temporal network, defined as a directed, partially weighted/labeled simple graph G^h, called *Case History Graph*, where:

- Each time element t in T^h is denoted as a directed arc of the graph labeled by t that is bounded by a pair of nodes, which are called the *tail-node*, and the *head-node*, of the arc, respectively.
- Each relation $Meets(t_i, t_j)$ in $Meets^h$ is represented by means of merging the *head-node* of t_i and the *tail-node* of t_j as a common node, of which t_i is an in-arc and t_j is an out-arc, respectively (see Fig. 1).
- Each formula $Holds(c^h{}_i, t^h{}_i)$ in $Holds^h$ is represented by means of simply adding $c^h{}_i$ as an additional label to the arc labeled by the corresponding $t^h{}_i$. For any time element t in T^h, if there is no *Holds* knowledge, it will be labeled by the empty state {}.
- Each piece of duration knowledge $Dur(t) = r$ in Dur^h is expressed as a real number, r, alongside the corresponding arc t.

In what follows, we shall simply assume $|F| = n$. Corresponding to case history graph G^h with m nodes, we define a m*m-matrix M^h, named the characteristic matrix, where $M^h(u, v)$ is a $(n+1)$-dimension vector $l_{uv} \in R^{n+1}$, such that:

(a) For any adjacent pair of nodes u and v in G^h, if (u, v) is an arc representing time element t, then $l_{uv}(k) = 1$ if Holds(f_k, t), otherwise $l_{uv}(k) = 0$, $1 \leq k \leq n$; and $l_{uv}(n+1) = Dur(t)$.

(b) For any non-adjacent pair of nodes u and v in G^h, $l_{uv} = <w, w, \ldots, w>$, where w is a negative real number, which will be use to adjust the edit-distance of deleting operations in graph matching process.

In this paper, we shall use M^h_k to denote the matrix whose u-v-entry is the k-th element of u-v-entry in M^h.

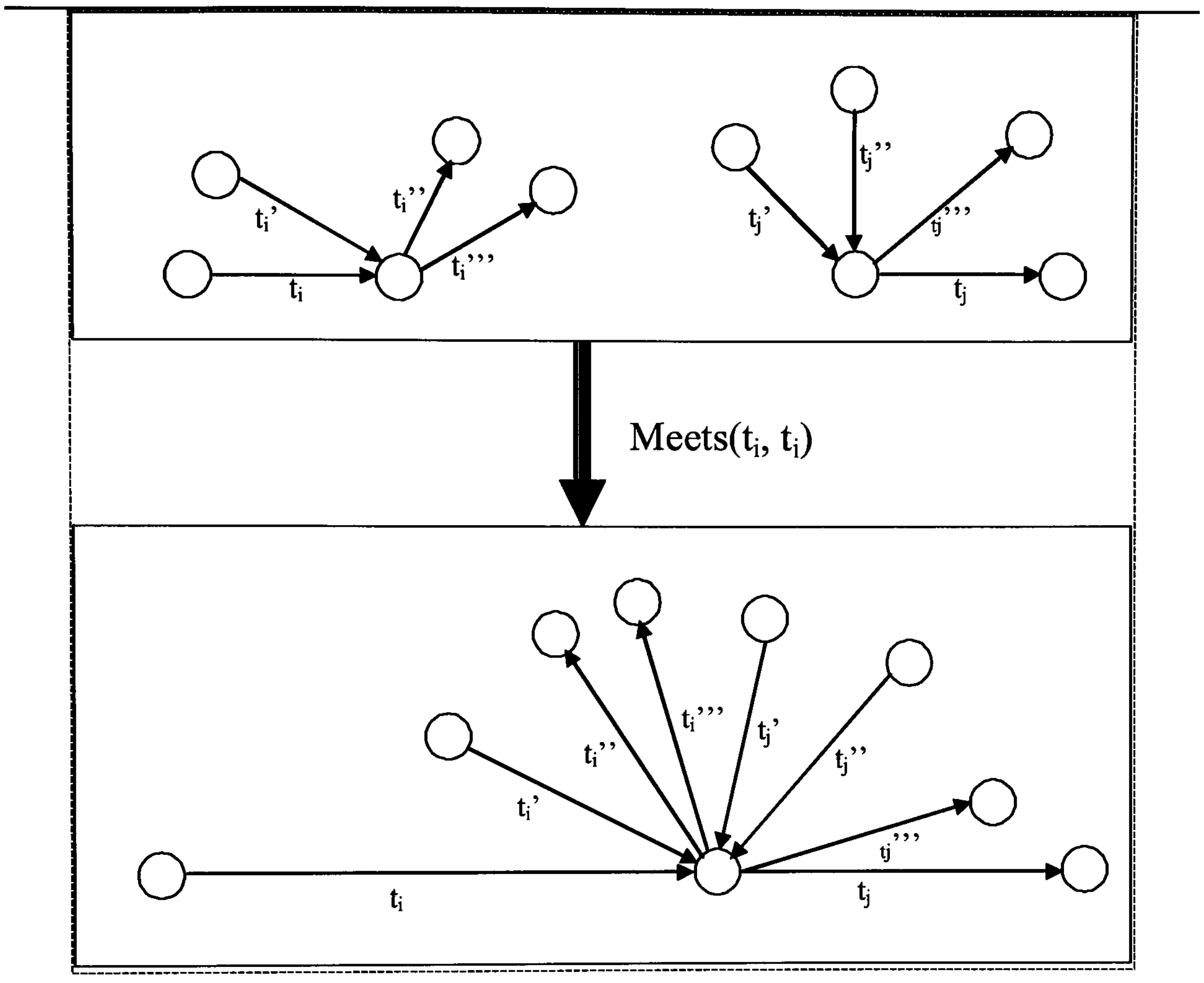


Fig. 1. Merging the *head-node* of t_i and the *tail-node* of t_j as a common node if *Meets*(t_i, t_j)

4 Eigen-decomposition graph matching algorithm

Spectral graph theory is a branch in mathematics which aims to characterize the properties of graphs using the eigen-values and eigen-vectors of the adjacency matrix or the closely related Laplacian matrix [17]. However, conventional spectral-based approaches usually deal only with symmetric real matrices, where the adjacency matrices of directed graphs, like the case history graph introduced in this paper, are in general asymmetric. Moreover, the entries of the characteristic matrix of a given case history graph are (n+1)-dimension vectors, rather than single real values as in conventional spectral-based models. In what follows, we shall extend the so-called eigen-decomposition graph matching algorithm, proposed by Umeyama [18], to match case history graphs.

APPENDIX C MATCHING CASE HISTORY PATTERNS IN CASE-BASED REASONING

4.1 Definition of case history similarity

In what follows in this paper, if M is a complex number matrix, we shall use $\|M\|$ to denote the Frobenius-norm of M, and $|M|$ to denote the matrix whose elements are the module of the corresponding elements of M.

Given two *case histories* h_1 and h_2, assume the characteristic matrices are M^{h1} and M^{h2}, with size m_1*m_1 and m_2*m_2. Without losing the generality, we assume $m_1 = m_2 = m$. In fact, if $m_1 < m_2$, we can simply add $m_2 - m_1$ isolated dummy nodes to graph G^{h1} to get an extended graph $G^{h1'}$, whose characteristic matrix $M^{h1'}$ will have the same size as that of M^{h2}, i.e., m_2*m_2. Similar treatment can be applied to the case where $m_2 < m_1$.

The similarity degree between h_1 and h_2 is therefore universally defined by:

$$sim(h_1,h_2) = 1 - \frac{\min\limits_{p\in perm(m)} \sum\limits_{k=1}^{n+1} \left\| pM_k^{h1}p^T - M_k^{h2} \right\|^2}{\sum\limits_{k=1}^{n+1} (\left\|M_k^{h1}\right\|^2 + \left\|M_k^{h2}\right\|^2)}$$

where $perm(m)$ denotes the set of all $m*m$ permutation matrices. It is easy to see that $sim(h_1, h_2)$ falls in the range of [0, 1].

4.2 Calculating the similarity

The similarity degree between two case graphs defined in section 4.1 only involves calculating the minimal value with respect to all the possible permutation matrices. In what follows, we extend Umeyama's algorithm as define for a single pair of asymmetric matrices to m pairs of asymmetric matrices.

In fact, to calculate

$$\min_{p\in perm(m)} \sum_{k=1}^{n+1} \left\| pM_k^{h1}p^T - M_k^{h2} \right\|^2$$

we defined

$$E_k^{h1} = \frac{M_k^{h1} + (M_k^{h1})^T}{2} + \sqrt{-1}\,\frac{M_k^{h1} - (M_k^{h1})^T}{2}$$

$$E_k^{h2} = \frac{M_k^{h2} + (M_k^{h2})^T}{2} + \sqrt{-1}\,\frac{M_k^{h2} - (M_k^{h2})^T}{2}$$

where $k = 1, 2, \ldots, n, (n+1)$.

Since E_k^{h1} and E_k^{h2} are Hermitian matrices, we can get the eigen-compositions of E_k^{h1} and E_k^{h2} as:

$$E_k^{h1} = U_k^{h1} D_k^{h1} (U_k^{h1})^*$$

$$E_k^{h2} = U_k^{h2} D_k^{h2} (U_k^{h2})^*$$

where U_k^{h1} and U_k^{h2} are unitary matrices, and D_k^{h1} and D_k^{h2} are diagonal matrices formed from the ordered eigen-values of E_k^{h1} and E_k^{h2}, respectively. N.B. Here, * denotes the Hermitian transposition.

Let

$$V_1 = < \left|U_1^{h1}\right|, \left|U_2^{h1}\right|, \cdots, \left|U_{n+1}^{h1}\right| >$$

$$V_2 = < \left|U_1^{h2}\right|, \left|U_2^{h2}\right|, \cdots, \left|U_{n+1}^{h2}\right| >$$

Then, we get the optimized permutation matrix p by means of using the Hungarian algorithm:

$$p = Hungarian\ (V_2 V_1^T)$$

4.3 Experimental results

The algorithm has been implemented in MatLab. What follows describes some experiments conducted, where the corresponding weight w was set as -0.3.

As shown in Fig. 2 – Fig. 8, for the 7 pairs of case history graphs, G_i^{hl} and G_i^{h2} (i = 1, 2, ..., 7), the corresponding results computed from the algorithm provide a well stable similarity measurement as expected.

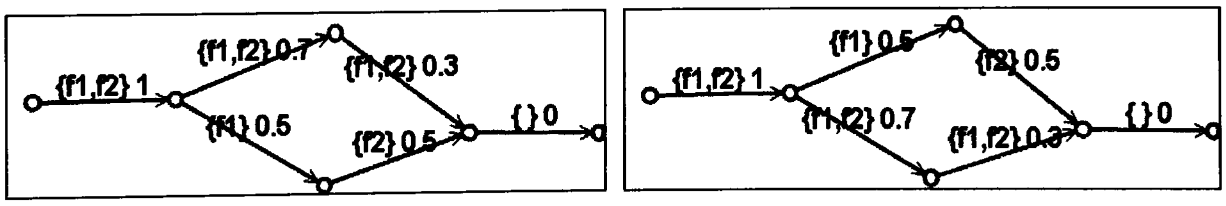


Fig. 2. The similarity between G_1^{h1} and G_1^{h2} is 1

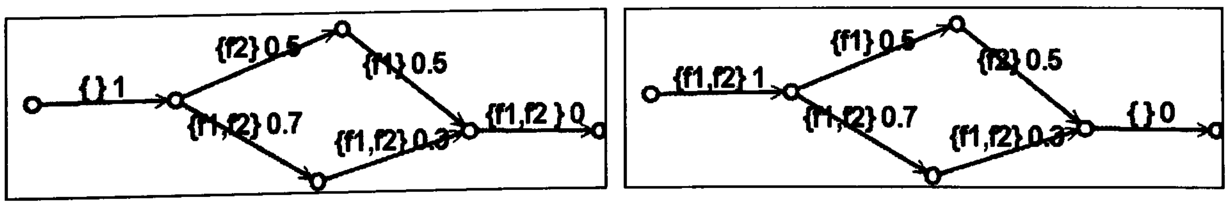


Fig. 3. The similarity between G_2^{hl} and G_2^{h2} is 0.7800

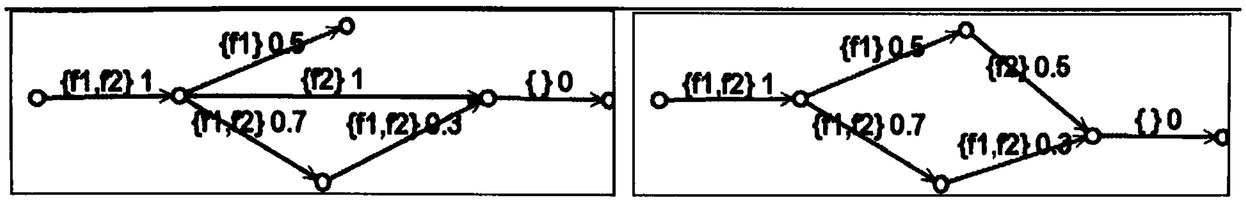


Fig. 4. The similarity between G_3^{h1} and G_3^{h2} is 0.8413

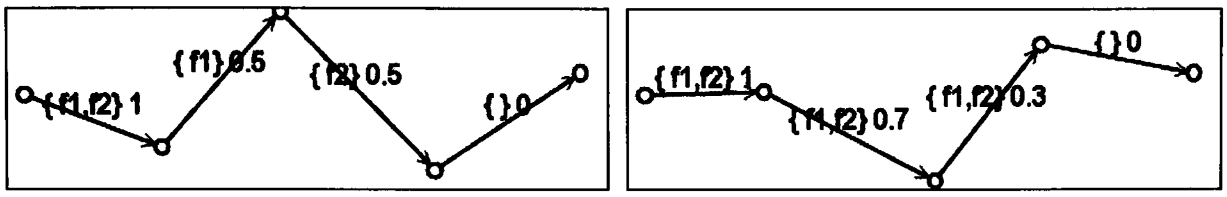


Fig. 5. The similarity between G_4^{h1} and G_4^{h2} is 0.9148

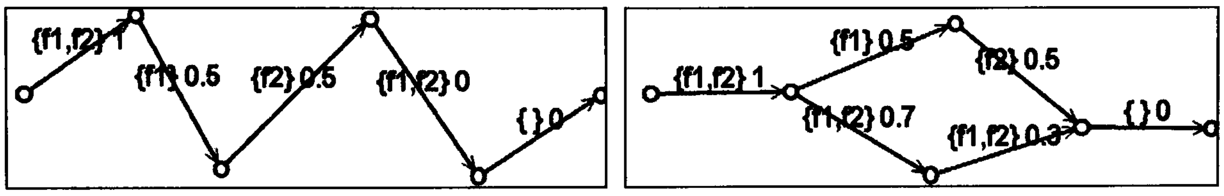


Fig. 6. The similarity between G_5^{h1} and G_5^{h2} is 0.7301

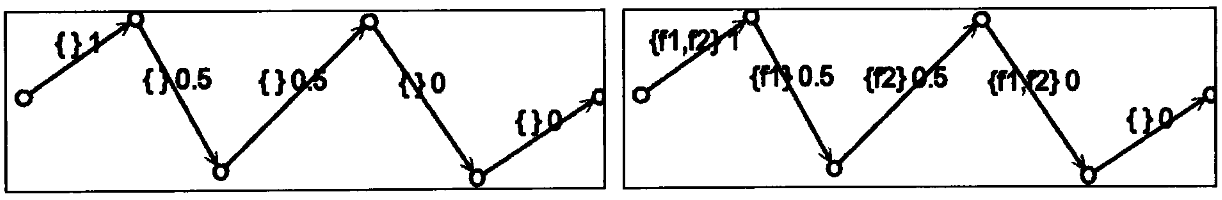


Fig. 7. The similarity between G_6^{h1} and G_6^{h2} is 0.7669

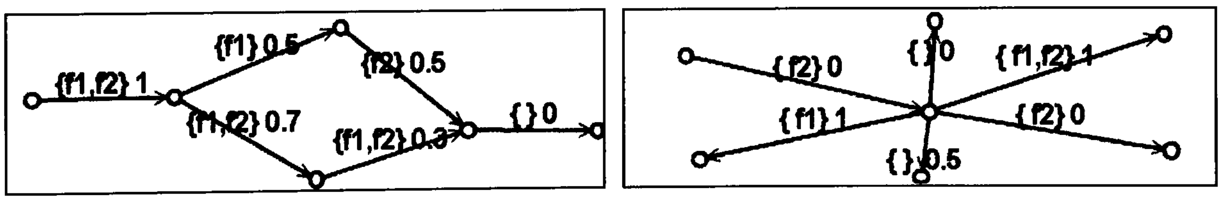


Fig. 8. The similarity between G_7^{h1} and G_7^{h2} is 0.5711

4.4 Computational complexity

For two given case history graph G^{h1} and G^{h2} with m nodes and n fluents, the computation consists of two significant parts. The first part involves $2*(n+1)$ calculations as for the eigen-decomposition of matrix with size $m*m$, giving a complexity of $O(nm^3)$. The second part involves in applying Hungarian algorithm to a matrix of size $m*m$, giving a computational complexity of $O(m^3)$. Therefore, the overall complexity of matching two case history graphs is $O(nm^3)$.

5 Conclusions

In this paper, we have introduced a mechanism for representing and recognizing case histories with rich internal temporal aspects, in the domain of case-based reasoning. The formalism includes two equivalent schemas for case history formalization and a graphical representation corresponding to the unified second schema. It is shown that case history pattern recognition and matching can be simply transformed into graph matching. By means of extending the eigen-decomposition algorithm from weighted graph to vector labeled graph, we can get ideal results in matching pairs of case history graphs for most states of affairs. However, in some special states of affairs, the algorithm may fail to work as expected. The future work of this research includes identifying the reason of the failure, and improving the algorithm for general real life applications.

References

1. Nakhaeizadeh, G.: Learning Prediction of Time Series: A Theoretical and Empirical Comparison of CBR with Some Other Approaches. In Proceedings of the Workshop on Case-Based Reasoning, AAAI-94. Seattle, Washington (1994) 67-71
2. Branting, L. and Hastings, J.: An empirical evaluation of model-based case matching and adaptation. In Proceedings of the Workshop on Case-Based Reasoning, AAAI-94. Seattle, Washington (1994) 72-78
3. Jaczynski, M.: A Framework for the Management of Past Experiences with Time-Extended Situations. In Proceedings of the 6th International Conference on Information and Knowledge Management (CIKM'97), Las Vegas, Nevada, USA, November 10-14, (1997) 32-39
4. Hansen, B.: Weather reasoning predication using case-based reasoning and Fuzzy Set Theory, MSc Thesis, Technical University of Nova Scotia, Halifax, Nova Scotia, Canada (2000)
5. Jare, M., Aanodt, A. and Shalle, P.: Representing Temporal Knowledge for Case-Based Reasoning. Proceedings of the 6th Euroupean Conference, ECCBR 2002, Aberdeen, Scotland, UK, September 4-7, (2002) 174-188
6. Tveter, D.: The Pattern Recognition Basis of Artificial Intelligence. Wiley-IEEE Computer Society Press (1998)
7. Theodoridis, S. and Koutroumbas, K.: Pattern Recognition. Second Edition, Academic Press (2003)
8. Ma, J. and Knight, B.: Reified Temporal logic: An Overview, Artificial Intelligence Review, Vol. 15. (2001) 189-217
9. Allen, J.: Maintaining knowledge about temporal intervals. Communications of the ACM, Vol. 26 (11), (1983) 832-843
10. Allen, J. and Hayes, P.: Moments and Points in an Interval-based Temporal-based Logic. Computational Intelligence, Vol. 5. (1989) 225-238
11. Ma, J. and Hayes, P.: Primitive Intervals Vs Point-Based Intervals: Rivals Or Allies?", the Computer Journal, Vol.49(1). (2006) 32-41
12. Ma, J. and Knight, B.: A General Temporal Theory. The Computer Journal, Vol. 37(2). (1994) 114-123

13. Ma, J. and Knight, B.: Representing The Dividing Instant. The Computer Journal, Vol. 46(2). (2003) 213-222
14. Shoham, Y.: Temporal Logics in AI: Semantical and Ontological Considerations, Artificial Intelligence Vol. 33. (1987) 89-104
15. Shanahan, M.: A Circumscriptive Calculus of Events, Artificial Intelligence, Vol. 77. (1995) 29-384
16. Knight, B. and Ma, J.: A General Temporal Model Supporting Duration Reasoning, Artificial Intelligence Communication, Vol. 5(2). (1992) 75-84
17. Chung, F.: Spectral Graph Theory, CBMS series 92, American Mathematical Society, Province, RI, 1997
18. Umeyama, S.: An Eigendecomposition Approach to Weighted Graph Matching Problems, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10(5). (1988), 695-703

Matching Scenarios Patterns by Using Linear Programming

Guoxing Zhao[1], Bin Luo[2] and Jixin Ma[1]
[1]*School of Computing and Mathematical Sciences, University of Greenwich, U.K.*
[2]*School of Computer Science and Technology, AnHui University, China*
{j.ma, g.zhao}@gre.ac.uk, luobin@ahu.edu.cn

Abstract

This paper continues the work presented previously at ICNC-FSKD-05 for representing and matching scenario patterns. A unified scheme is presented to replace the two previous equivalent schemas for formalizing scenario patterns. In the unified scheme, a scenario is denoted in terms of a collection of states with the corresponding temporal constraints, where a state is defined as a set of Boolean-valued time-dependent fluents. The concept of a scenario graph is formally introduced as a directed, partially weighted and labeled simple graph. Based on such a graphical representation, an extended linear programming graph matching algorithm is proposed for recognizing scenario patterns.

1. Introduction

Like the concept of case in case-based reasoning, the notion of state in state-based systems is fundamental for many real-time applications. In conventional state-based systems, various states of the world in the discourse are usually represented in terms of isolated snapshots, while the state histories (or temporal scenarios) with rich internal temporal aspects are neglected in most approaches. Over the past three decades, it has been noted that temporal representation and reasoning is essential for many areas of Artificial Intelligence, where one is interested not only in the representation of distinct episodes of an enterprise, but also in the history of earlier/future situations [13, 14, 23, 24, 27]. In particular, an appropriate representation and reasoning for temporal knowledge is necessary for many state-based systems, where the history of states, rather than distinct episodes, plays an important role in solving problems including explanation/diagnosis, prediction/forecast, planning/scheduling, process management, and history reconstruction, etc.

A natural approach to represent the temporal constraints on certain states is to associate the states with time elements. Generally speaking, there are three known choices as for the sort of objects to be taken as time elements: (1) points, i.e., instant without duration; (2) intervals, i.e., periods with positive duration; and (3) both points and intervals. In addition, in temporal systems where time intervals are modeled as time elements, there are two different approaches. In the first, intervals are modeled as derived objects constructed from points, e.g., as sets of points [8, 22], or as pairs of points [7, 9, 10, 16, 26]. However, it has been argued in the literature that defining intervals as objects derived from points may lead to the so-called Dividing Instant Problem [8, 11, 19]. The second treatment takes intervals as primitive objects, without insisting on the existence of "ending-points", "internal-points", or any points at all. Allen's interval logic [1, 2, 3, 4], Vilain's temporal system [31], and Ma and Knight's general time theory [17] are examples that treat intervals as primitive.

Generally speaking, pattern recognition aims at the operation and design of technologies to pick up meaningful patterns in data [29]. While pattern classification is about putting a particular instance of a pattern in a category, the goal of pattern matching is to determine how similar a pair of patterns are [28]. In state-based systems, certain states may be associated with specific time elements, where various temporal relations between the involved time elements will characterize different scenario patterns.

In [20], a formal method is proposed for representing and recognizing scenario patterns with rich internal temporal aspects. This paper continues such a work by means of unifying the two previous equivalent schemas for representing temporal scenarios, and implementing a linear programming graph matching algorithms for matching scenarios patterns. In section 2, we introduce the theoretical background of the formalism, where the graphical representation of temporal scenarios is formally described in section 3. Section 4 presents an extended linear programming algorithm for matching scenario graphs, and provides some experimental results. Finally, section 5 concludes the paper.

2. The background formalism

As proposed in [20], a simple point-based time model is adopted as the temporal basis. In such a time model, time elements are defined as typed point-based intervals, allowing expression of both absolute time values and relative temporal relations [21]. Following the notations taken in [20], we shall use R to denote the set of real numbers, and T, the set of time elements. Each time element t is defined as a typed (left-open & right-open, left-closed & right-open, left-open & right-closed, left-closed & right-closed) subset of the set of real numbers R. I.e., each time element must be in one of the following four forms:

$(p_1, p_2) = \{p \mid p \in R \wedge p_1 < p < p_2\}$

$[p_1, p_2) = \{p \mid p \in R \wedge p_1 \leq p < p_2\}$

$(p_1, p_2] = \{p \mid p \in R \wedge p_1 < p \leq p_2\}$

$[p_1, p_2] = \{p \mid p \in R \wedge p_1 \leq p \leq p_2\}$

In the above, p_1 and p_2 are real numbers, and are called the left-bound and right-bound of time element t, respectively. The absolute values as for the left and/or right bounds of some time elements might be unknown. In this case, real number variables are used for expressing relative relations to other time elements (see later).

In addition, if the left-bound and right-bound of time element t are the same, t is called a time point, otherwise it is called a time interval. Without confusion, time element [p, p] is taken as identical to point p. Also, if a time element is not specified as open or closed at its left (right) bound (that is, the left (right) type of the time element is unknown), we shall use "<" (or ">") instead of "(" and "[" (or ")" and "]") as for its left (or right) bracket. Also, the duration of a time element t, D(t), is defined as the distance between its left bound and right bound. In other words:

$t = \langle p_1, p_2 \rangle \Leftrightarrow D(t) = p_2 - p_1$

Following Allen's terminology [1, 2, 3], "Meets" is use to denote the immediate predecessor order relation over time elements:

$\text{Meets}(t_1, t_2) \Leftrightarrow \exists p_1,p,p_2 \in R(t_1 = (p_1, p) \wedge t_2 = [p, p_2)$
$\vee\ t_1 = [p_1, p) \wedge t_2 = [p, p_2)) \vee t_1 = (p_1, p) \wedge t_2 = [p, p_2]$
$\vee\ t_1 = [p_1, p) \wedge t_2 = [p, p_2] \vee t_1 = (p_1, p] \wedge t_2 = (p, p_2)$
$\vee\ t_1 = [p_1, p] \wedge t_2 = (p, p_2) \vee t_1 = (p_1, p] \wedge t_2 = (p, p_2]$
$\vee\ t_1 = [p_1, p] \wedge t_2 = (p, p_2])$

It is easy to see that the intuitive meaning of Meets(t_1, t_2) is that, on the one hand, time elements t_1 and t_2 don't overlap each other (i.e., they don't have any part in common, not even a point); on the other hand, there is not any other time element standing between them.

Analogous to the 13 relations introduced by Allen for intervals [1, 2, 3], there are 30 exclusive temporal order relations over time elements including both time points and time intervals, which can be classified into the following 4 groups:

- Relations that relate points to points:
  {Equal, Before, After}
- Relations that relate points to intervals:
  {Before, After, Meets, Met_by, Starts, During. Finishes}
- Relations that relate intervals to points:
  {Before, After, Meets, Met_by, Started_by, Contains, Finished_by}
- Relations that relate intervals to intervals:
  {Equal, Before, After, Meets, Met_by, Overlaps, Overlapped_by, Starts, Started_by, During, Contains, Finishes, Finished_by}

N.B. The definition of these derived temporal order relations in terms of the single relation Meets is straightforward. E.g.:

$\text{Equal}(t_1, t_2) \Leftrightarrow \exists t_3,t_4 \in T(\text{Meets}(t_3, t_1) \wedge \text{Meets}(t_3, t_2) \wedge \text{Meets}(t_1, t_4) \wedge \text{Meets}(t_2, t_4))$

A fluent is a statement (or proposition) whose truth-value is dependent on time elements. We use F to denote the set of fluents.

In order to associate a fluent with a time element, we use a meta-predicate [18, 26], Holds, to substitute the formula Holds(f, t) for each pair of a fluent f and a time element t, denoting that fluent f holds true over time t.

(H1) $\text{Holds}(f_1 \vee f_2, t) \Leftrightarrow \text{Holds}(f_1, t) \vee \text{Holds}(f_1, t)$

(H2) $\text{Holds}(f, \langle p_1, p_2 \rangle) \Leftrightarrow \forall p_3,p_4 \in R(p_1 \leq p_3 \wedge p_4 \leq p_2 \Rightarrow \text{Holds}(f, \langle p_3, p_4 \rangle))$

(H3) $\text{Holds}(f, \langle p_1, p_2 \rangle)$
$\wedge\ \text{Holds}(f, \langle p_2, p_3 \rangle)$
$\wedge\ \text{Meets}(\langle p_1, p_2 \rangle, \langle p_2, p_3 \rangle)$
$\Rightarrow \text{Holds}(f, \langle p_1, p_3 \rangle)$

It is worth pointing out that the time model and the formulae introduce in the above allows temporal knowledge with absolute values, as well as temporal knowledge expressed in terms of relative relations. An example can be found in [20].

We shall represent the static state of the world in the discourse is defined as a collection of fluents, and denote the set of all the states as S. In addition, we use Belongs(f, s) to represent fluent f belongs to state s [24]:

In [20], two equivalent schemas have been proposed for representing temporal scenarios. In this paper, by means of introducing the concept of states, each given scenario, st, can be formalized in terms of a unified scheme, represented as a quadruple:

st = $<State^{st}, Holds^{st}, Meets^{st}, Dur^{st}>$, such that

$State^{st} = \{s^{st}_i \mid s^{st}_i \in S, i = 1, \ldots, m\}$,

$Holds^{st} = \{Holds(s^{st}_i, t^{st}_i) \mid \; t^{st}_i \in T, 1 \leq i \leq m\}$

$Meets^{st} = \{Meets(t^{st}{}', t^{st}{}'') \mid$ for some $t^{st}{}', t^{st}{}'' \in T^{st}\}$

$Dur^{st} = \{Dur(t) = r \mid$ for some $t \in T^{st}, r \in R\}$

where T^{st} is the minimal subset of T closed under the following rules:

- $t^{st}_i \in T^{st}$, i = 1, ..., m.
- $t \in T^{st} \Leftrightarrow \exists t' \in T^{st}(Meets(t, t') \vee Meets(t', t))$

3. Scenario graphs

In [15], a graphical representation for expressing temporal knowledge has been introduced in terms of a directed and partially weighted graph. It can be extended to express scenarios presented in the scheme introduced in section 2. In fact, a given scenario st can be represented in terms of a scenario graph, which can be formally defined as a directed, partially weighted/labeled simple graph G^{st}, where:

- Each time element t in T^{st} is denoted as a directed arc of the graph labeled by t that is bounded by a pair of nodes, which are called the tail-node, and the head-node, of the arc, respectively.
- Each relation Meets(ti, tj) in $Meets^{st}$ is represented by means of merging the head-node of t_i and the tail-node of t_j as a common node, of which ti is an in-arc and tj is an out-arc, respectively (see Figure 1).
- Each formula $Holds(s^{st}_i, t^{st}_i)$ in $Holds^{st}$ is represented by means of simply adding s^{st}_i as an additional label to the arc labeled by the corresponding t^{st}_i. For any time element t in T^{st}, if there is no Holds knowledge, it will be labeled by the empty state {}.
- Each piece of duration knowledge Dur(t) = r in Dur^{st} is expressed as a real number, r, alongside the corresponding arc t.

In what follows, we shall simply assume |F| = n, that is, the total number of fluents is n. Corresponding to scenario graph G^{st} with m nodes, we define a m*m-matrix M^{st}, named the characteristic matrix, where $M^{st}(u, v)$ is a (n+1)-dimension vector $l_{uv} \in R^{n+1}$, such that:

(a) For any adjacent pair of nodes u and v in G^{st}, if (u, v) is an arc representing time element t, then $l_{uv}(k) = 1$ if $Holds(f_k, t)$, otherwise $l_{uv}(k) = 0$, $1 \leq k \leq n$; and $l_{uv}(n+1) = Dur(t)$.

(b) For any non-adjacent pair of nodes u and v in G^{st}, $l_{uv} = <w, w, \ldots, w>$, where w is a negative real number, which will be use to adjust the edit-distance of deleting operations in graph matching process.

In this paper, we shall use M_k^{st} to denote the matrix whose u-v-entry is the k-th element of u-v-entry in M^{st}.

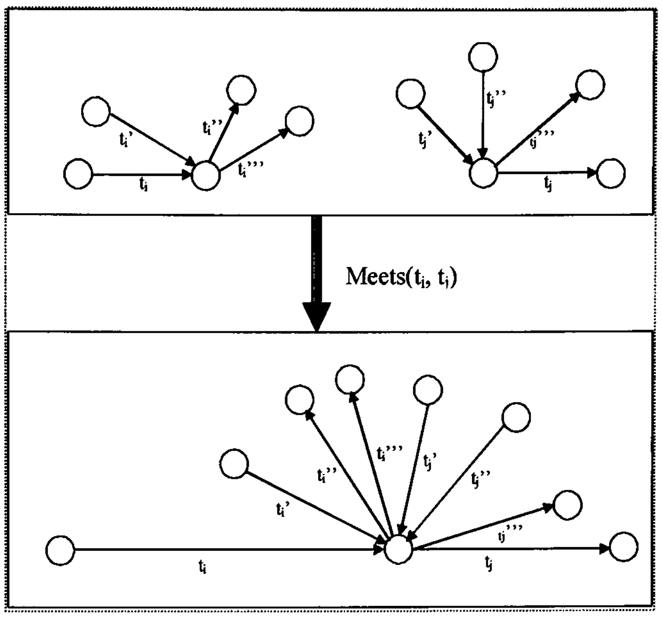


Figure. 1. Merging the *head-node* of t_i and the *tail-node* of t_j as a common node if *Meets*(t_i, t_j)

4. Extended linear programming approach for matching scenario graphs

Based on the graph representation of scenarios, we can match two temporal scenarios by means of matching their corresponding scenario graphs. We have explored various graph matching algorithms for our scenario graph matching, including the eigen-decomposition approach of Umeyama [30], the polynomial transform approach of Almohamad [5] and the linear programming approach of Almohamad and Duffuaa [6]. The experimental results show that the linear programming approach gets the most accurate result within acceptable computational time. Since each of entries of the characteristic matrix of a given scenario graph is a (*n*+1)-dimension vector, rather than a single real value as in conventional spectral-based models, we need to extend the so-called linear programming graph matching algorithm for matching scenario graphs.

4.1. Definition of scenario similarity

In what follows in this paper, if M is a real number matrix, we shall use $||M||_1$ to denote the L_1-norm of M [6]. Given two scenarios st_1 and st_2, assume the characteristic matrices are M^{st1} and M^{st2}, with size m_1*m_1 and m_2*m_2, respectively. Without losing the generality, we assume $m_1 = m_2 = m$. In fact, if $m_1 < m_2$, we can simply add $m_2 - m_1$ isolated dummy nodes to graph G^{st1} to get an extended graph $G^{st1'}$, whose characteristic matrix $M^{st1'}$ will have the same size as that of M^{st2}, i.e., m_2*m_2. Similar treatment can be applied to the case where $m_2 < m_1$.

The similarity degree between st_1 and st_2 is then defined by:

$$sim(st_1, st_2) = 1 - \frac{\min\limits_{p \in perm(m)} \sum_{k=1}^{n+1} \left\| pM_k^{st1} p^T - M_k^{st2} \right\|_1}{\sum_{k=1}^{n+1} (\left\| M_k^{st1} \right\|_1 + \left\| M_k^{st2} \right\|_1)}$$

where perm(m) denotes the set of all m-by-m permutation matrices. It is easy to see that sim(st$_1$, st$_2$) falls within the range of [0, 1].

4.2. Calculating the similarity

The similarity degree between two scenario graphs defined in section 4.1 only involves calculating the minimal value with respect to all the possible permutation matrices. In what follows, we extend linear programming graph matching algorithm as define for a single pair of asymmetric matrices to n+1 pairs of asymmetric matrices.

In fact, to calculate $\min\limits_{p \in perm(m)} \sum_{k=1}^{n+1} \left\| pM_k^{st1} p^T - M_k^{st2} \right\|_1$,

we only need to calculate $\min\limits_{p \in perm(m)} \sum_{k=1}^{n+1} \left\| pM_k^{st1} - M_k^{st2} p \right\|_1$.

We shall use ⊗ and Vec to denote the Kronecker product and the vectorization of a matrix formed by stacking its columns into a single column vector.

From the fact $Vec(ABC) = (C^T \otimes A)Vec(B)$, we have:

$$\min_{p \in perm(m)} \sum_{k=1}^{n+1} \left\| pM_k^{st1} - M_k^{st2} p \right\|_1$$

$$= \min_{p \in perm(m)} \sum_{k=1}^{n+1} \left\| ((M_k^{st1})^T \otimes I - I \otimes M_k^{st2}) Vec(p) \right\|_1$$

In order to find the matrix p satisfying the above minimum requirement, we employ 2m*(n+1)*(n+1) additional variables $Z_k^{i,j}$, $Y_k^{i,j}$, where $1 \le k \le m$ and $1 \le i, j \le n+1$, to approximately solve the problem in terms of the following linear programming:

$$\min_{p, Y_k, Z_k} \sum_{i,j,k} Z_k^{i,j} + Y_k^{i,j}$$

Such that

$$(M_k^{st1})^T \otimes I - I \otimes M_k^{st2}) Vec(p) + Vec(Z_k) - Vec(Y_k) = 0^{(n+1)*(n+1)}$$

$$\sum_i p_{i,j} = 1 \quad ,$$

and $\sum_j p_{i,j} = 1$

$$p, Z_k, Y_k \ge 0$$

However, the entries of matrix p obtained from the above linear programming may not be all 0 or 1. If this is the case, we need to apply the so-called Hungarian algorithm [12] to matrix p. That is, Hungarian(p) will be the approximate solution to the corresponding graph match problem.

4.3. Experimental results

The algorithm has been implemented in MatLab. What follows describes some experiments conducted, where the corresponding weight w was set as -0.3.

As shown in Figure. 2 - Figure. 8, for the 7 pairs of scenario graphs, st$_{i,1}$ and st$_{i,2}$ (i = 2, 4 ··· 8), the corresponding results computed from the algorithm provide a well stable similarity measurement as expected.

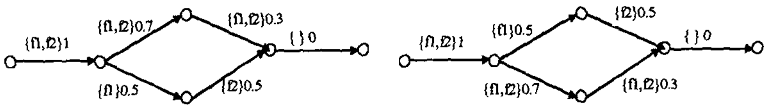


Figure 2. Sim(st$_{2,1}$, st$_{2,2}$) = 1

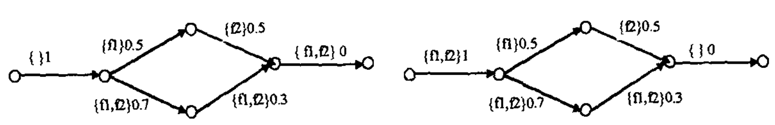


Figure 3. Sim(st$_{3,1}$, st$_{3,2}$) =0.8947

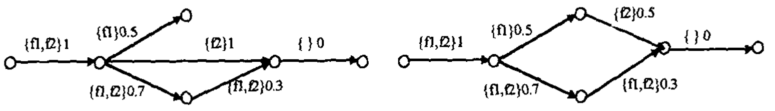


Figure 4. Sim(st$_{4,1}$, st$_{4,2}$) =0.9307

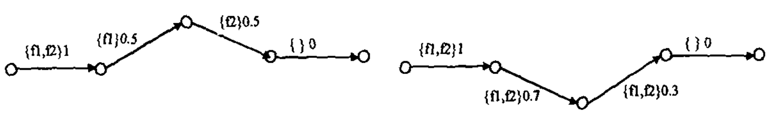


Figure 5. Sim(st$_{5,1}$, st$_{5,2}$) =0.9537

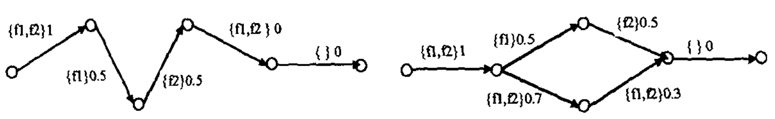


Figure 6. Sim(st$_{6.1}$, st$_{6.2}$) =0.8823

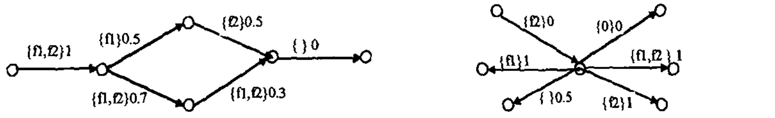


Figure 7. Sim(st$_{7,1}$, st$_{7,2}$) =0.7271

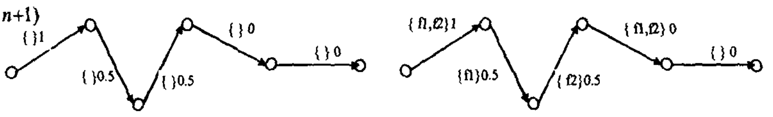


Figure 8. Sim(st$_{8,1}$, st$_{8,2}$) =0.9088

4.4. Computational complexity

For two given scenario graphs G^{st1} and G^{st2} with m nodes and n fluents, the complexity of the computation is O(n^6*m^3*L), since the linear graph matching problem can be solved in O(k^3*L), where k is the number of variables (here k = $(2m+1)*n^2$), L is the size of the LP problem, for details please see [6].

5. Conclusions

In this paper, we have extended the work presented at ICNC-FSKD05. By means of introducing the concept of states, we have unified the two previous equivalent schemas for scenario formalization. In order to graphically represent a temporal scenario, we have provided a formal definition of a scenario graph, in terms of a directed, partially weighted and labeled simple graph. Therefore, the problem of matching temporal scenarios is transformed into conventional graph matching. By means of extending the linear programming graph matching algorithm from weighted graph to vector labeled graph, we obtained ideal results in matching pairs of scenario graphs for most states of affairs. However, if the number of time elements is larger than 30, the CPU time is unacceptable. The future work of this research includes accelerating and improving the algorithm for general real life applications.

6. References

[1] J. Allen, "An interval-based representation of temporal knowledge", *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981, pp.221-226.

[2] J. Allen, "Maintaining knowledge about temporal intervals", *Communications of the ACM 26:11*, 1983, pp.832-843,.

[3] J. Allen, "Towards a General Theory of Action and Time", *Artificial Intelligence 23*, 1984, pp.123-154.

[4] J. Allen and P. Hayes, "Moments and Points in an Interval-based Temporal-based Logic", *Computational Intelligence 5*, 1989, pp.225-238.

[5] H. Almohamad, "A Polynomial Transformfor Matching Pairs of Weighted Graphs", *Applied Mathematical Modelling,15(44)*, 1991, pp.216-222,

[6] H. Almohamad and S. Duffuaa, "A Linear Programming Approach for the Weighted Graph Matching Problem", *IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(5)*, May 1993, pp.522-525,

[7] van P. Beek, "Reasoning about qualitative temporal information", *Artificial Intelligence 58*, 1992, pp.297-326.

[8] van J. Benthem. *The Logic of Time*, Kluwer Academic, Dordrech, 1983.

[9] B. Bruce, "A Model for Temporal References and Its Application in a Question Answering Program", *Artificial Intelligence 3*, 1972, pp.1-25.

[10] R. Dechter, I. Meiri, I and J. Pearl, "Temporal Constraint networks", *Artificial Intelligence* 49, 1991, pp.61-95.

[11] A. Galton, "A Critical Examination of Allen's Theory of Action and Time", *Artificial Intelligence 42*, 1990, pp.159-188.

[12] W. Harold, "The Hungarian Method for the assignment problem", *Naval Research Logistic Quarterly* 2, 1995, pp.83-97.

[13] M. Jare, A. Aanodt and P. Shalle, "Representing Temporal Knowledge for Case-Based Reasoning", *Proceedings of the 6th Euroupean Conference, ECCBR 2002, Aberdeen, Scotland, UK, September 4-7*, 2002, pp.174-188.

[14] E. Keravnou, "Modelling medical concepts as time-objects" Eds. M. Stefanelli and J. Watts, eds., *Lecture Notes in artificial Intelligence* 934, 1995, Springer pp.67-90.

[15] B. Knight and J. Ma, "A General Temporal Model Supporting Duration Reasoning", *Artificial Intelligence Communication, Vol.5:2*, 1992, pp.75-84.

[16] P. Ladkin, "Effective solutions of qualitative intervals constraint problems", *Artificial Intelligence 52*, 1992, pp.105-124.

[17] J. Ma and B. Knight, "A General Temporal Theory", *the Computer Journal 37:2*, 1994, pp.114-123.

[18] J. Ma and B. Knight, "Reified Temporal logic: An Overview", *Artificial Intelligence Review 15*, 2001, pp.189-217.

[19] J. Ma and B. Knight, "Representing the Dividing Instant", *the Computer Journal 46:2*, 2003, pp.213-222.

[20] J. Ma and B. Luo, "Representing and Recognizing Scenario Patterns", *Lecture Notes in Computer Science, Vol.3614*, 2005, pp.140-149.

[21] J Ma and P Hayes "Primitive Intervals Vs Point-Based Intervals: Rivals or Allies", *the Computer Journal, Vol.49:1*, 2006, pp.32-41.

[22] D. McDermott, "A Temporal Logic for Reasoning about Processes and Plans", *Cognitive Science 6*, 1982, pp.101-155.

[23] G. Nakhaeizadeh, "Learning Prediction of Time Series: A Theoretical and Empirical Comparison of CBR with Some Other Approaches", In *Proceedings of the Workshop on Case-Based Reasoning, AAAI-94*. Seattle, Washington, 1994, pp.67-71.

[24] Y. Shahar, "Timing is everything: temporal reasoning and temporal data maintenance medicine", Eds. W Horn et al., *AIMDM'99, LNAI* 1620, Springer Verlag, 1999, pp.30-46.

[25] M. Shanahan, "A Circumscriptive Calculus of Events", *Artificial Intelligence* 77, 1995 pp.29-384.

[26] Y. Shoham, "Temporal Logics in AI: Semantical and Ontological Considerations", *Artificial Intelligence 33*, 1987, pp.89-104.

[27] R. Snodgrass and I. Ahn, "Temporal Databases", *IEEE, 0018-9162/86/0900-0035*, 1986, pp.35-42.

[28] S. Theodoridis and K. Koutroumbas, *Pattern Recognition. Second Edition,* Academic Press, 2003.

[29] D. Tveter. *The Pattern Recognition Basis of Artificial Intelligence*. Wiley-IEEE Computer Society Press, 1998.

[30] S. Umeyama, "An Eigendecomposition Approach to Weighted Graph Matching Problems", *IEEE Transactions on Pattern Analysis and Machine Intelligence 10:5*, 1988, pp.695-703.

[31] M. Vilain, "A System for Reasoning about Time", *Proceedings of the 1st National Conference on Artificial Intelligence,* 1982, pp.197-201.

APPENDIX E USING EIGEN-DECOMPOSITION METHOD FOR WEIGHTED GRAPH MATCHING

Using Eigen-decomposition Method for Weighted Graph Matching

Guoxing Zhao[1], , Bin Luo[2], Jin Tang[2] and Jinxin Ma[1]

[1]School of Computing and Mathematical Sciences, University of Greenwich, U.K.
{g.zhao, j.ma}@gre.ac.uk
[2]School of Computer Science and Technology, AnHui University, China
{luobin, jintang}@ahu.edu.cn

Abstract. In this paper, Umeyama's eigen-decomposition approach to weighted graph matching problems is critically examined. We argue that Umeyama's approach only guarantees to work well for graphs that satisfy three critical conditions: (1) The pair of weighted graphs to be matched must be nearly isomorphic; (2) The eigenvalues of the adjacency matrix of each graph have to be single and isolated enough to each other; (3) The rows of the matrix of the corresponding absolute eigenvetors cannot be very similar to each other. For the purpose of matching general weighted graph pairs without such imposed constraints, we shall propose an approximate formula with a theoretical guarantee of accuracy, from which Umeyama's formula can be deduced as a special case. Based on this approximate formula, a new algorithm for matching weighted graphs is developed. The experimental results demonstrate great improvements to the accuracy of weighted graph matching.

Keywords: Intelligent Computing, Pattern Recognition, Graph Matching.

1 Introduction

Graphs are a powerful and versatile tool used for the description of structural objects in many application areas such as case-based reasoning, semantic networks, document processing, image analysis, biometric identification, computer vision and video analysis, and so on. In general, in terms of their graph representation, objects can be represented by the vertices whilst the relationships between objects can be represented by the edges. Therefore, the task of calculating the similarity degree between two objects can be simply transferred into the problem of matching the corresponding pair of graphs.

Various algorithms for graph matching problems have been developed, which, according to [6], can be classified into two categories: (1) search-based methods which rely on possible and impossible pairings between vertices; and (2) optimization-

This research is supported in part by National Nature Science Foundation of China

based methods which formulate the graph matching problem as an optimization problem. Generally speaking, on one hand, search-based methods will find optimal solutions, but require exponential time in the worst case. On the other hand, normally, optimization-based methods only require polynomial bounded computation time, but in some cases may fail to find the optimal solution. Most search-based approaches use the idea of heuristics [11,14,15], where optimization-based methods have followed different approaches, including Bayesian methods [5], relaxation labeling [8], neural network [13], genetic algorithm [9], symmetric polynomials transformation (SPGM) [1], linear programming (LPGM) [2], and Kronecker Product Successive Projection methods [3], etc.

Another pioneer optimization-based method is Umeyama's eigen-decomposition approach (EDGM) [16]. This approach is based on matrix decomposition and norm from spectral theory. Over the past two decades, Umeyama's method has always been cited and compared with other approaches again and again. On one hand, it is noted to be easy to use and computationally efficient; on the other hand, it is criticized to be inaccurate in general since its mean error in graph matching is above average. However, the theoretical reason of these has been neglected in the literature and no investigation has been carried out to explore the scope of graph pairs in which the EDGM algorithm can provide efficient and effective matching with a high degree of accuracy.

In this paper, we shall critically examine Umeyama's EDGM approach. In section 2, we provide a brief introduction to the EDGM algorithm and, by means of statistical demonstration, we shall point out that Umeyama's approach only guarantees to work well for graphs that satisfy the following three critical conditions: (1) The pair of weighted graphs to be matched must be nearly isomorphic; (2) The eigenvalues of the adjacent matrix of each graph have to be single and isolated enough to each other; (3) The rows of the matrix of the corresponding absolute eigenvetors cannot be very similar to each other. For general treatments, an approximate formula is proposed in section 3 for matching any weighted graph pairs, together with a theoretical discussion of its accuracy. It is shown that, as a special case, Umeyama's original formula can be directly deduced from the approximate formula. In section 4, a new graph matching algorithm is proposed based on the approximate formula and experimental results are provided. Finally, section 5 concludes the paper.

2 The Eigen-decomposition Approach

In [16], an Eigen-decomposition approach was proposed for matching weighted graphs with the same number of nodes. A weighted graph G can be denoted as an ordered pair (N, w), where N is a set of nodes and w is a function which assigns a weight $w(v_i, v_j)$ to each pair of nodes (v_i, v_j) (edge of the graph). The adjacency matrix of a weighted graph $G = (V, w)$ is defined as $A_G = \{g_{ij}\}$, where $g_{ij} = w(v_i, v_j)$, $i, j = 1, 2, \ldots, n$, and n is the number of nodes in graph G.

APPENDIX E USING EIGEN-DECOMPOSITION METHOD FOR WEIGHTED GRAPH MATCHING

In this paper, for the reason of simply repression, without confusion, we shall not distinguish a weighted graph G and its corresponding adjacency matrix A_G. In other words, we shall simply express the adjacency matrix of G as G itself.

The problem of matching two weighted graphs G and H of n nodes is to find a one-to-one correspondence between the two corresponding sets of nodes that minimizes the distance between G and H, $d(G, H)$, which can be formulated in terms of the so-called Frobenius-norm (denoted as $\|\bullet\|_F$) as:

$$d(G,H) = \min_{P \in Perm(n)} \left\| PGP^T - H \right\|_F \qquad (2.1)$$

where G and H are the adjacent matrices of the weighted graphs to be matched and $Perm(n)$ is the set of all n-by-n permutation matrices.

From the definition, the adjacency matrix of any undirected graph G is symmetric. Therefore, there exists a real orthogonal matrix O such that $D_G = O^TGO$ is a diagonal matrix. However, for directed graphs, their adjacency matrices are in general asymmetric and therefore may be not "real-orthogonally" diagonalizable. To handled this problem, Umeyama uses the idea of decomposing a matrix uniquely into a sum of a symmetric and a skew-symmetric matrix. In fact, any real n-by-n matrix G can be transformed into a complex Hermitian matrix $Ht(G)$:

$$Ht(G) = \frac{G+G^T}{2} + \sqrt{-1}\frac{G-G^T}{2}$$

It is easy to get that, for any two n-by-n real matrices G and H:

$$\left\| PGP^T - H \right\|_F = \left\| PAP^T - B \right\|_F$$

where A=$Ht(G)$ and B=$Ht(H)$. Therefore, the problem of matching two matrices (symmetric or asymmetric) G and H is transformed into the problem of matching two Hermitian matrices.

From matrix theory, Hermitian matrices A and B can be decomposed as $A=VD_AV^*$, matrix $B=WD_BW^*$, where D_A and D_B are the diagonal matrices of the eigenvalues (in ascending order) of A and B, respectively, and V and W are two unitary matrices.

In [16], the following formula is used to solve general graph matching problems:

$$P\text{=Hungarian}(|W|\,|V|^T) \qquad (2.2)$$

where $|V|$ and $|W|$ are matrices whose entries are absolute values of the corresponding entries of V and W, Hungarian(*) denote the Hungarian algorithm [7], which is a combinatorial optimization algorithm which solves assignment problems in polynomial time ($O(n^3)$).

The Eigen-decomposition method has been noted to be easy to use and computationally efficient. On the other hand, it has also been pointed out to be inaccurate in general since its mean error in graph matching is above average compared with other approaches. However, no investigation has been carried out to explore the scope of graph pairs in which the EDGM algorithm can provide efficient and effective matching with a high degree of accuracy. Based on some theoretical and experimental analysis as shown below, we list the three constraints of the EDGM algorithm in graph matching.

1. "Nearly-Isomorphic": The distance $d(G,H)$ of the two graphs to be matched must be small enough.
2. "Isolating eigenvalues": All the eigenvalues of the matrix A, as well as matrix B, has to be single and the distance between two successive eigenvalues has to be big enough.
3. "Dissimilar rows": Any two rows of matrix $|V|$ in formula (2.2) cannot be very similar to each other, and the same requirement applies to $|W|$.

In fact, these three constraints are necessary and sufficient for the EDGM algorithm to get good approximations in general graph matching.

2.1 The Sufficiency of the Three Constraints

Firstly, we claim that the EDGM algorithm works very well for the graph pairs satisfy all the 3 constraints. 500 pairs of isomorphic graphs are generated randomly, which satisfy constraints 1 – 3. For each pair G and H, H is disturbed by adding a perturbation matrix E whose entries are uniformly random real numbers in the range from –e to +e. Graph size ranges from 5 to 12 and the noise amplitude e is fixed to 0.05, then the mean distance between each pair of graphs is calculated by three graph matching algorithm: EDGM, SPGM and LPGM. CPU times are also compared.

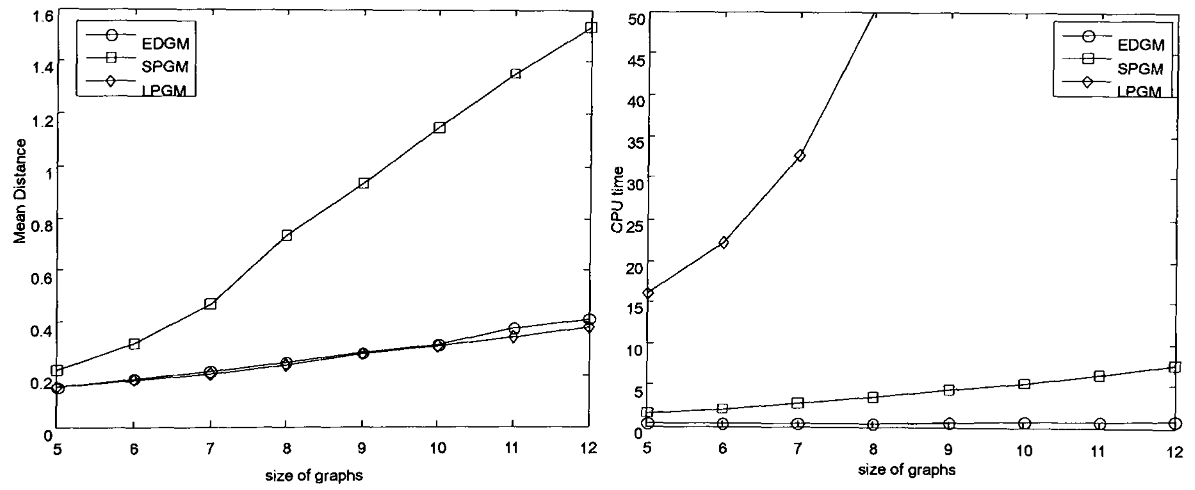


Fig. 1. Mean distance and calculating time of graph pairs satisfy all three constraints

In the above, Fig.1 shows that the EDGM algorithm obtains almost the same results as good as LPGM but uses significantly shorter CPU time for graphs satisfying the 3 constraints.

2.2 The Need of the "Nearly Isomorphic" Constraint

Secondly, we carry out tests to investigate the calculating error of the EDGM algorithm caused by increasing the distances between graph pairs.

The calculating error is defined as:

$$er = \left\| \bar{P} G \bar{P}^T - H \right\| - d(G,H)$$

where $\bar{P}$ is the solution calculated by the EDGM algorithm.

We also generate 500 pairs of isomorphic graphs *G* and *H* which satisfy both constraint 2 and constraint 3. For each pair *G* and *H*, we make them no longer isomorphic to each other by means of perturbing *H* with a noise e, ranging from 0 to 0.15.

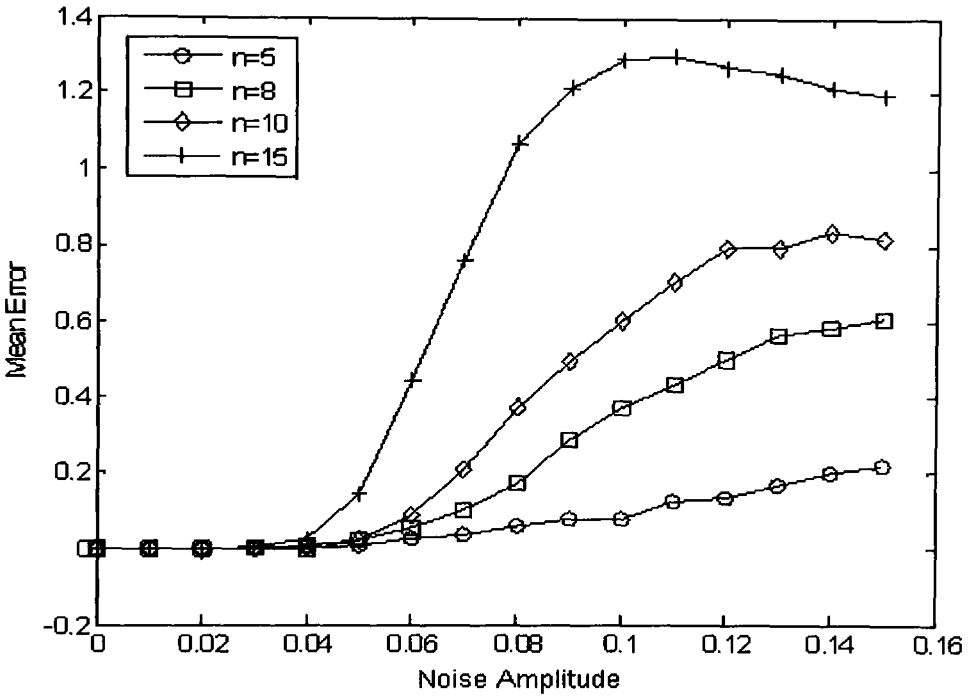


Fig. 2. Calculating error of EDGM algorithm relative to noise

From Fig. 2, we can see that the calculating error of the EDGM algorithm grows quickly when the noise amplitude or the size of graph increases, which confirms our claim that the "nearly isomorphic" property is needed for EDGM algorithm.

2.3 The Need of the "Isolating Eigenvalues" Constraint

Here, we demonstrate by example that without the "Isolating eigenvalues" condition, the EDGM method may fail to work. Consider

$$G=\begin{bmatrix}2&0&0&0\\0&2&0&0\\0&0&2&3\\0&0&0&2\end{bmatrix}\quad H=\begin{bmatrix}2&0&0&0\\3&2&0&0\\0&0&2&0\\0&0&0&2\end{bmatrix}\quad P_0=\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}$$

G and *H* are isomorphic because of $H=P_0GP_0^T$. Let $A=Ht(G)$ and $B=Ht(H)$, the eigenvalues of *A* and *B* are $\lambda(\mathrm{A})=\lambda(\mathrm{B})$ =[-0.1213, 2, 2, 4.1213]

By formula 2.2, we get the approximate solution:

$$P=Hungarian(|W||V|^T)=\begin{bmatrix}0&0&1&0\\0&0&0&1\\0&1&0&0\\1&0&0&0\end{bmatrix}$$

$\left\|PGP^T-H\right\|_F=4.2426$, the EDGM algorithm fails to find the best solution, that is, an isomorphic correspondence between *G* and *H* which gives a distance of 0 instead.

2.4 The Need of the "Dissimilar Rows" Constraint

Now, we show that "dissimilar rows constraint" is also needed. For instance, let

$$G=\begin{bmatrix}0&1&0&0\\0&0&0&0\\0&0&3&0\\0&0&0&1\end{bmatrix}\quad H=\begin{bmatrix}1&0&0&0\\0&3&0&0\\0&0&0&1\\0&0&0&0\end{bmatrix}\quad P_0=\begin{bmatrix}0&0&0&1\\0&0&1&0\\1&0&0&0\\0&1&0&0\end{bmatrix}$$

Again, *G* and *H* are isomorphic since $H=P_0GP_0^T$. Let $A=Ht(G)$ and $B=Ht(H)$. The eigenvalues of *A* and *B* are $\lambda(\mathrm{A})=\lambda(\mathrm{B})$ =[-1.2153, 2.6386, 3.6255, 5.9512], which are all single and well isolated.

The absolute matrix of *V* and *W* are:

$$|V|=\begin{bmatrix}\sqrt{2}/2&\sqrt{2}/2&0&0\\\sqrt{2}/2&\sqrt{2}/2&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\qquad |W|=\begin{bmatrix}0&0&1&0\\0&0&0&1\\\sqrt{2}/2&\sqrt{2}/2&0&0\\\sqrt{2}/2&\sqrt{2}/2&0&0\end{bmatrix}$$

And the solution from EDGM algorithm is:

$$P=Hungarian(|W||V|^T)=\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}$$

$\|PGP^T - H\|_F = 1.4142$, the algorithm still fails to find the best permutation because the matrices $|V|$ and $|W|$ both have two same rows.

3 A Formula for Graph Matching

In order to explain by theoretical reasons why the 3 constraints are necessary and extend Umeyama's algorithm for general treatments to cases where some of the constraints are not satisfied, we introduce here a new approximate formula to graph matching problems.

3.1 The Approximate Formula

Given a Hermitian matrix A with $\lambda(\mathrm{A}) = [\lambda_1 = \cdots = \lambda_{\mathrm{n1}} < \lambda_{\mathrm{n1+1}} = \cdots = \lambda_{\mathrm{n1+n2}} < \cdots \le \lambda_{\mathrm{n}}]$ as its eigenvalues. That is, matrix A has k distinct eigenvalues with repeating times $\mathrm{n}_1,\ldots,\mathrm{n}_\mathrm{k}$, respectively, where $\sum_{\mathrm{j=1}}^{\mathrm{k}} \mathrm{n_j} = \mathrm{n}$.

Then, we can decompose matrix $A=VD_AV^*$, where $V=[V_1,\ldots,V_k]$, and V_j is the eigen-space of the j-th distinct eigenvalue of matrix A..

A simple but important property for the eigen-decomposition is that $A=(VX)D_A(VX)^*$ is also an spectral decomposition of matrix A, for every unitary matrix $X\in U(n_1\ldots,n_k)$, where $U(n_1,\ldots,n_k)$ denotes the set of all block matrices whose j-th diagonal matrix a n_j-by-n_j unitary matrix.

Let $B=WD_BW^*$, it is easy to see that:

$$\|PAP^T - B\|_F \le \|PV - WX\|_F(\|D_A\|_F + \|D_B\|_F) + \|D_A - D_B\|_F$$

So it is reasonable to use the following approximate formula to solve graph matching problems:

$$\min_{\substack{P\in Perm(n)\\ X\in U(n_1,\cdots,n_k)}} \|PV - WX\|_F \tag{3.1}$$

3.2 An Error Estimation Theorem for the Approximate Formula

If Hermitian matrix $B=WD_BW^*$ is gained from A by adding some small perturbation, that is $B = P_0AP_0^T + E$ and $\|E\| \le \varepsilon$, then from the matrix perturbation theory [10,12], it is easy to get:

(T3.1) $\|D_A - D_B\|_F \le \varepsilon$

(T3.2) There exists a unitary matrix $X_0\in U(n_1\ldots,n_k)$ such that

$$\|P_0V - WX_0\|_F \le \sqrt{2}\frac{\varepsilon}{\delta}$$

where δ is a real number only depending on the eigenvalues of matrices A and B.

N.B. We omit the proof of here which can be deduced from Hoffman-Wielandt theorem [12] and Davis-Kahan $sin\theta$ theorem [4].

Theorem (T3.3): Given two nearly isomorphic graphs G and H, if $(\overline{P},\overline{X})$ is the argument that minimize the value of (3.1), then

$$\left\|\overline{P}G\overline{P}^T - H\right\|_F \le (\sqrt{2}\frac{\|D_A\|_F + \|D_B\|_F}{\delta} + 1)\varepsilon$$

We omit the proof here, which follows from (T3.1) and (T3.2).

From (T3.3), we can see that if the distance between graph G and H is small enough, then the solution from formula (3.1) will be satisfactory. In other word, Theorem (T3.3) guarantees that the accuracy of the approximate formula proposed here.

3.3 Deducing Umeyama's Formula

In fact, formula (3.1) is an optimization on the space of permutation matrices and unitary matrices, which is a mixed 0-1 non-linear programming. Thus, even all the eigenvalues of matrix A and B are single, it is still not easy to reach the optimization for all graph matching problems. For the case where all the eigenvalues of matrix A and B are single, formula (3.1) can be specified as:

$$\min_{\substack{P\in Perm(n)\\ x_1,\cdots,x_n\in U(1)}} \|P[v_1,\cdots,v_n] - [w_1x_1,\cdots,w_nx_n]\|_F \tag{3.2}$$

where $U(1)$ is the set of all unit complex numbers.

To reach (3.2), we can minimize the distance of the absolute values as an approximation:

$$\min_{P\in Perm(n)} \left\|P|[v_1,\cdots,v_n]| - |[w_1x_1,\cdots,w_nx_n]|\right\|_F \tag{3.3}$$

In formula (3.3), all the numbers x_j an be eliminated. In this way, we get Umeyama's formula (2.2).

The above induction shows the relationship between Umeyama's method and the approximate formula proposed here, and therefore provides a theoretical support to the claims made in section 2. In fact, on one hand, formula (3.1) provide a approximate solution to nearly-isomorphic graph matching with a guaranteed accuracy as specified by (T3.3); on the other hand, with the additional "isolating eigenvalues" constraint, formula (3.1) turns out to be formula (3.2), which, with another additional constraint, i.e., "Dissimilar rows", leads to formula (3.3) that is equivalent to Umeyama's formula (2.2).

APPENDIX E USING EIGEN-DECOMPOSITION METHOD FOR WEIGHTED GRAPH MATCHING

4 Improved Algorithm

In this section, we shall introduce a new graph matching algorithm which can be used for more general cases where the graph pairs just need to be nearly isomorphic.

4.1 Meta-basis for Euclid Space

In formula (3.1), the optimization on both permutation matrices and unitary matrices makes the problem hard to solve. However, if the unitary matrix *X* can be determined somehow beforehand, the problem will become much easier.

The requirement of the unitary matrix *X* for formula (3.1) is due to fact that there are infinite orthonormal basis for a given Euclid space, rather than a unique one. We shall use a n-by-m matrix *V* to denote the orthonormal basis of m-dimensional Euclid space in n-dimensional complex space C^n, where each column of *V* is a vector of the basis. Obviously, each matrix VX, $X \in U(m)$ is also an orthonormal basis of the Euclid space. If we can define a meta-basis which is unique for each Euclid space, then *X* will be eliminated from formula (3.1). In fact,

Let F: $C^{n*n} \to C^n$ be a function which maps an n-by-n matrix to a vector , provided:

$$F(PGP^T) = PF(G) \tag{4.1}$$

for all $P \in Perm(n), G \in C^{n \times n}$

We shall call such functions as edge-to-node attribute functions. A simple case of this kind of function is $F_1(G)=[1,1,..,1]^T$ which maps all n-by-n matrices to a constant vector.

Given a edge-to-node attribute function *F*, we define a new function

$$f : \bigcup_{j \in N} C^{n \times j} \to C^n \quad \text{such that} \quad f(V) = VV^T F(VV^T)$$

where N denotes the set of natural numbers.

It is easy to see that $f(V)$is a vector of Euclid space *V* and $f(VX)$=f(V), for all $V \in C^{n \times j}, X \in U(j)$. Thus, for the given Euclid space V, $f(V)$ is a vector in *V* which is independent on its orthonormal basis. Based on function *f*, we then define a unique meta-basis for any given Euclid space *V* in terms of the recursive manner as described in Table 1.

We call the matrix *Y* defined here a meta-basis of the given Euclid space *V*. It is important to note that, in some cases, *Y* may be a real orthonormal basis of the given Euclid space, while in other cases, *Y* is just a group of orthonormal vectors of the given Euclid space (not necessarily to be a basis – it even can be empty). Obviously, for each Euclid space *V*, the Meta-basis defined in this way is unique.

Table 2. Algorithmic definition of the meta-basis.

```
Function V'=meta-basis(V)
  [n, m]=size(V);          // V is a n-by-m matrix.
  $v = VV^T f(VV^T)$;
  if  norm(v)==0
    V'=[];          // V' is empty. fail to find.
    return;
  else if m ==1
            V' =v/norm(v);
            return;
        else
            v=v/norm(v);
            $V = v \oplus Z$    ;        //orthogonal
decomposition.
            V'        =[v,        meta-basis(Z)];
```

4.2 Graph Matching Using Meta-basis

Formula 3.1 can be rewritten as

$$\min_{\substack{P \in Perm(n) \\ U_j \in U(n_j)}} \left\| P[V_1, V_2, \cdots, V_k] - [W_1 U_1, W_2 U_2, \cdots, W_k U_k] \right\|_F \tag{4.2}$$

where V_j is the eigen-space of the j-th eigenvalue of matrix $A=Ht(G)$, and W_j is the corresponding block matrix formed in the same manner as that of V_j, rather than the eigen-space of the j-th eigenvallue of $B=Ht(H)$.

To eliminated U_j in formula (4.2), we use the meta-basis V_j' of V_j and meta-base $W_j'$of W_j , rather than V_j and W_j themselves. Inthis way, since the meta-base is not dependent on unitary transformation, therefore, formula (4.2) can be simplified as:

$$\min_{P \in Perm(n)} \left\| P[V_1', V_2', \cdots, V_k'] - [W_1', W_2', \cdots, W_k'] \right\| \tag{4.3}$$

N.B. In the case where the meta-basis of V_j and W_j have different numbers of columns, columns from the bigger one will be deleted to make them same.

Experiments have been conducted in applying formula (4.3) to “nearly isomorphic” graphs that do not satisfy constraint 2 and/or constraint 3, with respects to both the calculating accuracy and computational speed.

On one hand, as shown in Fig. 3 and Fig. 4, for nearly-isomorphic graphs, the new algorithm makes great improvements compared with Umeyama’s original algorithm. It reaches matching results as good as that of LPGM. On the other hand, Fig. 5 shows that the computational speed of the new algorithm is very close to that of the EDGM algorithm, but much faster than that of LPGM.

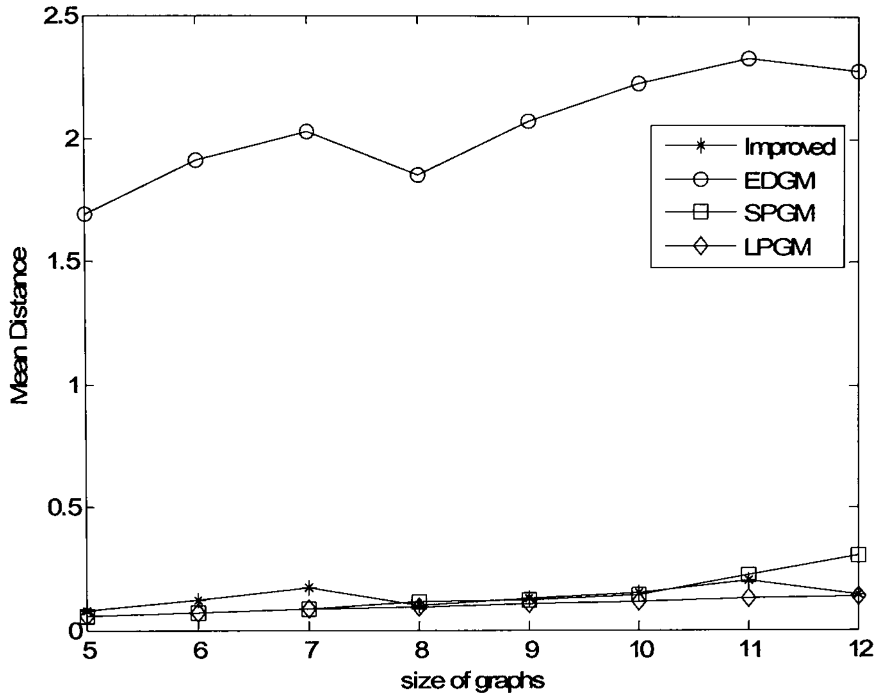


Fig. 3. Improved algorithm for graphs with multiple eigenvalues

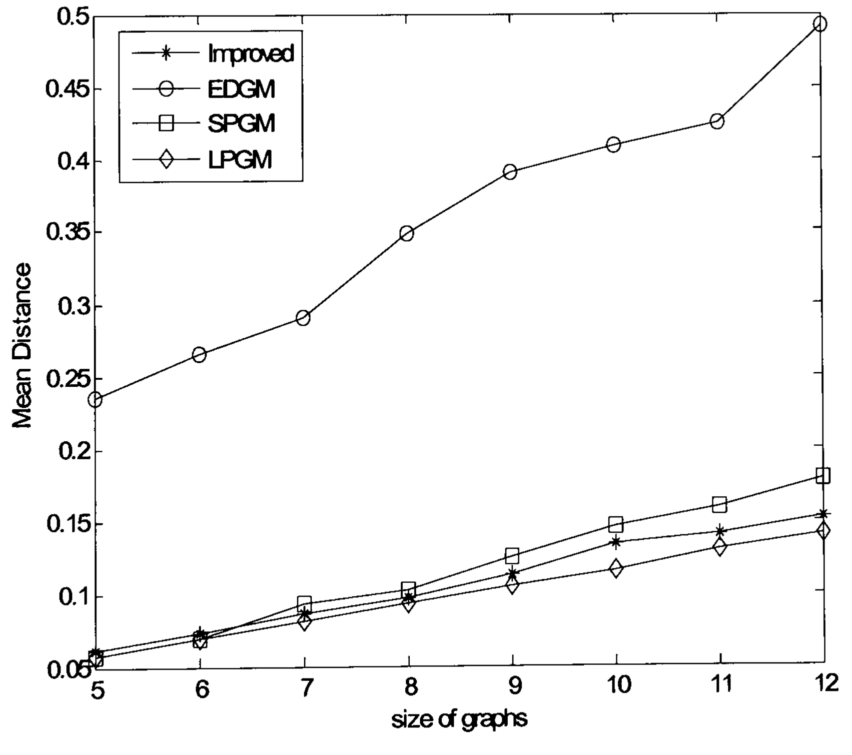


Fig. 4. Improved algorithm for graph pairs with similar rows

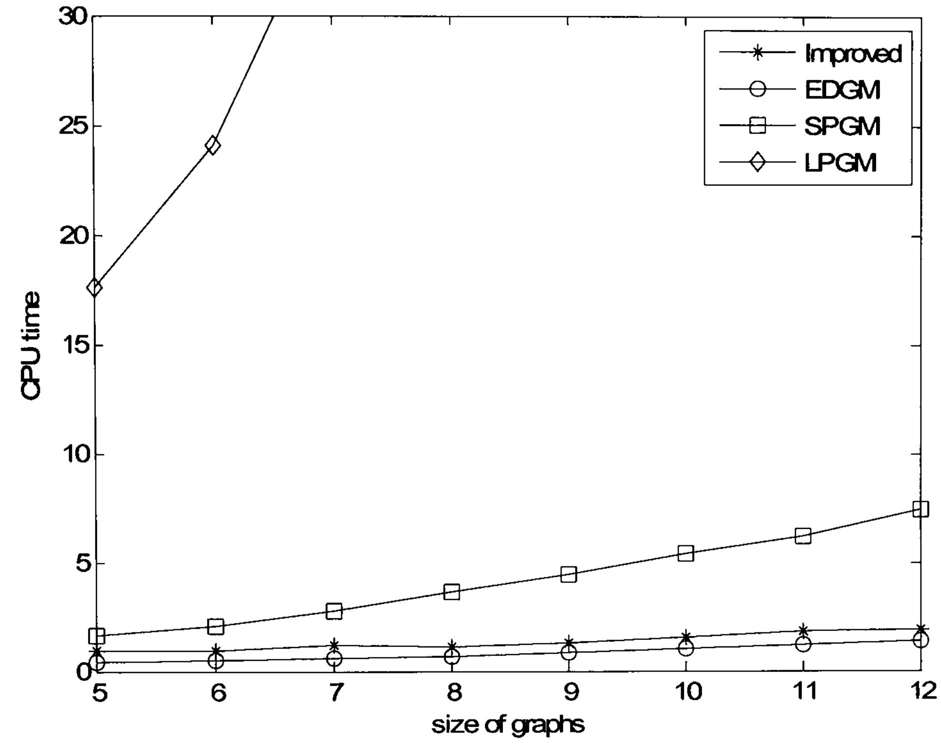


Fig. 5. Calculating time of Improved algorithm

5 Conclusion and Future Work

In this paper, we have specified the three conditions under which Umeyama's approach will work well for graph matching. The approximate formula proposed here can be seen as an extension to Umeyama's formula. Experimental results have shown that, on one hand, for general treatments, the new approach is more accurate that the EDGM method, and on the other hand, it is more efficient than LPGM. Also, it is believed that the new algorithm can be further improved by means of using better edge-to-node attribute functions, rather than the simplest one we have adopted in this paper. Due to the length limit of the paper, we leave this as for future work.

References

1. Almohamad H.A.: A Polynomial Transformfor Matching Pairs of Weighted Graphs. Applied Mathematical Modelling,15(44) (1991) 216-222, Elsevier Science
2. Almohamad H.A. and Duffuaa S.O.: A Linear Programming Approach for the Weighted Graph Matching Problem. IEEE Trans. PAMI, 15(5) (1993) 522-525
3. Barend Jacobus van Wyk.: Kronecker Product Successive Projection and Related Graph Matching Algorithms. Ph.D. diss., University of the Witwatersrand, Johannesburg (2002)

4. Davis, C. and Kahan, W.M.: The Rotation of Eigenvectors by a Perturbation. III, SIAM J. Numer. Anal. 7 (1970)1-46
5. Finch, A.M., Wilson R.C. and Hancock, E.R.: Matching delaunay triangulations by probabilistic relaxation. Proc. of Computer Analysis of Images and Patterns (1995) 350-358
6. Gold S. and Rangarajan A.: A Graduated Assignment Algorithm for Graph Matching. IEEE Trans. PAMI, 18(4) (1996) 377-388
7. Harold W. K.: The Hungarian Method for the assignment problem. Naval Research Logistic Quarterly, 2 (1955) 83-97, Kuhn's original publication
8. Hummel, R. and Zuker, S.: On the Foundations of Relaxation labeling processes. IEEE Trans. PAMI 5 (1983) 267-287
9. Krcmar, M. Dhawan, A.P: Application of genetic algorithms in graph matching. Proc. of the International conference on Neural Networks 6 (1994) 3872-3876. IEEE Press
10. Ninoslav, T.: Relative Perturbation Theory for Matrix Spectral Decompositions. Ph.D. diss., Dept. of Mathematics, Univ. of Zagreb (2000)
11. Sanfeliu, A. and Fu, K.S.: A Distance Measure between Attributed Relational Graphs for Pattern Recognition. IEEE Trans. SMC 13 (1983) 53-363
12. Stewart, G. W. and Sun, J.: Matrix Perturbation Theory. Academic Press, Inc., San Diego (1990)
13. Suganthan, P., Teoh, E. and Mital, D.: Pattern Recognition by Graph Matching Using the Potts MFT Neural Networks. Pattern Recognition 28 (1995) 997-1009
14. Tasi, W.H. and Fu, K.S.: Error-Correcting Isomorphisms of Attributed Relational Graphs for Pattern Recognition. IEEE Trans. SMC 9 (1979) 757-768
15. Ullman, J.R.: An Algorithm for Subgraph Isomorphism. Journal of the Association for Computing Machinery 23(1) (1976) 31-42
16. Umeyama S.: An Eigendecomposition Approach to Weighted Graph Matching Problems. 1988. IEEE Trans. PAMI, 10(5) (1988) 695-703