Reasoning with preferences over temporal, uncertain, and conditional statements

Kristen Brent Venable

Technical Report UBLCS-2005-06

March 2005

Department of Computer Science

University of Bologna

Mura Anteo Zamboni 7
40127 Bologna (Italy)

Recent Titles from the UBLCS Technical Report Series

2003-7 Intersection Types, Lambda Abstraction Algebras and Lambda Theories (Ph.D. Thesis), Lusin, S., March 2003.

2003-8 Towards an Ontology-Guided Search Engine, Gaspari, M., Guidi, D., June 2003.

2003-9 An Object Based Algebra for Specifying A Fault Tolerant Software Architecture, Dragoni, N., Gaspari, M., June 2003.

2003-10 A Scalable Architecture for Responsive Auction Services Over the Internet, Amoroso, A., Fanzieri F., June 2003.

2003-11 WSSecSpaces: a Secure Data-Driven Coordination Service for Web Services Applications, Lucchi, R., Zavattaro, G., September 2003.

2003-12 Integrating Agent Communication Languages in Open Services Architectures, Dragoni, N., Gaspari, M., October 2003.

2003-13 Perfect load balancing on anonymous trees, Margara, L., Pistocchi, A., Vassura, M., October 2003.

2003-14 Towards Secure Epidemics: Detection and Removal of Malicious Peers in Epidemic-Style Protocols, Jelasity, M., Montresor, A., Babaoglu, O., November 2003.

2003-15 Gossip-based Unstructured Overlay Networks: An Experimental Evaluation, Jelasity, M., Guerraoui, R., Kermarrec, A-M., van Steen, M., December 2003.

2003-16 Robust Aggregation Protocols for Large-Scale Overlay Networks, Montresor, A., Jelasity, M., Babaoglu, O., December 2003.

2004-1 A Reliable Protocol for Synchronous Rendezvous (Note), Wischik, L., Wischik, D., February 2004.

2004-2 Design and evaluation of a migration-based architecture for massively populated Internet Games, Gardenghi, L., Pifferi, S., D'Angelo, G., March 2004.

Dottorato di Ricerca in Informatica
Università di Bologna, Padova

Reasoning with preferences over temporal, uncertain, and conditional statements

Kristen Brent Venable

March 2005

Abstract

In this thesis we consider reasoning with preferences in a number of different AI problems. We start with a specific area as, temporal reasoning, for which there is specialized reasoning machinery. In particular, we consider quantitative temporal constraints and we add preferences in such a context. We discuss the complexity for solving temporal constraint satisfaction problems with preferences in general, and we identify a tractable subclass: simple temporal problems with semi-convex preference functions. For this subclass, we consider finding optimal solutions with respect to the (fuzzy) maximin criterion and to the Pareto criterion. In the first case, we propose two solvers based on two different approaches. One enforces the notion of path consistency adapted to deal also with preferences. We show that enforcing such a property is sufficient to solve an STPP and it is a polynomial time algorithm. The second solver, reduces the problem to that of solving several hard constraint problems. We show that also this solver is polynomial. We have also designed a solver that finds Pareto optimal solutions. It applies one of the solvers for fuzzy-optimal solutions several times, to problems obtained from the original one by changing some constraints. Also this solver is polynomial. We have implemented and tested the solvers for fuzzy optimals. The tests, performed on randomly generated problems, show that the second solver is much faster, although it allows to represent less general preferences. We also consider the problem of actually obtaining the preferences on all the temporal constraints. Practically, it may not be feasible to retrieve all such preferences by a user. We,

thus, apply a machine learning technique, inductive learning, to overcome this problem. In particular, we consider inducing local preferences (i.e., preferences on the constraints) from global preferences on complete solutions. We have implemented and tested the learning module. The experimental results are encouraging since they show that the learned constraint problem approximates well the original preference functions defined on the solutions.

Next we study the more general class of temporal constraint satisfaction problems under uncertainty. We tackle the problem of finding optimal schedules that have desirable properties w.r.t. the occurrence of uncertain events. In order to do so, we take the notions of controllabilty (strong, dynamic and weak), from the context of Simple Temporal Problems with Uncertainty and we extend them in order to handle preferences as well. This leads to the definition of Optimal Strong, Dynamic and Weak Controllability. Such definitions require different levels of robustness with respect to uncontrollable events and optimality with respect to preference in all possible situations. We give algorithms to test such properties and we also consider the execution of optimal schedules in the context of dynamic control of plans. The key result in this respect is to show that adding preferences does not make the problem harder. This is important, since in many application the tradeoff between quality and time is a key issue.

We then consider more general scenarios characterized by the coexistence of preference represented in different ways, qualitative (as in CP-nets) and quantitative (as in soft constraints). We also consider problems in which there are hard constraints as well. We show that the two approaches, CP-nets and soft constraints, are complementary and none can substitute the other. However, we give a mapping algorithm that approximates the ordering induced on the outcomes by a set of conditional preference statements via a soft constraint satisfaction problem. We show that it is possible to do so by linearizing the partial ordering of acyclic CP-nets in a consistent way and in a way that respects the fundamental properties of the semantics. The motivation for such an approximation is the need to overcome the bad complexity for dominance testing in the qualitative framework of CP-nets. Next we focus on outcome optimization, that is, finding undominated outcomes, in a constrained environment. In particular, we define a new semantics for constrained CP-nets and we reduce the problem of finding the optimals with respect to this semantics to that of solving a hard constraint problem. We also extend this approach to problems where there are soft constraints as well, taking the position of giving a higher priority to the soft constraints rather than to optimality in the CP-net. Finally, we try to use the hard constraint reduction in the context of another semantics defined for CP-nets. In this respect we give an algorithm able to detect the presence of undominated outcomes even in cyclic CP-nets which is based on solving a hard constraint problem. We use such a technique in order to find solutions that are both undominated in the CP-net and feasible with respect to a set of hard constraints. This solver, whose complexity is exponential, has the advantage of dealing with cyclic CP-nets as well and of allowing the use of powerful state of the art constraint solvers.

Finally, we enlarge our target even more, encompassing also multi-agent scenarios. We consider preference aggregation when the users reason about their own preferences either with CP-nets or soft constraints. In particular, we define the multiple agent framework of mCP-nets and we consider different aggregation criteria. Such criteria are mutuated from the field of voting theory. It is not possible to apply them directly due to the incompleteness of the orderings specified by the agents in our context. In particular the issue of fairness of the aggregation schemes rises. We reconsider the main results on this topic from social welfare theory, namely Arrow's impossibility theorem and Sen's possibility theorems, with respect to partial orders. We show that the results continue to hold if the properties required are conveniently adapted to partial orders.

Acknowledgements

This thesis summarizes many years of research. During this time I have had the privilege of working with many extraordinary people. Each of them have contributed in many different ways to this thesis.

I would like start by thanking my Ph.D. supervisor: Francesca Rossi. She has supported me with strength and dedication for all these years. She has helped me in many different ways. First of all in getting a Ph.D. thesis, indicating interesting lines of research, allowing me to participate in many doctoral programs and graduate schools. The contribution she has given in time and thought to our joint research is extraordinary. We have spent many (great) days discussing and studying many different issues. Finally by being close to her I have learned a lot on how a good AI researcher should behave. In this respect, she has been an outstanding example in many ways. She has taught me to develop new ideas, write good papers and give clear presentations. She showed me how to be a fair reviewer and an good supervisor for undergraduate students. I have acquired many things from her, but I still have many to learn.

Secondly, I would like to thank Toby Walsh: he has been a great co-author. His intuitions have opened for me new perspectives and he has shown me how to look at problems from unusual point of views. He also introduced me to many important people in the field and has supported me in many different occasions.

A special thanks goes to all the scientists with which have shared this research: Alessandro Sperduti from University of Padova (Italy); Lina Khatib and Bob Morris and Paul Morris from NASA Ames (USA); Neil Yorke-Smith from SRI (USA); Carmel Domshlak from Technion (Israel) and Steve Prestwich from 4C (Ireland).

I would also like to thank my reviewers: Thomas Schiex (INRA, France) and Fahiem Bacchus (University of Toronto, Canada). Their comments have been very helpful both for improving the thesis and for suggesting new topics of research.

Contents

List of Figures

Chapter 1

Introduction

The aim of this Ph.D. thesis is to identify classes of problems in which adding preferences gives a substantial gain in terms of expressive power and to develop techniques to handle the representation and reasoning on such preferences in a flexible and efficient way.

1.1 Motivation and Main Goal

Preferences have an important role in many aspects of reasoning, decision making and knowledge representation. In particular preferences are the key to understanding the non crisp aspect of many human behaviors. The representation and handling of soft requirements as well as the representation of priorities, rankings, different levels of importance and hierarchies should be available and efficient in any sophisticated automated reasoning tool. In mathematical decision theory, preferences (often expressed as utilities) are used to model people's economic behavior. In Artificial Intelligence, preferences help to capture agents' goals. In databases, preferences help in reducing the amount of information returned in response to user queries. In philosophy, preferences are used to reason about values, desires, and duties. Surprisingly, there has been so far very little interaction between those areas. The difference source fields, as well as variations in terminology, make the results obtained in one area difficult to use in another.

Indeed preferences are gaining more and more attention in AI and in particular in the constraint programming area. Moreover, preferences are essential to treat conflicting information in nonmonotonic reasoning, reasoning about action and time, planning diagnosis, configuration, and other areas in knowledge representation and reasoning.

AI permits complex preference representations and thus allows to reason with and about preferences. Hence, AI provides a new perspective for formalizing information that is essential for many decision making problems, e.g. web-based configuration, scheduling, robot planners [Jun02]. A rich language for encoding preference information would include qualitative and quantitative representations as well, and it should span from comparative preferences to ordinary utility functions [Doy96].

There are many issues to be addressed about preferences from an AI point of view.

The first is preference specification and representation, i.e., which formalisms can be used to model the preferences of an agent. In this respect, contributions have been brought from studying the axiomatic properties of preferences, as well as logics of preferences or their topological and algebraic structures. In a multiagent scenario, instead, core issues are preference composition, merging and aggregation, as well as preference elicitation and learning priorities, conflict resolution and belief revision.

Here we tackle the problem of adding preferences to AI problems from a constraint satisfaction point of view. In constraint programming preferences in the form of soft constraints are used to reduce search efforts. Preferences are complementary to constraints and represent an AI counterpart to objective or utility functions. In particular, we identify classes of problems to which adding preferences constitutes a consistent gain in expressiveness and does not cause a drawback in terms of complexity. In particular, we will consider semiring-based soft constraints as a preference reasoning framework [BMR95, BMR97]. We will show how such an approach makes it possible to add the expressiveness power of preferences in a natural way to different AI problems without causing a complexity blowup. We also want to investigate how such preference reasoning techniques can coexist and cooperate with other preference representation frameworks, such has conditional preference networks (CP-nets) [BBD$^+$04a]. From the multiagent point of view, we will consider how the proposed methods of preference representation impact on preference aggregation rules. In particular, we will reconsider results taken from social welfare theory in the context of automated aggregation of preferences in societies of artificial agents.

1.2 Objectives

Given a well-defined mechanism to express and combine preferences, the goal is to find the best solution. Therefore, solving soft constraints becomes an optimization task, harder than the satisfaction tasks in classical CSP. Sometimes optimization becomes multi-objective, making the solving process even more complex. The interest of the constraint programming community on soft constraints has increased in the last years. Several theoretical frameworks have been developed to allow soft constraints in problem modeling. For these frameworks, new algorithms have been built. Generally speaking, a soft constraint is just a classical constraint plus a way to associate, either to the entire constraint or to each assignment of variables, a certain element, which is usually interpreted as a level of preference or importance. Such levels are usually ordered and the order reflects the idea that some levels are better than others. Moreover, one has also to say, via suitable combination operators, how to obtain the level of preference of a global solution from the preferences in the constraint.

As said above, many formalism have been developed to describe one or more classes of soft constraints. For instance consider fuzzy CSPs [RHZ76, DDP93, Rut94], where the crisp constraints are extended with a level of preference represented by a real number between 0 and 1, or probabilistic CSPs [FL93], where the probability to be in the real problem is assigned to each constraint. Some other examples are partial CSPs [FW92] or valued CSPs [SFV95] where a preference is assigned to each constraint, in order to also solve over constrained problems.

As mentioned before, we have chosen to use one of the most general frameworks to deal with soft constraints [BMR95, BMR97]. The framework is based on a semiring structure that is equipped with the operations needed to combine the constraints present in the problem and to choose the best solutions. According to the choice of the semiring, this framework is able to model all the specific soft constraint notions mentioned above. The semiring-based soft constraint framework provides a structure capable of representing in a compact way problems with preferences.

We start adding preferences to a specific class of problems, that is, to problems concerning reasoning with time. In particular, we consider the quantitative approach of Temporal Constraint Satisfaction Problems (TCSPs) [DMP91]. In such a framework temporal constraints are defined as sets of intervals containing the allowed duration or interleaving times of events. We choose to start from this specific problem class since its structure is particularly suitable to being augmented with preferences. Moreover, specific tractable subclasses (TCSPs are hard in general) have been identified and ad hoc machinery has been developed to reason efficiently over them. In particular, we will focus on the tractable subclass of Simple Temporal Problems (STPs), that is problems where there is a single interval per constraint. We will add preferences to such a constraint problem simply by mapping each element of the interval into a preference. We call these new soft constraint satisfaction problems Simple Temporal Problems with Preferences (STPPs). The semiring-based structure allows to combine and compare such preferences. Thus, using the underlying structure, we consider designing solvers that find optimal solutions. In particular, we identify the tractability assumptions needed to ensure polynomial time complexity to solvers for such a class of problems. Such assumptions are defined on the way preferences are aggregated which must be idempotent, but also on the shape of the preference functions, which must be semi-convex. We thus focus on soft constraints based on the fuzzy semiring [BMR95, BMR97], which meets the assumption on the combining operator since it aggregates preferences with min and compares them using max. We consider two different ways of finding fuzzy optimal solutions. The first one is based on the extension of the local consistency rule known as path consistency [Mon74], the second takes a "divide and conquer" approach in the sense that it decomposes the problem with preferences in several hard constraint sub-problems. We also investigate finding optimals with respect to another criterion, that is, Pareto optimals. Any such optimal assignments is characterized by the fact that no other assignment has a higher preference on every constraint. We propose a solver for finding a Pareto optimal solution that iteratively applies one of the solvers for fuzzy optimals to problems obtained from the original one by changing each time some constraint. The complexity of all these solvers is polynomial. We have tested the two solvers that find fuzzy optimal solutions on randomly generated problems and we have compared their performance. The second solver is faster than the first which is penalized by the discrete representation of constraints it requires. Thus we conclude that adding preferences in this temporal scenario does not change the complexity of the classical version without preferences, while incrementing in a considerable way the expressive power.

STPPs give us a structure capable of representing in a compact way a problem where there are some preferences over the execution time of temporal events. However, if the problems are of a certain size, as often happens in real life, it is infeasible to ask the user to define all the preferences involved. The user will be more incline to give more general, or global, information on his or her preferences. We have identified in Machine Learning (ML) techniques a powerful tool to embed in the system the capability of inducing local information on preferences from global information. We use such a technique to induce local temporal preferences from preferences given on entire solutions. After the learning phase has terminated, the obtained problem is solved using one of the algorithms we have proposed and the optimal solutions are shown to be optimal also with respect to the original preferences given on the complete assignments. We have tested the learning module with very good results on the behavior of the learned problems w.r.t. the original ranking functions defined on solutions.

Preferences are one type of soft information present in AI problems. Another very important one is uncertainty. We investigate the relation between these two aspects by considering temporal problems with uncertainty and embedding soft constraints as a preference representation in such structures. In particular we focus on Simple Temporal Problems with Uncertainty [VF99, MMV01], a quantitative approach similar to STPs with the exception that constraints on uncontrollable events are allowed. The uncertainty taken into consideration in such a class of problems is the lack of knowledge on when an uncontrollable event will happen (only an upper and lower bound of the time of occurrence is given). We add preferences in a similar way to STPPs, i.e., mapping each element of the intervals into a preference. In this way we define a class of problems which we call Simple Temporal Problems with Preference and Uncertainty (STPPUs). In such problems, the goal is not to establish a consistent assignment to all the variables that satisfies all the constraints, since some of the variables cannot be decided by the agent, but rather to find assignments of the controllable variables that feature certain levels of robustness with respect to the uncertain events. This notion is known as controllability. Here we extend such a notion by adding optimality with respect to preferences. In particular we extend the three notions of controllability, (that is, strong, dynamic and weak [VF99]), that differ on the strength of the robustness to uncertainty they require, by defining optimal strong, optimal dynamic and optimal weak controllability. In particular, we look for solutions that are optimal in one (optimal weak), some (optimal dynamic), or all (optimal strong) scenarios. For each property, we give an algorithm to test if a given STPPU possess it, and we show that the complexity of this test is in the same class as the hard constraint case. For optimal dynamic controllability we also consider the execution of an optimal schedule in an incremental way.

Next we take a broader perspective and consider more general preference scenarios. We focus on studying the relation that may occur between semiring-based soft constraints and other preference representation framework. In particular we consider Conditional Preference networks (CP-nets) [BBHP99, BBD$^+$04a], a qualitative framework proposed to handle conditional preference statements. Such statements are expressed over values of given features which are used to decompose the space of outcomes. In particular, such statements are "ceteris paribus", that is, the conditional dependence of the preferences on a given attribute from the values assigned to other attributes is given "all else being equal", that is, under the assumption that all other feature's values remain fixed. We first show that CP-nets and soft constraints are incomparable and thus complementary as far as representational power. We then propose to approximate the ordering induced by CP-nets on the outcomes using soft constraints. The main motivation is to overcome the bad complexity of testing whether an outcome dominates another one in CP-nets. We show that such a linearization is possible while maintaining both the consistency with the ordering of the CP-net and the ceteris paribus assumption. Moreover, such an approximation is obtained by a mapping algorithm which is polynomial in the size of the CP-net. We then focus our attention to finding optimal outcomes in scenarios where hard and soft constraints coexist with conditional preferences. In such a context we propose to reduce the problem of finding optimal outcomes to a hard constraint satisfaction problem. In particular, we study this method for finding optimals according to two different semantics. The first one has attractive complexity features but in general induces a weak ordering which leaves a lot of incompatibility among outcomes (thus allowing many optimals). The second one, instead, induces a more intuitive ordering.

Finally we generalize our reasoning to scenarios where there is more than one agent, and the agents interact in terms of their preferences. In particular, we define a framework, mCP-nets, which handles the representation of each agent's preferences using an extended version of CP-nets where agents are allowed to be indifferent to some features. The proposed framework is equipped with several voting semantics used to aggregate the individual preferences into global preferences. Such aggregation criteria are derived from the most popular ones in social welfare theory [Str80, Kel87], which we then adapt to the context of societies of artificial agents. First, we study them in terms of computational complexity and formal mathematical properties, as subsumption and transitivity. Then, we study other properties such as fairness, and we consider classical results on fairness of social welfare functions, as Arrow's impossibility theorem [Arr51, Kel78] and Sen's possibility theorem [Sen70], in this new context. The main difference is that, in contrast to what is assumed in social welfare scenarios, our agents describe their preference using partial orders and not total orders. We show that similar results still hold, although some of the assumptions need to be suitably adapted to deal with incomparability.

1.3 Publications

The work presented in this thesis has been developed with many different research groups and it has partially been published in the proceedings of international conferences, as we describe in what follows.

The work on STPPs has been developed in collaboration with Francesca Rossi and Alessandro Sperduti from University of Padova. The implementation and testing was performed during two Student Summer Programs at NASA Ames, Moffett Field, California, with Lina Khatib, Robert Morris and Paul Morris (NASA Ames Research Center). The following papers have been published on STPPs:

- F. Rossi, K. B. Venable, A. Sperduti, L. Khatib, P. Morris, R. Morris. *Solving and Learning Soft Temporal Constraints: Experimental Scenario and Examples.* CP-01 Workshop on Soft Constraints (Soft'01), pages 69-81, Cyprus, 2001.

- F. Rossi, K. B. Venable, A. Sperduti, L. Khatib, P. Morris, R. Morris. *Two solvers for tractable constraints with preferences.* AAAI Workshop on Preferences in AI and CP: Symbolic Approaches, July 28th-29th, Edmonton, Alberta, Canada; 2002.

- F. Rossi, K. B. Venable, A. Sperduti, L. Khatib, P. Morris, R. Morris. *Solving and Learning Soft Temporal Constraints: Experimental Scenario and Examples.* In Proc. of the Eighth International Conference on Principles and Practice of Constraint Programming (CP 2002); Sept. 8th- 13th, Cornell University, Ithaca, NY, USA; pages 249-263, Springer LNCS 2470, 2002.

- L. Khatib, P. Morris, R. Morris and K. B. Venable. *Tractable Pareto Optimization of Temporal Preferences.* In Proc. of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003); August 2003, Acapulco, Mexico; pages 1289-1294, Morgan Kauffman, 2003.

- National Prize for Best Laurea Thesis in AI 2001: F. Rossi, K. B. Venable, A. Sperduti, L. Khatib, P. Morris, R. Morris. *Solving and Learning Soft Temporal Constraints.* AI*IA Notizie, Volume 4, pages 22-26, 2002.

The research on STPPUs is in collaboration with Neil Yorke-Smith from IC-Parc London (presently at SRI International, California) and Francesca Rossi. The following papers have been published on this subject:

- K. B. Venable, Neil Yorke-Smith. *Simple Temporal Problems with Preferences and Uncertainty.* In Proc. of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003); June 9-13, Trento, Italy; pages 132-136, AAAI, 2003.

- F. Rossi, K. B. Venable, and N Yorke-Smith, *Preferences and Uncertainty in Simple Temporal Problems*. In Proc. CP03 Workshop: Online-2003 (International Workshop on Online Constraints Solving - Handling Change and Uncertainty) , Kinsale, Ireland, 2003.

- F. Rossi, K. B. Venable, Neil Yorke-Smith. *Temporal Reasoning with Preferences and Uncertainty*. Poster paper in Proc. of the Eighteenth International Joint

Conference on Artificial Intelligence (IJCAI 2003), pages 1385-1386, Morgan Kaufmann, 2003.

- F. Rossi, K. B. Venable and N. Yorke-Smith. *Controllability of Soft Temporal Constraint Problems*. In Proc. of the 10th International Conference on Principles and Practice of Constraint Programming (CP-04); September 2004, Toronto, Canada; pages 588-603, Springer LNCS 3258, 2004.

The work on qualitative and quantitative preference statements is a joint research with Francesca Rossi, Toby Walsh from NICTA (Australia), Carmel Domshlak from Technion (Israel), and Steve Prestwich from 4C (Ireland). Our joint publications are:

- C. Domshlak, F. Rossi, K. B. Venable and T. Walsh. *Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques.* In Proc. of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003); August 2003, Acapulco, Mexico; pages 215-220, Morgan Kauffman, 2003.

- F. Rossi, K. B. Venable and T. Walsh. *CP-networks: semantics, complexity, approximations and extensions.* CP'02 Workshop: Soft'02 (Fourth International Workshop on Soft Constraints), Cornell University, Ithaca, NY, USA; Sept. 8th, 2002.

- F. Rossi, K. B. Venable, T. Walsh. *mCP-nets: representing and reasoning with preferences of multiple agents*. In Proc. of the Nineteenth National Conference on Artificial Intelligence (AAAI-04); July 25-29, San Jose, California; pages 729-734, AAAI Press / The MIT Press, 2004.

- S. Prestwich, F. Rossi, K. B. Venable and T. Walsh. *Constrained CP-nets*. Proceedings of the Joint Annual Workshop of ERCIM/CoLogNet on Constraint Solving and Constraint Logic Programming (CSCLP'04); Lausanne, 2004.

- S. Prestwich, F. Rossi, K. B. Venable, T. Walsh. *Softly Constrained CP-nets*. Doctoral Paper in Proc. of the 10th International Conference on Principles and Practice of Constraint Programming (CP-04); September 2004, Toronto, Canada; page 806, Springer LNCS 3258, 2004.

- F. Rossi, K. B. Venable and T. Walsh. *Aggregating Preferences cannot be fair*. In Proc. of the Workshop "Agenti e Vincoli Modelli e Tecnologie per Dominare la Complessità " colocated with Convegno Associazione Italiana Intelligenza Artificiale (AI*IA 2004), Perugia, Italy, 2004.

- F. Rossi, K. B. Venable and T. Walsh. *Aggregating Preferences cannot be fair*. To appear: Intelligenza Artificiale, 2005.

1.4 Structure of the Thesis

The thesis is organized as follows:

- **Chapter 2**. We give the background on Temporal Constraint Satisfaction Problems, Soft Constraints and Inductive learning. We define a new framework called Temporal Constraint Satisfaction Problems with Preferences and we identify some tractability assumptions. This allows us to characterize the tractable subclass of problems called Simple Temporal Problems with Preferences and Uncertainty. We then give three solvers, where the first two find fuzzy-optimal solutions and the last one finds optimal solutions according to the Pareto semantics. Next we give experimental results on randomly generated problems for the solvers that find fuzzy-optimals, and we compare their performance. Finally we consider inducing local preferences on constraints from global preferences given on solutions using the machine learning technique of gradient descent. Experimental results for the learning module are given.

- **Chapter 3**. We consider the coexistence of preferences and uncertainty in temporal problems and we start by giving the background on Simple Temporal Problems with Uncertainty. We then augment such approach with preferences, defining STPUs with Preferences (STPPUs). Next we consider the three classical notions of controllability (strong, dynamic and weak) and we extend them in order to handle preferences. For each extended notion, we give an algorithm that tests whether a given STPPU satisfies the property. Moreover, we show that such tests in the context of preferences remain in the same complexity class of the tests with only hard constraints. Finally we outline a guiding strategy for applying the proposed algorithms.

- **Chapter 4**. The combination of qualitative and quantitative preference representation in constrained environments is considered. We start giving the background on decision theory, conditional preference statements and conditional preference networks. We then give a mapping algorithm which allows us to approximate the ordering induced by a set of conditional statements on outcomes with a soft constraint problem. We show that such approximation is consistent with the original ordering and respects the properties of the original semantics. The main gain from this approximation is allowing dominance testing in linear time rather than in exponential time as it is in CP-nets. Next we consider the problem of finding optimal outcomes in problems with conditional preference statements, hard and soft constraints and we give two different algorithms for finding optimals according to two different semantics.

- **Chapter 5**. We consider multiagent scenarios and propose a framework in which agents reason about their preference using soft constraints or CP-nets. We start by giving the background on social welfare theory. We consider various voting semantics taken among the most popular in social welfare theory. We examine the formal properties of each aggregation criteria. Finally we consider some of the most important results on fairness in the context of preference aggregation in societies of artificial agents and we extend some of the well-known impossibility and possibility results on fairness to our context.

- **Chapter 6**. We summarize the results of the thesis, highlighting the lessons that have been learned. We then discuss future directions for further work.

Chapter 2

Temporal Reasoning with Preferences

This thesis is about preferences in AI problems. Thus, we will start with a specific problem (temporal reasoning) which has been widely studied and for which specific machinery has been developed. We believe this is an interesting starting point, since the problem at hand is well defined and structured. In the following chapters we will move into considering more and more general scenarios in which adding preferences constitutes an advantage.

Often we need to work in scenarios where events happen over time and preferences are associated to event distances and durations. Soft temporal constraints allow one to describe in a natural way problems arising in such scenarios.

In general, solving soft temporal problems require exponential time in the worst case, but there are interesting subclasses of problems which are polynomially solvable. In this chapter we identify one of such subclasses giving tractability results. Moreover, we describe three solvers for this class of soft temporal problems, and we show some experimental results. The random generator used to build the problems on which tests are performed is also described. We also compare the two solvers highlighting the tradeoff between performance and robustness.

Sometimes, however, temporal local preferences are difficult to set, and it may be easier instead to associate preferences to some complete solutions of the problem. To model everything in a uniform way via local preferences only, and also to take advantage of the existing constraint solvers which exploit only local preferences, we show that machine learning techniques can be useful in this respect. In particular, we present a learning module based on a gradient descent technique which induces local temporal preferences from global ones. We also show the behavior of the learning module on randomly generated examples.

2.1 Motivation and Chapter Structure

Preferences on temporal events and duration are commonly used and crucial in many areas as planning and scheduling. In particular, several real world problems involving the manipulation of temporal information in order to find an assignment of times to a set of activities or events can naturally be viewed as having preferences associated with local temporal decisions, where by a local temporal decision we mean one associated with how long a single activity should last, when it should occur, or how it should be ordered with respect to other activities.

For example, an antenna on an earth orbiting satellite such as Landsat 7 must be slewed so that it is pointing at a ground station in order for recorded science or telemetry data to be downlinked to earth. Assume that as part of the daily Landsat 7 scheduling activity a window $W = [s, e]$ is identified within which a slewing activity to one of the ground stations for one of the antennae can begin, and thus there are choices for assigning the start time for this activity. Antenna slewing on Landsat 7 has been shown to occasionally cause a slight vibration to the satellite, which in turn might affect the quality of the image taken by the scanning instrument if the scanner is in use during slewing. Consequently, it is preferable for the slewing activity not to overlap any scanning activity, although, since the detrimental effect on image quality occurs only intermittently, this disjointness is best not expressed as a hard constraint. Thus if there are any start times t within W such that no scanning activity occurs during the slewing activity starting at t, then t is to be preferred. Of course, the cascading effects of the decision to assign t on the scheduling of other satellite activities must be taken into account as well. For example, the selection of t, rather than some earlier start time within W, might result in a smaller overall contact period between the ground station and satellite, which in turn might limit the amount of data that can be downlinked during this period. This may conflict with the preference for attaining maximal contact times with ground stations, if possible.

Reasoning simultaneously with hard temporal constraints and preferences, as illustrated in the example just given, is crucial in many situations. However, in many temporal reasoning problems it is difficult or impossible to specify a local preference on durations. In real world scheduling problems, for example, it is sometimes easier to see how preferable is a solution, but it may be virtually impossible to specify how specific ordering choices between pairs of events contribute to such global preference value.

This scenario is typical in many cases. For example, it occurs when there is no precise function which describes the assignment of a preference value to a global solution. This may happen for example when we just have an expert, whose knowledge is difficult to code as local preferences, but who can immediately recognize a good (or bad) global solution. Another typical case occurs when the environment in which the solver will work presents some level of uncertainty. In this case, we could have the local preferences, but their effect on a global solution could depend on events which are not modeled within the problem.

On the other hand, if all the knowledge could be coded as local preferences, it could be used as heuristics to guide the scheduler to prefer local assignments that were found to yield better solutions. Here we propose to solve this problem by automatically generating local temporal preference information, from global preferences, via a machine learning approach, and using a representation of local preferences in terms of soft constraints.

The work presented in this chapter has appeared in the proceedings of the following conferences and workshops.

- F. Rossi, K. B. Venable, A. Sperduti, L. Khatib, P. Morris, R. Morris. *Solving and Learning Soft Temporal Constraints: Experimental Scenario and Examples.*

CP01 Workshop on Soft Constraints (Soft'01), Cyprus, 2001.

- F. Rossi, K. B. Venable, A. Sperduti, L. Khatib, P. Morris, R. Morris. *Two solvers for tractable constraints with preferences.* AAAI Workshop on Preferences in AI and CP: Symbolic Approaches. Edmonton, Alberta, Canada; July 28th-29th, 2002.

- F. Rossi, K. B. Venable, A. Sperduti, L. Khatib, P. Morris, R. Morris. *Solving and Learning Soft Temporal Constraints: Experimental Scenario and Examples.* Proc. Eighth International Conference on Principles and Practice of Constraint Programming (CP 2002), Springer LNCS 2470, pp. 249-263, Cornell University, Ithaca, NY, USA; Sept. 8th- 13th, 2002,

- National Prize for Best Laurea-Thesis in AI 2001: F. Rossi, K. B. Venable, A. Sperduti, L. Khatib, P. Morris, R. Morris. *Solving and Learning Soft Temporal Constraints.* AI*IA Notizie, Volume 4, pp. 22-26, Anno 2002.

- L. Khatib, P. Morris, R. Morris and K. B. Venable. *Tractable Pareto Optimization of Temporal Preferences.* Proc. Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003), pp. 1289-1294; Acapulco, Mexico; Morgan Kauffman, 2003.

This Chapter is organized as follows. In Section 2.2 we describe the background on which our research is based. In particular, as a temporal reasoning system, Temporal Constraints are considered, while Soft Constraints are the framework used to handle preferences. In Section 2.3 we give the formal definition of Temporal Constraint Problems with Preferences. After showing that TCSPPs are NP-hard, Simple Temporal Problems with Preferences (STPPs), that is, TCSPPs with one interval on each constraint, are studied. In particular, a subclass of STPPs, characterized by assumptions on both the underlying semiring and the shape of the preference functions, is shown to be tractable. In Section 2.4 we describe two solvers for STPPs, one using a local consistency technique (Path-solver), and one based on a decomposition approach (Chop-solver). After describing the solvers, we give details about the implementation and experimental testing of both. In Section 2.4.3 we describe an enhanced solver that finds schedules that are optimal both globally and locally (Pareto undominated). In Section 2.6 learning local temporal preferences is taken into consideration. After a brief introduction on inductive learning techniques, we describe the module we have designed and give the experimental results. In Section 2.7 we consider the present literature related to our work.

2.2 Background

In this section we give an overview of the background on which our work is based. First we describe Temporal Constraint Satisfaction Problems [DMP91], a well known framework used for handling time constraints. We will use a similar approach to handle constraints on durations of events and on interleaving times between events. Secondly, we consider semiring-based Soft Constraint Satisfaction Problems introduced in [BMR97]. We will use this framework, to handle preferences over the durations of events or their interleaving times.

2.2.1 Temporal Constraint Satisfaction Problems (TCSPs) and Simple Temporal Problems (STPs)

Researchers coming from many different areas, as philosophy, psychology and physics, have been fascinated with the concept of time and have been struggling to capture its essence in order to find a suitable formal representation. Recently, time has shown to play an important role in Artificial Intelligence as well. In fact, almost any area within AI, from planning to knowledge representation, from commonsense reasoning to qualitative reasoning, involves reasoning about time in some way.

A temporal reasoning system should consist of a temporal knowledge base, procedures to check its consistency, a query-answering mechanism and a inference capability to discover new information. Temporal Constraint Satisfaction Problems [DMP91], built on the solid bases of Constraint Satisfaction Problems, have been shown to fulfill all the above requirements. We will know describe the fundamentals of this temporal reasoning system.

Constraint Satisfaction Problems (CSPs)

A *constraint* is a relation which is defined on a set of variables each having a domain containing the values that it can take. The role of a constraint is to restrict the possible simultaneous assignments of the variables. *Constraints* are characterized by the number of variables they involve. A constraint is said to be *unary* if it involves only one variable, *binary* if it involves two variables and, in general, *n-ary* the number of constrained variables is n. In this thesis we will are mainly interested in binary constraints.

Definition 2.1 *A constraint satisfaction problem (CSP) is a triple* $\langle X, D, C \rangle$ *where:*

- $X = \{X_1, \ldots, X_n\}$ *is the set of variables;*
- $D = \{D(X_1), \ldots, D(X_n)\}$ *is the set containing the domain of each variable;*
- C *is a set of constraints defined on the variables in* X*, where each constraint* $c(X_h, \ldots, X_k)$ *on variables* $X_h, \ldots, X_k$ *is a subset of the Cartesian product* $D(X_h) \times \cdots \times D(X_k)$ *containing all the allowed tuples.*

A *variable assignment* to X_i is its instantiation to a value in its domain $D(X_i)$. Given CSP $P = \langle X, D, C \rangle$ an assignment to a subset of variables $A \subset X$ is said to be *partial*, while an assignment to all the variable in X is said to be $\mathtt{total}$. Given a total or a partial assignment t and a set of variables $B = \{X_i, \ldots, X_j\}$, the projection of assignment t on the variables in B, written $t_{\downarrow B}$, is the set of values assigned by t to the variables in B. An assignment t is said to be *consistent* if it satisfies all the constraints on the variables it involves. A total assignment s to the variables in X is said to be a *solution* if it is a consistent assignment, that is, if it satisfies *all* the constraints in C. Solving a CSP, thus, means finding an assignment to all the variables satisfying all the constraints. It has been shown that, in general, solving a CSP is NP-hard and is thus intractable [Mac77].

The definition of consistency given above is *global*. However, weaker notions of consistency have been shown to be very useful in practice.

Definition 2.2 (k-consistency) *Given a CSP $P = \langle X, D, C \rangle$, P is said to be k-consistent iff given any consistent instantiation of any $k-1$ distinct variables, there exists an instantiation of any kth variable such that the k values taken together satisfy all of the constraints among the k variables.*

In this thesis, we will be interested in particular in 3-consistency, also known as $\mathit{path\ consistency}$. It processes networks of size 3 (triangles), checking if every consistent assignment to two variables can be extended consistently to an assignment to the third variable.

Despite the underlying intractability, CSPs have been widely proven to be very useful both in modeling and solving many real-life applications [Sim01, Ros00, Wal96]. In fact, many algorithms and procedures are dedicated to solving CSPs, and from the $\mathit{Constraint\ Programming}$ corpus.

Temporal Constraint Satisfaction Problems (TCSPs)

In a Temporal Constraint Satisfaction Problem information is represented as a CSP where variables denote event times and constraints represent the possible temporal relations between them. Two types of tasks are of interest: (1) checking the consistency of the given temporal information and (2) finding a solution, i.e. a schedule satisfying all the constraints. TCSPs have been divided into three main classes *qualitative point constraints*, *qualitative interval constraints* and *metric point constraints* [SV98]. We will briefly describe the first two, and give more details on the third since it will be the representation used in this thesis.

Qualitative Point Constraints Qualitative Point TCSPs have been introduced in [VKB89]. In such problems variables represent time points and the relations used in the most important of the algebras defined in this context, the *Point Algebra (PA)*, are $\{\emptyset, <, =, >, \geq, \leq, ?, \neq\}$. The main contributions to PA have been IxTeT [GA89], the work by van Beek [vB89] and TimeGraph-II[GS95].

In [GA89] a TCSP based on the PA is translated into a graph with only $\leq$ and $\neq$ relations. The consistency is checked collapsing each $\leq$ loop into a single vertex and then checking if any two collapsed vertices are connected by a $\neq$ edge. If so, then the TCSP is inconsistent.

This is equivalent to finding the Strongly Connected Components as proposed in [vB89]. Using Tarjan's algorithm [Tar72] for computing the SCC allows to check the consistency and to find a solution of a PA-TCSP in linear time ($O(n+e)$ where n is the number of variables and e the number of edges).

Systems such as IxTeT and TimeGraph-II use internal representations that take advantage of the inherent structure of temporal information. In particular, IxTeT uses *indexed spanning trees*, while TimeGraph-II arranges time points into *chains*.

Qualitative Interval Constraints In the framework introduced by Allen in [All83] variables represent time intervals and the set of basic temporal relations between such intervals is { *before, after, meets, met by, overlaps, overlaps_by, during, contains, equals, starts, started_by, finishes, finished_by* }. Such interval relations can be expressed by conjunctions of PA relations defined on the start and end points of the intervals. In general, deciding the consistency of an $\mathsf{Interval}$-TCSP is NP-complete, as shown in [VKB89]. A number of tractable subclasses, however, have been identified. For example Pointisable TCSPs [VKB89] and the more general Horn-clause based class described in [NB95].

Metric Point Constraints As anticipated, we will now devote more space to describe *metric point constraints*, a *quantitative* framework for handling temporal information first introduced in [DMP91]. This is of particular interest for us, since the models we propose to handle temporal preferences in certain (Chapter 2) and uncertain (Chapter 3) scenarios, is based on this representation of time.

Also in metric TCSPS the variables represent time points, however the relations among them are expressed using quantitative constraints

Definition 2.3 (metric TCSP) *A Temporal Constraint Satisfaction Problem (TCSP) consists of a set of variables* $\{X_1, \ldots, X_n\}$ *and a set of unary and binary constraints over pairs of such variables. The variables have continuous or discrete domains; each variable represents a time point. Each constraint is represented by a set of intervals* [1] $\{I_1, \ldots, I_k\} = \{[a_1, b_1], \ldots, [a_k, b_k]\}$. *A unary constraint* T_i *restricts the domain of variable* X_i *to the given set of intervals; that is, it represents the disjunction* $(a_1 \leq X_i \leq b_1) \vee \ldots \vee (a_k \leq X_i \leq b_k)$. *A binary constraint* T_{ij} *over variables* X_i *and* X_j *constrains the permissible values for the distance* $X_j - X_i$*; it represents the disjunction* $(a_1 \leq X_j - X_i \leq b_1) \vee \ldots \vee (a_k \leq X_j - X_i \leq b_k)$. *Constraints are assumed to be given in the canonical form in which all intervals are pair-wise disjoint.*

A TCSP can be represented by a directed *constraint graph* where nodes represent variables and an edge $X_i \longrightarrow X_j$ indicates constraint T_{ij} and it is labeled by the interval set. A special time point X_0 is introduced to represent the "beginning of the world". All times are relative to X_0; thus, we can treat each unary constraint T_i as a binary constraint T_{0i}. Given a TCSP, a tuple of values for its variables, say $\{v_1, \ldots, v_n\}$, is called a $\mathtt{solution}$ if the assignment $\{X_i = v_1, \ldots, X_n = v_n\}$ does not violate any constraint. A TCSP is said to be *consistent* if it has at least a solution.

Consider the following example, taken from [DMP91, Dec03]:

Example 2.1 John travels to work either by car (30-40 minutes) or by bus (at least 60 minutes). Fred travels to work either by car (20-30 minute) or in a carpool (40-50 minutes). Today John left home between 7:10 and 7:20 A.M., and Fred arrived at work between 8:00 and 8:10 A.M.. We also know that John arrived at work 10-20 minutes after Fred left home.

[1]For simplicity, we assume closed intervals; however the same applies to semi-open intervals.

The TCSP corresponding to the example has 5 variables: *John-leaves-home, John-arrives-work, Fred-leaves-home, Fred-arrives-work, 7:00 A.M.* The last variable corresponds to the special time point x_0 mentioned above. The constraint graph corresponding to the example is shown in Figure 2.1.

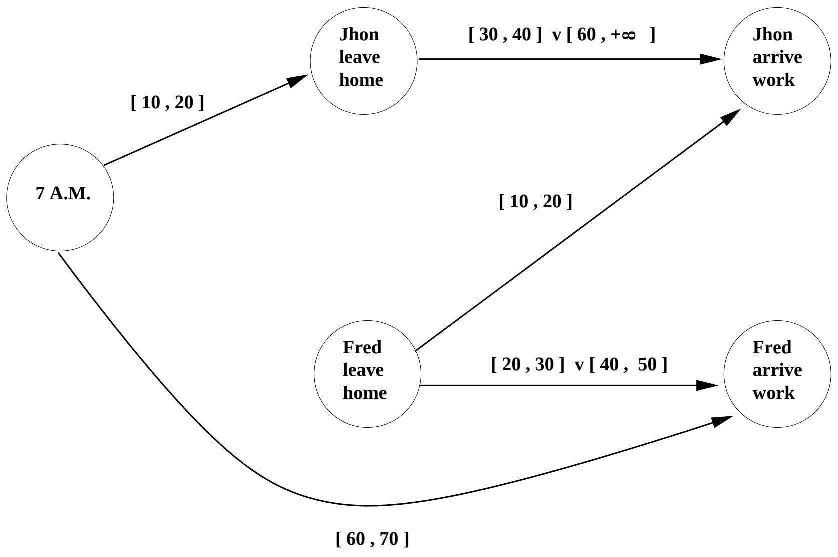


Figure 2.1: Constraint graph of the TCSP described in Example 2.1

Given a TCSP, a value v_i is a *feasible value* for variable X_i if there exists a solution in which $X_i = v_i$. The set of all feasible values for a variable is called its *minimal domain*. A minimal constraint T_{ij} between X_i and X_j is the set of feasible values for $X_j - X_i$. A TCSP is minimal if its domains and constraints are minimal. It is *decomposable* if every assignment of values to a set of its variables which does not violate the constraints among such variables can be extended to a solution.

Constraint propagation over TCSPs is defined using three binary operations on constraints: union, intersection and composition.

Definition 2.4 *Let* $T = \{I_1, \ldots, I_l\}$ *and* $S = \{J_1, \ldots, J_m\}$ *be two temporal constraints defined on the pair of variables* X_i *and* X_j*. Then:*

- *The Union of T and S, denoted* $T \cup S$*, is*

$$T \cup S = \{I_1, \ldots, I_l, J_1, \ldots, J_m\}.$$

- *The Intersection of T and S, denoted* $T \oplus S$*, is*

$$T \oplus S = \{K_1, \dots, K_n\}, \ \textit{where } K_k = I_i \cap J_j, \ \textit{for some } i, j.$$

- *The Composition of T and S, denoted by* $T \otimes S$ *is*

$$T \otimes S = \{K_1, \dots, K_n\}, K_k = [a + c, b + d], \exists I_i = [a, b], J_j = [c, d].$$

These three operations correspond to the usual operations of union, intersection and composition of constraints [Mon74]. In fact, the union allows only values which satisfy either one of the constraint, while the intersection allows only values which satisfy both constraints. Furthermore, the composition of two temporal constraints, say S and T, defined respectively on the pairs of variables (X_i, X_k) and (X_k, X_j), is a constraint defined on the pair (X_i, X_j) which allows only pairs of values, say (v_i, v_j), for which there exists a value v_k, such that (v_i, v_k) satisfies S and (v_k, v_j) satisfies T.

Figure 2.2 shows a graphical example of how the three operators work.

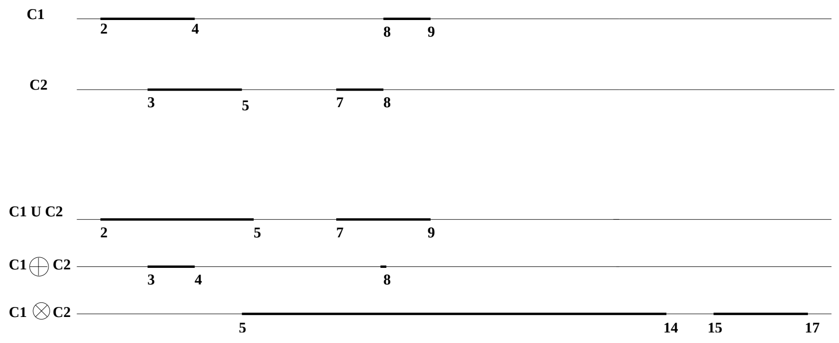


Figure 2.2: Temporal constraint operations: example of Union, Intersection and Composition of two temporal constraints.

Given a TCSP, the first interesting problem is to determine its consistency. If the TCSP is consistent, we may wish to find some solutions, or to answer queries concerning the set of all solutions. All these problems are NP-hard [DMP91].

Simple Temporal Problems A TCSP in which all constraints specify a single interval is called a *Simple Temporal Problem*. In such a problem, a constraint between X_i and X_j is represented in the *constraint graph* as an edge $X_i \longrightarrow X_j$ labeled by a single interval $[a_{ij}, b_{ij}]$ that represents the constraint $a_{ij} \leq X_j - X_i \leq b_{ij}$. An STP can also be associated with another directed edge-weighted graph $G_d = (V, E_d)$, called the *distance graph*, which has the same set of nodes as the constraint graph but twice the number of edges: for each binary constraint over variables X_i and X_j, the distance graph has an edge $i \longrightarrow j$ which is labeled by weight b_{ij}, representing the linear inequality $X_j - X_i \leq b_{ij}$, as well as an edge $X_j \longrightarrow X_i$ which is labeled by weight a_{ij}, representing the linear inequality $X_i - X_j \leq -a_{ij}$.

Example 2.2 Consider Example 2.1 and assume that John cannot take the bus to work and that Fred cannot take the carpool. This leaves only one interval per constraints and thus the example can be represented as a Simple Temporal Problem.

In Figure 2.3 we show both the constraint graph and the distance graph relative to Example 2.2.

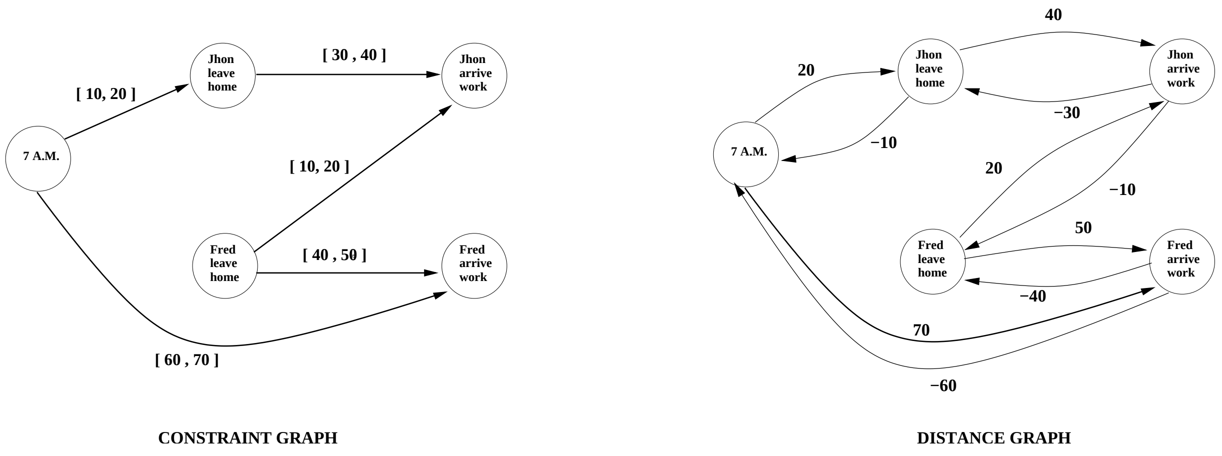


Figure 2.3: Constraint Graph and Distance Graph of Example 2.2.

Each path from X_i to X_j in the distance graph G_d, say through variables $X_{i_0} = X_i, X_{i_1}, X_{i_2}, \ldots, X_{i_k} = X_j$ induces the following *path constraint*: $X_j - X_i \leq \sum_{h=1}^{k} b_{i_{h-1} i_h}$.

The intersection of all induced path constraints yields the inequality $X_j - X_i \leq d_{ij}$, where d_{ij} is the length of the shortest path from X_i to X_j, if such a length is defined, i.e., if there are no negative cycles in the distance graph. An STP is consistent if and only if its distance graph has no negative cycles [Sho81, LS88]. This means that enforcing path consistency is sufficient for solving STPs [DMP91]. It follows that a given STP can be effectively specified by another complete directed graph, called a *d-graph*, where each edge $X_i \longrightarrow X_j$ is labeled by the shortest path length d_{ij} in the distance graph G_d.

In [DMP91] it is shown that any consistent STP is backtrack-free (that is, decomposable) relative to the constraints in its *d-graph*. Moreover, the set of temporal constraints of the form $[-d_{ji}, d_{ij}]$ is the *minimal STP* corresponding to the original STP and it is possible to find one of its solutions using a backtrack-free search that simply assigns to each variable any value that satisfies the minimal network constraints compatibly with previous assignments. Two specific solutions (usually called the *latest* and the *earliest* one) are given by $S_L = \{d_{01}, \ldots, d_{0n}\}$ and $S_E = \{d_{10}, \ldots, d_{n0}\}$, which assign to each variable respectively its latest and earliest possible time [DMP91].

The $d\text{-}graph$ (and thus the $minimal\ network$) of an STP can be found by applying Floyd-Warshall's All-Pairs-Shortest-Path algorithm to the distance graph with a complexity of $O(n^3)$ where n is the number of variables. Such an algorithm initializes a $n \times n$ matrix D to the values of the distance graph. That is, element d_{ij} is initialized to the weight of the constraint from variable X_i to variable X_j in the distance graph, i.e., $d_{ij} := b_{ij}$, while element d_{ji} is initialized to $-a_{ij}$. The diagonal elements of the matrix are instead initialized to 0. The main loop of the algorithm updates each element d_{ij} with $min(d_{ij}, d_{ik} + d_{kj})$ for every k until quiescence. In the end, either some diagonal element d_{ii} is negative, in which case it is possible to conclude that there is a negative cycle and thus the STP is not consistent, or the elements of the matrix contain the minimum distances. Since, given the d-graph, a solution can be found in linear time, the overall complexity of solving an STP is polynomial.

Pseudocode of All-Pairs-Shortest-Path

```
1. for $i := 1$ to $n$ do $d_{ii} \longleftarrow 0$;
2. for $i, j := 1$ to $n$ do $d_{ij} \longleftarrow a_{ij}$
3. for $k := 1$ to $n$ do
4.     for $i, j := 1$ to $n$ do
5.         $d_{ij} \leftarrow \min\{d_{ij}, d_{ik} + d_{kj}\}$;
```

Figure 2.4: Algorithm All-Pairs-Shortest-Path.

In Table 2.1 we show the minimal network corresponding to Example 2.2.

	0	**1**	**2**	**3**	**4**
0	[0]	[10,20]	[40,50]	[20,30]	[60,70]
1	[-20,-10]	[0]	[30,40]	[10,20]	[50,60]
2	[-50,-40]	[-40,-30]	[0]	[-20,-10]	[20,30]
3	[-30,-20]	[-20,-10]	[10,20]	[0]	[40,50]
4	[-70,-60]	[-60,-50]	[20,30]	[-50,40]	[0]

Table 2.1: Minimal network of Example 2.2.

Path-consistency in TCSPs An interesting property to consider in the context of TCSPs is path-consistency (see Definition 2.2). For example the Floyd-Warshall's algorithm used to solve STPs enforces this kind of local consistency. The definition of path-consistency in TCSPs uses the $\otimes$ and $\oplus$ operators as follows.

Definition 2.5 *A temporal constraint,* T_{ij}, *is* path consistent *iff*

$$T_{ij} \subseteq \oplus(T_{ik} \otimes T_{kj}) \forall k.$$

A TCSP is path consistent if all its constraints are path consistent.

In [DMP91] several algorithms are proposed for enforcing path consistency on a TCSP. In Figure 2.5 we show algorithm PC-2, the most efficient one.

Algorithm PC-2

```
1.  Input: TCSP P;
2.  Queue $Q \leftarrow \{(i,j,k) \mid (i < j)\text{and}(k \neq i,j)\}$;
3.  while $Q \neq \emptyset$ do
4.    select and delete a path $(i,j,k)$ from Q
5.    if $T_{ij} \neq T_{ik} \otimes T_{kj}$ then
6.      $T_{ij} \longleftarrow T_{ij} \oplus (T_{ik} \otimes T_{kj})$
7.      if $T_{ij} = \emptyset$ then exit(inconsistency)
8.      $Q \longleftarrow Q \cup \{(i,j,k) | 1 \leq k \leq n, i \neq k \neq j\}$
9.    end-if
10. end-while
11. Output: Path consistent network of P;
```

Figure 2.5: Pseudocode of algorithm PC-2.

In [DMP91] it is shown that PC-2 is a sound and complete algorithm for enforcing path consistency. Moreover its complexity is of $O(n^3R)$ composition steps and $O(n^3R^3)$ arithmetic operations, where R is the range of the TCSP expressed in the coarsest possible time units. It is important to notice that while for STPs enforcing path consistency is enough for solving them, in general it is not guaranteed to compute the minimal network of a TCSP. In fact, the $\mathsf{d\text{-}graph}$ of an STP is obtained directly from the bounds of the intervals on the constraints obtained applying PC-2. It should be noticed that, when applied to STPs in order to find the minimal network, PC-2 is more expensive than All-Pairs-Shortest-Path. The reason is that PC-2 can change the bounds of the interval of a constraint (Figure 2.5, line 6) up to R times in the worse case scenario in which each change decreases the interval of only one time unit and the constraint becomes empty in the end. However, as mentioned, in [DMP91] Floyd-Warshall's algorithm can be considered as a relaxation algorithm, one of the many based on the same original principle (path consistency) introduce by Montanari in [Mon74]. While Floyd Warshall can be applied only in the case of STPs, path consistency applies directly to the more general class of TCSPs.

2.2.2 Soft Constraint Satisfaction Problems (SCSPs)

In the literature there are many formalizations of the concept of *soft constraints* [RHZ76, SFV95]. Here we refer to the one described in [BMR97, BMR95], which however can be shown to generalize and express many of the others [BMR97, BFM$^+$96]. In a few words, a soft constraint is just a classical constraint where each instantiation of its variables has an associated value from a partially ordered set. Combining constraints will then have to take into account such additional values, and thus the formalism has also to provide suitable operations for combination ($\times$) and comparison (+) of tuples of values and constraints. This is why this formalization is based on the concept of semiring, which is just a set plus two operations.

Definition 2.6 (semirings and c-semirings) *A semiring is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ such that:*

- *A is a set and $\mathbf{0}, \mathbf{1} \in A$;*
- *$+$ is commutative, associative and $\mathbf{0}$ is its unit element;*
- *$\times$ is associative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element.*

A c-semiring is a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ *such that:*

- $+$ *is defined over possibly infinite sets of elements of* A *in the following way:*
  - $\forall a \in A$, $\sum(\{a\}) = a$;
  - $\sum(\emptyset) = \mathbf{0}$ *and* $\sum(A) = \mathbf{1}$;
  - $\sum(\bigcup A_i, i \in S) = \sum(\{\sum(A_i), i \in S\})$ *for all sets of indices* S *(flattening property);*
- $\times$ *is commutative.*

Let us consider the relation $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$. Then it is possible to prove that (see [BMR95]):

- $\leq_S$ is a partial order;
- $+$ and $\times$ are monotone on $\leq_S$;
- $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum;
- $\langle A, \leq_S \rangle$ is a complete lattice and, for all $a, b \in A$, $a + b = lub(a, b)$.

Moreover, if $\times$ is idempotent, then $\langle A, \leq_S \rangle$ is a complete distributive lattice and $\times$ is its glb. Informally, the relation $\leq_S$ gives us a way to compare (some of the) tuples of values and constraints. In fact, when we have $a \leq_S b$, we will say that *b is better than a*.

Definition 2.7 (constraints) *Given a c-semiring* $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$*, a finite set* D *(the domain of the variables), and an ordered set of variables* V*, a constraint is a pair* $\langle def, con \rangle$ *where* $con \subseteq V$ *and* $def : D^{|con|} \rightarrow A$.

Therefore, a constraint specifies a set of variables (the ones in con), and assigns to each tuple of values of D of these variables an element of the semiring set A. This element can then be interpreted in several ways: as a level of preference, or as a cost, or as a probability, etc. The correct way to interpret such elements depends on the choice of the semiring operations.

Given a constraint $c = \langle def, con \rangle$, we call $type(c)$ the set con of the variables of c. Given a constraint set C, $type(C) = \{type(c)/c \in C\}$ and $V(C) = \bigcup_{c \in C} type(c)$ is the set of all the variables appearing in the constraints of C.

Constraints can be compared by looking at the semiring values associated to the same tuples. In fact, consider two constraints $c_1 = \langle def_1, con \rangle$ and $c_2 = \langle def_2, con \rangle$, with $|con| = k$. Then $c_1 \sqsubseteq_S c_2$ if for all k-tuples t, $def_1(t) \leq_S def_2(t)$. The relation $\sqsubseteq_S$ is a partial order.

Definition 2.8 (Soft Constraint Satisfaction Problem (SCSP)) *A soft constraint satisfaction problem is a pair $\langle C, con \rangle$ where $con \subseteq V$ and C is a set of constraints.*

We also assume that there are not two constraints with the same type.

It is possible to extend relation $\sqsubseteq_S$, defined above, to sets of constraints, and also to problems (seen as sets of constraints). Therefore, given two SCSPs P_1 and P_2 with the same graph topology, we will write $P_1 \sqsubseteq_S P_2$ if, for each constraint c_1 in P and the corresponding constraint c_2 in P_2, we have that $c_1 \sqsubseteq_S c_2$.

Note that a classical CSP is a SCSP where the chosen c-semiring is:

$$S_{CSP} = \langle \{false, true\}, \vee, \wedge, false, true \rangle.$$

In fact, since constraints in CSPs are crisp, that is, they either allow a tuple or not, it is possible to model them via semiring domain with only two values, say $false$ and $true$: allowed tuples will have associated value $true$ and not allowed ones the values $false$. Moreover, in CSPs, constraint combination is achieved via a join operation among allowed tuple sets. This can be modeled by choosing logical and ($\wedge$) as the multiplicative operator. Finally to model the projection over some of the variables, as the k-tuples for which there exists a consistent extension to an n-tuple (where n is the total number of variables), it is enough to assume the additive operation to be logical or ($\vee$).

Fuzzy CSPs [RHZ76, Sch92, Rut94] extend the notion of classical CSPs by allowing non crisp constraints, that is, constraints which associate a preference level with each tuple of values. Such level is always between 0 and 1, where 1 represents the best value and 0 the worst one. The solution of a fuzzy CSP is then defined as the set of tuples of values (for all the variables) which have the maximal value. The way they associate a value with an n-tuple is by minimizing the values of all its subtuples. The motivation for such a max-min framework relies on the attempt to maximize the value of the least preferred tuple. It is easy to see that Fuzzy CSPs can be modeled in the SCSP framework by choosing the c-semiring:

$$S_{FCSP} = \langle [0,1], max, min, 0, 1 \rangle.$$

Example 2.3 Figure 2.6 shows a fuzzy CSP. Variables are inside circles, constraints are represented by undirected arcs, and semiring values are written to the right of the corresponding tuples. Here we assume that the domain D of the variables contains only elements a and b.

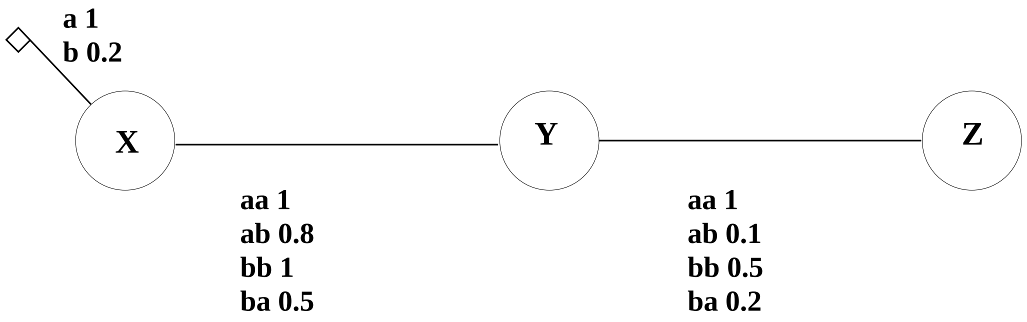


Figure 2.6: A fuzzy CSP

Definition 2.9 (combination) *Given two constraints* $c_1 = \langle def_1, con_1 \rangle$ *and* $c_2 = \langle def_2, con_2 \rangle$*, their* combination $c_1 \otimes c_2$ *is the constraint* $\langle def, con \rangle$ *defined by* $con = con_1 \cup con_2$ *and* $def(t) = def_1(t \downarrow^{con}_{con_1}) \times def_2(t \downarrow^{con}_{con_2})$[2]. *The combination operator*

[2]By $t \downarrow^X_Y$ we mean the projection of tuple t, which is defined over the set of variables X, over the set of variables $Y \subseteq X$.

$\otimes$ can be straightforward extended also to sets of constraints: when applied to a set of constraints C, we will write $\bigotimes C$.

In words, combining two constraints means building a new constraint involving all the variables of the original ones, and which associates to each tuple of domain values for such variables a semiring element which is obtained by multiplying the elements associated by the original constraints to the appropriate subtuples.

Using the properties of $\times$ and $+$, it is easy to prove that:

- $\otimes$ is associative, commutative, and monotone over $\sqsubseteq_S$;
- if $\times$ is idempotent, $\otimes$ is idempotent as well.

Definition 2.10 (projection) *Given a constraint $c = \langle def, con \rangle$ and a subset I of V, the* projection *of c over I, written $c \Downarrow_I$, is the constraint $\langle def', con' \rangle$ where $con' = con \cap I$ and $def'(t') = \sum_{t/t\downarrow^{con}_{I\cap con}=t'} def(t)$.*

Informally, projecting means eliminating some variables. This is done by associating to each tuple over the remaining variables a semiring element which is the sum of the elements associated by the original constraint to all the extensions of this tuple over the eliminated variables. Given two sets S_1 and S_2, we have $(c \Downarrow_{S_1}) \Downarrow_{S_2} = c \Downarrow_{S_1 \cap S_2}$.

Definition 2.11 (solution) *The* solution *of a SCSP problem $P = \langle C, con \rangle$ is the constraint $Sol(P) = (\bigotimes C) \Downarrow_{con}$.*

That is, to obtain the solution of an SCSP, we combine all constraints, and then project over the variables in con. In this way we get the constraint over con which is "induced" by the entire SCSP.

Example 2.4 For example, each solution of the fuzzy CSP of Figure 2.6 consists of a triple of domain values (that is, a domain value for each of the three variables) and an associated semiring element (here we assume that $\mathtt{con}$ contains all variables). Such an element is obtained by looking at the smallest value for all the subtuples (as many as the constraints) forming the pair. For example, for tuple $\langle a, b, a\rangle$ (that is, $X = a, Y = b, Z = a$), we have to compute the minimum between 1 (which is the value for $X = a$), 0.8 (which is the value for $\langle X = a, Y = b\rangle$) and 0.2 (which is the value for $\langle X = a, Y = b\rangle$). Hence, the resulting value for t his tuple is 0.2.

The best level of consistency of a SCSP problem P is defined by $\mathtt{blevel}(P) = \mathtt{Sol}(P) \Downarrow_{\emptyset}$. We say that: P is α-consistent if $\mathtt{blevel}(P) = \alpha$; P is consistent iff there exists $\alpha >_S \mathbf{0}$ such that P is α-consistent; P is inconsistent if it is not consistent.

Definition 2.12 (optimal solution) *Given an SCSP problem P, consider $\mathtt{Sol}(P) = \langle \mathtt{def}, \mathtt{con}\rangle$. An optimal solution of P is a pair $\langle t, v\rangle$ such that $\mathtt{def}(t) = v$, and there is no t' such that $v <_S \mathtt{def}(t')$.*

Therefore optimal solutions are solutions which have the best semiring element among those associated to solutions. The set of optimal solutions of an SCSP P will be written as $\mathtt{Opt}(P)$.

Example 2.5 Consider again the fuzzy SCSP described in Example 2.4 and depicted in Figure 2.6. A solution is $\langle X = b, Y = b, Z = b\rangle$ with an associated preference of 0.2. The best solution, instead, is $\langle X = a, Y = a, Z = a\rangle$ which is the only complete assignment with preference 1.

Note that in the case of partially ordered preferences the best level of consistency of a problem is an upper bound of the semiring elements associated with its optimal solutions, while the case of a total order it *is* the value associated to the optimal solutions.

Definition 2.13 (problem ordering and equivalence) *Consider two problems P_1 and P_2. Then $P_1 \sqsubseteq_P P_2$ if $\mathtt{Sol}(P_1) \sqsubseteq_S \mathtt{Sol}(P_2)$. If $P_1 \sqsubseteq_P P_2$ and $P_2 \sqsubseteq_P P_1$, then they have the same solution, thus we say that they are equivalent and we write $P_1 \equiv P_2$.*

The relation $\sqsubseteq_P$ is a preorder. Moreover, $P_1 \sqsubseteq_S P_2$ implies $P_1 \sqsubseteq_P P_2$. Also, $\equiv$ is an equivalence relation. Consider two problems $P_1 = \langle C_1, con \rangle$ and $P_2 = \langle C_1 \cup C_2, con \rangle$. Then $P_2 \sqsubseteq_P P_1$ and $blevel(P_2) \leq_S blevel(P_1)$.

Local Consistency We remind here the basic notions and properties of local consistency over SCSPs. This is of particular interest in this thesis, since, as we have mentioned before we will extend and use path consistency in order to deal with preferences (Chapters 2 and 3).

Definition 2.14 (local inconsistency) *We say that the SCSP problem* $P = \langle C, con \rangle$ *is* locally inconsistent *if there exist* $C' \subseteq C$ *such that* $blevel(C') = \mathbf{0}$.

Consider a set of constraints C and $C' \subseteq C$. If C is α-consistent then C' is β-consistent with $\alpha \leq_S \beta$. As a corollary: if a problem is locally inconsistent, then it is inconsistent. Of course also the converse holds, since if a problem $P = \langle C, con \rangle$ is inconsistent then there exists a subset of constraints $C' \subseteq C$ such that $blevel(C') = \mathbf{0}$.

SCSP problems can be solved by extending and adapting the technique usually used for classical CSPs. For example, to find the best solution we could employ a branch-and-bound search algorithm (instead of the classical backtracking), and also the successfully used propagation techniques, like arc-consistency [Mac77], can be generalized to be used for SCSPs.

The detailed formal definition of *propagation* algorithms (sometimes called also *local consistency* algorithms) for SCSPs can be found in [BMR95]. For the purpose of this thesis, what is important to say is that a *propagation rule* is a function which, taken an SCSP, solves a subproblem of it. It is possible to show that propagation rules are idempotent, monotone, and intensive functions (over the partial order of problems) which do not change the solution set. Given a set of propagation rules, a local consistency algorithm consists of applying them in any order until stability. It is possible to prove that local consistency algorithms defined in this way have the following properties if the multiplicative operation of the semiring is idempotent: equivalence, termination, and uniqueness of the result.

Thus we can notice that the generalization of local consistency from classical CSPs to SCSPs concerns the fact that, instead of deleting values or tuples, obtaining local consistency in SCSPs means changing the semiring values associated to some tuples or domain elements. The change always brings these values towards the worst value of the semiring, that is, the **0**. Thus, it is obvious that, given an SCSP problem P and the problem P' obtained by applying some local consistency algorithm to P, we must have $P' \sqsubseteq_S P$.

In what follows we will formalize the local consistency process. In [BMR95, BMR97] the concept of local consistency rule is introduced. The application of such a rule consists in solving one of the the subproblems. An additional notion, used in the definition of local consistency rule is the following.

Definition 2.15 (location) *A* typed location l *is a set of variables. Given a problem* $P = \langle C, con\rangle$*, the value* $[l]_P$ *of the location* l *in* P *is* $\langle def, l\rangle$ *if it belongs to* C*,* $\langle \mathbf{1}, l\rangle$ *otherwise. The value* $[\{l_1, \cdots, l_n\}]_P$ *of the set of locations* $\{l_1, \cdots, l_n\}$ *is the set* $\{[l_1]_P, \cdots, [l_n]_P\}$*.*

Informally a typed location is a location that has, as type, variables con and can thus be assigned only a constraint $c = \langle f, con\rangle$ defined on con.

Definition 2.16 (assignment) *An* assignment *is a pair* $l := c$ *where* $c = \langle def, l\rangle$*. Given a problem* $P = \langle C, con\rangle$*, the result of the assignment* $l := c$ *is defined as* $[l := c](P) = \langle\{\langle def', con'\rangle \in C / con' \neq l\} \cup c, con\rangle$*.*

The assignment $l := c$ can be seen as a function mapping constraint problems into constraint problems, which modifies a given problem by changing just one constraint, the one with type l, replacing it with c.

Definition 2.17 (local consistency rule) *A* local consistency rule *is* $l \leftarrow L$ *where* l *is a location,* L *a set of locations and* $l \notin L$*.*

Definition 2.18 (rule application) *The result of applying the rule* $l \leftarrow L$ *to the problem* P *is:*

$$[l \leftarrow L](P) = [l := \mathtt{Sol}(\langle [L \cup \{l\}]_P, l \rangle)](P)$$

The application of a sequence of rules $r; R$ *is defined by* $[r; R](P) = [R]([r](P))$.

Thus, applying r to P means assigning to location l the constraint obtained by solving the subproblem of P containing the constraints specified by the locations in $L \cup l$.

An important Theorem that we will use, adapting it to temporal preferences is the following:

Theorem 2.1 *Given a SCSP* P *and a rule r for* P *we have that* $P \equiv [r](P)$ *if* $\times$ *is idempotent.*

Definition 2.19 (stable problem) *Given a problem* P *and a set* R *of rules for* P, P *is said to be* stable *w.r.t* R *if, for each* $r \in R$, $[r](P) = P$.

Definition 2.20 (strategy) *Given a set* R *of rules, a* strategy *for* R *is an infinite sequence of rules, that is, a value from* R^∞. *A strategy* T *is fair if each rule of* R *occurs in* T *infinitely often.*

Definition 2.21 (local consistency algorithm) *Given a problem* P, *a set of rules* R *and a fair strategy* T *for* R, *a* local consistency algorithm *applies to* P *the rules in* R *in the order given by* T. *The algorithm stops when the current problem is stable w.r.t* R. *In that case, we note* $lc(P, R, T)$ *the resulting problem. We denote* $ac(P) = lc(P, AC, T)$ *where AC is the set of AC rules.*

We conclude observing that any local consistency algorithm satisfies the following properties:

- it terminates if the set of preferences assigned to at least a tuple in a constraint of the SCSP is contained in a set I which is finite and + and $\times$ are I-closed;

- given a problem P and a value v assigned to a tuple in a constraint of P, consider $P' = lc(P, R)$ and the value v' assigned to the same tuple of the same constraint in P', then $v' \leq_S v$.
- If the semiring operator $\times$ is idempotent and $P' = lc(P, R, T)$, then $P \equiv P'$ and $lc(P, R, T)$ does not depend on strategy T, (so we will write $lc(P, R)$ instead of $lc(P, R, T)$);

Examples of semiring based SCSPs We will now give an overview of the most common examples of semirings used as underlying structure of the SCSPs.

- **Classical SCSPs**. The semi-ring is

  $$S_{CSP} = \langle \{false, true\}, \vee, \wedge, false, true \rangle.$$

  The only two preferences that can be given are $true$, indicating that the tuple is allowed and $false$, indicating that the tuple is forbidden. Preferences are combined using logical and and compared using logical or. Optimization criterion: any assignment that has preference $true$ on all constraints is optimal.

- **Fuzzy SCSPs**. The semiring is

  $$S_{FCSP} = \langle [0, 1], max, min, 0, 1 \rangle.$$

  Preferences are values between 0 and 1. They are combined using min and compared using max. Optimization criterion: maximize minimal preference.

- **Weighted SCSPs**. The semiring is

  $$S_{WCSP} = \langle \Re^{+}, min, +, +\infty, 0 \rangle.$$

  Preferences are interpreted as costs from 0 to $+\infty$. Costs are combined with $+$ and compared with min. Optimization criterion: minimize sum of costs.

- **Probabilistic SCSPs**. The semiring is

$$S_{PCSP} = \langle [0,1], max, \times, 0, 1 \rangle.$$

Preferences are interpreted as probabilities ranging from 0 to 1. As expected, they are combined using $\times$ and compared using max. Optimization criterion: maximize joint probability.

- **Set-based CSPs** The semiring is

$$S_{set} = \langle P(A), \cup, \cap, \emptyset, A \rangle.$$

where A is any set, and P(A) is the powerset of A. Preferences are sets and are combined using $intersection$ and compared using $union$. Optimization criterion: maximize intersection with respect to inclusion.

We conclude this part of background on Soft Constraints mentioning a very useful result given in [BMR97] on N-$dimensional$ SCSPs. Consider, for example, two semirings $S_1 = \langle A_1, +_1, \times_1, 0_1, 1_1 \rangle$ and $S_2 = \langle A_2, +_2, \times_2, 0_2, 1_2 \rangle$. In [BMR97] it is shown that the structure that has:

- as preferences the set of pairs $A_1 \times A_2$
- as additive operator $+$, that works as $+_1$ on the first element of the pairs and as $+_2$ on the second element of the pairs, and
- as multiplicative operator $\times$, defined in terms of $\times_1$ and $\times_2$ in a similar way as +,

is a semiring. This structure can be used to model multiple-criteria optimization, in which each tuple is associated with more than one value, each representing the preference of the tuple with respect to a different aspect.

In Figure 2.7 we show an example, where $S_1 = S_{FCSP}$ and $S_1 = S_{WCSP}$. Preferences are pairs where the first element is a "fuzzy" preference in [0,1] and the second element is a cost. Consider solution $\langle X = a, Y = a, Z = a \rangle$. Its projections on the three constraints are associated with the following preferences $(1, 20)$ on the unary constraint on X, $(1, 20)$ on the binary constraint on X and Y, and $(1, 10)$ on the constraint on Y and Z. The global preference is the pair obtained taking the minimum of the first components and the sum of the second components, i.e. $(1, 50)$. Notice that for example solution $\langle X = a, Y = a, Z = a \rangle$ and $\langle X = b, Y = b, Z = b \rangle$ are incomparable since their preferences are $(1, 50)$ and $(0.2, 30)$, which means that $\langle X = a, Y = a, Z = a \rangle$ wins on the fuzzy preferences but at greater cost.

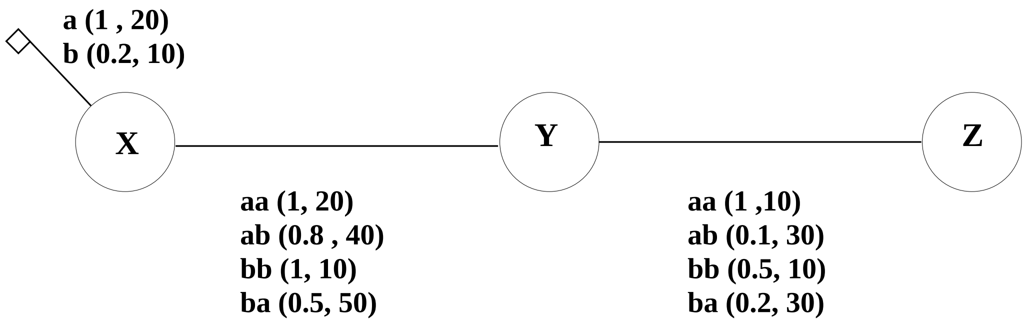


Figure 2.7: A 2-dimensional SCSP.

2.2.3 Inductive learning

The problem of learning preferences in STPPs, from examples of solutions ratings, can be formally described as an inductive learning problem [RN95, Mit97]. Inductive learning can be defined as the ability of a system to induce the correct structure of a map $t(\cdot)$ which is known only for particular inputs. More formally, defining an example as a pair $(x, t(x))$, the computational task is as follows: given a collection of examples of $t(\cdot)$, i.e., the *training set*, return a function $h(\cdot)$ that approximates $t(\cdot)$. Function $h(\cdot)$ is called a hypothesis.

A common approach to inductive learning, especially in the context of neural networks, is to evaluate the quality of a hypothesis h (on the training set) through an *error function* [Hay94]. An example of popular error function, that can be used over the reals, is the sum of squares error [Hay94]:

$$SSE = \frac{1}{2}\sum_{i=1}^{n}(t(x_i) - h(x_i))^2$$

where $(x_i, t(x_i))$ is the i-th example of the training set. Other error functions that can be used to evaluate the quality of an hypothesis are the absolute maximum error and absolute mean error, respectively defined as:

$$E_{max} = max_{1,\dots,n}|t(x_i) - h(x_i)|$$

and

$$E_{med}\frac{\sum_{i=1\dots n}|t(x_i) - h(x_1)|}{n}.$$

Given a starting hypothesis h_0, the goal of learning is to minimize the chosen error function by modifying h_0. This can be done by using a definition of h which depends on a set of internal parameters W, i.e., $h \equiv h_W$, and then adjusting these parameters. This adjustment can be formulated in different ways, depending on whether the domain is isomorphic to the reals or not. The usual way to be used over the reals, and if h_W is continuous and derivable, is to follow the negative of the *gradient* of the error function with respect to W. This technique is called *gradient descent* [Hay94]. Specifically, the set of parameters W is initialized to small random values at time $\tau = 0$ and updated at time $\tau + 1$ according to the following equation, known as $\Delta - rule$: $W(\tau + 1) = W(\tau) + \Delta W(\tau)$, where $\Delta W(\tau) = -\eta\frac{\partial E}{\partial W(\tau)}$, and η is the step size used for the gradient descent called the $learning\ rate$. Learning is usually stopped when a minimum of the error function is reached. Note that,in general, there is no guarantee that the found minimum is global.

As far as how the examples are used, learning techniques can be divided in two categories: stochastic (also called online) and batch (also called offline) learning. Batch supervised learning is the classical approach in machine learning: a set of examples is obtained and used in order to learn a good approximating function (i.e. train the network), before the network is used in the application. On the other hand, in online learning, data gathered during the normal operation of the system are used to continuously adapt the learned function. Batch training is consistent with the theory of unconstrained optimization, since the information from all the training set is used. For example in batch learning, when minimizing sum of squares error the sum would be computed as in $SSE = \frac{1}{2}\sum_{i=1}^{n}(t(x_i) - h(x_i))^2$ where $x_1, \dots, x_n$ are all the examples of the training set.

In stochastic training, the network weights are updated after the presentation of each training example, which may be sampled with or without repetition. This corresponds to the minimization of the instantaneous error which, in the case of sum of squares error, would be $SSE = \frac{1}{2}(t(x_i) - h(x_i))^2$ when computed on the i-th example. It can be shown that, for sufficiently small values of the learning rate η, the stochastic gradient descent converges to the minimum of a batch error function [Hay94].

The learning module presented in Section 2.6 performs a stochastic gradient descent on SSE. In fact, stochastic training may be chosen for a learning task either because of the very large (or even redundant) training set or for modeling a slowly timevarying system. Although batch training seems faster for small training sets and networks, stochastic training is probably faster for large training sets, it helps escaping local minima and provides a more natural approach for learning nonstationary tasks. Given the inherent efficiency of stochastic gradient descent, various schemes have been proposed recently [ALAP98, Saa98, Sut92]. Moreover, stochastic learning has several advantages over batch learning. Stochastic methods seem more robust as errors, omissions or redundant data in the training set can be corrected or ejected during the training phase. Additionally, training data can often be generated easily and in great quantities when the system is in operation, whereas it is usually scarce and precious before. Finally, stochastic training is necessary in order to learn and track time varying functions and to continuously adapt in a changing environment. In a broad sense, stochastic learning is essential if the goal is to obtain a learning system as opposed to a merely learned one, as pointed out in [SW93].

2.3 Temporal CSPs and STPs with Preferences (TCSPPs and STPPs)

Although very expressive, TCSPs are able to model just *hard* temporal constraints. This means that all constraints have to be satisfied, and that the solutions of a constraint are all equally satisfying. However, in many real-life scenarios these two assumptions do not hold. In particular, sometimes some solutions are preferred with respect to others. Therefore the global problem is not to find a way to satisfy all constraints, but to find a way to satisfy them optimally, according to the preferences specified.

To address such problems we propose a framework, where each temporal constraint is associated with a preference function, which specifies the preference for each distance or duration. This framework is based on both TCSPs and semiring-based soft constraints. The result is a class of problems which we will call Temporal Constraint Satisfaction Problems with Preferences (TCSPPs).

Definition 2.22 (soft temporal constraint) *A soft temporal constraint is a 4-tuple* $\langle (X, Y), I, A, f \rangle$ *consisting of*

- *an ordered pair of variables* (X, Y) *over the integers, called the scope of the constraint;*
- *a set of disjoint intervals* $I = \{[a_1, b_1], \ldots, [a_n, b_n]\}$*, where all* a_i*'s and* b_i*'s are integers, and* $a_i \leq b_i$ *for all* $i = 1, \ldots, n$*;*
- *a set of preferences* A*;*
- *a preference function* f*, where* $f : \bigcup_{i=1}^{n} [a_i, b_i] \rightarrow A$*, which is a mapping of the elements belonging to an interval of* I *into preference values, taken from set* A*.*

When the variables and the preference set will be obvious, we will omit them and we will define a soft temporal constraint just as a pair $\langle I, f \rangle$*.*

Given an assignment of the variables X *and* Y*, say* ν_x *and* ν_y*, we say that this assignment satisfies the constraint* $\langle (X, Y), I, A, f \rangle$ *iff there is* $[a_i, b_i] \in I$ *such that* $a_i \leq \nu_y - \nu_x \leq b_i$*. In such a case, the preference associated to the assignment by the constraint is* $f(\nu_y - \nu_x)$*.*

Definition 2.23 (TCSPP) *Given a semiring* $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$*, a Temporal Constraint Satisfaction Problems with Preferences (TCSPPs) over* S *is a pair* $\langle V, C \rangle$*, where* V *is a set of variables and* C *is a set of soft temporal constraint over pairs of variables in* V *and with preferences in* A*.*

Definition 2.24 (solution) *Given a TCSPP* $\langle V, C \rangle$ *over a semiring* S*, a solution is an assignment to all thee variables in* V*, say* t*, that satisfies all the constraints in* C*. An assignment* t *is said to satisfy a constraint* c *in* C *with preference* p *if the projection of* t *over the pair of variables of* c *satisfies* c *with an associated preference equal to* p*. We will write pref(t,c) = p.*

Each solution has a *global preference value*, obtained by combining, via the $\times$ operator of the semiring, the preference levels at which the solution satisfies the constraints in C.

Definition 2.25 (preference of a solution) *Given a TCSPP* $\langle V, C \rangle$ *over a semiring* S *and one of its solutions* $t = \langle \nu_1, \ldots, \nu_n \rangle$*, its preference, denoted by val(t), is computed by* $\Pi_{c \in C}\, \mathtt{pref}(s, c)$*, where the product here is perfomed by using the multiplicative operation of semiring* S*.*

The optimal solutions of a TCSPP are those solutions which are not dominated by any other solution in preference terms.

Definition 2.26 (optimal solutions) *Given a TCSPP* $P = \langle V, C \rangle$ *over the semiring* S*, a solution* t *of* P *is optimal if for every other solution* t' *of* P*,* $t' \not\geq_S t$*.*

For example, consider the semiring $S_{\mathrm{fuzzy}} = \langle [0,1], max, min, 0, 1 \rangle$, used for fuzzy constraint solving [Sch92]. The global preference value of a solution will be the minimum of all the preference values associated with the distances selected by this solution in all constraints, and the best solutions will be those with the maximal value.

Another example is the semiring $S_{\mathrm{csp}} = \langle \{false, true\}, \vee, \wedge, false, true \rangle$, which allows to describe hard constraint problems [Mac92]. Here there are only two preference values: *true* and *false*, the preference value of a complete solution will be determined by the logical *and* of all the local preferences, and the best solutions will be those with preference value *true* (since *true* is better than *false* in the order induced by using the logical or as the $+$ operator of the semiring). This semiring thus recasts the classical TCSP framework [DMP91] into a TCSPP.

A special case occurs when each constraint of a TCSPP contains a single interval. We call such problems *Simple Temporal Problems with Preferences* (STPPs), due to the fact that they generalize Simple Temporal Problems (STPs) [DMP91]. This case is interesting because, as noted above, STPs are polynomially solvable, while general TCSPs are NP-hard, and the computational effect of adding preferences to STPs is not immediately obvious.

STPPs are also expressive enough to represent many real life scenarios. Consider the Landsat 7 example given in the Introduction. In Figure 2.8 we show an STPP that models the scenario in which there are 3 events to be scheduled on Landsat 7: the start time (**Ss**) and ending time (**Se**) of a slewing procedure and the (**Is**) starting time of an image retrieval. The slewing activity in this example can take from 3 to 10 units of time, but it is preferred that it takes the shortest time possible. The image taking can start any time between 3 and 20 units of time after the slewing has been initiated. The third constraint, on variables **Is** and **Se**, models the fact that it is better for the image taking to start as soon as the slewing has stopped.

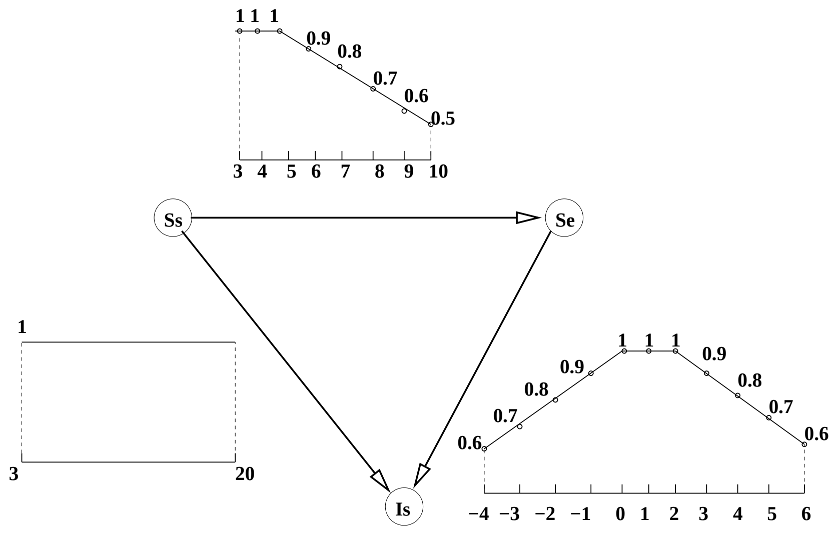


Figure 2.8: An STPP representing a Landsat 7 scenario in which there are 3 activities to be scheduled: the start time (**Ss**) and ending time (**Se**) of a slewing procedure and the (**Is**) starting time of an image retrieval.

2.3.1 Complexity of solving TCSPPs and STPPs

As noted in Section 2.2.1, TCSPs are NP-hard problems. Since the addition of preference functions can only make the problem of finding the optimal solutions more complex, it is obvious that TCSPPs are NP-hard problems as well. In fact, a TCSP is a special instance of a TCSPP over the S_{CSP} semiring.

We now turn our attention to the complexity of general STPPs. We recall that STPs are polynomially solvable, thus one might speculate that the same is true for STPPs. However, it is possible to show that, in general, STPPs fall into the class of NP-hard problems.

Theorem 2.2 (Complexity of STPPs) *The class of STPPs is NP-hard.*

Proof: We prove this result by reducing an arbitrary TCSP to an STPP. Consider a TCSP, and take any of its constraints, say $I = \{[a_1, b_1], \ldots [a_n, b_n]\}$. We will now obtain a corresponding soft temporal constraint containing just one interval (thus belonging to an STPP). The semiring that we will use for the resulting STPP is the classical one: $S_{csp} = \langle\{false, true\}, \vee, \wedge, false, true\rangle$. Thus the only allowed preference values are false and true. Assuming that the intervals in I are ordered such that $a_i \leq a_{i+1}$ for $i \in \{1, \ldots, n-1\}$, the interval of the soft constraint is just $[a_1, b_n]$. The preference function will give value 1 to all elements belonging to an interval in I and 0 to the others. Thus we have obtained an STPP whose set of solutions with value 1 (which are the optimal solutions, since $0 \leq_S 1$ in the chosen semiring) coincides with the set of solutions of the given TCSP. Since finding the set of solutions of a TCSP is NP-hard, it follows that the problem of finding the set of optimal solutions to an STPP is NP-hard.

In the following section we will show that a class of STPPs is polynomially solvable: a sufficient condition is having semi-convex preference function and a semiring with a total order of preference values and an idempotent multiplicative operation [KMMR01]. In [DP85] it has been shown that the only aggregation operator on a totally ordered set that is idempotent is min, i.e. the multiplicative operator of the S_{FCSP} semiring.

2.3.2 Path consistency for TCSPPs

Given a constraint network, it is often useful to find the corresponding minimal network in which the constraints are as explicit as possible. This task is normally performed by enforcing various levels of local consistency. For TCSPPs, in particular, we can define a notion of $path\ consistency$ by just extending the notion of path consistency for TCSPs [DMP91]. Given two soft constraints, $T_1 = \langle I_1, f_1\rangle$ and $T_2 = \langle I_2, f_2\rangle$ and a semiring S, we define:

- the *intersection* of two soft constraints T_1 and T_2, written $T_1 \oplus_S T_2$, as the soft temporal constraint $\langle I_1 \oplus I_2, f\rangle$, where:
  - $I_1 \oplus I_2$ is the pairwise intersection of intervals in I_1 and I_2, and
  - $f(a) = f_1(a) \times_S f_2(a)$ for all $a \in I_1 \oplus I_2$.

- the *composition* of two soft constraints T_1 and T_2, written $T_1 \otimes_S T_2$, as the soft constraint $T = \langle I_1 \otimes I_2, f\rangle$ where:
  - $r \in I_1 \otimes I_2$ iff there exists a value $t_1 \in I_1$ and $t_2 \in I_2$ such that $r = t_1 + t_2$, and
  - $f(a) = \sum\{f_1(a_1) \times_S f_2(a_2) | a = a_1 + a_2, a_1 \in I_1, a_2 \in I_2\}$.

The *path-induced* constraint on variables X_i and X_j is $R_{ij}^{path} = \oplus_S \forall k(T_{ik} \otimes_S T_{kj})$, i.e., the result of performing $\oplus_S$ on each way of generating paths of length two from X_i to X_j. A constraint T_{ij} is *path-consistent* iff $T_{ij} \subseteq R_{ij}^{path}$, i.e., T_{ij} is at least as strict as R_{ij}^{path}. A TCSPP is path-consistent iff all its constraints are path-consistent.

It is interesting to study under which assumptions, by applying the path consistency operation $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ to any constraint of a given TCSPP, the resulting TCSPP is equivalent to the given one, that is, it has the same set of solutions with the same preferences. The assumptions can be derived directly from those which are sufficient in generic SCSPs, as stated by the following theorem.

Theorem 2.3 *Consider an TCSPP P defined on a semiring which has an idempotent multiplicative operator. Then, applying operation* $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ *for any* k *to any constraint* T_{ij} *of P returns an equivalent TCSPP.*

Proof: Consider TCSPPs P_1 and P_2 on the same set of variables $\{x_1, \ldots, x_n\}$ and defined over a semiring with an idempotent multiplicative operator. Assume also that the set of constraints of P_2 consists of the constraints of P_1 minus $\{T_{ij}\}$ plus $\{T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})\}$.

To show that P_1 is equivalent to P_2 we must show that every solution t of P_1 with preference $val(t)$ is a solution of P_2 with the same preference. Notice that P_1 and P_2 differ only for the constraint defined on variables X_i and X_j, which is $T_{ij} = \langle I, f\rangle$ in P_1 and $T'_{ij} = T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj}) = \langle I', f'\rangle$ in P_2, where $T_{ik} = \langle I_{ik}, f_{ik}\rangle$ and $T_{kj} = \langle I_{kj}, f_{kj}\rangle$ are the same in P_1 and P_2.

Now, since $T'_{ij} = T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj}) = \langle I', f' \rangle$, then $I' = I \oplus (I_1 \otimes I_2)$. This means that $I' \subseteq I$. Assuming $I' \subset I$, we will now show that no element $a \in$ I-I$'$ can be in a solution s of P_1. Assume to the contrary that s is a solution of P_1 such that $s \downarrow_{x_i,x_j} = (s_i, s_j)$ and $s_j - s_i = a \in$ I-I$'$. Then, since $a \notin I'$ means that there is no $a_1 \in I_{ik}$ nor $a_2 \in I_{kj}$ such that $a = a_1 + a_2$, then either $s_k - s_i = a_1 \notin I_{ik}$ or $s_j - s_k = a_2 \notin I_{kj}$. But this cannot be the case, since s is assumed to be a solution of P_1.

From the above argument we can conclude that, for any solution t of P_1, we have $t \downarrow_{x_i,x_j} \in I'$. Thus P_1 and P_2 have the same set of solutions.

Consider solution t in P_1. Then, as stated in the previous section, the global preference associated to t in P_1 is $val(t) = \times\{f_{pq}(v_q - v_p) | (v_p, v_q) = t \downarrow_{x_p,x_q}\}$, which can be rewritten, highlighting the preferences obtained on constraints T_{ij}, T_{ik} and T_{kj}, as:

$$val(t) = f(v_j - v_i) \times f(v_k - v_i) \times f(v_j - v_k) \times B$$

where $B = \times\{f_{pq}(v_q - v_p) | (v_p, v_q) = t \downarrow_{x_p,x_q}, (p, q) \notin \{(i, j), (k, i), (j, k)\}\}$.

Similarly, in P_2 the global preference of t is

$$val'(t) = f'(v_j - v_i) \times f(v_k - v_i) \times f(v_j - v_k) \times B.$$

We want to prove that $val(t) = val'(t)$. Notice that B appears in $val(t)$ and in $val'(t)$, hence we can ignore it.

By definition:

$$f'(v_j - v_i) = f(v_j - v_i) \times \sum_{v'_k | (v'_k - v_i) \in I_{ik}, (v_j - v'_k) \in I_{kj}} [f(v'_k - v_i) \times f(v_j - v'_k)].$$

Now, among the possible assignments to variable x_k, say v'_k, such that $(v'_k - v_i) \in I_{ik}$ and $(v_j - v'_k) \in I_{kj}$, there is the assignment given to x_k in solution t, say v_k. Thus we rewrite $f'(v_j - v_i)$ in the following way:

$$f'(v_j - v_i) = f(v_j - v_i) \times \{[f(v_k - v_i) \times f(v_j - v_k)] +$$

$$\sum_{v'_k | (v'_k - v_i) \in I_{ik}, (v_j - v'_k) \in I_{kj}, v'_k \neq v_k} [f(v'_k - v_i) \times f(v_j - v'_k)]\}.$$

At this point, the preference of solution t in P_2 is

$$val'(t) = f(v_j - v_i) \times \{[f(v_k - v_i) \times f(v_j - v_k)] +$$

$$\sum_{v'_k | (v'_k - v_i) \in I_{ik}, (v_j - v'_k) \in I_{kj}, v'_k \neq v_k} [f(v'_k - v_i) \times f(v_j - v'_k)]\} \times f(v_k - v_i) \times f(v_j - v_k) \times B.$$

We will now show a property that holds for any two elements $a, b \in A$ of a semiring with an idempotent multiplicative operator: $a \times (a + b) = a$. In [BMR97] it is shown that $\times$ is $intensive$ with respect to the ordering of the semiring, that is, for any $a, c \in A$ we have $a \times c \leq_S a$. In particular this holds for $c = (a + b)$ and thus $a \times (a + b) \leq_S a$. On the other hand, since $a + b$ is the lub of a and b, $(a + b) \geq_S a$, and by monotonicity of $\times$ we get that $a \times (a + b) \geq a \times a$. At this point we use the idempotency assumption on $\times$ and obtain $a \times a = a$ and, thus, $a \times (a + b) \geq_S a$. Therefore $a \times (a + b) = a$.

We now use this result in the formula describing $val'(t)$, setting

$$a = (f(v_k - v_i) \times f(v_j - v_k))$$

and

$$b = \sum_{v'_k | (v'_k - v_i) \in I_{ik}, (v_j - v'_k) \in I_{kj}, v'_k \neq v_k} [f(v'_k - v_i) \times f(v_j - v'_k)].$$

We obtain:

$$val'(t) = f(v_j - v_i) \times f(v_k - v_i) \times f(v_j - v_k) \times B$$

which is exactly $val(t)$.

Under the same condition, applying this operation to a set of constraints (rather than just one) returns a final TCSPP which is always the same independently of the order of application. Again, this result can be derived from the more general results that hold for SCSPs [BMR97], as shown by following theorem.

Theorem 2.4 *Consider an TCSPP* P *defined on a semiring which has an idempotent multiplicative operator. Then, applying operation* $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ *to a set of constraints of* P *returns the same final TCSPP regardless of the order of application.*

Proof: Applying operation $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ can be seen as applying a function F to an TCSPP P that returns another TCSPP $F(P) = P'$. The local consistency algorithm that applies this operation until quiescence can thus be seen as the repetitive application of the function. As in the case of generic SCSPs it is possible to use a result of classical chaotic theory [Cou77] that ensures the independence of order application for closure operators, that is for functions that are idempotent, monotone and intensive. We will now prove that function F satisfies such requirements if the multiplicative operator is idempotent. In particular, under such assumption, function F is:

- *idempotent*: in fact applying twice operation $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$

  to constraint T_{ij}, when constraints T_{ik} and T_{kj} have not changed, gives the same result as applying it only once;

- *monotone*: w.r.t. the ordering on SCSPs defined in [BMR97]. Such ordering can be redefined here as follows: given TCSPP P_1 and TCSPP P_2 we say that $P_1 \leq_P P_2$ iff for every constraint $T^1_{pq} = \langle I^1_{pq}, f^1_{pq} \rangle$ in P_1 and corresponding constraint $T^2_{pq} = \langle I^2_{pq}, f^2_{pq} \rangle$ in P_2, then $I^1_{pq} \subseteq I^2_{pq}$ and $\forall w \in I^1_{pq}$ we have $f^1_{pq}(w) \leq f^2_{pq}(w)$. If now we consider applying operation F to P_1 and P_2 we get two new TCSPPs, $F(P_1)$ and $F(P_2)$ that differ resp. from P_1 and P_2 only on constraint T^1_{ij} and T^2_{ij}. It is easy to see, using the idempotency of $\times$, that applying $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ to a constraint can only shrink its interval and lower the preferences corresponding to the remaining elements. Since such lowering of preference depends on the preferences on constraints T_{ik} and T_{kj} and by assumption we have that $T^1_{ik} \leq_P T^2_{ik}$ and $T^1_{kj} \leq_P T^2_{kj}$, the new preferences on constraint T^1_{ij} in $F(P_1)$ are smaller or equal than those on constraint T^2_{ij} in problem $F(P_2)$. Notice that it cannot happen that the preferences on T^2_{ij}

in $F(P_2)$ are lowered more than the preferences on T^1_{ij} in $F(P_1)$. In fact, when the path consistency reduction is applied, then for every element of I^1_{ij}, say x, the new preference of x in $F(P_1)$ is $min(f^1_{ij}(x), max(min(f^1_{ik}(x_1), f^1_{kj}(x_2))))$ where $x_1 \in I^1_{ik}$ and $x_2 \in I^2_{kj}$ and $x = x_1 + x_2$. By hypothesis, we have that $I^1_{ik} \subseteq I^2_{ik}$ and $I^1_{kj} \subseteq I^2_{kj}$. Moreover, $f^1_{ij}(x) \leq f^2_{ij}(x)$ $\forall x \in I^1_{ij}$, and for every x_1 in I^1_{ik} and x_2 in I^1_{kj} we have that $f^1_{ik}(x_1) \leq f^2_{ik}(x_1)$ and $f^1_{kj}(x_2) \leq f^2_{kj}(x_2)$. This implies that $max(min(f^1_{ik}(x_1), f^1_{kj}(x_2))) \leq max(min(f^2_{ik}(t_1), f^2_{kj}(t_2)))$ where t_1 is in I^2_{ik} and t_2 is in I^2_{kj} and $t_1 + t_2 = x$. In fact the set $\{(t_1, t_2) | t_1 \in I^2_{ik}, t_2 \in I^2_{kj}, t_1 + t_2 = x\}$ is a superset of $\{(x_1, x_2) | x_1 \in I^1_{ik}, x_2 \in I^1_{kj}, x_1 + x_2 = x\}$.

- *intensive*: that is $F(P_1) \leq_P P_1$ for any TCSPP P_1. In fact, as mentioned in the previous step, $F(P_1)$ differs from P_1 only constraint T_{ij}. However operation F ensures that constraint T_{ij} in $F(P_1)$ can only have a smaller or equal interval than that in P_1 and that remaining elements can have preferences smaller or equal than the ones in P_1.

Thus any TCSPP can be transformed into an equivalent path-consistent TCSPP by applying the operation $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ to all constraints T_{ij} until no change occurs on any constraint. We will call this path consistency enforcing algorithm TCSPP_PC-2 when applied to an TCSPP and STPP_PC-2 when applied to an STPP.

Path consistency is proven to be polynomial for TCSPs, with complexity $O(n^3 R^3)$, where n is the number of variables and R is the range of the constraints [DMP91]. However, applying it is not sufficient to find a solution. Again, since a TCSP is a special instance of TCSPP over the S_{CSP} semiring, applying path consistency is not sufficient to find an optimal solution of an TCSPP, as well. On the other hand, with STPPs over the same semiring, that is STPs, applying STPP_PC-2 is sufficient [DMP91]. It is easy to infer that the hardness result for STPPs, given in Section 2.3.1, derives either from the nature of the semiring or from the shape of the preference functions.

2.3.3 Semi-convexity and path consistency

When the preference functions are linear, and the semiring chosen is such that its two operations maintain such linearity when applied to the initial preference function, it can be seen that the initial STPP can be written as a linear programming problem, solving which is tractable [CLR90]. In fact, consider any given TCSPP. For any pair of variable X and Y, take each interval for the constraint over X and Y, say $[a, b]$, with associated linear preference function f. The information given by each of such intervals can be represented by the following inequalities and equation: $X - Y \leq b$, $Y - X \leq -a$ and $f_{X,Y} = c_1(X - Y) + c_2$. Then if we choose the fuzzy semiring $S_{FCSP} = \langle [0, 1], max, min, 0, 1 \rangle$, the global preference value V will satisfy the inequality $V \leq f_{X,Y}$ for each preference function $f_{X,Y}$ defined in the problem, and the objective is $max(V)$. If instead we choose the semiring $\langle R, min, +, \infty, 0 \rangle$, where the objective is to minimize the sum of the preference level, we have $V = f_1 + \ldots + f_n$[3] and the objective function is $min(V)$. In both cases the resulting set of formulas constitutes a linear programming problem.

Linear preference functions are expressive enough for many cases, but there are also several situations in which we need preference functions which are not linear. A typical example arises when we want to state that the distance between two variables must be as close as possible to a single value. Then, unless this value is one of the extremes of the interval, the preference function is convex, but not linear. Another case is one in which preferred values are as close as possible to a single distance value, but in which there are some subintervals where all values have the same preference. In this case, the preference criteria define a *step function*, which is not even convex.

We consider the class of semi-convex functions which which includes linear, convex, and also some step functions. Semi-convex functions are such that, if one draws a horizontal line anywhere in the Cartesian plane corresponding to the graph of the function, the set of X such that $f(X)$ is not below the line forms a single interval. More formally, a *semi-convex* function f is one such that, for all

[3]In this context, the "+" is to be interpreted as arithmetic "+"

$y \in \Re^+$, the set $F(X)\{x \in X$ such that $f(x) \geq y\}$ forms an interval. For example, the *close to k* criteria cannot be coded into a linear preference function, but it can be specified by a semi-convex preference function, which could be $f(x) = x$ for $x \leq k$ and $f(x) = 2k - x$ for $x > k$. Figure 2.9 shows some examples of semi-convex and non-semi-convex functions.

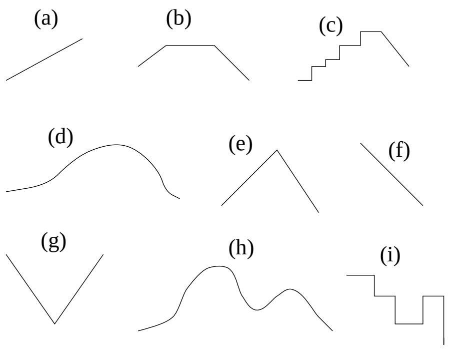


Figure 2.9: Examples of semi-convex functions [(a)-(f)] and non-semi-convex functions [(g)-(i)]

Semi-convex functions are closed under the operations of intersection and composition, when certain semirings are chosen. For example, this happens with the fuzzy semiring, where the intersection performs the min and composition performs max operation.

Theorem 2.5 (closure under intersection) *Given two semi-convex preference functions* f_1 *and* f_2 *which return values over a totally-ordered semiring* $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ *with an idempotent multiplicative operator* $\times$*, let* f *be defined as* $f(a) = f_1(a) \times f_2(a)$*. Then,* f *is a semi-convex function as well.*

Proof: Given any y, consider the set $\{x : f(x) \geq y\}$, which by definition coincides with $= \{x : f_1(x) \times f_2(x) \geq y\}$. Any semiring (even one with a partially ordered set and with a non-idempotent operator) is such that the multiplicative operator is intensive w.r.t. the ordering induced by the additive operator of the semiring. That is, $\forall a, b \in A$, $a \times b \leq a$ and $a \times b \leq b$. However, if $\times$ is idempotent then we also have $f_1(x) \times f_2(x) = glb(f_1(x), f_2(x))$. Moreover, if S is totally ordered, we have $glb(f_1(x), f_2(x)) = min(f_1(x), f_2(x))$, that is the glb coincides with one of the two elements, that is the minimum of the two [BMR97]. Thus, given the hypothesis of our theorem we have

$$\{x : f_1(x) \times f_2(x) \geq y\} = \{x : min(f_1(x), f_2(x)) \geq y\}.$$

Of course,

$$\{x : min(f_1(x), f_2(x)) \geq y\}$$

$$= \{x : f_1(x) \geq y \text{ and } f_2(x) \geq y\} = \{x : x \in [a_1, b_1] \text{ and } x \in [a_2, b_2]\}$$

since each of f_1 and f_2 is semi-convex. Now, by definition,

$$\{x : x \in [a_1, b_1] \text{ and } x \in [a_2, b_2]\}$$

$$= [a_1, b_1] \cap [a_2, b_2] =$$

$$= [max(a_1, a_2), min(b_1, b_2)].$$

We have thus proven the semi-convexity of f, since $\{x : f(x) \geq y\}$ is a unique interval.

A similar result holds for the composition of semi-convex functions:

Theorem 2.6 (closure under composition) *Given a totally ordered semiring with an idempotent multiplicative operation* $\times$, *let* f_1 *and* f_2 *be semi-convex functions which return values over the semiring. Define* f *as* $f(a) = \sum_{b+c=a}(f_1(b) \times f_2(c))$. *Then* f *is semi-convex.*

Proof: From the definition of semi-convex functions, it suffices to prove that, for any given y, the set $S = \{x : f(x) \geq y\}$ identifies a unique interval. If S is empty, then it identifies the empty interval. In the following we assume S to be not empty.

By definition of f:

$$\{x : f(x) \geq y\} = \{x : \sum_{u+v=x} (f_1(u) \times f_2(v)) \geq y\}.$$

In any semiring, + is the lub operator. Moreover, if the semiring has an idempotent $\times$ operator and is totally ordered, the lub of a finite set is the maximum element of the set. Thus,

$$\{x : \sum_{u+v=x} (f_1(u) \times f_2(v)) \geq y\} = \{x : max_{u+v=x}(f_1(u) \times f_2(v)) \geq y\}$$

which coincides with

$$\{x : \exists u, v \mid x = u + v \text{ and } (f_1(u) \times f_2(v)) \geq y\}.$$

Using the same steps as in the proof of Theorem 2.5, we have:

$$\{x : \exists u, v \mid x = u + v \text{ and } (f_1(u) \times f_2(v)) \geq y\}$$

$$= \{x : \exists u, v \mid x = u + v \text{ and } min(f_1(u), f_2(v)) \geq y\}$$

$$= \{x : \exists u, v \mid x = u + v \text{ and } f_1(x) \geq y \text{ and } f_2(x) \geq y\}$$

$$= \{x : \exists u, v \mid x = u + v \text{ and } \exists a_1, b_1, a_2, b_2 \mid u \in [a_1, b_1] \text{ and } v \in [a_2, b_2]\}$$

since each of f_1 and f_2 is semi-convex. This last set coincides with

$$\{x : x \in [a_1 + a_2, b_1 + b_2]\}.$$

We have thus shown the semi-convexity of a function obtained by combining semi-convex preference functions.

These results imply that applying the STPP_PC-2 algorithm to an STPP with only semi-convex preference functions, and whose underlying semiring contains a multiplicative operation that is idempotent, will result in a network whose induced soft constraints also contain semi-convex preference functions.

Consider now an STPP with semi-convex preference functions and defined on a semiring with an idempotent multiplicative operator, like $S_{FCSP} = \langle [0, 1]$

max, min, $0, 1\rangle$. In the following theorem we prove that if such an STPP is also path consistent then all its preference functions must have the same maximum preference value.

Theorem 2.7 *Consider a path consistent STPP* P *with semi-convex functions defined on a semiring with idempotent multiplicative operator. Then all its preference functions have the same maximum preference.*

Proof: All the preference functions of an STPP have the same maximum iff any pair of soft temporal constraints of the STPP is such that their preference function have the same maximum. Notice that the theorem trivially holds if there are only two variables. In fact, in this case, path consistency is reduced to performing the intersection on all the constraints defined on the two variables and a single constraint is obtained.

For the following of the proof we will denote with $T_{IJ} = \langle I_{IJ}, f_{IJ}\rangle$ the soft temporal constraint defined on variables I and J.

Let us now assume that there is a pair of constraints, say T_{AB} and T_{CD}, such that $\forall h \in I_{AB}, f_{AB}(h) \leq M$, and $\exists r \in I_{AB}, f_{AB}(r) = M$, and $\forall g \in I_{CD}, f_{CD}(g) \leq m$, and $\exists s \in I_{CD}, f_{CD}(s) = m$ and $M > m$.

Let us first note that any soft temporal constraint defined on the pair of variables (I, J) induces a constraint on the pair (J, I), such that $max_{h\in I_{IJ}} f_{IJ}(h) = max_{g\in I_{JI}}$ $f_{JI}(g)$. In fact, assume that $I_{IJ} = [l, u]$. This constraint is satisfied by all pairs of assignments to I and J, say v_I and v_J, such that $l \leq v_J - v_I \leq u$. These inequalities hold iff $-u \leq v_I - v_J \leq -l$ hold. Thus the interval of constraint T_{IJ} must be $[-u, -l]$. Since each assignment v_J to J and v_I to I which identifies element $h \in [l, u]$ with a preference $f_{IJ}(x) = p$ identifies element $-x \in [-u, -l]$, it must be that $f_{IJ}(x) = f_{JI}(-x)$. Thus f_{IJ} and f_{JI} have the same maximum on the respective intervals.

Consider now the triangle of constraints T_{AC}, T_{AD} and T_{DC}. To do this, we assume that, for any three variables, there are constraints connecting every pair of them. This is without loss of generality because we assume to work with a path-consistent STPP.

Given any element a of I_{AC}, since the STPP is path consistent it must be that:

$$f_{AC}(a) \leq \sum_{a_1+a_2=a} (f_{AD}(a_1) \times f_{DC}(a_2))$$

where a_1 is in I_{AD} and a_2 is in I_{DC}.

Let us denote with $max(f_{IJ})$ the maximum of the preference function of the constraint defined on variables I and J on interval I_{IJ}. Then, since $\times$ and + are monotone, the following must hold:

$$f_{AC}(a) \leq \sum_{a_1+a_2=a} ((f_{AD}(a_1) \times f_{DC}(a_2))) \leq max(f_{AD}) \times max(f_{DC}).$$

Notice that this must hold for every element a of I_{AC}, thus also for those with maximum preference, thus:

$$max(f_{AC}) \leq max(f_{AD}) \times max(f_{DC}).$$

Now, since $\times$ is intensive, we have $max(f_{AD}) \times max(f_{DC}) \leq max(f_{DC})$. Therefore, $max(f_{AC}) \leq max(f_{DC}) = m$.

Similarly, if we consider the triangle of constraints T_{CB}, T_{CD}, and T_{DB} we can conclude that $max(f_{CB}) \leq max(f_{CD}) = m$.

We now consider the triangle of constraints T_{AB}, T_{AC} and T_{CB}. Since the STPP is path consistent, then $max(f_{AB}) \leq max(f_{AC})$ and $max(f_{AB}) \leq max(f_{CB})$. But this implies that $max(f_{AB}) \leq m$, which contradicts the hypothesis that $max(f_{AB}) = M > m$.

Consider an STPP P that satisfies the hypothesis of Theorem 2.7 having, hence, the same maximum preference M on every preference function. Consider the STP P' obtained by P taking the subintervals of elements mapped on each constraint into preference M. In the following lemma we will show that, if P is a path consistent STPP, then P' is a path consistent STP.

Lemma 2.1 *Consider a path consistent STPP P with semi-convex functions defined on a semiring with idempotent multiplicative operator. Consider now the STP* P' *obtained from P by considering on each constraint only the subinterval mapped by the preference function into its maximal value for that interval. Then,* P' *is a path consistent STP.*

Proof: An STP is path consistent iff all its constraints are path consistent, that is, for every constraint T_{AB}, we have $T_{AB} \subseteq \oplus_K(T_{AK} \otimes T_{KB})$, where K varies over the set of variables of the STP [DMP91]. Assume that P' is not path consistent. Then there must be at least a hard temporal constraint of P', say T_{AB}, defined on variables A and B, such that there is at least a variable C, with $C \neq A$ and $C \neq B$, such that $T_{AB} \not\subset T_{AC} \otimes T_{CB}$. Let $[l_1, u_1]$ be the interval of constraint T_{AB}, $[l_2, u_2]$ the interval of constraint T_{AC} and $[l_3, u_3]$ the interval of constraint T_{CB}. The interval of constraint $T_{AC} \otimes T_{CB}$ is, by definition, $[l_2 + l_3, u_2 + u_3]$. Since we are assuming that $T_{AB} \not\subset T_{AC} \otimes T_{CB}$, it must be that $l_1 < l_2 + l_3$, or $u_2 + u_3 < u_1$, or both.

Let us first assume that $l_1 < l_1 + l_2$ holds. Now, since l_1 is an element of an interval of P', by Theorem 2.7 it must be that $f_{AB}(l_1) = M$, where f_{AB} is the preference function of the constraint defined on A and B in STPP P and M is the highest preference in all the constraints of P. Since P is path consistent, then there must be at least a pair of elements, say a_1 and a_2, such that $a_1 \in I_{AC}$ (where I_{AC} is the interval of the constraint defined on A and C in P), $a_2 \in I_{CB}$ (where I_{CB} is the interval of the constraint defined on C and P in P), $l_1 = a_1 + a_2$, and $f_{AC}(a_1) \times f_{CB}(a_2)) = M$. Since $\times$ is idempotent and M is the maximum preference on any constraint of P, it must be that $f_{AC}(a_1) = M$ and $f_{CB}(a_2) = M$. Thus, $a_1 \in [l_2, u_2]$ and $a_2 \in [l_3, u_3]$. Therefore, it must be that $l_2 + l_3 \leq a_1 + a_2 \leq u_2 + u_3$. But this is in contradiction with the fact that $l_1 = a_1 + a_2$ and $l_1 < l_1 + l_2$.

Similarly for the case in which $u_2 + u_3 < u_1$.

Notice the the above lemma, and the fact that an STP is path consistent iff it is consistent (i.e., it has at least a solution) [DMP91] allows us to conclude that if an STPP is path consistent, then there is at least a solution with preference M. In the theorem that follows we will claim that M is also the highest preference assigned to any solution of the STPP.

Theorem 2.8 *Consider a path consistent STPP P with semi-convex functions defined on a semiring with idempotent multiplicative operator and with maximum preference* M *on each function. Consider now the STP* P' *obtained from P by considering on each constraint only the subinterval mapped by the preference function into* M*. Then, the set of optimal solutions of P is the set of solutions of* P'*.*

Proof: Let us call $\mathtt{Opt}(P)$ the set of optimal solutions of P, and assume they all have preference $\mathtt{opt}$. Let us also call $\mathtt{Sol}(P')$ the set of solutions of P'. By Lemma 2.1, since P is path consistent, we have $\mathtt{Sol}(P') \neq \emptyset$.

First we show that $\mathtt{Opt}(P) \subseteq \mathtt{Sol}(P')$. Assume that there is an optimal solution $s \in \mathtt{Opt}(P)$ which is not a solution of P'. Since s is not a solution of P', there must be at least a hard constraint of P', say I'_{ij} on variables X_i and X_j, which is violated by s. This means that the values v_i and v_j which are assigned to variables X_i and X_j by s are such that $v_j - v_i \notin I'_{ij}$. We can deduce from how P' is defined, that $v_j - v_i$ cannot be mapped into the optimal preference in the corresponding soft constraint in P, $T_{ij} = \langle I_{ij}, f_{ij} \rangle$. This implies that the global preference assigned to s, say $f(s)$, is such that $f(s) \leq f_{ij}(v_j - v_i) < \mathtt{opt}$. Hence $s \notin \mathtt{Opt}(P)$ which contradicts out hypothesis.

We now show $\mathtt{Sol}(P') \subseteq \mathtt{Opt}(P)$. Take any solution t of P'. Since all the intervals of P' are subintervals of those of P, t is a solution of P as well. In fact, t assigns to all variables values that belong to intervals in P′ and are hence mapped into the optimal preference in P. This allows us to conclude that $t \in \mathtt{Opt}(P)$.

2.4 Solving Simple Temporal Problems with Preferences

In this section we describe two solvers for STPPs. Both find an STP such that all its solutions are optimal solutions of the STPP given in input. Finding a single optimal can, thus, be done without backtracking. Both solvers require some tractability assumptions to hold, one on the shape of the preference functions and the others on the underlying semiring. The first solver applies to the STPP an extended version of path consistency which takes into account also preferences. The second solver, instead, decomposes the search into testing the consistency of some related STPs, which can be extracted form the STPP.

2.4.1 Path-solver

The theoretical results of the previous section can be translated in practice as follows: to find an optimal solution for an STPP, we can first apply path-consistency and then use a search procedure to find a solution without the need to backtrack. Summarizing, we have shown that:

- Semi-convex functions are closed w.r.t. path-consistency: if we start from an STPP P with semi-convex functions, and we apply path-consistency, we get a new STPP P' with semi-convex functions (by Theorems 2.5 and 2.6); the only difference in the two problems is that the new one can have smaller intervals and worse preference values in the preference functions.
- After applying path-consistency, all preference functions in P' have the same best preference level (by Theorem 2.7).
- Consider the STP obtained from the STPP P' by taking, for each constraint, the sub-interval corresponding to the best preference level; then, the solutions of such an STP coincide with the best solutions of the original P (and also of P'). Therefore, finding a solution of this STP means finding an optimal solution of P.

Our first solving module, which we call Path-solver, relies on these results. In fact, the STPP solver takes as input an STPP with semi-convex preference functions, and returns an optimal solution of the given problem, working as follows and as shown in Figure 2.10. First, path-consistency is applied to the given problem, by function STPP_PC-2, producing a new problem P'; then, an STP corresponding to P' is constructed, applying Reduce_to_Best to P', by taking the

```
Pseudocode for Path-solver
1. input STPP $P$;
2. STPP $P'$ ←STPP_PC-2(P);
3. if $P'$inconsistent then exit;
4. STP $P''$ ← Reduce_to_Best($P'$);
5. return Earliest_Best($P''$).
```

Figure 2.10: Path-solver.

subintervals corresponding to the best preference level and forgetting about the preference functions. Finally, a backtrack-free search is performed to find a solution of the STP, specifically the earliest one is returned by function Earliest_Best. All these steps are polynomial, so the overall complexity of solving an STPP with the above assumptions is polynomial, as proven in the following theorem.

Theorem 2.9 *Given an STPP with semi-convex preference functions, defined on a semiring with an idempotent multiplicative operator, with n variables, maximum size of intervals r, and l distinct totally ordered preference levels, the complexity of Path-solver is* $O(n^3rl)$.

Proof: Let us follow the steps performed by Path-solver to solve an STPP. First we apply STPP_PC-2. This algorithm must consider n^2 triangles. For each of them, in the worst case, only one of the preference values assigned to the r different elements is moved down of a single preference level. This means that we can have $O(rl)$ steps for each triangle. After each triangle is visited, at most n new triangles are added to the queue. In fact, consider the triangle of constraints T_{ij}, T_{ik} and T_{kj}. When this triangle is visited the path consistency reduction is applied to T_{ij} which, thus, is the only constraint of the triangle that might change. If it does change we must consider all the other triangles which include T_{ij}. Notice that there is one such triangle of every variable X_h, where $h \neq i, j$. Thus there are

$O(n)$ of them. If we assume that each step which updates T_{ij} needs constant time, we can conclude that the complexity of STPP_PC-2 is $O(n^3 rl)$.

After STPP_PC-2 has finished, the optimal solution must be found. This has the same complexity of finding a solution of STPP P'', obtained from P'considering only intervals mapped into maximum preference M on each constraint. In [DMP91] it has been shown that this can be done without backtracking in n steps (see also Section 2.2.1). At each step a value is assigned to a new variable not violating the constraints that relate this variable with the previously assigned variables. Assume there are d possible values left in the domain of the variable, then the compatibility of each value must be checked on at most $n-1$ constraints. Since for each variable the cost of finding a consistent assignment is $O(rd)$, the total cost of finding a complete solution of $P'$is $O(n^2 d)$. The complexity of this phase is clearly dominated by that of STPP_PC-2. This allows us to conclude that the total complexity of finding an optimal solution of P is $O(n^3 rl)$.

A more realistic measure of the complexity can be obtained by counting the arithmetic operations. Every time STPP_PC-2 performs relaxation $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ on a triangle of constraints, the total number of operations required is $O(r^3)$. The reason for this is that each constraint of the triangle has at most r elements in the interval, and for each element of the constraint T_{ij} the preference of $O(r^2)$ decompositions must be checked. With this new measure, the total complexity of finding an optimal solution is $O(r^4 n^3 l)$. From this complexity it is also possible to see why Path-solver is not very efficient if the instance given in input is a hard constraint problem, that is an STPP defined on the S_{FCSP} semiring in which all the preference functions are the constant function equal to 1. In fact, the complexity of algorithm PC-2, designed to enforce path consistency on hard temporal constraint problems, is $O(r^3 n^3)$ when expressed in terms of arithmetic operations [DMP91] (see also Section 2.2.1) . This means that the complexity of Path-solver on hard constraint problems is that of PC-2 multiplied by a factor equal to $2r$. In fact, path solver would perform the path consistency operations on each constraint ignoring that all the preferences are equal to 0 or 1. Obviously

Path-solver is even more inefficient on this type of problems when compared to All-Pairs-Shortest-Paths which has a complexity of $O(n^3)$ (see Section 2.2.1).

In Figure 2.11 we show the effect of applying Path-solver on the example described in Section 2.3 and depicted in Figure 2.8. As it can be seen, the interval on the constraint on variables **Ss** and **Is** has been reduced from [3,20] to [3,14] and some preferences on all the constraints have been lowered. It is easy to see that the optimal preference of the STPP is 1 and the minimal STP containing all optimal solutions restricts the duration of the slewing to interval [4,5], the interleaving time between the slewing start and the image start to [3,5] and the interleaving time between the slewing stop and the image start to [0,1].

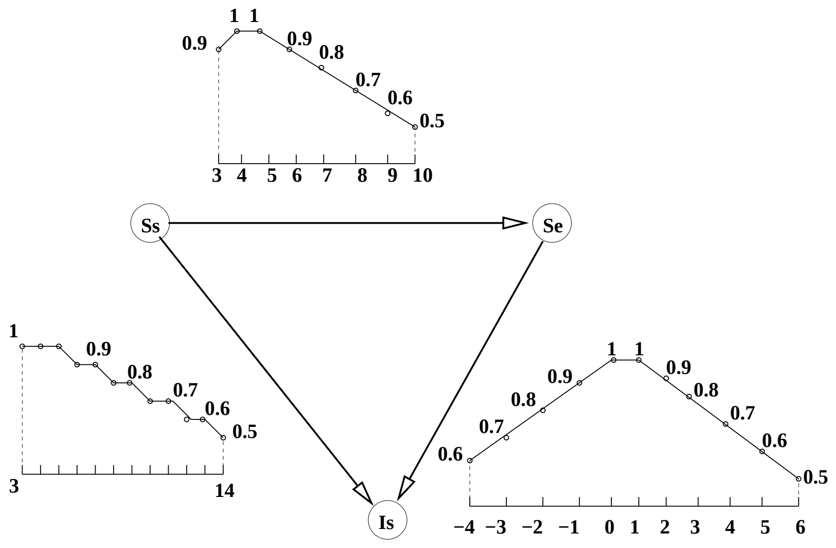


Figure 2.11: The STPP representing the Landsat 7 scenario depicted in Figure 2.8 after applying STPP_PC-2.

2.4.2 Chop-solver

Given an STPP and an underlying semiring with A the set of preference values, let $y \in A$ and $\langle I, f \rangle$ be a soft constraint defined on variables X_i and X_j in the STPP, where f is semi-convex. Consider the interval defined by $\{x : x \in I \cap f(x) \geq y\}$

(because f is semi-convex this set defines a single interval for any choice of y). This concept is similar to that of α-cut in Fuzzy Set theory [Zad75, DOP00]. In particular, in that context $\{x : x \in I \cap f(x) \geq y\}$ is called the y-level cut of function f. Let this interval define a constraint on the same pair X_i and X_j. Performing this transformation on each soft constraint in the original STPP results in an STP, which we refer to as STP_y. (Notice that not every choice of y will yield an STP that is solvable.) Let opt be the highest preference value (in the ordering induced by the semiring) such that STP_{opt} has a solution. We will now prove that the solutions of STP_{opt} are the optimal solutions of the given STPP.

Theorem 2.10 *Consider any STPP with semi-convex preference functions over a totally ordered semiring with $\times$ idempotent. Take opt as the highest y such that STP_y has a solution. Then the solutions of STP_{opt} are the optimal solutions of the STPP.*

Proof: First we prove that every solution of STP_{opt} is an optimal solution of STPP. Take any solution of STP_{opt}, say t. This instantiation t, in the original STPP, has global preference $val(t) = f_1(t_1) \times \ldots \times f_n(t_n)$, where t_i is the distance $v_j - v_i$ for an assignment to variable X_i, X_j, $(v_i, v_j) = t \downarrow_{X_i,X_j}$, and f_i is the preference function associated with the soft constraint $\langle I_i, f_i \rangle$, with $v_j - v_i \in I_i$. Now assume for the purpose of contradiction that t is not optimal in STPP. That is, there is another instantiation t' such that $val(t') > val(t)$. Since $val(t') = f_1(t'_1) \times \ldots \times f_n(t'_n)$, by strict monotonicity of $\times$, we can have $val(t') > val(t)$ only if each of the $f_i(t'_i)$ is greater or equal than the corresponding $f_i(t_i)$ and the inequality is strict for at least an i. But this means that we can take the smallest such value $f_i(t'_i)$, call it w', and construct the $STP_{w'}$. It is easy to see that $STP_{w'}$ has at least one solution, t', therefore opt is not the highest value of y, contradicting our assumption.

Next we prove that every optimal solution of the STPP is a solution of STP_{opt}. Take any t optimal for the STPP, and assume it is not a solution of the STP_{opt}. This means that, for some constraint, $f(t_i) < opt$. Therefore if we compute $val(t)$ in STPP. we have that $val(t) < opt$. Then take any solution t' of STP_{opt}, since

$\times$ is idempotent, we have that $val(t') \leq opt$, thus t was not optimal as initially assumed.

This result implies that finding an optimal solution of the given STPP with semi-convex functions reduces to a two step search process, consisting of iteratively choosing a w and then solving STP_w, until STP_{opt} is found. Both phases can be performed in polynomial time, and hence the entire process is tractable. This approach allows us to relate the tractability of STPPs with semi-convex functions with the polynomial class induced by the LP described at the beginning of Section 2.3.3. Such mapping can be done by considering each STP_w as having constant preference functions equal to w. Thus, solving the STPP reduces to solving a finite number of linear programs. Since semi-convexity is preserved only if the multiplicative operator of the semiring is idempotent then only S_{FCSP} can be considered as the underlying semiring.

The second solver for STPPs that we have implemented [RSV$^+$02], and that we will call Chop-solver, is based on this last proof of tractability for STPPs.

Chop-solver works with STPPs with semi-convex functions based on the fuzzy semiring. This means that the set of preferences we are considering are contained in the interval $[0, 1]$. The solver finds an optimal solution of the STPP identifying first STP_{opt} and then returning its earliest or latest solution. Preference level opt is found by performing a binary search in $[0, 1]$. The bound on the precision of a number, that is the maximum number of decimal coded digits, implies that the number of search steps is always finite. In Figure 2.12 we show the pseudocode for this solver.

Three variables are maintained during the search: ub containing the lowest level at which an inconsistent STP was found, lb containing the highest level at which a consistent STP was found, and y for the current level at which we need to perform the "chopping". It is easy to see that ub and lb are the upper and lower bound of the portion of the [0,1] interval to which we can restrict our search.

The algorithm takes in input an STPP P and the desired precision (lines 1 and 2). Then a counter, which will be used to store the current level of precision

Pseudocode for Chop-solver

```
1. input STPP $P$;
2. input precision;
3. integer n $\leftarrow$ 0;
4. real lb$\leftarrow 0$, ub$\leftarrow 1$, y$\leftarrow 0$, STP $STP_0 \leftarrow$ Chop($P$,y);
5. if(Consistency($STP_0$))
6.   y$\leftarrow 1$, STP $STP_1 \leftarrow$ Chop($P$,y);
7  if ($STP_0 = STP_1$) return solution of $STP_0$;
8.   if (Consistency($STP_1$)) return solution;
9.   else
10.    y$\leftarrow 0.5$, n $\leftarrow$n+1;
11.    while (n<=precision)
12.     if(Consistency(Chop($P$,y)))
13.       lb $\leftarrow$ y, y $\leftarrow$ y+(ub-lb)/2, n$\leftarrow$ n+1;
14.     else
15.       ub$\leftarrow$y, y$\leftarrow$y-(ub-lb)/2, n$\leftarrow$n+1;
16.    end of while;
17.    return solution;
18. else exit.
```

Figure 2.12: Chop-solver.

achieved in each step of the algorithm, is initialized to 0 (line 3). In line 4 variable ub is initialized to 1, variable lb to 0 and the search for the optimal preference level starts with $y = 0$. Function Chop applied to an STPP P and a preference level w returns the STP STP_w obtained chopping P at w. For each constraint of P, it considers three types of functions: a constant, a line or a semi-convex parabola. It then finds the intersection of the function with the constant function at the chopping level. Since STP_0 is the STP we would obtain by considering all the soft constraints as hard constraints, that is, with preference function equal to 1 on the elements of the interval and to 0 everywhere else, the algorithm first checks if the hard part of the problem is consistent. If it is found not to be consistent the algorithm stops, informing the user that the whole problem is inconsistent (line 5). Otherwise the search goes on considering preference level 1 and the STP, STP_1, obtained chopping the STPP at level 1 (line 6). If STP_1 is the same (in the sense it has exactly the same constraints) as STP_0 then a solution of STP_0 is returned (line 7). This happens when the algorithm is given in input a consistent hard constraint problem. If, instead, STP_0 and STP_1 are different but STP_1 is consistent, then, since 1 is the highest preference in the preference set, a solution of STP_1 is returned (line 8). Otherwise the search proceeds updating the three values, ub, lb and y, depending on the outcome of the consistency test (lines 9-17). Function Consistency receives, as input, an STP, and it checks if it is consistent. Such test is performed using All-Pairs-Shortest-Path (see Section 2.2.1). We recall that such algorithm uses a $n \times n$ matrix D, where n is the number of variables of P, called the distance matrix. Such matrix is initialized with the values of the bounds found performing the intersection. For example, consider a constraint of P on variables X_i and X_j with interval $[a, b]$ and preference function f_{ij} and a given preference level $y \in [0, 1]$. Let us denote with $[a', b']$ the set $\{x \in [a, b] | f_{ij}(x) \geq y\}$. Then, the following elements of D would be initialized as follows: $d_{ij} = b'$ and $d_{ji} = -a'$. By running Floyd-Warshall's All-Pairs-Shortest-Path on D it is possible test if the STP is consistent. In fact it is consistent if and only if there are no negative diagonal elements(d_{ii}). If the number of decimal digits given in input is reached, then Chop-solver returns a solution (either the earliest or the latest solution, respectively corresponding to the assignments $x_i = -d_{i0}$ and $x_i = d_{0i}$). If instead the last STP considered is inconsistent a solution of the last consistent one is returned. The solution returned is always made of integers, that is, in the case of the earliest solution, the real numbers found intersecting the preference functions with the chopping level are approximated to the first larger integer while, for the latest, the approximation is to the largest smaller integer. For example Chop-solver when applied to the triangular STPP shown in Section 2.3 will stop the search when it reaches preference level 1 (line 8) and finds that the STP obtained chopping the STPP at 1 is consistent and has as its minimal network the same found by Path-solver: [4,5] on the constraint on **Ss** and **Se**, [3,5] on the constraint on **Ss** and **Is**, and [0,1] on the constraint **Is** and **Se**.

The following theorem shows that Chop-solver allows to find an optimal solution of an STPP respecting the tractability assumptions in polynomial time.

Theorem 2.11 *The complexity of Chop-solver is $O(\mathtt{precision} \times n^3)$ where n is the number of variables and $\mathtt{precision}$ is the maximum number of steps allowed in the binary search.*

Proof: At each preference level considered within the binary search, the actual chopping of the preference functions is linear in the number of constraints and hence takes $O(n^2)$ where n is the number of variables. At each step Floyd-Warshall's algorithm is applied to solve STP_y with complexity $O(n^3)$ (see Section 2.2.1). Thus, we can conclude that the global complexity of Chop-solver is $O(\mathtt{precision} \times n^3)$ where $\mathtt{precision}$ is the maximum number of steps allowed in the binary search.

Notice that it is reasonable to consider $\mathtt{precision}$ bounded since it is directly connected to the number of decimal digits which can be represented by a machine. Moreover, if we denote with l the number of different preferences assigned in the constraints of an STPP then the iterations performed by Chop-solver, during the binary search, are $O(\log(l))$.

The version of Chop-solver that we have implemented works with quadratic functions, i.e. convex parabolas and linear functions. Moreover, it allows the user to specify at the beginning of the solving process the number n of digits he or she wants for the optimal solution's preference level. In contrast with Path-solver, Chop-solver when given in input a hard constraint problem, has the same complexity as All-Pairs-Shortest-Paths ($O(n^3)$). In fact, if the STP is inconsistent Chop-solver will find the inconsistency in line 5 if in instead the hard problem is consistent then it will stop in line 7. In both cases it will have applied All-Pairs-Shortest-Paths once.

2.4.3 WLO+: finding Pareto optimal solutions

The method proposed in the previous sections for evaluating the global temporal preference of a solution of an STPP follows the fuzzy semantics and thus is based on maximizing the minimally preferred local preference. This is sometimes called the *maximin* approach. This means that the global preference of a solution is the minimal preference which is assigned to any of its projections on the constraints.

Definition 2.27 (Weakest link w.r.t. a solution) *Consider an STPP* P *defined on the* S_{FCSP} *semiring, and a solution of* P, s. *Let the global preference assigned to* s *be* $p \in [0, 1]$. *Then a constraints,* c, *of* P *such that* $\mathtt{pref}(s, c) = p$ *is a* weakest link *with respect to* s.

In other words, given a solution, the weakest links with respect to that solution are those constraints on which the preference assigned to the projection of the solution is the same as the global preference of that solution. While its major advantage is computational efficiency, maximin optimization offers an insufficiently fine grained method for comparing solutions, for it is based on a single value. In this section we propose an approach that allows to find solutions that are optimal according to the Pareto criteria, which is a refinement of maximin, by combining Chop-solver with an iterative strategy for solving STPPs [KMMV03].

Overcoming the drowning effect As we have just mentioned, the maximin criterion offers what amounts to a coarse method for comparing solutions, based on the minimal preference value over all the projections of the solutions to local preference functions. This is known as the “drowning effect”. Consequently, following this criterion may be insufficient to find solutions that are intuitively more globally preferable.

For example, consider the following simple Mars rover planning problem, illustrated in Figure 2.13. The rover has a sensing instrument and a CPU. There are two sensing events, of durations 3 time units and 1 time unit. The sensing events are represented by the hard constraints between the variable representing the start time of the first event (resp. second event), namely ins_1^s (resp. ins_2^s),

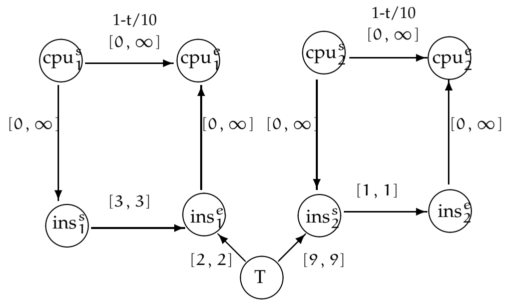


Figure 2.13: The STPP for the Rover Science Planning Problem where T is any timepoint.

and the variable representing the end of the first event (resp. second event) , ins_1^e (resp. ins_2^e). Moreover, the first sensing event must end exactly 2 units of time after we start counting, while the second event must start at 9 units of time. This is expressed by the hard constraints defined between variable "beginning of the world", T in the figure, and resp. the end time of the first event and the start time of the second event. There is a hard temporal constraint that the CPU must be on while the instrument is on. This is represented by four hard constraints. The first two are between the start times of the activity of the CPU, cpu_1^s and cpu_2^s, and the start times of the sensing events, ins_1^s and ins_2^s, and ensure that when the sensing event starts the CPU is on. The other two constraints are between the end times of the events, ins_1^e and ins_2^e and the end times of the CPU activity, cpu_1^e and cpu_2^e, which ensure that the CPU is turned off only after each sensing event has occurred. There is a soft constraint that the CPU should be on as little as possible, to conserve power. This is implemented defining two soft constraints, respectively on variables cpu_1^s and cpu_1^e and cpu_2^s and cpu_2^e. The preference function maps temporal values, t, indicating the duration that the CPU is on, to preference values. For simplicity, we assume that the preference function on the CPU duration constraints is $f(t) = 1 - t/10$, so, for example, if the CPU stays on for 1 time unit the preference will be 0.9, if it stays on for 3 time units it will be 0.7 and so on. Because the CPU must be on at least as long as the sensing events, any globally preferred solution using maximin criterion has preference value 0.7.

Notice however that for the second sensing event the CPU needs to be on only for 1 unit of time. However, the solutions in which the CPU is turned on for the second event for 1, 2 or 3 units of time all have preference 0.7 and are, thus, all optimal with respect to the maximin criterion. The fact that the fuzzy semantics is unable to discriminate between the global values of these solutions, despite the fact that the one with 1 time unit is obviously preferable to the others, is a clear limitation of this optimization criterion.

One way of formalizing this drawback is to observe that a maximin policy is not *Pareto Optimal*. To see this, we reformulate the set of preference functions of a STPP, $f_1, \dots, f_m$ as criteria requiring simultaneous optimization, and let $s = [t_1, \dots, t_n]$ and $s' = [t'_1, \dots t'_m]$ be two solutions to a given STPP. We say that s' dominates s if for each j, $f_j(t_j) \leq f_j(t'_j)$ and for some k, $f_k(t_k) < f_k(t'_k)$. In a Pareto optimization problem, the *Pareto optimal set* of solutions is the set of non-dominated solutions. Similarly, let the *fuzzy-optimal set* be the set of optimal solutions that result from applying the chopping technique for solving STPPs described in Section 2.4.2. Clearly, applying Chop-solver to an STPP does not guarantee that the set of Fuzzy-optimal solutions is a Pareto optimal set. In the rover planning problem, for example, suppose we consider only solutions where the CPU duration for the first sensing event is 3. Then the solution in which the CPU duration for the second sensing event is 1 time unit dominates the solution in which it is 2 time units, but both are Fuzzy-optimal, since they have the same preference value: 0.7.

Assuming that Pareto-optimality is a desirable objective in optimization, a reasonable response to this deficiency is to replace Chop-solver with an alternative strategy for evaluating solution tuples. A natural, and more robust alternative evaluates solutions by summing the preference values, and ordering them based on preferences towards larger values. This strategy would also ensure Pareto optimality, since every maximum sum solution is Pareto optimal. This policy might be called "utilitarian." The main drawback to this alternative is that the ability to solve STPPs tractably is no longer apparent. The reason is that the formalization of utilitarianism as a semiring forces the multiplicative operator (in this case, sum), not to be idempotent (i.e., $a + a \neq a$), a condition required in the proof that a local consistency approach is applicable to the soft constraint reasoning problem. Recently, however, in [MMK+04] it has been shown that finding utilitarian optimal solutions of an STPP can be done in polynomial time if the preference functions are piecewise linear and convex (see Section 2.7).

Of course, it is still possible to apply a utilitarian framework for optimizing generic preferences, using either local search or a complete search strategy such as branch and bound. Rather than pursuing this direction of resolving the drowning effect, we select another approach, based on an algorithm that interleaves flexible assignment with propagation using Chop-solver.

An algorithm for Pareto Optimization The proposed solution is based on the intuition that if a constraint solver using the maximin criterion could iteratively "ignore" the weakest link values (i.e. the values that contributed to the global solution evaluation) then it could eventually recognize solutions that dominate others in the Pareto sense. For example, in the Rover Planning problem illustrated in Figure 2.13, if the weakest link value of the global solution could be "ignored," the Chop-solver could recognize that a solution with the CPU on for 1 time unit during the second instrument event is to be preferred to one where the CPU is on for 2 or 3 time units.

We formalize this intuition by a procedure wherein the original STPP is transformed by iteratively selecting what we shall refer to as a *weakest links* w.r.t. the set of optimal solutions, changing the constraint in such a way that it can effectively be "ignored," and solving the transformed problem.

Definition 2.28 (Weakest link w.r.t. the set of optimal solutions) *Given an STPP* P *defined on the* S_{FCSP} *semiring and the set of all its optimal solutions,* $\mathrm{Optsol}(P)$, *all with preference* opt, *a* weakest link *w.r.t.* $\mathrm{Optsol}(P)$ *is a constraint* c *such that* $\mathrm{pref}(s, c) = \mathrm{opt}, \forall s \in \mathrm{Optsol}(P)$.

For example, after applying Chop-solver to the problem in Figure 2.13, the CPU duration constraint associated with the first sensing event will be a weakest link, since it now has a fixed preference value of 0.7. However, the CPU constraint for the second event will not be a weakest link since its preference value can still vary from 0.7 to 0.9.

We also define a weakest link constraint to be *open* if v is not the "best" preference value (i.e., $v < \mathbf{1}$, where $\mathbf{1}$ is the designated "best" value among the values in the semi-ring).

Formalizing the process of "ignoring" weakest link values is a two-step process of restricting the weakest links to their intervals of optimal temporal values, while eliminating their restraining influence by resetting their preferences to a single, "best" value (1 in the fuzzy context). Formally, the process consists of:

- Reducing the temporal domain to include all and only those values which are optimally preferred; and
- Replacing the preference function by one that assigns the most preferred value (i.e. $\mathbf{1}$) to each element in the new domain.

The first step ensures that only the best temporal values are part of any solution, and the second step allows Chop-solver to be re-applied to eliminate Pareto-dominated solutions from the remaining solution space.

The pseudocode of an algorithm which follows this technique, called WLO+, is shown in Figure 2.14.

The algorithm takes as input an STPP P satisfying the usual tractability assumptions (step 1). Then, it applies to P Chop-solver (step 3). This implies finding the optimal preference value of P, say $\mathtt{opt}_1$, and the intervals on the constraints containing elements that belong to optimal solutions. The algorithm searches for the constraints such that the maximum preference obtained on elements belonging to at least one solution is exactly $\mathtt{opt}_1$. These constraints are the weakest links and must be replaced. This is performed in steps 4-6. In step 5, in particular, we denote the soft constraint that results from the two-step process described above

Pseudocode for WLO+ solver

```
1. input STPP P;
2. do {
3.    STPP P' ← Chop-solver(P);
4.    Delete from P' all weakest links;
5.    For each deleted weakest link constraint ⟨[a, b], f⟩ :
6.      add ⟨[a_opt, b_opt], f_best⟩ to P';
7. until (3)-(6) leave P' unchanged;
8. Return STP(P) P'.
```

Figure 2.14: WLO+ Solver: finds Pareto undominated solutions of a given STPP.

as $\langle [a_{opt}, b_{opt}], f_{best} \rangle$, where $[a_{opt}, b_{opt}]$ is the interval of temporal values that are optimally preferred, and f_{best} is the preference function such that $f_{best}(v) = \mathbf{1}$ for any input value v.

Notice that the run time of WLO+ is $O(|X|^2)$, where X is the set of variables of P, multiplied by the time it takes to execute Chop-solver(P), which is a polynomial in $|X|$.

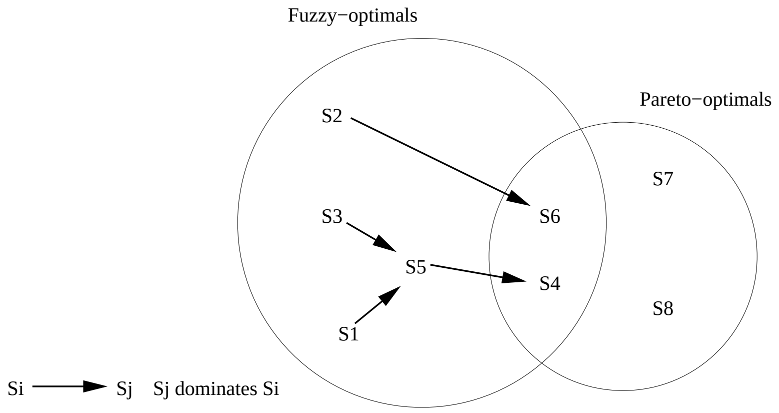


Figure 2.15: Relationships between Pareto-optimal and fuzzy-optimal solutions.

We now proceed to prove the main result, in two steps. We assume the existence of weakest links at every iteration of the WLO+ algorithm, and show that the subset of solutions of the input STPP returned by WLO+ is contained in the intersection of WLO-optimal and Pareto-optimal solutions. Then we show that, given additional restrictions on the shape of the preference functions, such weakest links can be shown to always exist.

Given an STPP P, let $\mathtt{Sol}_F(\mathtt{P})$ (resp. $\mathtt{Sol}_{PAR}(\mathtt{P})$) be the set of fuzzy-optimal (respectively, Pareto-optimal) solutions of P, and let $\mathtt{Sol}_{WLO+}(\mathtt{P})$ be the set of solutions to $\mathtt{P}$ returned by WLO+. Then the result can be stated as follows.

Theorem 2.12 *If a weakest link constraint is found at each stage of WLO+, then* $\mathtt{Sol}_{WLO+}(\mathtt{P}) \subseteq \mathtt{Sol}_F(\mathtt{P}) \cap \mathtt{Sol}_{PAR}(\mathtt{P})$*. Moreover, if* $\mathtt{P}$ *has any solution, then* $\mathtt{Sol}_{WLO+}(\mathtt{P})$ *is nonempty.*

Proof: First note that after an open weakest link is processed in steps (4) to (6), it will never again be an open weakest link (since its preference is reset to $\mathtt{f}_{\mathtt{best}}$). Since the theorem assumes a weakest link constraint is found at each stage of WLO+, the algorithm will terminate when the weakest link constraint is not open, i.e., when all the soft constraints in $\mathtt{P}$ have all preferences that equal the best ($\mathbf{1}$) value.

Now assume $s \in \mathtt{Sol}_{WLO+}(\mathtt{P})$. Since the first iteration reduces the set of solutions of $\mathtt{P}$ to $\mathtt{Sol}_F(\mathtt{P})$, and each subsequent iteration either leaves the set unchanged or reduces it further, it follows that $s \in \mathtt{Sol}_F(\mathtt{P})$. Now suppose $s \notin \mathtt{Sol}_{PAR}(\mathtt{P})$. Then s must be dominated by a Pareto optimal solution s'. Let c be a soft constraint in $\mathtt{P}$ for which the preference associated to s' is strictly higher than that associated to s. Thus, the preference value of the duration assigned by s to c cannot be $\mathbf{1}$. It follows that at some point during the course of the algorithm, c must become an open weakest link. Since s is in $\mathtt{Sol}_{WLO+}(\mathtt{P})$, it survives until then, and so it must provide a value for c that is equal to the chop level. However, since s' dominates s, s' must also survive until then. But this contradicts the assumption that c is a weakest link constraint, since s' has a value greater than the optimal chop level found by Chop-solver. Hence, s is in $\mathtt{Sol}_{PAR}(\mathtt{P})$, and so in $\mathtt{Sol}_F(\mathtt{P}) \cap \mathtt{Sol}_{PAR}(\mathtt{P})$.

Next suppose the original STPP P has at least one solution. To see that $\mathtt{Sol}_{WLO+}(P)$ is nonempty, observe that the modifications in steps (4) to (6). Consider the first iteration. Chop-solver is applied to P and the optimal preference $\mathtt{opt}_1$ is found. The STP P_1, obtained solving the STP derived chopping P at $\mathtt{opt}_1$, allows in the constraints only elements contained in at least a solution with preference $\mathtt{opt}_1$, that is, each interval of P, $[a, b]$ is reduced to $[a_{\mathtt{opt}_1}, b_{\mathtt{opt}_1}]]$ in P_1. WLO+ identifies the weakest links and replaces them with the corresponding intervals in P_1. The new STPP P'will be like P except for the weakest links which have been replaced by the intervals in P_1 and preference function equal to **1**. Notice that the intervals of P_1 are all included in the corresponding ones in P', thus since P_1 is consistent, P' must at least a solution, and thus an optimal solution. It is easy to see that the above reasoning hold for all the iterations of the algorithm. Since we are assuming that first P has a solution, it follows by induction that $\mathtt{Sol}_{WLO+}(P)$ is nonempty.

The theorem shows that it is possible to maintain the tractability of fuzzy-based optimization while overcoming some of the restrictions it imposes. In particular, it is possible to improve the quality of the flexible solutions generated within an STPP framework from being fuzzy-optimal to being Pareto-optimal.

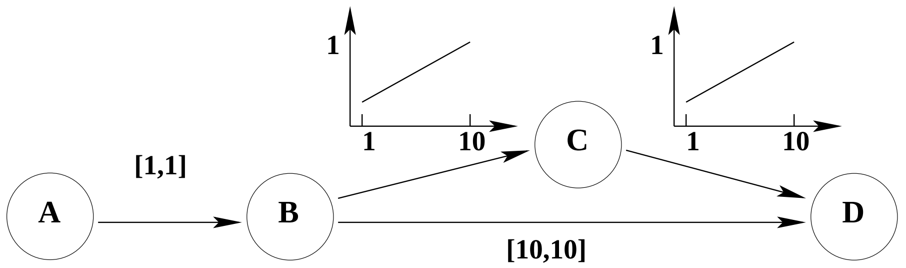


Figure 2.16: A unique WLO+ Solution.

Although the algorithm determines a nonempty set of solutions that are both fuzzy optimal and Pareto optimal, the set might not include all such solutions. Consider the example in figure 2.16. Assume the preference function for all soft constraints is given by $f(t) = t/10$ for $t = 1, \ldots, 9$, and $f(10) = 1$, i.e., longer durations are preferred. The WLO+ algorithm will retain a single solution where

BC and CD are both 5. However, the solution where BC $= 2$ and CD $= 8$, which is excluded, is also Pareto optimal. Note that AB, with a fixed value of 1, is the weakest link.

Existence of Weakest Links In this section we show that under suitable conditions, a weakest link constraint always exists. This involves a stronger requirement than for the tractability of a fuzzy STPP: the preference functions must be convex, not merely semi-convex. This would include linear functions, cycloids, and upward-pointing parabolas, for example, but not Gaussian curves, or step functions. Later on, we will see this requirement can be relaxed somewhat so that Gaussians can be permitted.

Before proceeding, we note that while a solution s of an STP P is defined in terms of an assignment to each variable, it also determines a value $s(e)$ for each constraint e, given by $s(e) = s(Y) - s(X)$ where X and Y are the start and end variables of e, respectively. We will use this notation in what follows.

Now consider any consistent STP P. The *minimal network* [DMP91] corresponding to P is another STP P'. The constraints between any two points X and Y in P' are formed by intersecting the constraints induced by all possible paths between X and Y in P. It has been shown in [DMP91] that such intervals contain only elements that belong to at least one solution.

In the following, a preference function f is said to be *convex* if $\{< x, y > | y \leq f(x)\}$ is a convex set. The claim of the existence of weakest links can be stated as follows:

Theorem 2.13 *Let P be an STPP with continuous domains and convex preference functions. Then there will be at least one weakest link constraint for the set of fuzzy-optimal solutions.*

Proof: Consider the (minimal) STP P_{opt} that corresponds to the optimal chopping level for P (as described in the Chop-solver algorithm in Section 2.4.2). Suppose there is no weakest link constraint. Then for each constraint e there is a solution s_e to P_{opt} such that $f(s_e(e)) > opt$, where f is the preference function for the edge.[4]

Let $\bar{s}$ be the average of all the s_e solutions, i.e.

$$\bar{s}(X) = 1/|E| \sum_{e \in E} s_e(X) \;\; \text{for all } X \in V$$

where E is the set of constraints and V is the set of temporal variables. By the linearity of the constraints, it is easy to show that $\bar{s}$ is also a solution to P_{opt}. For example, if $s_e(Y) - s_e(X) \geq a$ for all e, then $\bar{s}(Y) - \bar{s}(X) = 1/|E| \sum_{e \in E}(s_e(Y) - s_e(X)) \geq (1/|E|)|E|a = a$.

Since $\bar{s}$ is a solution, we must have $f(\bar{s}(e)) \geq opt$ for all edges e. Notice, however, that there must be some edge e such that $f(\bar{s}(e)) = opt$, otherwise $\bar{s}$ would be a solution with value greater than the optimal value. It follows that $\bar{s}(e) \neq s_e(e)$, since we already know that $f(s_e(e)) > opt$.

Thus, either $\bar{s}(e) < s_e(e)$ or $\bar{s}(e) > s_e(e)$. We consider only the case where $\bar{s}(e) < s_e(e)$. (The proof is similar in the other case.) Note also that $\min_{c \in E} s_c(e) < \bar{s}(e)$, since the minimum of a set of numbers can only be equal to the average if all the numbers are equal. It follows that $\min_{c \in E} s_c(e) < \bar{s}(e) < s_e(e)$. However, $f(\min_{c \in E} s_c(e)) \geq opt$, $f(\bar{s}(e)) = opt$, and $f(s_e(e)) > opt$. This violates the convexity of f, which establishes the theorem.

The operations in the WLO+ algorithm preserve the convexity property of the preference functions. Each stage of WLO+ applies Chop-solver. Thus, theorem 2.13 implies the following corollary.

Corollary 2.1 *Suppose P is an STPP with continuous domains and convex preference functions. Then a weakest link is found at each iteration of WLO+.*

An example of why the existence result does not apply more generally to semi-convex functions is found in figure 2.17. The STPP in the figure contains two semi-convex step-like functions with optimal preference values associated with

[4]To avoid excessive subscripting, we suppress the implied e subscript on f here and in what follows. In all cases, the applicable preference function will be clear from the context.

durations t and t'. Assume the STPP is minimal, and that the assignment $e = t, e' = t'$ is inconsistent. Then the highest possible chop point is p, and no weakest link exists, i.e., for neither e nor e' is it the case that, for every solution s, p is the value returned by the preference function associated with that constraint for the duration assigned by s.

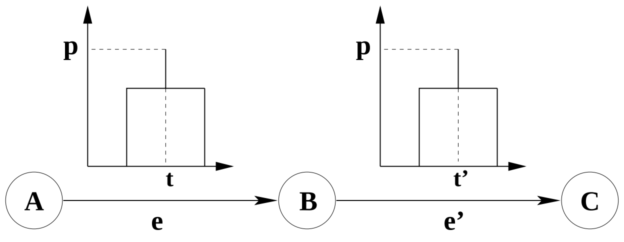


Figure 2.17: An STPP with no weakest link.

An examination of the proof of Theorem 2.13 shows that the weakest link constraint exists under a somewhat less restrictive condition than convexity: it is enough, assuming semi-convexity, to require that plateaus (subintervals of non-zero length where the preference function is constant) can only occur at the global maximum of the preference function. This means, for example, that the theorem is applicable in principle to any semi-convex smooth function such as a Gaussian curve.

However, in the practical setting of a computer program where numbers are computed to a finite precision and continuous curves are approximated, some adjustments may need to be made. Note that a representation as a discretized step function does not satisfy the no-plateau condition. An alternative is to treat a discretized function as corresponding to a piecewise linear function where the linear segments join successive points on the discretized graph. Even there, the long tails of a Gaussian curve may get approximated by horizontal segments. However, generally we can trim the domain of the curve to eliminate the flat tails without excluding all the solutions. In that case, the discretized Gaussian is acceptable. (Note that figure 2.17 could be simulated by an example involving extreme Gaussians where the tails are essential for the solution.)

Note that preferences such as longest or shortest durations, or closest to a fixed time, which appear to be the most useful in practice, can be easily modeled within this framework.

WLO+ has been implemented and tested on randomly generated problems, where each semi-convex preference function is a quadratic $ax^2 + bx + c$, with randomly selected parameters and $a \leq 0$. We compared the best solution found after applying WLO+ with the quality of the earliest solution found using the chop solver, using the utilitarian measure of quality (i.e., summing preference values). An average improvement of between 6 and 10% was observed, depending on constraint density (more improvement on lower density problems). Future research will focus on the application of WLO+ to the rover science planning domain.

2.5 Experimental Results for the Solvers

In this section we report and discuss the experimental results of the solvers for fuzzy-optimal solutions: Path-solver and chop-solver. We start by describing a random generator for fuzzy STPPs, then we consider the results obtained by each solver on randomly generated problems. We then conclude by comparing the solvers in terms of their performance.

2.5.1 Generating random STPPs

This STPP solver has been tested both on toy problems and on randomly generated problems. The random generator we have developed focuses on a particular subclass of semi-convex preference functions: convex quadratic functions of the form $ax^2 + bx + c$, with $a \leq 0$. The choice has been suggested both by the expressiveness of such a class of functions and also by the facility of expressing functions in this class by just three parameters: a, b, and c. Moreover, it generates fuzzy STPPs, thus preference values are between 0 and 1.

An STPP is generated according to the value of the following parameters:

- number n of variables;
- range r for the initial solution: to assure that the generated problem has at least one solution, we first generate such a solution, by giving to each variable a random value within the range $[0, r]$;
- density: percentage of constraints that are not universal (that is, with the maximum range and preference 1 for all interval values);
- maximum expansion from initial solution (max): for each constraint, the bounds of its interval are set by using a random value between 0 and max, to be added to and subtracted from the timepoint identified for this constraint by the initial solution;
- perturbation of preference functions (pa, pb, pc): we recall that each preference function can be described by three values (a, b, and c); to set such values for each constraint, the generator starts from a standard quadratic function which passes through the end points of the interval, with value 0, and the middlepoint, with value 0.5, and then modifies it according to the percentages specified for a, b, and c.

For example, if we call the generator with the parameters $\langle 10, 20, 30, 40, 20, 25, 30\rangle$, it will generate a fuzzy STPP with 10 variables. Moreover, the initial solution will be chosen by giving to each variable a value between 0 and 20. Among all the constraints, 70% of them will be universal, while the other 30% will be specified as follows: for each constraint, consider the timepoint specified by the initial solution, say t; then the interval will be $[t - t_1, t + t_2]$, where t_1 and t_2 are random numbers between 0 and 40. Finally, the preference function in each constraint is specified by taking the default one and changing its three parameters a, b, and c, by, respectively, 20%, 25%, and 30%.

To compare our generator with the usual one for classical CSPs, we notice that the maximum expansion (max) for the constraint intervals roughly corresponds to the tightness. However, we do not have the same tightness for all constraints, because we just set an upper bound to the number of values allowed in a constraint. Also, we do not explicitly set the domain of the variables, but we just set the constraints. This is in line with other temporal CSP generators, like the one in [SD93].

2.5.2 Experimental results for Path-solver

In Figures 2.18, 2.19 we show some results for finding an optimal solution for STPPs generated by our generator using Path-solver, which has been developed in C++ and tested on a Pentium III 1GHz.

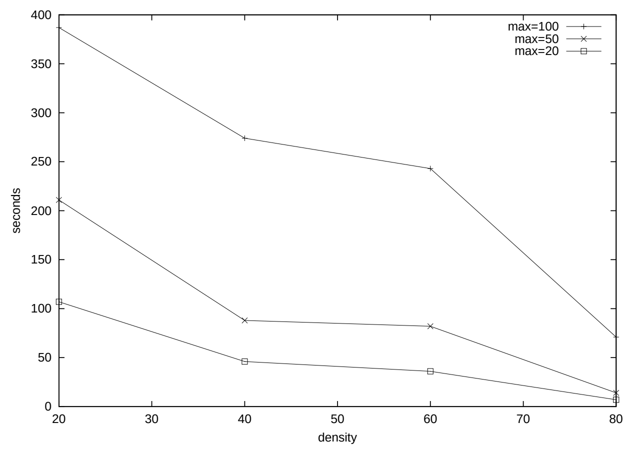


Figure 2.18: Time needed by Path-solver to find an optimal solution (in seconds), as a function of density (d). The other parameters are: n=20, r=100, pa=20, pb=20, and pc=30. Mean on 3 examples.

Among the input parameters of the random generators we have chosen to vary the number of variables n, $n = 20$ in Figure 2.18 and $n = 50$ in Figure 2.19, the density, from 20% to 80% on the x-axis, and the maximum range of intervals, 20, 50 and 100 elements.

We have chosen to leave parameter r, representing the range of the first solution, fixed to 100, since it clearly does not effect the speed of execution. In fact

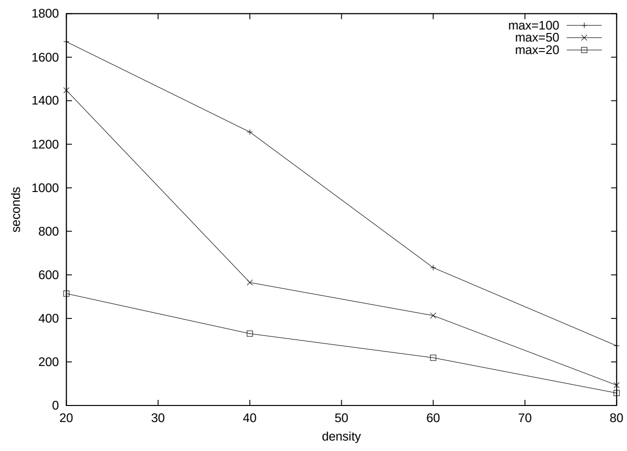


Figure 2.19: Time needed by Path-solver to find an optimal solution (in seconds), as a function of density (d). The other parameters are: n=50, r=100, pa=20, pb=20, and pc=30. Mean on 3 examples.

changing r, leaving all other parameters fixed corresponds to a "translation" of the generated problem. The other parameters which are fixed in our experiments are the distortion parameters of the parabolas pa, pb and pc. Such values are connected to the shape of the parabolas and, indirectly, determine the maximum number of distinct preference levels. However the percentages we have chosen, $pa = 20$, $pb = 20$ and $pc = 30$, have proved to allow a wide variety of preference levels between 0 and 1.

The curves depicted in Figures 2.18 and 2.19 represent the mean on 30 generated problems. By looking at these experimental results, we can see that:

- The time is directly proportional to the size of the intervals (max). This is due to the point-wise representation of constraints used by the algorithm;
- it is instead inverse proportional to the density of constraints introduced by the generator. The reason for this is that the constraints generated have smaller intervals than the universal constraints;
- Moreover, the smaller the maximum range max, the weaker the impact on

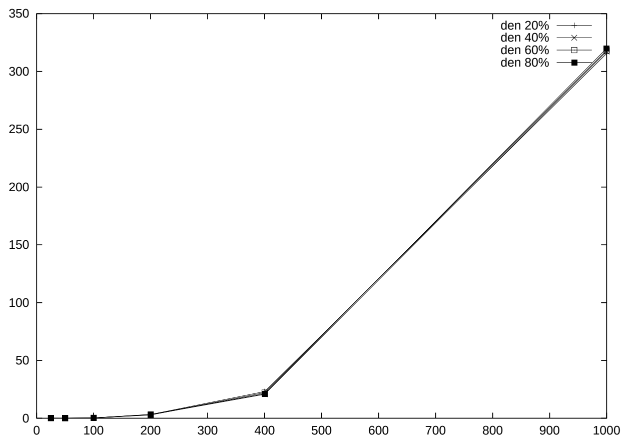


Figure 2.20: Time, in seconds, (y-axis) required by Chop-solver to solve, varying the number of variables (x-axis) and the density, with r=100000 max=50000, pa=5, pb=5 e pc=5. Mean on 10 examples.

performance of varying the density. This is another clear consequence of the relevance of the size of intervals w. r. t. performance.

As it can be seen, this solver is very slow. For example, it takes 200 seconds to solve a problem with 20 variables, maximum size of the interval 50, and the density of defined constraints equal to 20%. The main reason is that it uses a pointwise representation of the constraint intervals and the preference functions. This makes the solver more general, since it can represent any kind of preference functions, even those that don't have an analytical representation via a small set of parameters. In fact, even starting from convex quadratic functions, which need just three parameters, the first solving phase, which applies path-consistency, can yield new preference functions which are not representable via three parameters only. For example, we could get semi-convex functions which are generic step functions, and thus not representable by giving new values to the initial three parameters.

2.5.3 Experimental results for Chop-solver

Figure 2.20 shows some experimental results for Chop-solver. We have used basically the same random generator used to test the solver described in Section 3, although it has been slightly modified since the two solvers use two different representation of a constraint.

We have tested Chop-solver by varying the number of variables, from a minimum of 25 up to a maximum of 1000, and the density from 20% to 80%.

From Figure 2.20 we can conclude that Chop-solver is only slightly sensitive also to variations in the density and, in this sense, it finds more constrained problems a little more difficult. This fact can be partially explained by the way problems are generated. Having a higher density means having more constraints with non trivial parabolas, i.e. $a \neq 0$. The intersection procedure in this case is a little more complicated than in the case of constant or lines.

Chop-solver is indeed sensitive to the number of variables since it yields an increase of the number of constraints on which the intersection procedure must be performed.

The choice of maintaining a fixed maximum enlargement of the intervals, that can be interpreted as a fixed tightness, is justified by the continuous representation of the constraint this solver uses. In fact, each constraint is represented by only two integers for the left and right ends of the interval and three doubles as parameters of the function. Increasing max affects this kind of representation of a constraint only making these values bigger in modulo. This change however does not affect any of the operations performed by Chop-solver.

2.5.4 Path-solver vs. Chop-solver

In Table 2.2, 2.3 and 2.4 we can see a comparison between algorithm Chop-solver and Path-solver.

It appears clear that Chop-solver is much faster than Path-solver. It is also true that, in a sense, it's also more precise since it can find an optimal solution

	D=20	D=40	D=60	D=80
Path-solver	515.95	235.57	170.18	113.58
Chop-solver	0.01	0.01	0.02	0.02

Table 2.2: Time in seconds, used by Path-solver and Chop-solver to solve problems with $n = 30$, $r = 100$, $max = 50$, $pa = 10$, $pb = 10$, and $pc = 5$ and varying density D. Results are mean on 3 examples.

	D=20	D=40	D=60	D=80
Path-solver	1019.44	516.24	356.71	320.28
Chop-solver	0.03	0.03	0.03	0.03

Table 2.3: Time in seconds, used by Path-solver and Chop-solver to solve problems with $n = 40$, $r = 100$, $max = 50$, $pa = 10$, $pb = 10$, and $pc = 5$ and varying density D. Results are mean on 3 examples.

	D=20	D=40	D=60	D=80
Path-solver	2077.59	1101.43	720.79	569.47
Chop-solver	0.05	0.05	0.06	0.07

Table 2.4: Time in seconds, used by Path-solver and Chop-solver to solve problems with $n = 50$, $r = 100$, $max = 50$, $pa = 10$, $pb = 10$, and $pc = 5$ and varying density D. Results are mean on 3 examples.

with a higher precision. The point-to-point representation of the constraints that Path-solver uses is to be blamed for its poor performance. However a discrete representation allows to use any kind of semi-convex function even those that cannot be easily compactly parametrized. It is also true that, in general, time is dealt with as a discretized quantity, which means that, once the measuring unit that is most significant for the involved events is fixed, the problem can be automatically cast in the point-to-point representation. In general in order to use Chop-solver the functions must be parametrized with a low number of parameters and intersecting the constant at chopping level and the function must be easy. The continuous representation used by Chop-solver is, undoubtedly, more natural because it reflects the most obvious idea of temporal preferences that is, an interval plus a preference function over it.

2.6 Inductive Learning of Local Temporal Preferences

It is not always easy to specify the preference functions in each temporal constraint in a way that the real-life problem at hand is faithfully modeled. This happens because sometimes it is easier, or possible, to specify only global preference functions, to be associated to entire solutions, rather than local preference functions to be attached to the constraints. For this reason, and since the whole TCSPP machinery is based on local preference functions, we propose here a method to induce local preferences from global ones.

We now describe our methodology to learn preferences in STPPs from examples of solution ratings. Notice that here we focus on STPPs, which are tractable, rather than general TCSPPs.

2.6.1 A general strategy for learning soft temporal constraints

Learning in our context can be used to find suitable preference functions to be associated to the constraints of a given STP. More precisely, let $P = (V, C)$ be an

STP where V is a set of variables and C is a set of distance constraints of the form $l \leq X - Y \leq u$. Let also t be a function $t : S \rightarrow A$, where S is the set of solutions of P and A is a set of values indicating the "quality" of the solution.

The learning task consists of transforming the STP into an STPP, with each constraint $c_{i,j} \in C$ replaced by a soft constraint $\langle c_{i,j}, f_{i,j} \rangle$, where $f_{i,j}$ is the local preference function for $c_{i,j}$.

The examples to be used in the learning task consist of pairs $(s, t(s))$, where s is a solution to the original STP and $t(s)$ is its "score". In the following, we use P to denote an STP and P' to denote a corresponding STPP.

Let P and f be as defined above, and suppose a set of examples $TR = \{(s_1, t(s_1)), \ldots, (s_m, t(s_m))\}$ is given. To infer the local preferences, we must also be given the following: a semiring whose element set A contains the values $t(s_i)$ in the examples; a distance function over such a semiring. For example, if the score values are positive real numbers, we could choose the semiring $\langle \mathbb{R}^+, min, +, +\infty, 0 \rangle$ and, as distance function, the usual one over reals $dist(\ val_{P'}\ (s)\ , t(s)) = |\ val_{P'}(s) - t(s)\ |$. Given all of the above, the goal is to define a corresponding STPP P' over the same semiring such that P and P' have the same set of variables, variable domains and interval constraints, and for each t such that $(s, t(s))$ is an example, $dist(val_{P'}(s), t(s)) < \epsilon$, where $\epsilon > 0$ and small.

Once the semiring is decided, the only free parameters that can be arbitrarily chosen are the values to be associated to each distance. For each constraint, $c_{ij} = I_{ij} = [l_{ij}, u_{ij}]$ in an STP P, the idea is to associate, in P', a free parameter w_d, where $d = X_j - X_i$, to each element d in I_{ij}. This parameter will represent the preference over that specific distance. With the other distances, those outside I_{ij}, we associate the constant **0**, (the lowest value of the semiring (w.r.t. $\leq_S$)).

If I_{ij} contains many time points, we would need a great number of parameters. To avoid this problem, we can restrict the class of preference functions to a subset which can be described via a small number of parameters. For example, linear functions just need two parameters a and b, since they can be expressed as $a \cdot (X_j - X_i) + b$. In general, we will have a function which depends on a set of parameters W, thus we will denote it as $f_W : (W \times I_{ij}) \to A$.

The value assigned to each solution s in P' is

$$val_{P'}(s) = \prod_{c_{ij} \in P'} [\sum_{d \in I_{ij}} check(d, s, i, j) \times f_W(d)] \tag{2.1}$$

where $\prod$ generalizes the $\times$ operation, $\sum$ generalizes $+$, I_{ij} is the set of intervals associated to constraint c_{ij}, and $check(d, s, i, j) = \mathbf{1}$ if $d = s \downarrow_{X_j} - s \downarrow_{X_i}$ and $\mathbf{0}$ otherwise. Note that, for each constraint c_{ij}, there is exactly one distance d such that $check(d, s, i, j) = \mathbf{1}$, namely $d = t \downarrow_{X_j} - t \downarrow_{X_i}$. Thus, $val_{P'}(s) = \prod_{c_{ij} \in P'} f_W(s \downarrow_{X_j} - s \downarrow_{X_i})$. The values of the free parameters in W may be obtained via a minimization of the error function, which will be defined according to the distance function of the semiring.

In this learning framework, a user should provide the system with some examples (that is, rated solutions). Of course, the higher the number of examples, the higher the probability of a successful learning phase. However, it is not feasible to ask the user to provide too many examples. To overcome this, an incremental strategy which aims at reducing the number of the examples the user has to provide has been proposed in in [BRS00]. Using this strategy, the user just gives an initial small set of examples, over which the system performs the first learning phase. Then, the user checks the resulting system on a set of new solutions, and collects those that are mis-rated by the system, which will be given as new examples for the next learning phase. This process iterates until the user is satisfied with the current state of the system. We will use this approach in the temporal framework described in this chapter.

Suppose we are given a class of STPs to be "softened" via the learning approach defined above. As we know, STPs are tractable [DMP91]. However, in general we may end up with STPPs which are not tractable, since there is no guarantee that our learning approach returns preference functions which are semi-convex. For this reason the technique described in in [BRS00] cannot be used directly and a new learning algorithm is introduced, in order to guarantee the tractability of the STPP produced in output.

Moreover, one needs to choose a semiring which preserves semi-convexity.

To force the learning framework to produce semi-convex functions, we can specialize it for a specific class of functions with this property. For example, we could choose convex quadratic functions (parabolas) of the form $f(d) = a \cdot d^2 + b \cdot d + c$, where $a \leq 0$. In this case we just have three parameters to consider: $W = \{a, b, c\}$.

Of course, by choosing a specific class of semi-convex functions f_W, not all local preference shapes will be representable. Therefore, there will be cases in which the desired solution ratings, as specified in the training set, cannot be matched. For example, the user could have specified a set of examples which is not consistent with any soft temporal constraint problem using that class of semi-convex preference functions. Even if one chooses f_W to cover any semi-convex function, there is no guarantee that the desired solution ratings will be matched. In general, the learning process will return a soft temporal problem which will approximate the desired rating as much as possible, considering the chosen class of functions and the error function. But we will gain tractability for the solution process of the resulting problems: starting from the class of STPs, via the learning approach we will obtain a class of STPPs which is tractable as well.

2.6.2 The learning module

We will now describe in detail how the gradient descent technique can be implemented when all the preference functions are convex quadratics and the underlying semiring is the fuzzy semiring $S_{FCSP} = \{[0, 1], max, min, 1, 0\}$. If the problem we are considering has n variables only $\nu = (\frac{(n-1)n}{2}) - n$ of the n^2 constraints need to be considered, since the remaining are just reciprocal. Given solution s let s_i be its projection on the i-th constraint. The global preference assigned to s is:

$$h(s) = val_{P'}(s) = \prod_{i=1}^{\nu}\{f_i(s_i)\},$$

where f_i is the preference function of the i-th constraint. Since the multiplicative operator of the fuzzy semiring is min and the function s are all semi-convex quadratics, we can rewrite the above as:

$$h(s) = min_{i=1,\cdots,\nu}\{a_i s_i^2 + b_i s_i + c_i\}.$$

We can now substitute what we have obtained in the sum of squares error:

$$SSE = \frac{1}{2}\sum_{s\in Tr}(t(s) - [min_{i=1,\cdots,\nu}\{a_i s_i^2 + b_i s_i + c_i\}])^2.$$

The module performs a $stochastic$ gradient descent (see Section 2.2.3), which means that the error and the update will be computed after considering each example of the training set. We must therefore ignore the summation on all the examples, obtaining:

$$SSE(s) = \frac{1}{2}(t(s) - [min_{i=1,\cdots,\nu}\{a_i s_i^2 + b_i s_i + c_i\}])^2. \tag{2.2}$$

For the sake of notations we will, from now on, indicate $SSE(s)$ simply with E. Our aim is to derive the error function w. r. t. the parameters of the problem, $\{a_1, b_1, c_1, \cdots, a_\nu, b_\nu, c_\nu\}$, and to modify the parameters in a way such that the error decreases, in other words following the opposite direction at which the gradient points. In order to be able to compute $\frac{\partial E}{\partial a_i}$, $\frac{\partial E}{\partial b_i}$, and $\frac{\partial E}{\partial c_i}$, we must replace min with a continuous approximation [Gol97]:

$$min_{i=1,\cdots,\nu}(\alpha_i) \simeq min_\nu^\beta(\alpha_i) = -\frac{1}{\beta}\ln(\frac{1}{\nu}\sum_{i=1}^{\nu} e^{-\beta\alpha_i})$$

where parameter $\beta \geq 0$ determines the goodness of the approximation. Remembering that in our context $\alpha_i = a_i s_i^2 + b_i s_i + c_i$, we obtain the final formulation of the error:

$$E = \frac{1}{2}(t(s) + \frac{1}{\beta}\ln(\frac{1}{\nu}\sum_{i=1}^{\nu} e^{-\beta(a_i s_i^2 + b_i s_i + c_i)})])^2. \tag{2.3}$$

From this expression we can obtain the updated values for the parameters of the parabolas on all the constraints following the Δ $rule$:

$$\tilde{a}_i = a_i + \Delta a_i \text{ with } \Delta a_i = -\eta\frac{\partial E}{\partial a_i} \tag{2.4}$$

$$\tilde{b_i} = a_i + \Delta b_i \text{ with } \Delta b_i = -\eta \frac{\partial E}{\partial b_i} \tag{2.5}$$

$$\tilde{c_i} = c_i + \Delta c_i \text{ with } \Delta c_i = -\eta \frac{\partial E}{\partial c_i}. \tag{2.6}$$

In these formulas, parameter η is the learning rate which regulates the magnitude of the update. It is an important parameter since stochastic gradient descent converges to true gradient descent for $\eta \to 0$ [Mit97]. Thus, η should be as small as possible inducing a slow learning rate.

Notice that, since we are performing a stochastic gradient descent, η should be as small as possible in order to approximate true gradien

In detail $\frac{\partial E}{\partial a_i}$ is:

$$\frac{\partial E}{\partial a_i} = (t(s) - h(s))(-(\frac{\partial h(s)}{\partial a_i})). \tag{2.7}$$

in which only $h(s)$ depends on $\{a_1, b_1, c_1, \cdots, a_\nu, b_\nu, c_\nu\}$ while $t(s)$ is the target preference of s in the training set and it's totally independent from the above parameters. Considering $h(s)$ rewritten using the continuous approximation of min:

$$h(s) = -\frac{1}{\beta} \ln(\frac{1}{\nu} \sum_{j=1}^{\nu} e^{-\beta(a_j s_j^2 + b_j s_j + c_j)})$$

and the expression for $\frac{\partial h(s)}{\partial a_i}$, which can be obtained through a few easy steps,

$$\frac{\partial h(s)}{\partial a_i} = \frac{1}{(\sum_{j=1}^{\nu} e^{-\beta(a_j s_j^2 + b_j s_j + c_j)})} (s_i^2 e^{-\beta(a_i s_i^2 + b_i s_i + c_i)}) \tag{2.8}$$

we obtain:

$$\frac{\partial E}{\partial a_i} = (t(s) - h(s))(\frac{1}{(\sum_{j=1}^{\nu} e^{-\beta(a_j s_j^2 + b_j s_j + c_j)})} (-s_i^2 e^{-\beta(a_i s_i^2 + b_i s_i + c_i)})). \tag{2.9}$$

Similarly we can compute the derivatives of the error w. r. to b_i and c_i:

$$\frac{\partial E}{\partial b_i} = (t(s) - h(s))(\frac{1}{(\sum_{j=1}^{\nu} e^{-\beta(a_j s_j^2 + b_j s_j + c_j)})} (-s_i e^{-\beta(a_i s_i^2 + b_i s_i + c_i)})) \tag{2.10}$$

$$\frac{\partial E}{\partial c_i} = (t(s) - h(s))(\frac{1}{(\sum_{j=1}^{\nu} e^{-\beta(a_j s_j^2 + b_j s_j + c_j)})}(- e^{-\beta(a_i s_i^2 + b_i s_i + c_i)})). \tag{2.11}$$

Finally we can write the complete expression for the update:

$$\tilde{a_i} = a_i - \eta[(t(s) - h(s))(\frac{1}{\sum_{j=1}^{\nu} e^{-\beta(a_j s_j^2 + b_j s_j + c_j)})}(-s_i^2 e^{-\beta(a_i s_i^2 + b_i s_i + c_i)}))] \tag{2.12}$$

$$\tilde{b_i} = b_i - \eta[(t(s) - h(s))(\frac{1}{(\sum_{j=1}^{\nu} e^{-\beta(a_j s_j^2 + b_j s_j + c_j)})}(-s_i e^{-\beta(a_i s_i^2 + b_i s_i + c_i)}))] \tag{2.13}$$

$$\tilde{c_i} = c_i - \eta[(t(s) - h(s))(\frac{1}{(\sum_{j=1}^{\nu} e^{-\beta(a_j s_j^2 + b_j s_j + c_j)})}(- e^{-\beta(a_i s_i^2 + b_i s_i + c_i)}))]. \tag{2.14}$$

For each solution S, following the formulas illustrated above, we compute the error E and the update $\tilde{a_i}$, $\tilde{b_i}$ and $\tilde{c_i}$. In changing the parameters of the preference functions, two conditions must hold at all time: (1) all of the functions must be semi-convex and (2) the image of the function ($f_i(I)$ if I is the interval of the temporal constraint), must be contained in $[0, 1]$, due to the fuzzy semiring. In order to understand how (1) is maintained by the module it is useful to notice how controlling parameter a is sufficient to guarantee this property. In fact a parabola is semi convex iff $a \leq 0$. The algorithm maintains this property for all the parabolas, simply replacing an updated parameter a that is strictly positive with 0.

This method is, in some way, similar to projective methods [PR94], often used in practice. Lets consider in detail how it affects the update of a parameter a_i

$$\tilde{a_i} = a_i - \eta[(t(s) - h(s))(\frac{1}{(\sum_{j=1}^{\nu} e^{-\beta(a_j s_j^2 + b_j s_j + c_j)})}(-s_i^2 e^{-\beta(a_i s_i^2 + b_i s_i + c_i)}))]$$

Suppose this is the first update of a_i to a strictly positive $\tilde{a_i}$. We can then assume that $a_i < 0$. In order for $\tilde{a_i}$ to be positive it must be that:

$$-\eta[(t(s) - h(s))(\frac{1}{(\sum_{j=1}^{\nu} e^{-\beta(a_j s_j^2 + b_j s_j + c_j)})}(-s_i^2 e^{-\beta(a_i s_i^2 + b_i s_i + c_i)}))] \geq 0 \tag{2.15}$$

or equivalently:

$$\eta[(t(s)-h(s))(\frac{1}{(\sum_{j=1}^{\nu} e^{-\beta(a_j s_j^2+b_j s_j+c_j)})}(s_i^2 e^{-\beta(a_i s_i^2+b_i s_i+c_i)}))] \geq 0. \qquad (2.16)$$

This can only happen if all the factors are positive. But this implies that the error $(t(s)-h(s))$ must be positive, which means that the hypothesized preference is smaller than the true one, $t(s) \geq h(s)$. Not only, the gap between the $t(s)$ and $h(s)$ must be big enough to allow the second addend to be in modulo bigger than $|a_i|$. Forcing $\tilde{a}_i = 0$ means behaving as if the error would be smaller, only sufficient to make the second addend equal to a_i.

In order to maintain during all the learning phase compatibility of the functions with the fuzzy framework we introduce the notion of *truncated*, or *fuzzy*, parabola. To any semi-convex parabola the corresponding fuzzy parabola is the semi-convex function that coincides with the original functions on all the elements that are mapped into values between 0 and 1, and that maps all the elements originally mapped to strictly negative values to 0 and all the elements mapped to values greater than 1 to 1. An example of parabola and it's corresponding fuzzy parabola are depicted in figure 2.21.

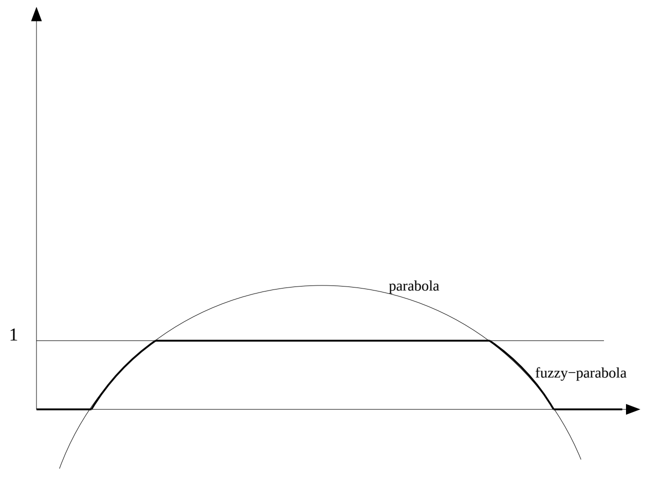


Figure 2.21: Parabola and corresponding fuzzy-parabola (bold).

Fuzzy parabolas fit well into the fuzzy schema but, on the other side, are not differentiable. For the implementation of the algorithm, two different possibilities have been considered: (a) use fuzzy parabolas for the update keeping the derivatives of the continuous parabolas; (b) use the parabolas and let them free to evolve during the learning phase even out of the [0,1] interval, and consider fuzzy parabolas only after the completion of the learning phase. Hypothesis (a) has been implemented but a detailed study of the consequences of the distortion of the gradient descent it creates has convinced us to drop it. The final module is implemented according to hypothesis (b).

We compute the following errors:

- the error used by the learning module: sum of squares error with parabolas, E, and with fuzzy parabolas, E^f.

- The absolute maximum error and absolute mean error with both parabolas, E_{max} and E_{med}, and fuzzy parabolas, E^f_{max} ed E^f_{med}. The maximum error tells us how much the hypothesized preference value of a solution is far from the target in the worse case. It can thus be defined in general as $max_s|t(s) - h(s)|$, where in E_{max} $h(s)$ is computed using parabolas, while in E^f_{max} it is computed using fuzzy parabolas. In the case of parabolas this allows us to see immediately if functions outside the $[0, 1]$ interval are learned, while in the case of fuzzy parabolas it gives a strict upper bound to the worst performance on any solution. The absolute mean error, instead, gives an overall measure of the quality of the learned functions. It gives the average over all solutions of the distance of the hypothesized preference and the true one. In general it can be defined as $\frac{\sum_{s \in T} |t(s)-h(s)|}{|T|}$, where $|T|$ is the number of solutions considered (e.g. those in the training set or those in the test set). We have chosen the improvement in terms of the absolute mean error as the criterion to stop the learning phase.

Once the learning phase has ended the STPP returned by the algorithm is with fuzzy parabolas. It is safe to do so since the errors computed with parabolas dominate those with fuzzy parabolas as stated in the following theorem.

Theorem 2.14 *Consider an STPP P with preference functions described as convex parabolas with positive values, and the corresponding STPP F with the same parabolas as P where all values > 1 have been lowered to 1 (i.e. with fuzzy parabolas). Then, given any solution of P (and F), s, and corresponding target, $t(s)$, with $0 \leq t(s) \leq 1$, the sum of squares error of s in P is greater or equal to that in F: $SSE^P(s) \geq SSE^F(s)$.*

Proof: Consider any constraint c in P and let $p(x) = ax^2 + bx + c$ be its preference function and $fp(x)$ the corresponding fuzzy parabola in F. Consider solution s and its projection on c, s_c. Now $p(s_c) = fp(s_c)$ if $p(s_c) \leq 1$, otherwise $p(s_c) > 1$ and $fp(s_c) = 1$ (by definition of fuzzy parabola). By definition we have that:

$$SSE^P(s) = \frac{1}{2}(t(s) - [min_{i=1,\cdots,\nu}\{p_i(s_i)\}])^2 \tag{2.17}$$

while:

$$SSE^F(s) = \frac{1}{2}(t(s) - [min_{i=1,\cdots,\nu}\{fp_i(s_i)\}])^2. \tag{2.18}$$

The only case in which $val_P(s) = min_{i=1,\cdots,\nu}\{p_i(s_i)\}$ and $val_F(s) = min_{i=1,\cdots,\nu}$ $\{fp_i(s_i)\}$ differ is if $p_i(s_i) > 1$, $\forall i$. In such case we will have $val_P(s) > 1$ and $val_F(s) = 1$. This means that $|t(s) - val_P(s)| > |t(s) - val_F(s)|$, since $0 \leq t(s) \leq 1$, which implies that $(t(s) - val_P(s))^2 > (t(s) - val_F(s))^2$. This leads us to $SSE^P(s) \geq SSE^F(s)$.

From the above theorem we can derive similar relations for the maximum absolute error and the mean absolute error, as stated in the following corollary.

Corollary 2.2 *Consider an STPP P with preference functions described as convex parabolas with positive values, and the corresponding STPP F with the same parabolas as P where all values > 1 have been lowered to 1 (i.e. with fuzzy parabolas). Then,*

1. *given a set of solutions with their corresponding target $T = \{(s, t(s))|$ s solution of $P, 0 \leq t(s) \leq 1\}$, then the maximum error on T computed in P, $E^P_{max} = max_{s \in T}|t(s) - val_P(s)|$ is greater or equal to that computed in F: $E^P_{max} \geq E^F_{max}$;*

2. *given a set of solutions with their corresponding target* $T = \{(s, t(s)) | \, s$ *solution of* $P, 0 \leq t(s) \leq 1\}$, *then the mean error on T computed in P,* $E^P_{med} = \frac{\sum_{s \in T} |t(s) - val_P(s)|}{|T|}$ *is greater or equal to that computed in F:* $E^P_{med} \geq E^F_{med}$*;*

Proof:

1. By what is stated above, we have that for any $s \in T$ either $|t(s) - val_P(s)| = |t(s) - val_F(s)|$ or $|t(s) - val_P(s)| > |t(s) - val_F(s)$. This means that $E^P_{max} = max_{s \in T}|t(s) - val_P(s)| \geq E^F_{max} = max_{s \in T}|t(s) - val_F(s)|$.

2. Again, since for any $s \in T$ either $|t(s) - val_P(s)| = |t(s) - val_F(s)|$ or $|t(s) - val_P(s)| > |t(s) - val_F(s)|$, when we sum the distances on all the solutions it must be that $\sum_{s \in T} |t(s) - val_P(s)| \geq \sum_{s \in T} |t(s) - val_F(s)|$.

Notice that, while in the derivatives the continuous approximation of min is used, everywhere else when $h(s)$ must be computed the actual discontinuous min function is used. The reason for this is that it's better to use approximations only when it is necessary.

The pseudocode of the learning module is shown in Figure 2.22. In line 1 the algorithm takes as input STPP P. P contains all the intervals, in some sense the "hard" constraints of the problems. Its preference functions are initialized according to some criterion that decides the a, b and c values for each of them. For example, the preference functions might be set to the constant function $f(x) = 1$ by setting $a = 0$, $b = 0$, and $c = 1$. Once the initialization is completed, the algorithm must start scanning the solutions and corresponding global preferences contained in the training set (lines 2,3). For each solution the algorithm updates the parameters of all the preference functions following the Δ $rule$, that is, the procedure described earlier in this paragraph, lines(4-6). Notice that function Update(P, E, η, β) updates the parameters on all the parabolas given the current problem P, the sum of squares error E, the learning rate η and the min approximation parameter β according to Equations 2.12, 2.13, and 2.14.

Once all the examples in the training set have been examined the errors E, E_{max}, E_{med}, E^f, E^f_{max}, and E^f_{med} are computed. The percentage of improvements of E^f_{med} w. r. t. its last value, $PLI^f_{E_{med}}$, is computed as well (line 7). This value is used as the stopping criterion for the learning module. In detail, if the values of E^f_{med} at iterations j, $j+1$, and $j+2$ are respectively, e_j, e_{j+1} and e_{j+2} then $PLI^f_{E_{med}}$ at iteration $j+2$ is $(e_{j+2} - e_{j+1})$, the value of threshold $Thres$ that appears in lines 8 and 9, is $\alpha \times (e_{j+1} - e_j)$, where α is a parameter chosen in $[0.5, 1[$. If for a certain number of iterations, $maxit$, $PLI^f_{E_{med}}$ fails to exceed threshold $Thres$, the algorithm stops (line 10).

At this point the parabolas are reduced to fuzzy parabolas and the testing phase starts (line 11). A set of examples, i.e. pairs (solution, target), none of which appears in the training set, namely the test set, is evaluated using the STPP produced by the module. By comparing the preferences assigned to the solutions by the STPP with their targets, errors E^{ts}, E^{ts}_{max}, and E^{ts}_{med} are computed. They are, obviously, the sum of squares error, the maximum absolute error and the absolute mean error on the test set. In line 12 the output is given. More precisely, the output will consist of the learned STPP and some data collected during the learning and the testing phase. Although the data to be collected can be specified at experimental setup time, usually it will contain all the errors computed at each iteration and the errors computed in the testing phase.

2.6.3 Experimental results

The learning module has been tested on some randomly generated problems: every test involves the generation of an STPP via our generator, and then the generation of some examples of solutions and their rating.

To generate problems on which to test the learning module we have used the random generator we have described in Section 2.5.1. Among the input parameters of the generator we have chosen to maintain fixed the domain of the first solution. In fact, the learning module is not sensitive to the position in the timeline of the timepoints around which the problem is generated. This is true even if a traslation of the STPP does change the parameters of the parabolas on the

```
Algorithm STPP_Learning_Module
1. initialize STPP $P$ ;
2. do
3.   for each $s \in Tr$;
4.     compute $E$;
5.     Update$(P, E, \eta, \beta)$;
6.   end of for;
7.   compute $E$, $E_{max}$, $E_{med}$, $E^f$, $E^f_{max}$, $E^f_{med}$, $PLI^f_{E_{med}}$ on training set;
8.   if $(PLI^f_{E_{med}} <$ Thres) stopcount++;
9.   if $(PLI^f_{E_{med}} \geq$ Thres) stopcount $\leftarrow 0$;
10. while(stopcount<maxit);
11. compute $E^{ts}_c$, $E^{ts}_{max}$, $E^{ts}_{med}$ on test set;
12. output;
```

Figure 2.22: Pseudocode of STPP_Learning_Module.

constraints. The learning is however not sensitive to the magnitude of such parameters. This is true for learning systems in general, since the regularity and shape of functions have a much bigger impact on the difficulty of learning than the size of the parameters, assuming an appropriated learning rate η. In all the experiments we have kept such a domain at $r = 40$, so all the problems have a solution within the first 40 units of time.

We have also maintained fixed the distortion parameters $pa = 10$, $pb = 10$ and $pc = 5$. The parameters have been chosen to allow for a wide range of different preference values.

We have instead decided to vary in the experimental phase the number of variables n. In particular, Table 2.5 shows results for problems with $n = 25$ variables, Table 2.6 for problems with $n = 20$ and Table 2.7 for problems with $n = 15$.

We have also considered three different sizes of intervals: $max = 20$, $max = 30$ and $max = 40$. According to the maximum size of the intervals we have also changed the size of the training and the test set. In particular we have given 500 examples to the training set and 500 to the test set for problems with $max = 15$; while 600 examples have been used in each set for problems with $max = 20$ and 700 examples for problems with $max = 25$. Notice that in all cases the size of the training and test set was less than 1% of the total number of solutions.

Another parameter which we have varied in the tests is the density of non default constraints. In particular we have considered densities $D = 40\%$, $D = 60\%$ and $D = 80\%$. These are fairly high values, but we have chosen to consider highly constrained networks since they have more non trivial preference functions and are, hence, more interesting from a learning point of view.

We have fixed the parameters of the learning module in the following way: $\beta = 8$ and $\eta = 10^{-7}$. The learning process starts with all the preference functions set to the constant $y = 1$. The stop criterion has been set to a 100 consecutive steps with an improvement of the absolute average error on the training set, E_{med}, smaller than the 70% of the previous one.

In Tables 2.5, 2.6 and 2.7 we show results on E^{ts}_{med}, that is the absolute mean error on the test set. The first value is the mean on 30 examples, while the values between brackets are respectively the minimum and the maximum absolute mean error obtained.

	D=40	D=60	D=80
max=20	0.017 (0.013,0.022)	0.007 (0.006,0.008)	0.0077 (0.0075,0.0081)
max=30	0.022 (0.017,0.025)	0.013 (0.01,0.017)	0.015 (0.005,0.028)
max=40	0.016 (0.011,0.019)	0.012 (0.012,0.013)	0.0071 (0.006,0.0079)

Table 2.5: E^{ts}_{med} on STPPs with $n = 25$, $r = 40$, $pa = 10$, $pb = 10$, $e\ pc = 5$.

From the experimental results we can draw the following conclusions:

- the error is reasonable since it ranges from 0.004 to 0.05 and the preference values are in $[0, 1]$;

	D=40	D=60	D=80
max=20	0.032 (0.022,0.043)	0.012 (0.01,0.016)	0.005 (0.004,0.006)
max=30	0.032 (0.021,0.040)	0.018 (0.016,0.021)	0.0074 (0.0073,0.0077)
max=40	0.033 (0.05,0.021)	0.023 (0.025,0.022)	0.016 (0.011,0.021)

Table 2.6: E^{ts}_{med} on STPPs with $n = 20$, $r = 40$, $pa = 10$, $pb = 10$, $e\ pc = 5$.

	D=40	D=60	D=80
max=20	0.018 (0.01,0.028)	0.009 (0.007,0.012)	0.009 (0.007,0.012)
max=30	0.021 (0.018,0.027)	0.014 (0.011,0.017)	0.016 (0.010,0.021)
max=40	0.024 (0.023,0.026)	0.019 (0.019,0.021)	0.0086 (0.007,0.01)

Table 2.7: E^{ts}_{med} on STPPs with $n = 15$, $r = 40$, $pa = 10$, $pb = 10$, $e\ pc = 5$.

- the error seems not to be affected by the number of variables; this means that there can be larger problems that are less difficult to learn;

- there is a weak connection between the density and the error; a reason for this could be that a greater density ensures a wider variety of preference values, which in turn translates into a training set with many different preference values, helping the module in the inference process.

We conclude giving some information on the number of iterations and the time used by the algorithm. All the tests have been performed on a machine with a Pentium III 1GHz processor and 512 Mb of RAM. The minimum number of iterations has been 357 while the maximum number has been 3812. The shortest time used has been of 2 minutes and 31 seconds while the longest 8 minutes and 18 seconds. Note that these results were obtained in 4 different problems since the time needed for a single iteration is not constant. We can conclude that the learning module uses a reasonable amount of time.

2.7 Related Work

In this section we consider several papers which are related with what we have presented in this chapter. We divide them in two groups: the first containing papers published before the thesis which represent the state of the art in the field, and the second group containing papers published during the thesis research period.

2.7.1 State of the art research

The lack of flexibility of hard temporal constraints has been addressed before. In the constraint satisfaction literature the maxi-min framework has been first suggested in [RHZ76] and fuzzy sets have been used to model local and global requirements in the form of flexible constraints in [Zad75].

In [VG94] **Fuzzy Temporal Constraint Networks** are defined. In such networks fuzzy intervals are used to model a *possibility* distribution (see [Zad75]) over intervals. Although our work and the work in [VG94] have two completely different goals, since ours is to model temporal flexibility in terms of preferences, while in [VG94] they consider vagueness of temporal relations there are many points in common in the approach. First, they introduce the notion of *fuzzy Temporal Constraint*, i.e. simple temporal constraint equipped with a possibility distribution mapping every element of the interval into a possibility value. There aim is to model imprecise pieces of temporal knowledge expressed in statements such as "I started to feel this pain in my knee between three and four weeks ago, more o less". Despite the different semantics the framework is close to ours since they give results for fuzzy Simple Temporal Problems with unimodal possibility distribution, where unimodal is a synonym of the semi-convexity, defined in Section 2.3.3. Thus they combine the possibilities with min and compare them with max, as we do with preferences. Given such problem they are interested in answering three fundamental questions:(1) is the problem consistent? (2) Is a given value for a variable *feasible*? i.e. does it belong to a solution with a certain possibility degree? and (3) Is a new constraint compatible with the ones in the network, and is it entailed by the constraints in the network? We too, have considered questions (1) and (2) in the context of preferences, however we are also interested in optimal solutions. To answer the first question they propose a generalization of the classical path consistency algorithm (PC-1)[Mon74], while we extend the more efficient version for TCSPs (PC-2, see Section 2.2.1) proposed in [DMP91]. They compute the minimal network, and of course, detect inconsistency if an interval becomes empty. However they do not prove that after path consistency the preference are all lowered to the same maximum and that the STP obtained considering only intervals mapped into that maximum is also minimal. This is what allows us to find an optimal solution in polynomial time. Besides, we have shown in Section 2.5.4 that, despite its generality, the path consistency approach is substantially slower in practice than the chopping procedure proposed in Section 2.4.2. Moreover, although addressing the feasibility issue for a value of a variable, only the formal definition is given and no explicit method is supplied to check such a property. Instead, within our framework, to see if a value is contained in a solution with a certain preference level, once either one of the solvers have been applied, it is sufficient to check if such a value belongs to the minimal network obtained by chopping that problem at the preference level considered.

Another related work on **Flexible Temporal Constraints** is presented in [BG00]. Here fuzzy temporal constraints are used to model preferences among feasible solutions while prioritize constraints are used to express the degree of necessity of their satisfaction. This work is , thus, related to what we have presented in this chapter but also to what is described in Chapter 3. Here we will focus on how preferences are represented and dealt with. The main difference in this respect is that they add preference to *qualitative*, "a la Allen", constraints (see paragraph *qualitative interval constraints* in Section 2.2.1). For example consider intervals I_1 and I_2, and the two relations { *before,meets* }. They associate a preference, i.e. a value in [0,1], to the relation I_1 `before` I_2, for example 0.8, and a value to I_1 `meets` I_2, for example 0.2, indicating that is preferred that the activity represented by interval I_1 is finished before the one represented by I_2 starts. We would model this simply adding a soft temporal constraint between the finishing time of the first activity and the start time of the second. An obvious difference is that they have disjunctions and are, thus, outside the tractable class we consider. In order to find optimal solutions they refer to an algorithm proposed by M. Cooper in [Coo94] which features some resemblance to Chop-solver. In fact it decomposes the problem into many subproblems each obtained considering a specific preference level. However it performs the search in a top-down fashion. This implies that in contrast to Chop-solver which follows a bottom-up binary search, is not an any-time algorithm. On the other hand it has the advantage that the first solution found is guaranteed to be optimal. We will consider again this paper in relation to temporal uncertainty in Chapter 3 Section 3.9.

In [DFP95] flexible constraints have been applied to the **Job-shop Scheduling** problem. In [DFP95] and in [DFP03] fuzzy sets are used to model preferences and activities of ill-known duration. In this sense this work is related to the results we have presented in this chapter in Sections 2.3 and 2.4 and to those we will present in Chapter 3. Here we will focus on the relation between the way they handle preferences and the framework we have presented.

The main difference is in the problems which are addressed. In [DFP95] and in [DFP03] a particular type of scheduling problem is considered: namely job-shop scheduling. In job-shop scheduling there are a set of jobs that must be accomplished in order to complete a project. Each job, i, is represented by a starting time s_i and a duration t_i. There are constraints involved in the description of the problem. In particular there are *precedence constraint* of the form $s_k \geq s_i + t_i$, meaning that job i must precede job k. Moreover, each machine can perform only one job at the time. This is called a *capacity constraint* and can be written in the following way: $s_k - s_i \geq t_1$ or $s_i - s_k \geq t_k$, meaning that job i and job k may not overlap (assuming they are performed by the same machine). Each job may require raw material and is thus subject to a *release time constraint* which specifies that its starting time must follow the delivery of the material ($s_i \geq r_i$, where r_i is the release time of job i). Similarly, there are *due dates constraints* specified in order to meet the delivery requirements of the customer ($s_i + t_i \geq d_i$, where d_i is the due date of job i). As shown in [DFP95], the job-shop scheduling problem can be easily represented by a TCSP whose variables are starting times of tasks and by adding the usual variable s_0 representing the "beginning of the world". The release and due date constraints can be represented as simple temporal constraints between each starting point and s_0, i.e. $s_i - s_0 \in [r_i, d_i - t_i]$. Precedence constraints can be represented as simple temporal constraints with intervals containing only positive values, $s_k - s_i \in [t_i, +\infty[$. Capacity constraints, however, cannot be represented as simple temporal constraints since they are disjunctive: $s_k - s_i \in [t_i, \infty[\vee]-\infty, -t_k]$. We can, thus, see how there is a fundamental difference in the type of problems that are addressed in in [DFP95] and those considered in this chapter. They consider a problem in which there are fixed types of constraints (precedence, capacity) while we allow any kind of constraint on durations and interleaving times which can be represented in a simple temporal problem. In this sense we are more general since we are considering generic scheduling problems. On the other hand they deal with disjunctive constraints which we do not allow in our framework. In addition to these fundamental differences on the structure of the problem, there are also differences on the preference representation. Although the context is the same (fuzzy preferences) and so is the goal (maximizing the minimal preference of solutions), in [DFP95] only a specific preferences are considered. In particular they model preferences on release and due dates constraints: the preference of the customer for the item to be delivered within the due date or as soon as possible after that, and the preference of the manufacturer to schedule the production possibly after the latest delivery date of the raw materials. In [DFP95] it is shown how these two types of preferences can be effectively represented by fuzzy intervals with trapezoidal functions. Specific preferences for durations, as "the shorter the better" or "the longer the better", are also taken into account. These preferences are clearly linear. The preference functions we allow, semi-convex, are more general. They include both trapezoids and linear functions but also different shapes as step functions or convex piece-wise linear functions. For example the last two types of functions are useful in practice since the users often describe their preference by grouping elements towards which they have a similar attitude. One of the problems considered and solved in this chapter, that is finding one or all optimal solution to an STPP, is similar to what is called ”consistency enforcing” in [DFP95]. In detail, consistency enforcing is the part of the solver for fuzzy job-shop scheduling that finds the optimal solution of the problem once the disjuncts of the capacity constraints have been resolved. However, the method they propose in tailored on job-shop scheduling problems and is specifically designed to cope only with precedence constraints and with preference functions expressed using fuzzy intervals.

Preferences in Job-Shop Scheduling have been considered in [SF89]. In their framework, activities in a plan are interconnected via temporal relation constraints a la Allen and are described via a temporal constraint graph (TGC). In this, although using a constraint based approach, they differ from the approach presented in this chapter since they represent temporal relations in a qualitative way instead that in a quantitative way. Additionally, in [SF89] there are capacity constraints restricting the use of each resource to only one activity at the time. The whole problem can thus be represented using a Temporal/Capacity Constraint Graph. The preferences considered are such as meeting due date, reducing order flow time, or selecting accurate machines and are represented as *utility functions* over the domain of possible start times and durations of activities and over the set of possible resources constraints (*preferential constraints*). As in our framework, there are also constraints that express absolute requirements (*requirement constraints* in [SF89] which correspond to our hard constraints). It should be noticed that we do not model directly preference which are not temporal (i.e. preferential capacity constraint). It is however possible to model such preferences indirectly as temporal preferences on the start times and durations of activities (as shown in [DFP95], it requires disjunctive temporal constraints). The utility of a complete schedule is the sum of the utilities obtained by the schedule on each preferential constraint. They are thus combining the preference with a different aggregation operator, i.e. *sum* instead of *min* which we use. *Sum* is the usual aggregation operator for utility functions, however it is not idempotent and thus it does not allow to use constraint propagation and it does not meet the tractability assumptions defined in Section 2.4. Another important difference between our approach and the one in [SF89] is that preferences are propagated within a probabilistic framework. This means that, in contrast to what we propose, probabilities derived from preferences, and not preferences, are propagated through the constraints. Also, the target of the preference propagation is different. We propose to find an optimal schedule statically while in [SF89] preference propagation is used to guide an incremental scheduler. An incremental scheduler works by iterating a two-phase process. In the first phase it analyzes the structure of the CSP resulting from the initial scheduling problem and the decisions that have been already made. In the second phase, new decisions are generated resulting in the expansion of the current schedule. If the scheduler reaches a deadend then it backtracks. In brief, their approach first produces an *a priori* probability distributions for the start times, durations and resource of each unscheduled activity based on the current partial scheduled as well as start time, duration and resource preferences. Then, these probabilities are propagated over the TGC and result in *a posteriori* probability distributions on the same variables. These probabilities are combined to compute activity individual demand densities, which represent for each activity, given a time t and a resource R, the probability that if the activity uses R at time t it will fulfill the resource requirements on R. Finally, individual demand densities are combined into aggregate demand densities. From a constraint point of view, a posteriori start time and duration distributions should be regarded locally as measures of value goodness, and globally as measures of variable looseness with respect to start times and durations. Summarizing, in [SF89] as in [DFP95] and [DFP03], a specific scheduling problem is considered. The representation of this problem is more general than the one we tackle since it considers explicitly resource constraints, but on the other hand it considers specific temporal constraints, [DFP95] and specific preference functions. Also the way preferences are dealt with is different. In [DFP95] a fuzzy interval based approach is used, while in [SF89] a probabilistic approach applied to utility functions is adopted. We use the fuzzy (max, min) operators applied to generic semi-convex functions which map elements in $[0, 1]$. The algorithms we propose for finding an optimal schedule are polynomial and do not require search. However they cannot be applied directly to the problems addressed in [DFP95] and [SF89] since either the problem features disjunctive temporal constraints or additional capacity constraints as in [DFP95, SF89], or a non idempotent aggregation operator, as in [SF89].

2.7.2 Current related research

A work that is related to what is presented in Section 2.4.3 is [MMK$^+$04]. As mentioned in Section 2.4.3, the WLO+ solver returns only a sub-set of Pareto undominated solutions. In [MMK$^+$04] the authors give a semantics, that is an ordering on the solutions, that characterizes exactly the set of optimal returned by the WLO+ algorithm. In [MMK$^+$04] the ordering is called **Stratigraphic Egalitarianism** (SE) and is defined in terms of the vector of preference $u_S = u_S^1, \ldots, u_S^m$ associated to solution S, where u_S^k is the preference associated to the projection of solution S on the kth constraint and m is he total number of constraints. Solution S' SE-dominates solution S at preference level α iff:

- $u_S^i < \alpha$ implies $u_{S'}^i \geq u_S^i$;
- $\exists i$ such that $u_S^i < \alpha$ implies $u_{S'}^i > u_S^i$;
- $u_S^i \geq \alpha$ implies $u_{S'}^i \geq \alpha$;

A solution S' SE-dominates S ($S' >_{SE} S$) iff it dominates S at any preference level. An interpretation of the SE semantics can be given in terms of an economic metaphor. The preference level α can be seen as a sort of "poverty line"and a solution S as policy. A policy S' has better overall quality than S if some members below the poverty line in S are improved in S', even if some members above the poverty line in S are made worse in S'. We thus conclude that algorithm WLO+, presented in Section 2.4.3, is a polynomial time algorithm for finding optimal solutions according to the SE-semantics, which is a refinement of the fuzzy semantics in the sense that it overcomes the unpleasant drowning effect.

In the literature of fuzzy reasoning there has been the awareness of the negative effect that characterizes maxi-min optimization. As mentioned in Section 2.4.3, the drowning effect consists of the fact that all solutions having the same minimum preference have the same global preference regardless on how they perform on other constraints. In [DF99] this problem is addressed by defining orderings that refine the one induced by the fuzzy semantics. In particular, three possible ordering on the solutions are considered. As SE, they are defined on the vector of preferences associated with a solution.

- **Discrimin ordering (D).** Consider the set $D(S', S) = \{c_i | u^i_{S'} \neq u^i_S\}$, that is the set of constraints on which the two solutions obtain different preferences. Then $S' >_D S$ iff $min_{c_i \in D(S',S)} u^i_{S'} > min_{c_i \in D(S',S)} u^i_S$. In other words, solution S' is better than solution S if its lowest preference among the constraints not equally satisfied is strictly greater than that of S.

- **No reason for regret (NR).** In [DF99] this semantics originally defined in [Beh77] is considered and compared against discrimin. Solution S' is NR-better than solution S ($S' >_{NR} S$) iff $\exists j$ such that $u^j_{S'} > u^j_S$ and $\forall i$ $(u^j_{S'} > u^i_S)$ $\vee$ $(u^i_{S'} \geq u^i_{S)}$. In [DF99] it is shown that NR is equivalent to D.

- **Leximin (L).** The leximin ordering of two solutions, S' and S, with associated preference vectors $u_{S'}$ and u_S, is defined in terms of the reordered vectors $u^*_{S'}$ and u^*_S in which the elements of $u_{S'}$ and u_S appear in increasing order. Solution S' is L-better than S iff: $\exists i \leq k$ such that $\forall j < i$, $u^{*j}_{S'} = u^{*j}_S$ and $u^{*i}_{S'} > u^{*i}_S$.

We can summarize the results in [DF99] and in [MMK+04] as:

$$S' >_{SE} S \Rightarrow S' >_{L} S \Rightarrow S' >_{D} S \Leftrightarrow S' >_{NR} S \Rightarrow S' >_{P} S \Rightarrow S' >_{F} S$$

where $S' >_P S$ is the Pareto ordering and $S' >_F S$ is the fuzzy ordering. This means that any optimal solution for SE is also optimal for L, D, NR, Pareto and fuzzy. In [DF99], algorithms are proposed to find both D-optimal solutions and L-optimal solutions. They are similar to that presented in Section 2.4.3 in the sense they identify the set of weakest links, which they call minimal subsets of saturated constraints, and then turn them into a hard (classical) constraint. The algorithms find solutions which may not be optimal according to SE. Moreover, as noted in [DF99], it is always possible to derive a D-optimal solution from a fuzzy-optimal solution. However the algorithm they propose for finding L-optimal solutions does not guarantee that from a fuzzy-optimal solution an L-optimal can be reached. Since in [MMK+04] it has been shown that any optimal solution for SE is optimal also for leximin we can conclude that the procedure we presented in Section 2.4.3 gives a way to find an L-optimal solution starting from any fuzzy-optimal solution.

The work we have described in this chapter and published in [RSV+02] and in [KMMV03] has obtained a considerable interest in the Constraint Programming community which has resulted in new works related to ours. For example, in [MMK+04] the authors consider preference in the **utilitarian** perspective, where local preferences are aggregated using sum and compared using max. They show that the problem of finding the set of all utilitarian-optimal solutions of an STP with piecewise linear convex functions is tractable. Basically they extend what we have mentioned in Section 2.3.2 in order to deal with piecewise linear convex functions and in order to find all utilitarian optimal solutions (instead of just one). The constraint problem is solved by transforming it into a linear program which has as variables the original temporal variables of the STPP plus the projections of a solution on the constraints and auxiliary variables, each corresponding to a linear piece of the function on a constraint. Moreover, in [MMK+04] they compare the performance of an optimized version of the implementation of the WLO+ solver described in Section 2.4.3 against the a simplex LP solver. It should be noticed that while WLO+ produces SE-optimal solutions, simplex produces utilitarian-optimal solutions. In [MMK$^+$04] they have compared both the run time for generating an optimal solution and the quality of the SE-optimal solution from the utilitarian point of view. WLO+ solver was shown to be faster than LP solver, on average, and the improvement increases with the size of the problem. However WLO+ seems to be worse than simplex on sparse problems. As far as quality, the SE-optimal solutions generated by WLO+ were within 90% of the utilitarian optimal value. The authors in [MMK$^+$04] thus suggest that WLO+ offers a feasible alternative to LP-based techniques since it allows an increase in speed at a modest solution quality cost.

In [PP04] our approach is extended to more general temporal constraint problems: DTPs (Disjunctive Temporal constraint Problems [SK00]). A DTP is defined on a set of temporal variables, that, just as in TCSPs, represent the start time or end time of an activity. In fact DTPs generalizes TCSPs by lifting the binary requirement and by allowing constraints of the following form:

$$(X_{j1} - X_{i1} \in [a_{i1j1}, b_{i1j1}] \vee X_{j2} - X_{i2} \in [a_{i2j2}, b_{i2j2}] \vee \ldots$$

$$\vee X_{jn} - X_{in} \in [a_{injn}, b_{injn}]).$$

As in our framework, the approach used for preferences is the max-min approach. In fact, in [PP04] they describe an algorithm that checks the consistency and finds optimal solutions of **DTPs with preferences** borrowing concepts from previous algorithms for solving TCSPs [SK00, OC00, TP03] and ours for solving STPPs. The algorithm proposed in [PP04] can be divided in two phases. In the first phase the preference values of the DTPP are discretized and a set of DTPs is projected from the DTPP, i.e. one DTP for each preference value. In the second phase, an attempt is made to solve each DTP searching for a consistent component STP. The goal is to find a consistent DTP such that its solutions have the highest preference value. The algorithm searches for a consistent STP component moving from the lowest preference up. The interesting thing is that, as the algorithm moves up in the preference level, it is able to detect which STPs at that level are "related" to the ones already searched at lower levels and can thus skipped them. This interesting feature of the algorithm allows the authors in [PP04] to conclude that the additional cost of adding preferences to DTPs is $|A| \times |X|^3$, where A is the discretized set of preferences and X is the set of variables.

2.8 Future Work

In this section we outline some possible directions we would like to pursue in order to extend the work we have presented in this chapter. In particular we consider applying learning techniques to the operators of the semirings and extending the Conditional Temporal Problems framework [TVP03] by adding preferences.

Learning the multiplicative operator of a semiring

The learning process that we have described in Chapter 2 Section 2.6 is meant to overcome the difficulty of the user to define precisely all the preference functions on all the constraints. In other words it induces local information from global information which may be easier to retrieve from the user.

Another possible application for machine learning techniques arises when considering the structure of the semiring and in particular the multiplicative operator which allows to combine preferences. In fact, two scenarios may occur: either it is left to the user to give sufficient and explicit information in order to derive "how" preferences should be combined, or the system must induce this from global information. Let us make an example that shows what knowing what the multiplicative operator is means. Suppose we have a problem with three variables, x_1, x_2 and x_3 and two soft constraints $c_1(x_1, x_2)$ and $c_2(x_2, x_3)$. Assume know that the preference assigned by c_1 to the partial assignment (v_1, v_2)

is p and that the preference assigned by c_2 to the partial assignment (v_2, v_3) is q. We assume that this information is available, in other words that the user is able to give p and q just looking at the partial assignments. The preference of the complete assignment (a solution, assuming consistency) will be $p \times q$ in the semiring soft constraint framework. What is very likely to happen is that, given p and q, the user will not be able to give $p \times q$ but it might be able to give his or her preference to the solution just looking at it. We think that a machine learning module that learns the multiplicative operator from some data given by the user could be very useful. Once learned the operator can be used directly, for example to compute the global preference of solutions given only partial ones as the pre-processing step for answering dominance queries. On the other hand, learning the operator can also be seen as the first step of the optimization solving process we have described in this chapter.

We plan to consider various machine learning approaches. Figure 2.23 shows a possible method.

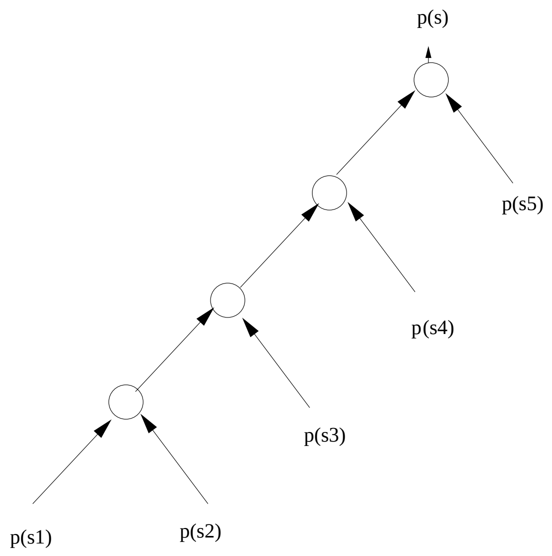


Figure 2.23: A possible neural network structure for learning the multiplicative operator.

Preference $p(s_i)$ is the preference assigned by the i-th constraint to the projection of solution s on the variables it constrains. Preference $p(s)$ is the global preference of the solution. In this case we assume only global preferences (on complete assignments) are given. The network is composed of a sequence of neurons, each of which represents the application of the multiplicative operator to two inputs. In fact the multiplicative operator is binary. Another property that must be satisfied is the symmetry of the operator. This is achieved imposing constraints on the weights of the neural network. One technique we plan to use is gradient descent since it is one of the most widely used. We hope to gain some interesting information, for example on the behavior of the system caused by the fact that different inputs might have the same output.

There are other possibilities, for example to use SVM. However the application of such a framework must take into account the recursion that this structure features. SVM for recursive cases have been proposed but they have not been very successful due to the heavy work load they cause.

One last possibility is to identify some appropriate structure in the data and to use techniques designed to handle learning for structured data.

Finally, we would like to extend this process of deriving the behavior of the multiplicative operator also to other semirings applied in other preference scenarios.

Conditional Temporal Problems with Preferences

Recently a framework for handling Conditional Temporal Problems has been proposed in [TVP03]. To each temporal variable a boolean formula is attached. Such formula represents the condition which must be satisfied in order for the execution of the event to be enabled. In such framework the conditional aspect is on the execution of the temporal variables. We, however, believe that it would be interesting to extend this approach in order to allow conditional preferences. By conditional preferences we mean allowing preference function on constraints to have different shapes according to the truth values of some formulas or the occurrence of some event at some time. We believe achieving this would constitute an additional gain in expressiveness. In fact, it would allow to express the dynamic aspect of preferences that change over time.

Chapter 3

Uncertainty and Preferences in Temporal Problems

Current Research on temporal reasoning, once exposed to the difficulties of real-life problems, can be found lacking both expressiveness and flexibility. In Chapter 2 we have addressed the lack of expressiveness of hard temporal constraints, adding preferences to the temporal framework. In this chapter we introduce a framework which allows us to handle both preferences and uncertainty in temporal problems. In particular, the type of uncertainty which is taken into consideration is the non-determinism on the time of occurrence of some events, which are said to be uncontrollable. The framework, which we describe here, is obtained by merging the two pre-existing models of Simple Temporal Problems with Preferences (STPPs), described in Chapter 2 and Simple Temporal Problems with Uncertainty (STPUs) introduced in [VF99]. We adopt the notion of controllability of STPUs, to be used instead of consistency because of the presence of the non-determinism of some variables, and we adapt it to handle preferences. The proposed model, Simple Temporal Problems with Preferences and Uncertainty (STPPUs), represents temporal problems with preferences and uncertainty via a set of variables, representing the starting or ending times of events (which can controllable or not), and a set of soft temporal constraints over such variables, each of which includes an interval containing the allowed durations of the event or the allowed interleaving times between events and a preference function associating each element of the interval with a value corresponding to how much its preferred. Such soft constraints can be defined on both controllable uncontrollable events.

The main results presented in this chapter are: the definition of STPPUs and the extension of the notions of controllability to notions which take into account also preferences. For each notion, that is, optimal strong controllability, optimal weak controllability and optimal dynamic controllability we give algorithms that check them and we show that adding preferences does not make the complexity of testing such properties worse than in the case without preferences. Moreover, we also consider the execution of dynamic optimal plans.

3.1 Motivation and Chapter Structure

Our research is motivated by the coexistence of uncertainty and preferences in many real-life temporal scenarios, as we will now illustrate considering several examples of possible applications.

Consider for example *production scheduling tasks in the industrial domain*. Many practical problems related to production planning are still solved in a heuristic way by a team of full-time schedulers with a long experience of the actual process. The main problem with the most methods available is the lack of ability to deal with the dynamic nature of the problem and with the underlying uncertainty. However using these heuristics can lead to strongly suboptimal solutions due to simplifications in the decision procedure. It is not difficult to see how using Temporal Problems with Preferences and Uncertainty would be beneficiary in this application. In fact, we propose to deal with uncertainty in an incremental fashion which is perfectly fitted for dynamic environments (see Optimal Dynamic Controllability, Section 3.7). The system simultaneously optimizes preferences avoiding sub-optimal solutions, i.e. always returning the best possible schedule. Let us see this on a specific and illustrative example from the production scheduling task in the paper converting industry [RHWI99]. The problem includes the allocation of the production runs so that the total production and inventory costs are minimized, and the customer delivery dates are met. Complicating factors in the scheduling environment are particularly the sequence-dependent setup times between different qualities, precedence constraint between production stages and a great number of production runs in relatively short-time periods. There is also some uncertainty on the completion time of a process on a specific unit due to human mistakes or machine failure that might occur and delay the process. This problem can be almost entirely modeled within our framework. In fact each run can be associated with two temporal variables indicating its starting and ending time. Similarly a variable can represent the delivery date to a customer. A binary (simple) temporal constraint with preferences between variables representing a run can model its allowed durations and the cost minimization assigning higher preferences to shorter production times. Of course such preferences can be changed to assign higher values to middle or long processing times if a higher quality is more cost effective than saving machine usage time. A soft contingent constraint can be used to represent the allowed interleaving time from when the paper leaves the plant to when it gets actually delivered to the customer. Of course although the actual delivery time is uncontrollable due to many factors as traffic, truck schedules etc., nevertheless it is preferable that the paper gets delivered as soon as possible. On the other side the customer might be willing to receive the paper early but prefers, however, to receive it just in time to minimize his inventory costs. Finally, also hard precedence and sequencing constraints can be naturally expressed within the temporal constraint network.

Another interesting software application is within *multimedia presentations*, in particular to help dealing with uncertainty in the duration of some multimedia objects [LSIR02]. This nondeterminism has been shown to have a negative effect on the quality of presentations and it is caused by external factors such as the access delay over Internet or the user interaction. In detail a multimedia presentation is a set of media objects rendered over a pre-established temporal scenario. The different objects are presented according to predefined durations set at the authoring stage. This scenario is contained in a multimedia document defined by an author and remotely accessed by users spread over the Internet. In such an environment some objects behave in a indeterministic manner and their duration does not necessarily match the predefined values. These delays can propagate in the scenario affecting the global synchronization. Predictive reasoning methods have been shown to have a limited effectiveness due to the lack of guarantees from the network and the system infrastructure which make desynchronizations unavoidable. We believe that in this context the flexibility introduced by the use of soft constraints coupled with a dynamic enforcement of consistency would be of great impact.

One of the most interesting and promising application of the work proposed here is *planning and scheduling for space missions*. In particular NASA has a wealth of scheduling problems in which STPs have been proved to be, on one side a successful tool to handle temporal reasoning, lacking however, of the capability to deal with uncertainty and preferences. Remote Agent [RBD+00], [MMPS98], represents one of the most interesting examples of this. The RAX experiment consisted in placing an artificial intelligence system on board to plan and execute spacecraft activities. Up to this experiment traditional spacecrafts were commanded through a tome-stamped sequence of extremely low level commands, such as "open valve-17" at 10:15 exactly". This low level direct commanding with rigid timestamps left the space craft little flexibility to shift around the time of commanding or to change around what hardware is used to achieve commands. One of the main features of Remote Agent was that it was commanded by a goal-trajectory consisting of high-level goals during different mission segments. A goal trajectory could contain goals such as optical navigation tasks, which specify the duration and frequency of time windows within which the spacecraft must take asteroid images to be used for orbit determination for the on-board navigator. Remote agent dealt with both flexible time intervals and uncontrollable events. The benefit of adding preferences to this framework would be to allow the planner to handle uncontrollables while maximizing the mission manager's preferences. In a sense, the application in this context of the model we propose would allow to specify goals of even a higher level as "open valve-17 within 10:00 and 10:30 and as early as possible". Reasoning on the feasibility of the goals maximizing preferences coupled with the uncontrollable events can be done as a preprocessing step, allowing the plan execution to proceed both not ruling out some possible occurrence times for uncontrollables and obtaining the best possible solution preference wise.

A more recent, but in some way similar, application is in the Rovers domain [DMR$^+$02]. In more detail, NASA is interested in the generation of optimal plans and schedule for rovers designed to explore some planets surfaces (e.g. Spirit and Opportunity for Mars). In [DMR$^+$02] the problem of generating plans for planetary rovers that handle uncertainty that involves continuous quantities as time and resources is described. Their approach involves first constructing a "seed" plan, and then incrementally adding contingent branches to this plan in order to improve utility. Notice that the way STPPUs are defined and handled is well suited for a continuous domain representation. Moreover, preferences can be used to embed utilities directly in the temporal model. For example if we consider a rover's task, e.g. moving forward, if the battery is very low a higher preference can be given to shorter durations of the motion event, while if the quality is the main goal, as in taking pictures, a longer time to allow higher resolution should be preferred. With linear functions over the intervals containing the allowed durations of the activities (moving, taking picture) these preferences can be incorporated in the model and maximized by the planner. This of course can be done also in presence of uncontrollable events and we will show this in more detail considering another space application: planning for Fleets of Earth Observing Satellites [FJMS01]. This planning problem is difficult since it involves multiple satellites, hundreds of requests, constraints on when and how to service each request, and resources such as instruments, recording devices, transmitters and ground stations. In detail scientist place their requests and the image data acquired by an EO Satellites can be either downlinked in real time or recorded on board for playback at a later time. Ground station or other satellites are available to receive downlinked images. Different satellites maybe able to communicate only with a subset of these resources, and transmission rates will differ from satellite to satellite and from station to station. Further, there may be different financial costs associated with using different communication resources. In [FJMS01] the EOS scheduling problem is dealt with using a Constraint-Based interval representation. Candidate plans are represented by variables and constraints which reflect the temporal relationship between actions, ordering decisions between actions, and constraints on the parameters of states or actions. This problem is of great interest to us since it features all the aspects we will address in this chapter:

- *Temporal constraints* which include duration and ordering constraints associated with the data collecting, recording, and downlinking tasks;

- *Preferences*: solutions are preferred based on objectives such maximizing the number of high priority requests serviced, maximizing the expected quality of the observations and minimizing the cost of downlink operations (notice that most of these preferences can be directly translated into preferences on durations of tasks);

- *Uncertainty*: the main uncertainty is due to weather and specifically to duration and persistence of cloud cover since image quality is obviously affected by the amount of clouds over the target. However, also some of the events that need to be observed may happen at unpredictable times and may have uncertain durations.

The content of this chapter has appeared in the proceedings of the following conferences and workshops:

- K. B. Venable, Neil Yorke-Smith. *Simple Temporal Problems with Preferences and Uncertainty.* In Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003), June 9-13, 2003, Trento, Italy. AAAI 2003.

- F. Rossi, K. B. Venable, and N Yorke-Smith, *Preferences and Uncertainty in Simple Temporal Problems*. in Proc. CP03 Workshop: Online-2003 (International Workshop on Online Constraints Solving - Handling Change and Uncertainty) , Kinsale, Ireland, 2003.

- F. Rossi, K. B. Venable, Neil Yorke-Smith. *Temporal Reasoning with Preferences and Uncertainty*. Poster paper in Proc. IJCAI03 (the Eighteenth International Joint Conference on Artificial Intelligence), Morgan Kaufmann, 2003, pag. 1385-1386.

- F. Rossi, K. B. Venable and N. Yorke-Smith. *Controllability of Soft Temporal Constraint Problems*. In Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP-04), LNCS 3258, pages 588-603, Springer, Toronto, Canada, September 2004.

This chapter is structured as follows. In Section 3.2 we give the background on STPUs, the formalism we will use to handle temporal uncertainty. In particular, we give the definition of the three notions of controllability (strong, dynamic and weak), along with the state of the art algorithms available to test them. As far as the preference handling system, no background is needed since the framework we refer to are STPPs which we have described in Chapter 2 of this thesis. In Section 3.3 we formally define Simple Temporal Problems with Preferences and Uncertainty merging the aforementioned approaches and obtaining both a new type of constraint problems and new notions of controllability. Algorithms to test such notions are described respectively in Section 3.5 for Optimal Dynamic Controllability, in Section 3.6 for Optimal Weak Controllability and in Section 3.7 for Optimal Dynamic Controllability. In Section 3.8 we summarize the results obtained in the previous sections and give a general strategy for dealing with temporal uncertainty and preferences. Finally, in Section 3.9, we consider the related work.

3.2 Background: Simple Temporal Problems with Uncertainty (STPUs)

Many research areas in Artificial Intelligence need to handle time in an explicit and highly expressive manner. This is particularly true when the reasoning concerns the activities that an agent performs interacting with an external world. Motivated by the fact that previous work disregarded the inherent uncertain nature of the duration of some tasks in real-life scenarios, T. Vidal and H. Fargier introduced in [VF99] a new temporal framework called Simple Temporal Problems with Uncertainty (STPUs).

The work is based on the necessity of distinguishing between *contingent* events, whose duration is controlled by nature, from *requirement* ("free" in [VF99]) events, which are under the control of the agent.

As in STPs, also in STPUs the activities have durations specified by intervals. The start times of all activities are assumed to be controlled by the agent (this brings no loss of generality). The end times, however, fall into two classes: requirement and contingent. The former, as in STP, are decided by the agent, but the latter are decided by 'nature' — the agent has no control over when the task will end; he observes it rather than executing it. The only information known prior to observation of a timepoint is that nature will respect the interval on the duration. Durations of contingent links are assumed to be independent. In an STPU the variables are thus divided into two sets, depending on the type of time points they represent. Formally:

Definition 3.1 (variables) *The variables of an STPU are divided into:*

- executable time-points*: are those points,* b_i*, whose date is assigned by the agent;*
- contingent time-points*: are those points,* f_i*, whose date is assigned by the external world.*

The distinction on variables propagates to constraints which are divided in requirement and contingent constraints depending on the type of variables they constrain. Note that as in STPs (see Section Chapter 2 Section 2.2.1) all the constraints are binary.

Definition 3.2 *In an STPU there are two types of constraints:*

- *a* requirement constraint (or link) r_{ij}*, on generic time-points* t_i *and* t_j [1]*, is an interval* $I_{ij} = [l_{ij}, u_{ij}]$ *such that* $l_{ij} \leq \gamma(t_j) - \gamma(t_i) \leq u_{ij}$ *where* $\gamma(t_i)$ *is a value assigned to variable* t_i

- *a* contingent link g_{hk}*, on executable point* b_h *and contingent point* f_k*, is an interval* $I_{hk} = [l_{ij}, u_{ij}]$ *which contains all the possible durations of the contingent event represented by* b_h *and* f_k*.*

The formal definition of an STPU is the following.

Definition 3.3 *A Simple Temporal Problem with Uncertainty is a 4-tuple* $N = \{X_e, X_c, R_r, R_{ct}\}$ *such that:*

- $X_e = \{b_1, \ldots, b_m\}$*: is the set of* m *executable time points;*

- $X_c = \{f_1, \ldots, f_l\}$*: is the set of* l *contingent time points;*

- $n = m + l$ *is the total number of variables;*

- $R_r = \{c_{i_1 j_1}, \ldots, c_{j_C j_C}\}$*: is the set of* C *requirement constraints;*

- $R_{ct} = \{g_{i_1 j_1}, \ldots, g_{i_G j_G}\}$*: is the set of G contingent constraints;*

We now give a short example, taken for [VF99], describing a scenario which can be modeled using an STPU.

[1]In general t_i and t_j can be either contingent or executable time-points

Example 3.1 Consider two activities as *Cooking* and *Having dinner*. Assume you don't want to eat your dinner cold. Also, assume, you can control when you start cooking and when the dinner starts but not when you finish cooking or when the dinner will be over. The STPU modeling this example is depicted in Figure 3.1. There are two executable timepoints { *Start-cook, Start-dinner* } and two contingent time-points { *End-cook, End-dinner* }. Moreover, the contingent constraint on variables { *Start-cook, End-cook* } models the uncontrollable duration of fixing dinner which can take anywhere from 20 to 40 minutes; the contingent constraint on variables { *Start-dinner, End-dinner* } models the uncontrollable duration of the dinner that can last from 30 to 60 minutes. Finally, there is a requirement constraint on variables { *End-cook, Start-dinner* } that simply bounds to 10 minutes the interleaving time between when the food is ready and when the dinner starts.

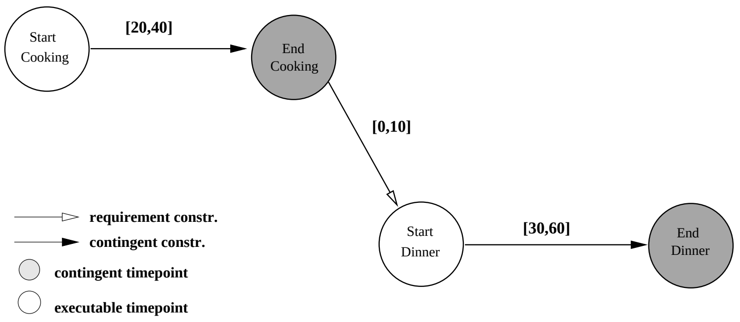


Figure 3.1: STPU corresponding to Example 3.1.

In [VF99] the authors distinguish also between assignments to executable variables and assignments to contingent variables. In detail:

Definition 3.4 (control sequence) *A* control sequence δ *is an assignment to executable time points. It is said to be* partial *if it assigns values to a subset of the executables in the STPU.*

Definition 3.5 (situation) *A situation ω is a set of durations on contingent constraints. If not all the contingent constraints are assigned a duration it is said to be* partial.

Moreover,

Definition 3.6 (schedule) *A* Schedule T *is a complete assignment to all the time-points in* X_e *and* X_c.

Sol(P) denotes the set of all schedules of STPU P. Given a schedule T, it identifies a control sequence, δ_T, consisting of all the assignments to the executable time points and a situation, ω_T, which is the set of all the durations identified by the assignments in T on the contingent constraints.

It is easy to see that to each situation corresponds an STP. In fact, once the durations of the contingent constraints are fixed, there is no more uncertainty in the problem, which, thus, becomes an STP. This is formalized by the notion of $\mathtt{projection}$.

Definition 3.7 (projection) *A* projection P_ω*, corresponding to a situation* ω*, is the STP obtained leaving all requirement constraints unchanged and replacing each contingent constraint* g_{hk} *with the constraint* $\langle[\omega_{hk}, \omega_{hk}]$*, where* ω_{hk} *is the duration of the event represented by* g_{hk} *in* ω*.*

Proj(P) is the set of all projections of an STPU P.

It is clear that in order to solve a problem with uncertainty all the possible situations must be considered. Clearly the notion of consistency defined for STPs (see Chapter 1 Section 2.2.1) does not apply since it requires the existence of a single schedule, which is clearly not sufficient in this case since all situations are equally possible. For this reason, in [VF99], they introduce the notion of controllability. *Controllability* of an STPU is, in some sense, the analogue of consistency of an STP. Controllable means the agent has a means to execute the timepoints under his control, subject to all constraints. The notion of controllability is expressed, thus, in term of the ability of the agent to find, given a situation, an appropriate control sequence. This ability is identified with having a strategy.

Definition 3.8 (strategy) Strategy S *is map* $S : \mathrm{Proj}(P) \to \mathrm{Sol}(P)$ *such that for every projection* P_ω, $S(P_\omega)$ *is a schedule which includes* ω*.*

We will write $[S(P_\omega)]_x$ to indicate the value assigned to executable time point x in schedule $S(P_\omega)$, and $[S(P_\omega)]_{<x}$ the *history* of x in $S(P_\omega)$, that is, the set of durations of contingent constraints which occurred in $S(P_\omega)$ before the execution of x.

Definition 3.9 (viable strategy) *A strategy* $S : \mathtt{Proj}(P) \to \mathtt{Sol}(P)$ *is viable if, for every projection* P_ω, $S(P_\omega)$ *is a solution of* P_ω, *i.e. it is consistent with all the constraints of* P_ω.

In [VF99] three notions of controllability are proposed, they are defined in the following sections.

3.2.1 Strong controllability

The first notion is, as the name suggests, the strongest in terms of requirements.

Definition 3.10 (Strong Controllability) *An STPU* P *is* Strongly Controllable *(SC) iff there is an execution strategy* S *s.t.* $\forall P_\omega \in \mathtt{Proj}(P)$, $S(P_\omega)$ *is a solution of* P_ω, *and* $[S(P_1)]_x = [S(P_2)]_x$, $\forall P_1, P_2$ *projections and for every executable timepoint* x.

In words, an STPU is *strongly controllable* if there is a fixed execution strategy that works in all situations. This means that there is a fixed control sequence that will be consistent with any possible scenario of the world. This is clearly a very strong requirement. As the authors suggest in [VF99], SC may be relevant in some applications where the situation is not observable at all or where the complete control sequence must be known beforehands (for example in cases in which other activities depend on the control sequence, as in the production planning area).

In [VF99] a polynomial time algorithm for checking if an STPU is strongly controllable is proposed. The main idea is to rewrite the STPU given in input as an equivalent STP only on the executable variables.

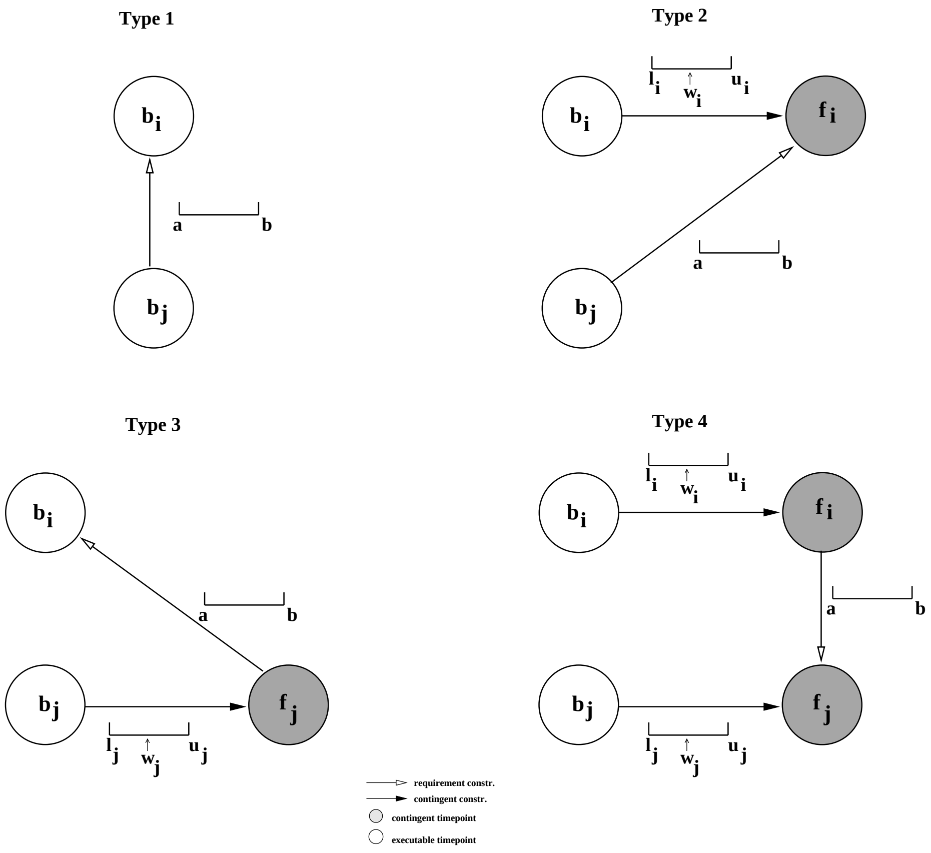


Figure 3.2: Possible types of requirement constraints.

Consider Figure 3.2 which shows all the requirement constraints that can be defined, given two contingent constraints defined on executable b_i and contingent f_i with interval $[u_i, l_i]$ (resp., executable b_j and contingent f_j with interval $[u_j, l_j]$).

We will indicate with ω_i (resp. ω_j) the duration on the contingent constraint defined on b_i and f_i (resp. on b_j and f_j). We will also, to simplify the notation, use the name of the variable to indicated the value assigned to it. Any requirement constraint, with interval $[a, b]$, is in one of the following four positions and imposes one of the corresponding inequalities[2]:

1. Type 1: requirement constraint on b_j and b_i which are both executable: $a \leq b_i - b_j$;

2. Type 2: requirement constraint on b_j and f_i: $b_i + \omega_i - b_j; \geq a$;

3. Type 3: requirement constraint on b_i and f_j: $b_i - b_j - \omega_j \geq a$;

4. Type 4: requirement constraint on e_i and f_j: $b_i + \omega_i - b_j - \omega_j \geq a$.

In [VF99] it is shown that if the above inequalities hold replacing the upper and lower bound of the contingent intervals then the STPU is strongly controllable. Moreover this is also a necessary condition.

In Figure 3.3 the pseudocode of algorithm Strong-Controllability [VF99] is given.

The algorithm takes an STPU P as input. Notice that in line 2 it initializes an STP Q with a simple temporal problem defined on the executable variables of P, X_e, and with the empty set of constraints. In fact all the constraints defined by the algorithm are only on executables, thus the output is such an STP. In [VF99] it is shown that STPU P is SC iff STP Q, returned by Strong-Controllability, is consistent and that any solution of Q is a control sequence satisfying SC. In [VF99] the authors also show that the complexity of Strong-Controllability is bounded by that of applying Floyd-Warshall, $O(n^3)$, or PC-2, $O(n^3 r)$, where n is the number

[2]The inequalities are given only for the lower bound since the ones for the upper bound can be obtained similarly.

Pseudocode of Strong-Controllability

```
1. input STPU $P=\langle X_e, X_c, R_r, R_{ct}\rangle$;
2. STP $Q \leftarrow \langle X_e, \emptyset\rangle$;
3. for every pair of contingent constraints $g_{b_i f_i}$ and $g_{b_j f_j}$:
4.     for all constraints of Type 2 in Q, replace constraint with : $b_i - b_j \leq a - l_i$;
5.     for all constraints of Type 3 in Q, replace constraint with : $b_i - b_j \leq a + u_j$;
6.     for all constraints of Type 4 in Q, replace constraint with : $b_i - b_j \leq a - l_i + u_j$;
7.     impose in Q the correspondent inequalities for upper bound $b$;
8. if Q consistent write "P SC" and return Q ;
9. else write " P not SC";
```

Figure 3.3: Algorithm Strong-Controllability proposed in [VF99] to test if an STPU is strongly controllable.

of variables and r is the range of the interval [DMP91] (see Chapter 2 Section 2.2.1).

3.2.2 Weak controllability

On the other hand, the notion of controllability with the least restrictions on the control sequences is Weak Controllability.

Definition 3.11 (Weak Controllability) *An STPPU is* Weakly Controllable *(WC) iff* $\forall P_\omega \in Proj(P)$ *there is a strategy* S_ω *s.t.* $S_\omega(P_\omega)$ *is a solution of* P_ω.

In other words, an STPU is *weakly controllable* if there is a viable global execution strategy: there exists at least one schedule for every situation.

This can be seen as a minimum requirement since if this property doesn't hold then there are some situation such that there is no way to execute the controllable events in a consistent way.

It also looks attractive since, once an STPU is shown to WC, as soon as one knows the situation, one can pick out and apply the control sequence that matches that situation. Unfortunately in [VF99] it is shown that this property is not so useful in classical planning. Nonetheless, WC may be relevant in specific applications (as large-scale warehouse scheduling) where the actual situation will be totally observable before (possibly *just before*) the execution starts, but one wants to know in advance that, whatever the situation, there will always be at least one feasible control sequence.

In [VF99] the conjecture that checking weak controllability is Co-NP-complete was initially formulated. Such conjecture was proven in [MM99].

The algorithm proposed for testing WC in [VF99] is based on a classical enumerative process and a lookahead technique based on the following property.

Definition 3.12 (weak controllability on bounds) *An STPU P is weakly controllable iff $\forall \omega \in \{l_1, u_1\} \times \cdots \times \{l_G, u_G\}$, P_ω is a consistent STP.*

In the definition, $\{l_j, u_j\}$ are respectively the lower and upper bound of the interval of a contingent constraint. So, with $\omega \in \{l_1, u_1\} \times \cdots \times \{l_G, u_G\}$, situations that identify durations all corresponding to the bounds of the contingent intervals are defined.

In [VF99] it shown that, obviously, Strong Controllability implies Weak Controllability. Moreover, an STPU can be seen as an STP if the uncertainty is ignored. If enforcing path consistency, for example applying algorithm PC-2 described in Chapter 2 Section 2.2.1, cuts some elements out from the contingent intervals, it means that such elements don't belong to any solution. If this is the case, it is possible to conclude that the STPU is not weakly controllable.

Definition 3.13 (pseudo controllability) *An STPU is pseudo controllable if applying PC-2 leaves the intervals on the contingent constraints unchanged.*

Unfortunately, if path consistency leaves the contingent intervals untouched it is not sufficient to conclude that the STPU is weakly controllable. In fact weak controllability requires that given any possible combination of durations of all contingent constraints the STP corresponding to that projection must be consistent. Pseudo controllability, instead, only guarantees that for each possible duration on a contingent constraint there is at least one projection that contains such duration and that is a consistent STP.

In Figure 3.4 we show the pseudo code of the algorithm proposed in [VF99] for testing weak controllability.

Pseudocode of WeakControllability

```
1.  input STPU $P=\langle X_e, X_c, R_r, R_{ct}\rangle$;
2.  if $P$ not pseudocontrollable write "not WC" and stop;
3.  if (Strong-Controllability(P)) write "WC" and stop;
4.  else if in $P$ $R_{ct} = \emptyset$ write "not WC" and stop;
5.  else
6.      choose and remove a $g_i$ from $R_{ct}$;
7.      replace the interval on $g_i$ with its lower bound $[l_i, l_i]$;
8.      if (WeakControllability(P)=WC) then ;
9.        replace the interval on $g_i$ with its upper bound $[u_i, u_i]$;
10.       WeakControllability(P);
```

Figure 3.4: Algorithm WeakControllability proposed in [VF99] to test whether an STPU is weakly controllable.

The algorithm takes as input an STPU P and checks if it is pseudo-controllable (lines 1-2). If the problem is not pseudo-controllable then it stops, since it cannot be weakly controllable. Next, it tests if the problem is strongly controllable by applying algorithm Strong-Controllability (line 3) . If the STPU is strongly controllable then it is weakly controllable as well. Finally (lines 6-10), it starts replacing the intervals of contingent constraints first with their lower bound and then with their upper bound, each time testing pseudo-controllability and strong controllability. If no inconsistency has been found and all the contingent constraints have been replaced by their bounds, then the STPU is weakly controllable.

3.2.3 Dynamic controllability

In dynamic applications domains, such as planning, the situation is built in the process of time, as far as one observes the situation at hand. This implies that decisions have to be made even if the situation remains partially unknown. Indeed the distinction between Strong and Dynamic Controllability is equivalent to that between conformant and conditional planning. In fact Conformant Planning is concerned with finding a sequence of actions that is guaranteed to achieve the goal regardless of the nondeterminism of the domain. In Conditional Planning, instead, plan steps are conditional on the current state of the world and are generated under partial observability conditions [GNT04, RN95].

The definition of Dynamic Controllability has been given first in [VF99]. Here we give the definition provided in [MMV01] which is equivalent but more compact.

Definition 3.14 (Dynamic Controllability) *An STPPU* P *is* Dynamic Controllable *(DC) iff there is a strategy* S *such that* $\forall P_1, P_2$ *in* $\mathrm{Proj}(P)$ *and for any executable time-point* x*:*

1. *if* $[S(P_1)]_{<x} = [S(P_2)]_{<x}$ *then* $[S(P_1)]_x = [S(P_2)]_x$*;*

2. $S(P_1)$ *is a solution of* P_1 *and* $S(P_2)$ *is a solution of* P_2*.*

In [VF99] they show that SC $\Rightarrow$ DC and that DC $\Rightarrow$ WC. In [MMV01] the complexity issue of testing this property has been solved. Dynamic Controllability, which is the most useful property in practice, is also the one that requires the most complicated algorithm which is surprisingly pseudo-polynomial. The work we will present in Section 3.7 is strongly related to the algorithm for testing DC presented in [MMV01], called DynamicallyControllable?. Such algorithm is based on some considerations on triangles of constraints. Consider the one depicted in Figure 3.5.

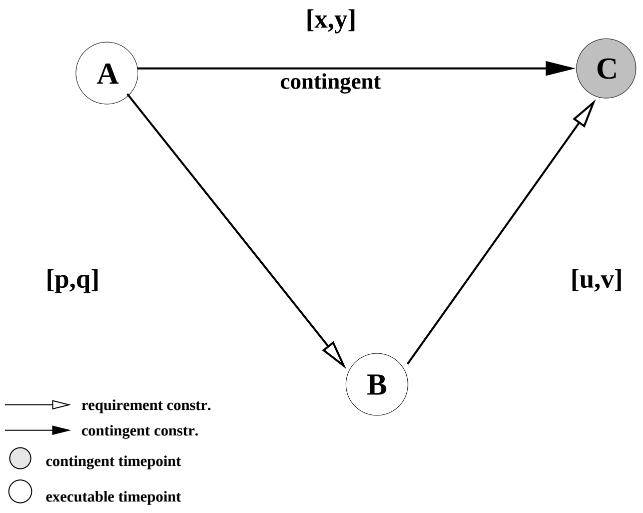


Figure 3.5: Triangular STPPU

The triangle shown in Figure 3.5 is a triangular STPU with one contingent constraint, AC, two executable timepoints, A and B, and a contingent time point C. Based on the sign of u and v, three different cases can occur:

- *Follow case* ($v < 0$)*:* B will always follow C. If the STPU is path consistent then it is also DC since, given the time at which C occurs after A, by definition of path consistency, it is always possible to find a consistent value for B.

- *Precede case* ($u \geq 0$)*:* B will always precede or happen simultaneously with C. Then the STPU is dynamically controllable if $y - v \leq x - u$ and the interval $[p, q]$ on AB should be replaced by interval $[y - v, x - u]$, that is by the sub-interval containing all the elements of $[p, q]$ that are consistent with each element of $[x, y]$.

- *Unordered case ($u < 0$ and $v \geq 0$):* B can either follow or precede C. To ensure dynamic controllability, B must wait either for C to occur first, or for $t = y - v$ units of time to go by after A. In other words, either C occurs and B can be activated at the first value consistent with C's time, or B can safely be executed t units of time after A's execution. This can be described by an additional constraint which is expressed as a *wait* on AB and is written $< C, t >$, where $t = y - v$. Of course if $x \geq y - v$ then we can raise the lb of AB, p, to $y - v$ (*Unconditional Unordered Reduction*). If $x > p$ we always raise p to x (*General Unordered reduction*) .

It can be shown that waits can be propagated (in [MMV01] they use the term "regressed") from one constraint to another, that is, a wait on AB induces a wait on another constraint involving A, e.g. AD, depending on the type of constraint DB. In particular, there are two possible ways in which the waits can be regressed.

- *Regression 1:* assume that the AB constraint has a wait $\langle C, t \rangle$. Then, if there is any DB constraint (including AB itself) with an upper bound, w, it is possible to deduce a wait $\langle C, t - w \rangle$ on AD. Figure 3.6 shows this type of regression.

- *Regression 2:* assume that the AB constraint has a wait $\langle C, t \rangle$, where $t \geq 0$. Then, if there is a contingent constraint DB with a lower bound, z, and such that $B \neq C$, it is possible to deduce a wait $\langle C, t - z \rangle$ on AD. Figure 3.7 shows this regression.

Assume for simplicity and without loss of generality that A is executed at time 0. Then, B can be executed before the wait only if C is executed first. After the wait expires, B can safely be executed at any time left in the interval. As Fig. 3.8 shows, it is possible to consider the Follow and Precede cases as special cases of the Unordered. In the Follow case we can put a "dummy" wait after the end of

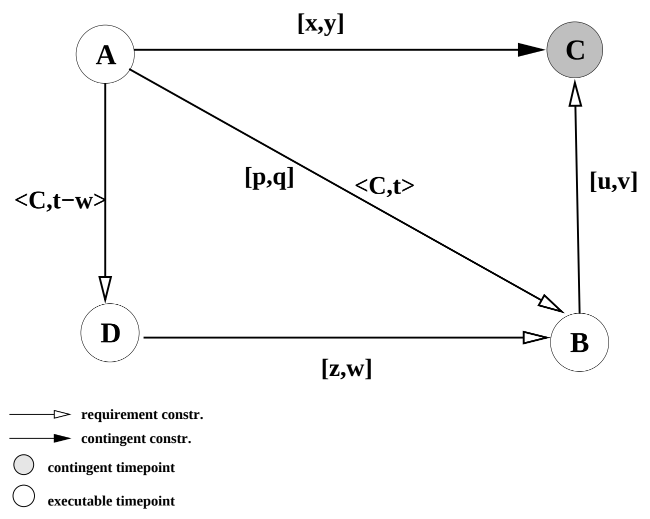


Figure 3.6: Regression 1.

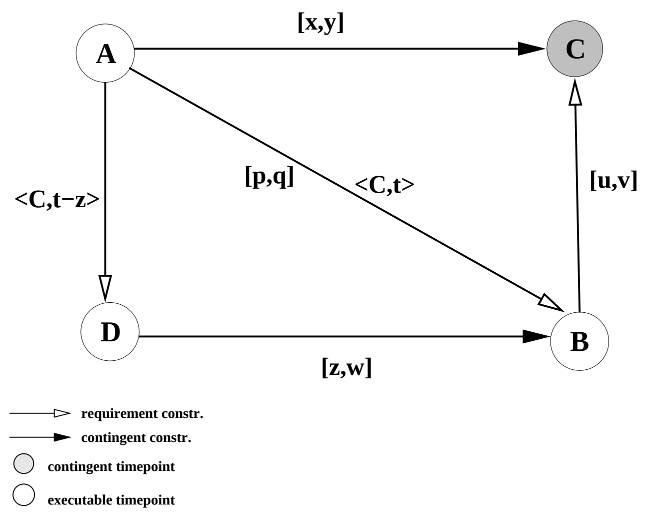


Figure 3.7: Regression 2.

the interval meaning that B must wait for C to be executed in any case (Fig.3.8 (a)). In the Precede case, we can set a wait that expires at the first element of the interval meaning that B will be executed before C and any element in the interval will be consistent with C (Fig.3.8 (b)). The Unordered case can, thus, be seen as a combination of the two previous states. The part of the interval before the wait can be seen as a Follow case (in fact, B must wait for C until the wait expires), while the second part including and following the wait can be seen as a Precede case (after the wait has expired, B can be executed and any assignment to B that corresponds to an element of this part of interval AB will be consistent with any possible future value assigned to C).

In Figure 3.8 we show the interval on constraint AB in the three cases.

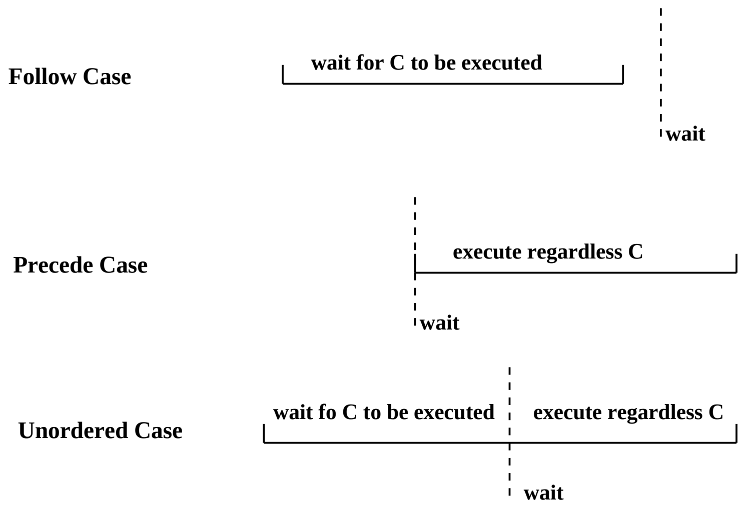


Figure 3.8: The resulting AB interval constraint in the three cases considered by the DynamicallyControllable? algorithm.

The algorithm proposed in [MMV01], shown in Fig. 3.9, applies these rules to all triangles in the STPU and regresses all possible waits. If no inconsistency is found, that is no requirement interval becomes empty and no contingent interval is squeezed, the STPU is DC and the algorithm returns an STPU where some constraints have waits to satisfy and the intervals contain elements that appear in at least a possible dynamic strategy.

This STPU can then be given to an execution algorithm which dynamically

Pseudocode of DynamicallyControllable?

```
1. input STPU W;
2. If W is not pseudo-controllable then write "not DC" and stop;
3. Select any triangle ABC, C uncontrollable, A before C, such that the upper
   bound of the BC interval, $v$, is non-negative.
4. Introduce any tightenings required by the Precede case and any waits re-
   quired by the Unordered case.
5. Do all possible regressions of waits, while converting unconditional waits
   to lower bounds. Also introduce lower bounds as provided by the general
   reduction.
6. If steps 3 and 4 do not produce any new (or tighter)
   constraints, then return true, otherwise go to 2.
```

Figure 3.9: Algorithm DynamicallyControllable? proposed in [MMV01] for checking Dynamic Controllability of an STPU.

assigns values to the executables according to the current situation. The pseudocode of the execution algorithm, DC-Execute, is shown in Figure 3.10.

The execution algorithm observes, as the time goes by, the occurrence of the contingent events and executes the controllables. For any controllable B, its execution is triggered if it is (1) $live$, that is, if current time is within its bounds, it is (2) $enabled$, that is, all the executables constrained to happen before have occurred, and (3) all the waits imposed by the contingent timepoints on B have expired. In [MMV01], the authors prove the soundness and completeness of DynamicallyControllable? by showing that DC-Execute produces dynamically a consistent schedule on every STPU on which DynamicallyControllable? reports success. Moreover, the authors show that the complexity of the algorithm is

Pseudocode for DC-Execute

```
1. **input** STPU $P$;
2. Perform initial propagation from the start timepoint;
3. **repeat**
4.    immediately execute any executable timepoints that
      have reached their upper bounds;
5.    arbitrarily pick an executable timepoint $x$ that
      is live and enabled and not yet executed, and whose waits,
      if any, have all been satisfied;
6.    execute $x$;
7.    propagate the effect of the execution;
8.    **if** network execution is complete **then** return;
9.    **else** advance current time,
      propagating the effect of any contingent timepoints that occur;
10. **until** false;
```

Figure 3.10: Algorithm that executes a dynamic strategy for an STPU.

$O(n^2r)$ where n is the number of variables and r is the number of elements in an interval. Since the value of r depends on the granularity chosen, the authors conclude that DynamicallyControllable? is *deterministic* polynomial.

3.3 Simple Temporal Problems with Preferences and Uncertainty (STPPUs)

Consider a temporal problem that we would model naturally with preferences in addition to hard constraints, but that also features uncertainty. Neither a STPP nor a STPU is adequate. Therefore we propose what we will call *Simple Temporal Problems with Preferences and Uncertainty*, or STPPU for short.

An initial definition of a STPPU is a STPP for which (end) timepoints are partitioned into two classes, requirement and contingent, just as in a STPU. Since some timepoints are not controllable by the agent, the notion of consistency of a STP(P) is replaced by that of controllability, just as in a STPU. Every solution to the STPPU has a global preference value, just as in a STPP, and we seek a solution which maximise this value.

More precisely, we can extend some definitions given for STPPs and STPUs to fit STPPUs in the following way.

Definition 3.15 *In a context with preferences:*

- executable time-point *is a variable,* b_i*, whose date is assigned by the agent;*
- contingent time-point *i a variable,* e_i*, whose uncontrollable date is assigned by the external world;*
- *a* soft requirement link r_{ij}*, on generic time-points* t_i *and* t_j [3]*, is a pair* $\langle I_{ij}, f_{ij}\rangle$*, where* $I_{ij} = [l_{ij}, u_{ij}]$ *such that* $l_{ij} \leq \gamma(t_j) - \gamma(t_i) \leq u_{ij}$ *where* $\gamma(t_i)$ *is a value assigned to variable* t_i*, and* $f_{ij} : I_{ij} \rightarrow A$ *is a preference function mapping each*

[3]In general, t_i and t_j can be either contingent or executable time-points.

element of the interval into an element of the preference set of the semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$*;*

- *a* soft contingent link g_{hk}*, on executable point* b_h *and contingent point* e_k*, is a pair* $\langle I_{hk}, f_{hk}\rangle$ *where interval* $I_{hk} = [l_{hk}, u_{hk}]$ *contains all the possible durations of the contingent event represented by* b_h *and* e_k *and* $f_{hk} : I_{hk} \rightarrow A$ *is a preference function that maps each element of the interval into an element of the preference set.*

Further, without loss of generality, and following the assumptions made for STPUs [MMV01], we may assume no two contingent constraints end at the same timepoint.

In both types of constraints, the preference function represents the preference of the agent on the duration of an event or on the distance between two events. The difference is that, while for soft requirement constraints the agent has control and can be guided by the preferences in choosing values for the timepoints, for soft contingent constraints the preference represents merely a desire of the agent on the possible outcomes of Nature: there is no control on the outcomes.

We can now state formally the definition of STPPUs, which combine preferences from the definition of a STPP with contingency from the definition of a STPU. Note that we consider links that are hard constraints to be soft constraints in which each element of the interval is mapped into the maximal element of the preference set.

Definition 3.16 (STPPU) *A* Simple Temporal Problem with Preferences and Uncertainty P *is a tuple* $P = (N_e, N_c, L_r, L_c, S)$ *where:*

- N_e *is the set of executable points;*
- N_c *is the set of contingent points;*
- L_r *is the set of soft requirement constraints over semiring S;*
- L_c *is the set of soft contingent constraints over semiring S;*

- $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ *is a c-semiring.*

Once we have a complete assignment to all timepoints we can compute its global preference. This is done according to the semiring-based soft constraint schema: first we project the assignment on all soft constraints, obtaining an element of the interval and the preference associated to that element; then we combine all the preferences obtained with the multiplicative operator of the semiring. Given two assignments with their preference, the best is chosen using the additive operator. An assignment is *optimal* if there is no other assignment with a preference which is better in the semiring's ordering.

In order to analyse the solutions to a STPPU, we need further preliminary definitions:

Definition 3.17 *Given an STPPU P:*

- *A* Schedule T *is a complete assignment to all the time-points in N_e and N_c;*
- Sol(P) *the set of all schedules of STPPU P;*
- *A* situation ω*, given a schedule, is the set of durations of all contingent constraints;*
- *A* control sequence δ*, given a schedule, is the set of assignments to executable time-points;*
- *A* projection P_ω *corresponding to a situation ω is the STPP obtained leaving all requirement constraints unchanged and replacing each contingent constraint g_{hk} with the soft constraint constraint $\langle [\omega_{hk}, \omega_{hk}], f(\omega_{hk})] \rangle$, where ω_{hk} is the duration of event represented by g_{hk} in ω, and $f(\omega_{hk})]$ is the preference associated to such duration;*
- *Given a projection P_ω we indicate with $\mathtt{opt}(P_\omega)$ the preference of any optimal solution of P_ω;*
- Proj(P) *is the set of all projection of an STPPU P;*

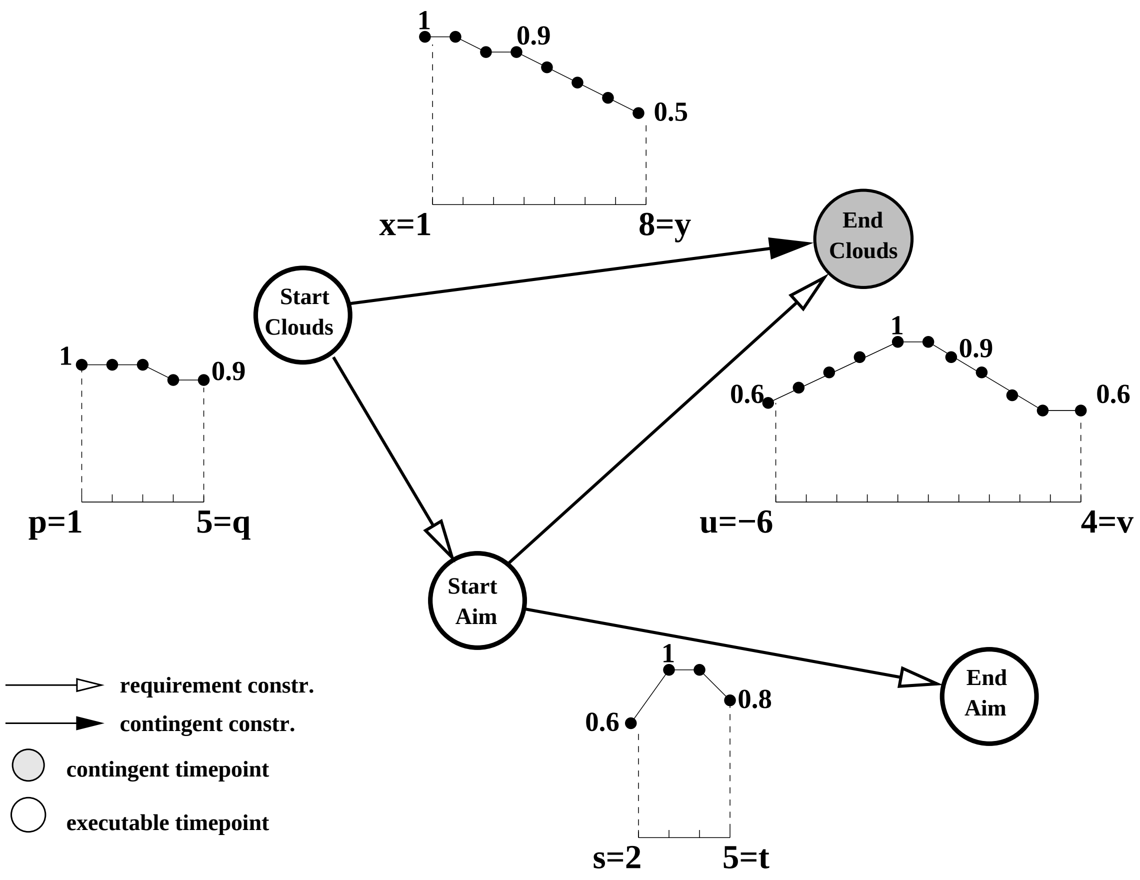


Figure 3.11: Example STPPU from the Earth Observing Satellites domain.

- *A* Strategy S *is map* $S : \text{Proj}(P) \longrightarrow \text{Sol}(P)$ *such that for every projection* P_ω, $S(P_\omega)$ *is a schedule which includes* ω.

- *A strategy is* viable *it* $\forall\omega$, $S(P_\omega)$ *is a solution of* $S(P_\omega)$.

Regarding notation, as in the case with hard constraints, given an executable timepoint x, we will write $[S(P_\omega)]_x$ to indicate the value assigned to x in $S(P_\omega)$, and $[S(P_\omega)]_{<x}$ to indicate the durations of the contingent events that finish prior to x in $S(P_\omega)$.

Example 3.2 Consider as an example the following scenario from the Earth Observing Satellites domain [FJMS01] described in the introduction of this chapter.

Suppose a request for observing a region of interest has been received and accepted. To collect the data, the instrument must be aimed at the target before images can be taken. It might be, however, that for a certain period during the time window allocated for this observation, the region of interest is covered by clouds. The earlier the cloud coverage ends the better, since it will maximise both the quality and the quantity of retrieved data; but coverage is not controllable.

Suppose the time window reserved for an observation is from 1 to 8 units of time and that we start counting time when the cloud occlusion on the region of interest is observable. Suppose, in order for the observation to succeed, the aiming procedure must start before 5 units after the starting time, ideally before 3 units, and it actually can only begin after at least 1 time unit after the weather becomes observable. Ideally the aiming procedure should start slightly before the cloud coverage will end. If it starts too early, then, since the instrument is activated immediately after it is aimed, clouds might still occlude the region and the image quality will be poor. On the other hand, if it waits until the clouds have disappeared then precious time during which there is no occlusion will be wasted aiming the instrument instead of taking images. The aiming procedure can be controlled by the mission manager and it can take anywhere between 2 and 5 units of time. An ideal duration is 3 or 4 units, since a short time of 2 units would put the instrument under pressure, while a long duration, like 5 units, would waste energy.

This scenario, rather tedious to describe in words, can be compactly represented by the STPPU shown in Fig. 3.11 with the following features:

- a set of executable timepoints *SC*, *SA*, *EA*;
- a contingent timepoint *EC*;
- a set of soft requirement constraints on $SC \rightarrow SA$, $SA \rightarrow EC$, $SA \rightarrow EA$;
- a soft contingent constraint $SC \rightarrow EC$;
- the fuzzy semiring $S_{FCSP} = \langle [0,1], \max, \min, 0, 1 \rangle$.

A solution of the STPPU in Fig. 3.11 is the schedule $s = \{SC = 0, SA = 2, EC = 5, EA = 7\}$. The situation associated with s is the projection on the only contingent constraint, $SC \rightarrow EC$, i.e. $\omega_s = 5$, while the control sequence is the assignment to the executable timepoints, i.e. $\delta_s = \{SC = 0, SA = 2, EA = 7\}$. The global preference is obtained considering the preferences associated with the projections on all constraints, that is pref$(2) = 1$ on $SC \rightarrow SA$, pref$(3) = 0.6$ on $SA \rightarrow EC$, pref$(5) = 0.9$ on $SA \rightarrow EA$ and pref$(5) = 0.8$ on $SC \rightarrow EC$. The preferences must then be combined using the multiplicative operator of the semiring, which is min, so the global preference of s is 0.6. Another solution $s' = \{SC = 0, SA = 4, EC = 5, EA = 9\}$ has global preference 0.8. Thus s' is a better solution than s according to the semiring ordering since $\max(0.6, 0.8) = 0.8$.

3.4 Controllability

We now consider how it is possible to extend the notion of controllability to accommodate preferences. In general we are interested in the ability of the agent to execute the timepoints under its control not only subject to all constraints but also in the best possible way w.r.t. preferences. It transpires the meaning of 'best possible way' depends on the types of controllability we introduced earlier. In particular, the concept of optimality must be reinterpreted due to the presence of uncontrollable events. In fact, the distinction on the nature of the events induces a difference on the meaning of the preferences expressed on them. In fact a preference on a situation (i.e. a set of projections on contingent constraints) should be interpreted as how favorable that scenario is for the agent. It should be noticed that, although the agent expresses a preference over the different situations she has no control over those events. Once a scenario with a certain level of desirability is given, then the agent often has several choices for the events dhe controls that are consistent with that scenario. Some of these choices might be preferable with respect to others. This is expressed by the preferences on the requirement constraints and such information should guide the agent in choosing the best possible actions to take. Thus, the concept of optimality is now "relative" to the specific scenario. The final preference of a complete assignment is an overall value which is the minimum of how much the corresponding scenario is desirable for the agent and how well the agent has reacted in that scenario. The concepts of controllability we will propose here are, thus, based on the possibility of the agent to execute the events under her control in the best possible way given the actual situation. Thus acting in an optimal way can be seen as not lowering further the preference given by the uncontrollable events.

For each of the three levels of controllability defined for STPUs two different ways of optimizing preferences are considered: the first one requires optimality in all situations, the second (weaker) one requires the final outcome not have a preference below a certain level unless the situation itself has a preference below such threshold.

3.4.1 Strong controllability with preferences

We start considering the strongest notion of controllability, Strong Controllability. We extend this notion, taking into account preferences, in two ways thus obtaining Optimal Strong Controllability and α-Strong Controllability, where α is a preference level. As we will see, the first is a stronger requirement since it assumes the existence of a fixed unique assignment for all the executable timepoints that is optimal in every projection. The second notion requires such fixed assignment to be optimal only in those projections that have a maximum preference value smaller than or equal to α, and to yield a preference $\geq \alpha$ in all other cases.

Definition 3.18 (Optimal Strong Controllability) *An STPPU* P *is said to be* Optimally Strongly Controllable *(OSC) iff there is an execution strategy S s.t.* $\forall P_\omega \in \mathrm{Proj}(P)$, $S(P_\omega)$ *is an optimal solution of* P_ω, *and* $[S(P_1)]_x = [S(P_2)]_x$, $\forall P_1, P_2$ *projections and for every executable timepoint* x.

In other words, an STPPU is OSC if there is a fixed control sequence that works in all possible situations and is optimal in each of them. In the definition, 'optimal' means that there is no other assignment the agent can choose for the executable timepoints that could yield a higher preference in any situation. Since this is a powerful restriction, as mentioned before, we can instead look at just reaching a certain quality threshold:

Definition 3.19 (α-Strong Controllability) *An STPPU* P *is* α-Strongly Controllable *(α-SC), with $\alpha \in A$ a preference, iff there is a strategy S s.t.:*

1. $\forall P_\omega \in \mathtt{Proj}(P)$, $S(P_\omega)$ *is a solution of* P_ω *such that* $[S(P_1)]_x = [S(P_2)]_x$, $\forall P_1, P_2$ *projections and for every executable timepoint* x*; and*

2. *the global preference of* $S(P_\omega)$ *is preference* $\mathtt{opt}(P_\omega)$ *if* $\mathtt{opt}(P_\omega) \leq \alpha$*, otherwise the global preference of* $S(P_\omega)$ *is* $\geq \alpha$*.*

In other words, an STPPU is α-SC if there is a fixed control sequence that works in all situations and results in optimal schedules for those situation where the optimal preference level of the projection is $\leq \alpha$ and results in a schedule with preference at least α in all other cases.

3.4.2 Weak controllability with preferences

Secondly, we extend similarly the less restrictive notion of controllability: Weak Controllability. Weak Controllability requires the existence of a solution in any possible situation, possibly a different one in each situation. We extend this definition requiring the existence of an optimal solution in every situation.

Definition 3.20 (Optimal Weak Controllability) *An STPPU is* Optimally Weakly Controllable *(OWC) iff* $\forall P_\omega \in \mathtt{Proj}(P)$ *there is a strategy* S_ω *s.t.* $S_\omega(P_\omega)$ *is an optimal solution of* P_ω*.*

In other words, an STPPU is OWC if, for every situation, there is a fixed control sequence that results in an optimal schedule for that situation.

Optimal Weak Controllability of an STPPU is equivalent to Weak Controllability of the corresponding STPU obtained by ignoring preferences, as we will formally prove in Section 3.6. The reason is that if a projection P_ω has at least one solution then it must have an optimal solution (i.e. one with the highest preference $\mathtt{opt}(P_\omega)$). This also implies that an STPPU is such that its underlying STPU is either WC or not. Hence it does not make sense to define a notion of α-Weak Controllability.

3.4.3 Dynamic controllability with preferences

Dynamic Controllability is seen as the more useful notion of controllability in practice. It addresses the ability of the agent to execute a schedule by choosing incrementally the values to be assigned to executable timepoints, looking only at the past. When preferences are available, it is desirable that the agent acts not only in a way that is guaranteed to be consistent with any possible future outcome but also in a way that ensures the absence of regrets w.r.t. preferences.

Definition 3.21 (Optimal Dynamic Controllability) *An STPPU* P *is* Optimally Dynamically Controllable *(ODC) iff there is a strategy* S *such that* $\forall P_1, P_2$ *in* $\mathtt{Proj}(P)$ *and for any executable timepoint* x*:*

1. *if* $[S(P_1)]_{<x} = [S(P_2)]_{<x}$ *then* $[S(P_1)]_x = [S(P_2)]_x$*;*

2. $S(P_1)$ *is a consistent complete assignment for* P_1 *and* $S(P_2)$ *is a consistent complete assignment for* P_2*;*

3. $\mathrm{pref}(S(P_1))$ *is optimal in* P_1 *and* $\mathrm{pref}(S(P_2))$ *is optimal in* P_2*.*

In other words, an STPPU is ODC is there exists a means of extending any current partial control sequence to a complete control sequence in the future in such a way that the resulting schedule will be optimal. As before, we also soften the optimality requirement to having a preference reaching a certain threshold.

Definition 3.22 (α-Dynamic Controllability) *An STPPU* P *is* α-Dynamically Controllable *(α-DC) iff there is a strategy* S *such that* $\forall P_1, P_2$ *in* $\mathrm{Proj}(P)$*:*

1. *if* $[S(P_1)]_{<x} = [S(P_2)]_{<x}$ *then* $[S(P_1)]_x = [S(P_2)]_x$*;*

2. $S(P_1)$ *is a consistent complete assignment for* P_1 *and* $S(P_2)$ *is a consistent complete assignment for* P_2*;*

3. $\mathrm{pref}(S(P_1))$ *is optimal in* P_1 *and* $\mathrm{pref}(S(P_2))$ *is optimal in* P_2 *if* $\mathrm{opt}(P_1) \leq \alpha$ *and* $\mathrm{opt}(P_2) \leq \alpha$*,*

4. $\mathrm{pref}(S(P_1)) \geq \alpha$ *and* $\mathrm{pref}(S(P_2)) \geq \alpha$ *if* $\mathrm{opt}(P_1) \geq \alpha$ *and* $\mathrm{opt}(P_2) \geq \alpha$*.*

In other words, an STPPU is α-DC if there is a means of extending any current partial control sequence to a complete sequence; but optimality is guaranteed only for situations with preference less or equal to α. For all other projections the resulting dynamic schedule will have preference at least α.

3.4.4 Comparing controllability notions

We will now consider the relation among the different notions of controllability we have given in Section 3.4.

In [VF99] it is shown that $SC \Rightarrow DC \Rightarrow WC$ (see Section 3.2).

We start giving a similar result that holds for the definitions of controllability with preferences. Intuitively, if there is a single control sequence that will be optimal in all situations, then clearly it can be executed dynamically, just assigning the values in the control sequence when the current time reaches them. Moreover if, whatever the final situation will be, we know we can consistently assign values to executables, just looking at the past assignments, and never having to backtrack, then it is clear that every situation has at least an optimal solution.

Theorem 3.1 *If an STPPU* P *is OSC then it is ODC and if it is ODC it is OWC.*

Proof: Let us assume that P is OSC. Then there is an execution strategy S such that $\forall P_\omega \in Proj(P)$, $S(P_\omega)$ is an optimal solution of P_ω, and $[S(P_1)]_x = [S(P_2)]_x$, $\forall P_1, P_2$ projections and for every executable timepoint x. This implies that also for any pair of projections P_1, P_2 and executable x, if $[S(P_1)]_{<x} = [S(P_2)]_{<x}$ then assigning $[S(P_1)]_x = [S(P_2)]_x$ will, at the end of any execution, give an optimal solution. Thus if P is OSC then it is also ODC and any strategy which is a witness of OSC is also a witness of ODC.

Let us now assume that P is ODC. Then there is a strategy S such that $\forall P_1, P_2$ in $Proj(P)$ and for any executable timepoint x: if $[S(P_1)]_{<x} = [S(P_2)]_{<x}$ then $[S(P_1)]_x = [S(P_2)]_x$ and $S(P_1)$ is an optimal solution of P_1 and $S(P_2)$ is an optimal solution of P_2. This clearly implies that every projection has at least an optimal solution. Thus P is OWC.

Clearly, the opposite implications do not hold in general. In fact, Figure 3.16 shows a triangular STPPU which is OWC, but is not ODC. In fact, it is easy to see, that any assignment to A and C, which is a projection of the STPPU can be consistently extended to an assignment of B. However, we will show in Section 3.7 that the STPPU is not ODC.

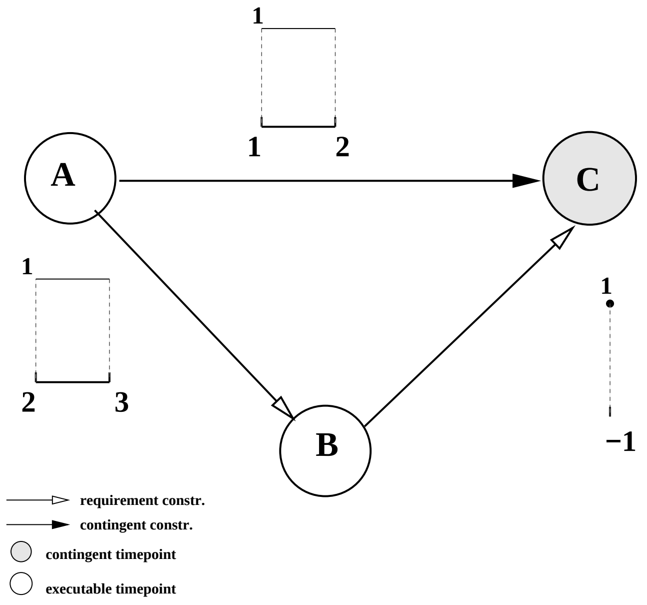


Figure 3.12: Triangular STPPU which is ODC but not OSC.

In Figure 3.12 an ODC triangular STPPU which is not OSC is shown. In the STPPU depicted, A and B are two executable time-points and C is a contingent time-point. There are only two projections, say P_1 and P_2, corresponding respectively to point 1 and point 2 in the AC interval. The optimal preference level for both is 1. In fact, $\langle A = 0, C = 1, B = 2\rangle$ is a solution of P_1 with preference 1 and $\langle A = 0, C = 2, B = 3\rangle$ is a solution of P_2 with preference 1. The STPPU is ODC. In fact, there is a dynamic strategy S that assigns to B value 2, if C occurs at 1, and value 3, if C occurs at 2 (assuming A is always assigned 0). However there is no single value for B that will be optimal (or even consistent) in both scenarios.

The following theorem shows that α-SC implies α-DC.

Theorem 3.2 *For any given preference level α, if an STPPU P is α-SC then it is α-DC.*

Proof: Assume that P is α-SC. Then there is a strategy S such that: $\forall P_\omega \in \mathtt{Proj}(P)$, $S(P_\omega)$ is a solution of P_ω such that $[S(P_1)]_x = [S(P_2)]_x$, $\forall P_1, P_2$ projections and for every executable timepoint x and $S(P_\omega)$ is an optimal solution of projection P_ω if $\mathtt{opt}(P_\omega) \leq \alpha$ and $\mathtt{pref}(S(P_z)) \geq \alpha$, for every other projection P_z. Since strategy S is such that $[S(P_1)]_x = [S(P_2)]_x$, $\forall P_1, P_2$ projections and for every executable timepoint x, then it must be so also for all pairs of projections, P_1 and P_2 such that $[S(P_1)]_{<x} = [S(P_2)]_{<x}$. Moreover, $S(P_1)$ (resp. $S(P_2)$) will be an optimal solution of P_1 (resp. P_2) if $\mathtt{opt}(P_1) \leq \alpha$, (resp. $\mathtt{opt}(P_2) \leq \alpha$), and, in any case, $\mathtt{pref}(S(P_1)) \geq \alpha$ and $\mathtt{pref}(S(P_2)) \geq \alpha$ will hold.

Again, the converse does not hold in general. As an example consider again the example shown in Figure 3.12. Such and STPPU is 1-DC but, as we have shown above, it is not 1-SC. This also shows that if we consider an STPPU P as an STPP, simply ignoring uncertainty, and find its optimal preference value $\mathtt{opt}$, then STPPU P is OSC iff it is $\mathtt{opt}$-SC and of course it is ODC iff it is $\mathtt{opt}$-DC. We will prove this formally in the following theorem.

Theorem 3.3 *Given an STPPU P with optimal preference $\mathtt{opt}$ then P is OSC (resp. ODC) iff it is $\mathtt{opt}$-SC (resp. $\mathtt{opt}$-DC).*

Proof: The result comes directly from the fact that $\forall P_i$, projection of P, $\texttt{opt}(P_i) \leq \texttt{opt}$, and there is always at least a projection, P_j, such that $\texttt{opt}(P_j) = \texttt{opt}$ (the one corresponding to an optimal solution of P).

Another useful result is that if a property holds at a given preference level, say β, then it holds also $\forall \alpha < \beta$.

Theorem 3.4 *Given an STPPU* P *and a preference level* β*, if* P *is* β*-SC (resp.* β*-DC), then it is* α*-SC (resp.* α*-DC),* $\forall \alpha < \beta$*.*

Proof: If P is β-SC then there is a strategy S such that: $\forall P_\omega \in Proj(P)$, $S(P_\omega)$ is a solution of P_ω such that $[S(P_1)]_x = [S(P_2)]_x$, $\forall P_1, P_2$ projections and for every executable timepoint x and $S(P_\omega)$ is an optimal solution of preference P_ω if $\texttt{opt}(P_\omega) \leq \beta$, and for every other projection, P_z, $\texttt{pref}(S(P_z)) \geq \beta$. But, of course, $\forall \alpha < \beta$ the set of projections that have optimal preference $\leq \alpha$ is included in that of projections having preference $\leq \beta$. Moreover, for all the other projections, P_z, $\texttt{pref}(S(P_z)) \geq \beta$ implies that $\texttt{pref}(S(P_z)) \geq \alpha$ since $\beta > \alpha$. Similarly for β-DC.

In Section 3.6 we formally prove that an STPPU is OWC iff the STPU obtained ignoring the preference functions is WC.

In Figure 3.13 we summarizes the results given in this section.

$$\begin{tikzcd}
\text{OSC} \arrow[r, leftrightarrow] \arrow[d] & \texttt{opt}\text{-SC} \arrow[r] \arrow[d] & \alpha\text{-SC} \arrow[r] \arrow[d] & \alpha_{min}\text{-SC} \arrow[r] \arrow[d] & 0\text{-SC} \arrow[r] \arrow[d] & \text{SC} \arrow[d] & & \\
\text{ODC} \arrow[r, leftrightarrow] & \texttt{opt}\text{-DC} \arrow[r] & \alpha\text{-DC} \arrow[r] & \alpha_{min}\text{-DC} \arrow[r] & 0\text{-DC} \arrow[r] & \text{DC} \arrow[r] & \text{WC} \arrow[r, leftrightarrow] & \text{OWC}
\end{tikzcd}$$

Figure 3.13: Comparison of controllability notions. α_{min} is the smallest preference over any constraint: $\texttt{opt} \geq \alpha \geq \alpha_{min}$.

Notice that chopping the STPPU at the minimum preference level $\alpha_{\min}$ or at preference level 0 gives the same STPU, that is, the STPU obtained ignoring the preference functions on the given STPPU.

3.5 Checking Optimal Strong Controllability

All the algorithms we will present in what follows rely on the following tractability assumptions, inherited from STPPs: (1)the underlying semiring is the fuzzy semiring S_{FCSP} defined in Chapter 2 Section 2.2.2, and (2) the preference functions are semi-convex.

We start considering the most restrictive notion of controllability: Strong Controllability. As it can be easily seen looking at the definitions of the previous section, it requires the existence of a fixed assignment for all the events under the agent's control that will be optimal in any scenario (OSC), or in all scenarios such that the optimal preference of the corresponding projection is smaller than or equal to a certain threshold α (for α-SC).

Strong Controllability is a notion that fits when off-line scheduling is allowed, in the sense that the fixed optimal control sequence is computed before execution begins. This approach is reasonable if the scheduling algorithm has no knowledge on the possible outcomes, other than the agent's preferences. Such a situation requires to find a fixed way to execute controllable events that will be consistent with any possible outcome of the uncontrollables and that will give the best possible final preference.

In this section we describe an algorithm that checks if an STPPU is OSC. If it is not OSC, the algorithm will detect this and will also return the highest preference level α such that the problem is α-SC.

The algorithm is based on a simple idea: for each preference level β it finds all the control sequences that guarantee strong controllability and a global preference $\leq \beta$ for all projections and optimality for those with optimal preference β. Then, it keeps only those control sequences that do the same for all preference levels $> \beta$.

We will call this algorithm Best-SC and its pseudocode is shown in Figure 3.14.

The algorithm takes in input an STPPU P (line 1). The first thing it does is to

Pseudocode for Best-SC

```
1. input STPPU $P$;
2. compute $\alpha_{min}$;
3. STPU $Q^{\alpha_{min}} \leftarrow$ Chop($P$,$\alpha_{min}$);
4. if (Strong-Controllability ($Q^{\alpha_{min}}$) inconsistent) write "not $\alpha_{min}$-SC" and stop;
5. else {
6.   STP $P^{\alpha_{min}} \leftarrow$ Strong-Controllability ($Q^{\alpha_{min}}$);
7.   preference $\beta \leftarrow \alpha_{min} + 1$;
8.   bool OSC←false, bool $\alpha$-SC←false;
9.   do {
10.     STPU $Q^{\beta} \leftarrow$ Chop($P$,$\beta$);
11.     if (PC-2($Q^{\beta}$) inconsistent) OSC←true;
12.     else {
13.      if (Strong-Controllability(PC-2($Q^{\beta}$)) inconsistent) $\alpha$-SC← true;
14.      else {
15.        STP $P^{\beta} \leftarrow P^{\beta-1} \bigcap$ Strong-Controllability(PC-2($Q^{\beta}$)) ;
16.        if ($P^{\beta}$ inconsistent) { $\alpha$-SC $\leftarrow$ true };
17.        else { $\beta \leftarrow \beta + 1$ };
18.          }
19.        }
20.     }while (OSC=false and $\alpha$-SC=false);
21.  if (OSC=true) write "P is OSC";
22.  if ($\alpha$-SC=true) write "P is" $(\beta - 1)$ "-SC";
23.  return $P^{\beta-1}$;
24.   };
```

Figure 3.14: Algorithm that tests if an STPPU is OSC and finds the highest α such that STPPU P is α-SC, returning the minimal network.

compute the lowest preference α_{min} [4]. The computation of α_{min} (line 2) is easy if we consider the semi-convexity of the preference functions. In general, if the preference function is in analytical form, it can be computed using usual calculus techniques. It would be, however, reasonable, to assume that α_{min} is given, that is, it is computed during the preference elicitation phase.

In (line 3) the STPU obtained from P by chopping it at preference level α_{min} is considered. Such STPU is obtained applying function Chop(STPPU G, preference β) with G=P and $\beta = \alpha_{min}$ [5]. In general, the result of chopping an STPPU P at a given preference level β is the STPU, Q^{β}, (i.e. a temporal problem with uncertainty but not preferences) defined as follows:

- Q^{β} has the same variables with the same domains as in P;
- for every soft temporal constraint (requirement or contingent) in P on variables X_i, and X_j, say $c = \langle I, f\rangle$, there is in Q^{β} a simple temporal constraint on the same variables defined as $\{x \in I | f(x) \geq \beta\}$.

Notice that the semi-convexity of the preference functions guarantees that set $\{x \in I | f(x) \geq \beta\}$ forms a unique interval. The intervals in Q^{β} contain all the durations of requirement and contingent events that have a local preference of at least β.

Once STPU $Q^{\alpha_{min}}$ is obtained, the algorithm checks if it is strongly controllable. If the STP obtained applying algorithm Strong-Controllability [VF99] to STPU $Q^{\alpha_{min}}$ is not consistent, then there is no hope for any higher preference and the algorithm can stop (line 4), saying that the STPPU is not α-SC $\forall\alpha \geq 0$ and thus is not OSC as well.

[4]In general the minimum preference, α_{min}, will be assigned to an element of a contingent constraint, since it is reasonable to leave in the requirement constraints only elements with a preference higher than that of the worst possible uncontrollable outcome.

[5]Notice that function Chop can be applied to both STPPs and STPPUs: in the first case the output is an STP, while in the latter case an STPU.

If, instead, no inconsistency is found, Best-SC stores the resulting STP (lines 5-6) and proceeds moving to the next preference level $\alpha_{min} + 1$ [6] (line 7).

At each preference level the following three steps are performed (lines 9-21):

- Chop STPPU P and obtain STPU Q^{β} (line 10);
- Apply path consistency to Q^{β} considering it as an STP: PC-2(Q^{β});
- Apply strong controllability to STPU PC-2(Q^{β}).

Let us now consider the last two steps in detail.

Applying path consistency (e.g. running algorithm PC-2 [DMP91]) to STPU Q^{β} means considering it as an STP, that is, treating contingent constraints as requirement constraints. Algorithm PC-2 returns the minimal network leaving in the intervals only values that are contained in at least one solution. This allows us to identify all the situations, ω, that identify contingent durations that locally have preference $\geq \beta$ and that are consistent with at least a control sequence of elements in Q^{β}. In other words, applying path consistency to Q^{β} leaves in the contingent intervals only durations that belong to situations such that their projections have optimal value at least β. If such a test gives an inconsistency, it means that the given STPU, seen as an STP, has no solution. This means that all the projections corresponding to a scenario of STPPU P have optimal preference $< \beta$ (line 11).

The third and last step applies Strong-Controllability to STPU PC-2(Q^{β}), reintroducing the information on uncertainty on the contingent constraints. Such algorithm rewrites all the contingent constraints in terms of constraints on only executable time-points. If the STPU is strongly controllable, Strong-Controllability will leave in the requirement intervals only elements that identify control sequences that are consistent with any possible situation. In our case applying Strong-Controllability to PC-2(Q^{β}) will find, if any, all the control sequences of

[6] By writing $\alpha_{min} + 1$ we mean that the next preference level higher than α_{min} is defined in terms of the granularity of the preferences in the [0,1] interval.

PC-2(Q^{β}) that are consistent with any possible situation in PC-2(Q^{β}). If STPU PC-2(Q^{β}) is strongly controllable, some of the control sequences found might not be optimal for scenarios with optimal preference lower than β. In order to keep only those control sequences that guarantee optimal strong controllability for all preference levels up to β, the STP obtained by Strong-Controllability(PC-2(Q^{β})) is intersected with the corresponding one found in the previous step (at preference level $\beta - 1$), that is $P^{\beta-1}$ (lines 15-16). By intersecting the two problems we mean considering each constraint, say c, in both problems, that is two intervals, and defining the new constraint as the intersection of the intervals. If the intersection becomes empty on some constraint or the STP obtained is inconsistent, we can conclude that there is no control sequence that will guarantee optimal strong controllability for preference level β and for all preferences $< \beta$ (line 16). If, instead, the STP obtained is consistent, algorithm Best-SC considers the next preference level, $\beta + 1$ [7], and performs the three steps again (line 17-21).

The algorithm will stop if one of the following events occurs at a given preference level γ:

- **Event 1.** Strong-Controllability($Q^{\alpha_{min}}$) is inconsistent (line 4), thus the STPPU is not α-DC $\forall \alpha \geq 0$ and, thus, it is not OSC;

- **Event 2.** PC-2(Q^{γ}) returns an inconsistency: since this STP has no solution there is no schedule of the original STPPU with preference $\geq \gamma$. Assuming this to be the first preference level γ at which such event occurs we can conclude that the problem is OSC and that the best preference level is $\gamma - 1$ (lines 11, 20, 21).

- **Event 3.** PC-2(Q^{γ}) is consistent but it is not strongly controllable. This means that there are scenarios such that their projection has optimal solutions with preference $\geq \gamma$ but there is no way to assign a time to the executable time-points such the schedule obtained will have preference at least

[7] By writing $\beta + 1$ we mean that the next preference level higher than β is defined in terms of the granularity of the preferences, which is assumed to be known beforehand.

γ in all those scenarios. However, since we are assuming that this is the first preference at which such event occurs we can conclude that the STPPU is not OSC but it is $(\gamma - 1)$-SC (lines 13, 20, 22).

- **Event 4.** PC-2(Q^γ) is strongly controllable, however the intersection of the STP obtained by Strong-Controllability(PC-2(Q^γ)) with the STP obtained at the previous preference level, $P^{\gamma-1}$, is empty. Evidently, the control sequences that guarantee optimal strong controllability up to $\gamma - 1$ don't work for γ. Assuming γ to be the first preference in the order at which such event occurs we can conclude that STPPU P is $(\gamma - 1)$-SC (lines 16, 20, 22).

Notice that in all but the first case, the algorithm returns STP $P^{\gamma-1}$ (line 23), that is, a simple temporal problem with no uncertainty and no preferences defined only on executables such that each of its solutions is a control sequence that guarantees OSC or $(\gamma - 1)$-SC depending on the event that caused the execution to stop.

Example 3.3 Before formally proving the correctness of algorithm Best-SC, let us consider its execution on the STPPU of Example 3.2. For simplicity we focus on the triangular sub-problem on variables SC, SA, and ES. In such an example, $\alpha_{min} = 0.5$. In Table 3.1 we show the STPUs Q^α obtained chopping the problem at each preference level $\alpha = 0.5, \dots, 1$ (line 10 in Fig. 3.14). In Table 3.2 the result of applying path consistency (line 11) to each of the STPUs shown in Table 3.1 is shown. As it can be seen, all of the STPUs are consistent. Finally in Table 3.3 we show the STPs defined only on executable variables SC and SA that are obtained applying algorithm Strong-Controllability to the STPUs in Table 3.2. By looking at Tables 3.2 and 3.3 it is easy to deduce that the algorithm will stop at preference level 1 because of the occurrence of Event 4. In fact, by looking more carefully at Table 3.3, we can see that STP $P^{0.9}$, obtained intersecting all the STPs Strong-Controllability(PC-2(Q^α)) with $\alpha = 0.5, \dots, 0.9$, consists of interval $[4, 4]$ on the constraint $SC \to SA$, while Strong-Controllability(PC-2(Q^1)) consist of interval $[3, 3]$ on the same constraint. Obviously intersecting the two gives an

Table 3.1: In this table each row corresponds to a preference level α and represents the intervals of STPU Q^{α} obtained by chopping the STPPU in Fig. 3.11 at level α.

STPU	$(SC \rightarrow EC)$	$(SC \rightarrow SA)$	$(SA \rightarrow EC)$
$Q^{0.5}$	$[1, 8]$	$[1, 5]$	$[-6, 4]$
$Q^{0.6}$	$[1, 7]$	$[1, 5]$	$[-6, 4]$
$Q^{0.7}$	$[1, 6]$	$[1, 5]$	$[-5, 2]$
$Q^{0.8}$	$[1, 5]$	$[1, 5]$	$[-4, 1]$
$Q^{0.9}$	$[1, 4]$	$[1, 5]$	$[-3, 0]$
Q^{1}	$[1, 2]$	$[1, 3]$	$[-2, -1]$

inconsistency, causing the condition in line 16 of Fig. 3.14 to be satisfied. The result of executing Best-SC on the example depicted in Fig.3.11 is that it is 0.9-SC. Let us now see why this is correct. Without loss of generality we can assume that SC is assigned value 0. From the last line of Table 3.3 we can see that the only value that can be assigned to SA that is optimal with both scenarios that have optimal preference 1, that is the scenario in which EC is assigned 1 and that in which it is assigned 2, is 3. However assigning 3 to SA is not optimal if EC happens at 6, since this scenario has optimal preference value 0.7, e.g if SA is assigned 5, while in this case it would have a global preference 0.6(given in constraint $SA \rightarrow EC$)[8].

We will now prove that algorithm Best-SC correctly identifies if an STPPU is OSC or not and it finds the highest preference level at which it is α-SC. Let us remind the reader that we are assuming the fuzzy semiring as preference structure and also semi-convex preference functions.

First we prove that the algorithm terminates.

[8]Let us remind the reader that in the fuzzy semiring context the global preference of any assignment is computed taking the minimum preference assigned to any of its projections.

Table 3.2: In this table each row corresponds to a preference level α and represents the intervals of STPU PC-2(Q^{α}) obtained applying path consistency to the STPUs in Table 3.1.

STPU	$(SC \rightarrow EC)$	$(SC \rightarrow SA)$	$(SA \rightarrow EC)$
PC-2$(Q^{0.5})$	$[1,8]$	$[1,5]$	$[-4,4]$
PC-2$(Q^{0.6})$	$[1,7]$	$[1,5]$	$[-4,4]$
PC-2$(Q^{0.7})$	$[1,6]$	$[1,5]$	$[-4,2]$
PC-2$(Q^{0.8})$	$[1,5]$	$[1,5]$	$[-4,1]$
PC-2$(Q^{0.9})$	$[1,4]$	$[1,5]$	$[-3,0]$
PC-2(Q^{1})	$[1,2]$	$[2,3]$	$[-2,-1]$

Theorem 3.5 *Given any STPPU P, the execution of algorithm Best-SC over P will terminate.*

Proof: Consider STPPU P as an STPP and take its optimal preference value opt, that is, the highest preference assigned to any of its solutions. By definition, Q^{opt+1} is not path consistent. This means that if the algorithm reaches level $opt+1$ the condition in line 11 will be satisfied and the execution will halt. By looking at lines 9-20 we can see that either one of the events that cause the execution to terminate occur or the preference level is incremented in line 17. Since there is a finite number of preference levels, this allows us to conclude that the algorithm will terminate in a finite number of steps.

Now we will prove that Best-DC is a sound and complete algorithm for checking if an STPPU is OSC and for finding the highest preference level at which it is α-SC. First we need the following two lemmas.

Lemma 3.1 *Consider an STPPU P and preference level γ, and consider the STPU Q^{γ} obtained by chopping P at γ. Then the solutions of STP T^{γ}= Strong-Controllability(PC-2(Q^{γ})) are all and the only control sequences that are consistent with all scenarios of PC-2(Q^{γ}) and optimal for any scenario of PC-2(Q^{γ}) that has optimal preference γ.*

Table 3.3: In this table each row corresponds to a preference level α and represents the intervals of STP Strong-Controllability PC-2(Q^{α}) obtained applying the strong controllability check to the STPUs in Table 3.2.

STP	$(SC \rightarrow SA)$
Strong-Controllability(PC-2($Q^{0.5}$))	$[4,5]$
Strong-Controllability(PC-2($Q^{0.6}$))	$[3,5]$
Strong-Controllability(PC-2($Q^{0.7}$))	$[4,5]$
Strong-Controllability(PC-2($Q^{0.8}$))	$[4,5]$
Strong-Controllability(PC-2($Q^{0.9}$))	$[4,4]$
Strong-Controllability(PC-2(Q^{1}))	$[3,3]$

Proof: Consider the STPU Q^{γ}. It is obtained by considering elements of both contingent and requirement intervals of P that are mapped into preferences greater than or equal to γ. Consider now any scenario of Q^{γ}. If it is consistent with some assignment to executables in Q^{γ}, then its projection has optimal preference at least γ and the scenario is part of one or more solutions of Q^{γ}. On the other side, if it is only consistent with assignments to executables with preference $< \gamma$ then its projection will have an optimal preference $< \gamma$ and the scenario will not belong to any solution of Q^{γ}. This means that the scenarios of STPU PC-2(Q^{γ}) are all and only the scenarios of P such that their projections have optimal value at least γ. Now, applying Strong-Controllability to STPU PC-2(Q^{γ}) means checking if their is at least a unique assignment in PC-2(Q^{γ}) for the executables that will be consistent with all the scenarios of PC-2(Q^{γ}). We know that if such condition holds then algorithm Strong-Controllability returns an STP defined only on the executable time points, such that all its solutions are consistent with any scenario of PC-2(Q^{γ}). This means that all the assignments in STP T^{γ}=Strong-Controllability(PC-2(Q^{γ})) are guaranteed to be consistent with any scenario of PC-2(Q^{γ}), but they will be optimal only for those scenarios that identify projections (STPPs) with optimal preference exactly γ. For all other scenarios of

PC-2(Q^γ), the preference of the complete schedule will be $\geq \alpha$. If instead the result of applying Strong-Controllability to STPU PC-2(Q^γ) is that the STPU is not SC, then there is no way to assign values to the executables in way such that it will be consistent with any scenario of PC-2(Q^γ).

Lemma 3.2 *Consider an STPPU P, and all preference levels from α_{min} to γ, and assume that the corresponding STPs, $T^{\alpha_{min}}, \ldots, T^\gamma$ obtained by chopping P at preference levels $\alpha_{min}, \ldots, \gamma$, and enforcing strong controllability are consistent. Then the solutions of STP $P^\gamma = \bigcap_{i=\alpha_{min},\ldots,\gamma} T^i$ obtained intersecting the corresponding intervals of all the T^i are all and the only assignments to executables of P consistent and optimal for all scenarios with optimal preference $\leq \gamma$.*

Proof: Clearly any solution of P^γ satisfies the condition. Consider an assignment a of to executables of P that is not a solution of P^γ. Then a cannot be a solution of all STPs T^i. This meas that there is at least a preference level β, $\alpha_{min} \leq \beta \leq \gamma$, such that a is not a solution of T^β. This means that there must be at least a scenario s such that the optimal preference of its projection, $\mathtt{Proj}(s)$, is β and such that a does not extend to an optimal solution of $\mathtt{Proj}(s)$. Hence a is not optimal for at least a scenario of P with optimal preference $\beta \leq \gamma$.

Theorem 3.6 *If the execution of algorithm* Best-SC *on STPPU P stops due to the occurrence of Event 1 (line 4), then P is not α-SC $\forall \alpha \geq 0$.*

Proof: Consider first preferences levels smaller than α_{min}. Since α_{min} is the minimum preference on any constraint, chopping P at any preference level between 0 and α_{min} gives the same STPU. The occurrence of Event 1 implies that $Q^{\alpha_{min}}$ is not strongly controllable. So it must be the same for all preferences lower than or equal to α_{min}.

Let us now consider preferences greater than α_{min}. $Q^{\alpha_{min}}$ contains the same set of possible scenarios of P. On the other hand, all the scenarios of $Q^{\alpha_{min}}$ identify projections that have optimal solutions with preference greater than or equal to α_{min}. Saying that $Q^{\alpha_{min}}$ is not strongly controllable means that there is no assignment to the executables that is consistent with all the scenarios of $Q^{\alpha_{min}}$ and optimal for those with optimal preference exactly α_{min}. By definition both OSC and α-SC, with $\alpha \geq \alpha_{min}$, require the existence of such an assignment for scenarios with optimal preference α_{min}. We can thus conclude that P is not OSC and is not α-SC for any $\alpha \geq 0$.

Theorem 3.7 *If the execution of algorithm Best-SC on STPPU P stops due to the occurrence of Event 2 (line 11) at preference level $opt + 1$, then P is OSC and any solution of STP P^{opt} returned by the algorithm is optimal in any scenario of P.*

Proof: If the condition of line 11 is satisfied by STPU Q^{opt+1}, it means that there are no schedules of P that have preference $opt + 1$. However, the same condition was not satisfied at the previous preference level, opt, which means that there are schedules with preference opt. This allows us to conclude that opt is the optimal preference for STPPU P seen as an STPP, that is, the highest preference assigned to any schedule of P. Since we are assuming that line 11 is executed by Best-SC at level $opt + 1$, the conditions in lines 13 and 14 must have not been satisfied at preference opt. This means that at level opt the STP P^{opt} (line 15) is consistent. By looking at line 15, we can see that STP P^{opt} satisfies the hypothesis of Lemma 3.2 at preference level opt. This allows us to conclude that any solution of P^{opt} is optimal in any scenario of P.

Theorem 3.8 *If the execution of algorithm Best-SC on STPPU P stops due to the occurrence of Event 3 (line 13) or Event 4 (line 16) at preference level γ then P is $(\gamma-1)$-SC and any solution of STP $P^{\gamma-1}$ returned by the algorithm is optimal in any scenario of P that has a projection with optimal preference $\leq \gamma - 1$.*

Proof: If Event 3 or Event 4 occurs the condition in line 13 must have not been satisfied. This allows us to conclude that there are schedules of P with preference γ. If Event 3 occurs, then the condition in line 11 must be satisfied. The STPU obtained by chopping P at preference level γ and applying path consistency is not strongly controllable. We can thus conclude that P is not OSC. However since the algorithm had executed line 11 at preference level γ, at $\gamma - 1$ it must have reached line 18. By looking at line 15 we can see that STP $P^{\gamma-1}$ satisfies the hypothesis of Lemma 3.2 at preference level $\gamma - 1$. This allows us to conclude that any solution of $P^{\gamma-1}$ is optimal in any scenario of P that has a projection with optimal preference $\leq \gamma - 1$.

If instead Event 4 occurs then it is P^{γ} to be inconsistent which means that there is no common assignment to executables that is optimal for all scenarios with preference $\leq$ $gamma$ and at the same time for those with preference γ. However since the execution has reached line 16 at preference level γ, again we can assume it had successfully completed the loop at preference $\gamma - 1$. Again, by looking at line 15, we can see that STP $P^{\gamma-1}$ satisfies the hypothesis of Lemma 3.2 at preference level $\gamma - 1$ and thus any of its solution is optimal in any scenario of P that has a projection with optimal preference $\leq \gamma - 1$.

We conclude this section considering the complexity of Best-SC.

Theorem 3.9 *The complexity of determining OSC or the highest preference level of α-SC of an STPPU with* n *variables,* l *preference levels and intervals of maximum size* r *is* $O(n^3lr)$.

Proof: The complexity of algorithm Best-SC depends on the granularity of the preferences. Assuming we have at most l different preference levels and at most r elements in the largest interval we can conclude that the complexity of Best-SC is bounded by that of applying l times Strong-Controllability, that is $\mathcal{O}(l \times n^3 \times r)$. This complexity is polynomial if r and k are assumed to be bounded (as it seems reasonable in most of the possible application domains).

Notice that we cannot use a binary search since the correctness of the procedure is based on the intersection of the result obtained at a given preference level, γ, with those obtained at all preference levels $< \gamma$.

The above theorem allows us to conclude that the cost of adding preferences, and thus a considerable expressiveness power, is low. In fact, the complexity w.r.t.

the case without preferences is still in P and it has grown only of a factor equal to the number of preference levels.

3.6 Checking Optimal Weak Controllability

In the following Theorem we show that if an STPPU is OWC then the STPU Q obtained from P by simply ignoring the preferences is WC.

Theorem 3.10 *A STPPU P is Optimally Weakly Controllable (OWC) if the STPU Q, obtained by simply ignoring the preference functions on all the constraints, is Weakly Controllable (WC). The converse, in general does not hold.*

Proof: "$\Rightarrow$". If P is OWC, then $\forall \omega \in \Omega$ there exists a control sequence δ such that schedule $T = (\delta_T, \omega)$ is consistent and optimal for projection $\mathrm{Proj}(\omega)$. At this point, it is enough to notice that if a schedule is "consistent" in the STPPU sense it means that the preference assigned to each duration is > 0, and this implies consistency in the STPU sense.

"$\not\Leftarrow$". If Q is WC, then every projection $\mathrm{Proj}(\omega)$ with $\omega \in \Omega$ is a consistent STP having at least one solution. Let us assume now that preferences are given in such a way that there is a projection $\mathrm{Proj}(\omega)$ that has, as consistent solutions (in the STPU sense), only assignments with global preference 0. This means that there is no consistent solution for projection $\mathrm{Proj}(\omega)$ in the STPPU sense (i.e. a solution with preference > 0).

The previous theorem takes in account the possibility of mapping some elements of the intervals into the preference level 0. However if all the elements are mapped into strictly positive preferences, then the correspondence becomes bijective, as stated in the following theorem.

Theorem 3.11 *A STPPU P, with strictly positive preference functions, is Optimally Weakly Controllable (OWC) iff the STPU Q, obtained by simply ignoring the preference functions on all the constraints, is Weakly Controllable (WC).*

Proof: "$\Rightarrow$". The proof is exactly the same as for Theorem 3.10.

"$\Leftarrow$". If Q is WC, then every projection $\mathtt{Proj}(\omega)$ with $\omega \in \Omega$ is a consistent STP having at least one solution. Let us assume now that only strictly positive preferences are given. Then for each projection we'll have a consistent solution with preference > 0. Now it is enough to say that either that it is the only solution, and hence it also optimal, or there are many solutions among which there is always one that has preference greater than or equal to the preferences of the others. In short: if there is a solution, then there must be an optimal solution.

Notice that the condition on preference functions used in this theorem is sufficient but not necessary. There could be a problem where some solutions of some projections are associated with preference 0 but the same projections have other solutions mapped into a strictly positive projection as well. In this situation, we would still have a bijective correspondence between the OWC of STPPU P and the WC of STPU Q.

This theorem allows us to conclude that, to check OWC, it is enough to apply algorithm Weak-Controllability as proposed in [VG96] and described in Section 3.2. Moreover, if we are given a scenario ω then we can find an optimal solution of its projection, STPP $\mathtt{Proj}(\omega)$, by simply using algorithm Chop-solver described in Chapter 2 Section 2.4.2.

Let us now consider Example 3.2 again. We have shown in Section 3.5 that the STPU obtained by chopping the STPPU shown in Fig. 3.11 at preference level α_{min} is strongly controllable. Since strong controllability implies weak controllability (see Section 3.2), we can conclude that the STPU is weakly controllable. Also, since the minimal preference is 0.5, we can conclude, by using Theorem 3.11, that the STPPU in Fig. 3.11 depicting Example 3.2 is Optimally Weakly Controllable.

3.7 Checking Optimal Dynamic Controllability

Optimal Dynamic Controllability (ODC) is the most interesting and useful property in practice.

In this section we describe an algorithm that tests whether an STPPU is ODC. We also give an execution algorithm that assigns values to executable timepoints dynamically while observing the current situation (i.e., the values of the contingent timepoints that have occurred).

3.7.1 Testing optimal dynamic controllability and α-dynamic controllability

In this section we define a necessary and sufficient condition for ODC, which is defined on the intervals of the STPPU. We will then propose an algorithm which tests such a condition, and we will show that it is a sound and complete algorithm for testing ODC. Moreover, in a similar way as algorithm Best-SC presented in Section 3.5, the algorithm we will propose, will, if the STPPU is not ODC, identify the highest preference level at which the STPPU is α-SC.

Our first claim is that, given an STPPU, the dynamic controllability of the STPUs obtained by chopping the STPPU and applying PC-2 at every preference level is a necessary but not sufficient condition for the optimal dynamic controllability of the STPPU. Figure 3.15 shows an example of an STPPU which is not ODC and such that there is a preference level at which chopping the STPPU gives an STPU that is not DC.

Let us call P the STPPU in Figure 3.15. First notice that the STPU obtained from P ignoring the preferences is an instance of the Precede Case (see Section 3.2), and so will be any STPU obtained by chopping P at preference levels higher than 0.8. If, for example, we chop the problem at preference level 0.9, we get interval $[0,4] = [p^{0.9}, q^{0.9}]$ on AB, $[3,6] = [x^{0.9}, y^{0.9}]$ on AC and $[2,4] = [u^{0.9}, v^{0.9}]$ on BC. The STPU obtained is path consistent and it contains all the situations (i.e., the elements of $[x, y]$) that have preference at least 0.9. Notice that it is not

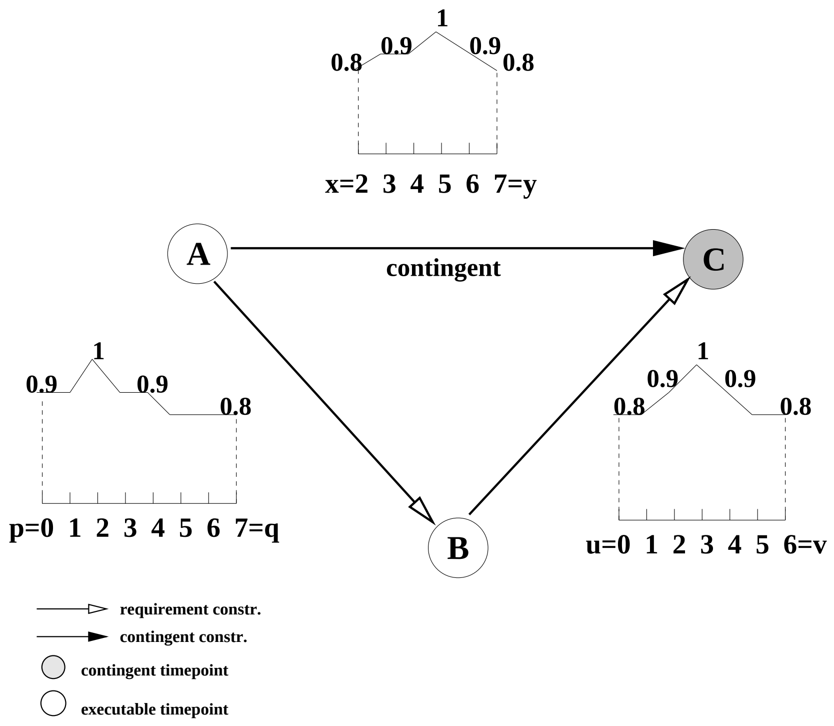


Figure 3.15: A triangular STPPU that is not DC if chopped at level 0.9.

DC since $y^{0.9} - v^{0.9} = 6 - 4 = 2 > x^{0.9} - u^{0.9} = 3 - 2 = 1$ (see Precede Case). Thus the STPU obtained by chopping P at level $\alpha = 0.9$ is not DC.

P is not ODC. In fact, we know that B must precede or happen simultaneously with C. Assume A is executed at time 0. For example, C might occur at time 5 with a preference = 1. The only choice for B which gives a solution with preference equal to 1 (i.e. an optimal solution) is B=2. However, this turns out not to be optimal if C happens at time 3 with preference =0.9, since then the projection on $[u, v]$ is 1 which is mapped into preference 0.8[9]. Summarizing, P is not ODC, since there is no way to assign a value to B which will be optimal both if C occurs at 3 or if it occurs at 5.

We will now prove formally that the DC of all STPUs obtained by chopping and applying PC-2 at all levels is necessary for global ODC.

Theorem 3.12 *Given an STPPU P, consider the STPU* Q^α *obtained by chopping* P *at*

[9]Since the aggregation operator of the preferences is min, the solution will have preference 0.8.

preference level α and applying **PC-2**. *If $\exists\alpha$ such that Q^α is not DC, then P is not ODC and it is not β-DC $\forall\beta \geq \alpha$.*

Proof: Assume that there is a preference level α at which chopping P gives an STPU, Q^α that is not DC. This means that there is no execution strategy S^α : $\mathrm{Proj}(Q^\alpha) \longrightarrow \mathrm{Sol}(Q^\alpha)$ such that $\forall P_1, P_2$ in $\mathrm{Proj}(Q^\alpha)$ and for any executable X, we have that if $[S(P_1)]_{<X} = [S(P_2)]_{<X}$ then $[S(P_1)]_X = [S(P_2)]_X$, $S(P_1)$ is a consistent complete assignment for P_1 and $S(P_2)$ is a consistent complete assignment for P_2. Notice that $\mathrm{Proj}(Q^\alpha) \subseteq \mathrm{Proj}(P)$ and that $\mathrm{Proj}(Q^\alpha)$ contains all the projections that have optimum preference at least α. This implies that there is no viable dynamic strategy for P, S', such that $S'(P_1)$ has optimal preference in P_1 and $S'(P_2)$ has optimal preference in P_2. The existence of such a strategy is required for all projections in the definition of ODC and for those with optimal preference $\leq \beta$ in the definition of β-DC. Thus, P is not ODC and is not β-DC $\forall\beta \geq \alpha$.

Unfortunately this condition is not sufficient, since an STPPU can still be not ODC even if at every preference level the STPU obtained after PC-2 is DC. An example is shown in Figure 3.16.

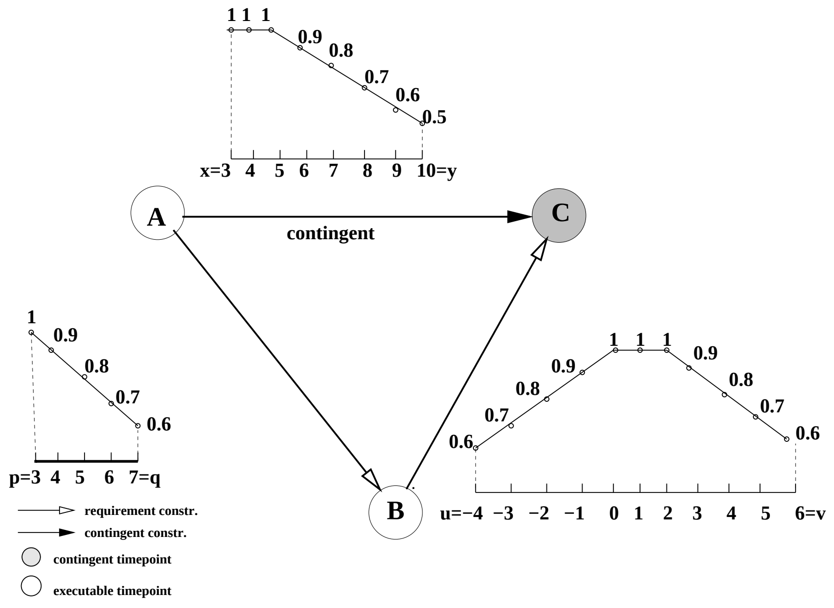


Figure 3.16: Triangular STPPU that is DC at every preference level but is not ODC.

If we apply Chop-solver to the triangular STPPU P shown in Figure 3.16, the highest preference level at which chopping the functions gives a consistent STP is 1 (in fact interval $[3,3]$ on AB, $[3,5]$ on AC and interval $[0,2]$ on BC is a consistent STP). The optimal solutions of this STPPU, regarded as an STPP, will have global preference 1.

Consider all the STPUs obtained by chopping at every preference level from the highest, 1, to the lowest 0.5. The minimum preference on any constraint in P is $\alpha_{min} = 0.5$ and, it is easy to see, that all the STPUs obtained by chopping P and applying PC-2 at all preference levels from 0.5 to 1 are DC. However, P is not ODC. In fact, the only dynamic assignment to B that is optimal with elements 3, 4 and 5 in $[x,y]$ is 3. But executing B at 3 will cause an inconsistency if C happens at 10, since 10-3=7 doesn't belong to $[u,v]$.

We will now elaborate more on this example to find a sufficient condition for ODC. Consider now the intervals on AB, $[p^{\alpha}, q^{\alpha}]$, and the waits $< C, t^{\alpha} >$ obtained applying the DC checking algorithm at preference level α. We have:

- $\alpha = 1$: $[p^1, q^1] = [3]$;
- $\alpha = 0.9$: $[p^{0.9}, q^{0.9}] = [3,4]$ with wait $< C, 3 >$;
- $\alpha = 0.8$: $[p^{0.8}, q^{0.8}] = [3,5]$ with wait $< C, 3 >$;
- $\alpha = 0.7$: $[p^{0.7}, q^{0.7}] = [3,6]$ with wait $< C, 3 >$;
- $\alpha = 0.6$: $[p^{0.6}, q^{0.6}] = [3,7]$ with wait $< C, 3 >$;
- $\alpha = 0.5$: $[p^{0.5}, q^{0.5}] = [3,7]$ with wait $< C, 4 >$;

If we look at the first and last intervals, resp., at $\alpha = 1$ and $\alpha = 0.5$, it appears clearly that there is no way to assign a value to B that at the same time induces a preference =1 on constraints AB and BC, if C occurs at 3, 4 or 5, and satisfies the wait $< C, 4 >$, ensuring consistency if C occurs at 10. This depends on the fact that the intersection of $[p^1, q^1]$, that is, $[3]$, and the sub interval of $[p^{0.5}, q^{0.5}]$ that satisfies $< C, 4 >$, that is, $[4,7]$, is empty.

We claim that the non emptiness of such an intersection, together with the DC of the STPUs obtained by chopping the problem at all preference levels is a necessary and sufficient condition for ODC.

Algorithm Best-DC

The algorithm we propose is, in some way, similar to the one shown in Section 3.5 for checking Optimal Strong Controllability. In fact, as done by Best-SC, it considers the STPUs obtained by chopping the STPPU at various preference levels. For each preference level, first it tests whether the STPU obtained considering it as an STP is path consistent. Then, it checks if the path consistent STPU obtained is dynamically controllable, using the algorithm proposed in [MMV01]. Thus, the control sequences that guarantee dynamic controllability for scenarios having different optimal preferences are found. Similarly to what has been done in the OSC case, the following step is to select only those sequences that satisfy the dynamic controllability requirement and are optimal at all preference levels.

We call this algorithm Best-DC. After describing it thoroughly, we will prove that it is a sound and complete algorithm for testing if an STPPU is ODC.

Figure 3.17 shows the pseudocode of Best-DC.

Algorithm Best-DC takes as input an STPPU P (line 1) and then computes the minimum preference, α_{min}, assigned on any constraint (line 2).

Once α_{min} is known, the STPU obtained by chopping P at α_{min} is computed (line 3). Such STPU can be seen as the STPU P with the same variables and intervals on the constraints as P but with no preferences. Such an STPU, which is denoted as $Q^{\alpha_{min}}$, is given as input to algorithm DynamicallyControllable?. If $Q^{\alpha_{min}}$ is not dynamically controllable, then P is not ODC nor γ-DC (for any $\gamma \geq \alpha_{min}$, hence for all γ), as shown in Theorem 3.12. The algorithm detects the inconsistency and halts (line 4). If, instead, $Q^{\alpha_{min}}$ is dynamically controllable, then the STPU that is returned in output by DynamicallyControllable? is saved and denoted with $P^{\alpha_{min}}$ (line 6). Notice that such STPU is minimal, in the sense that in the intervals there are only elements belonging to at least a dynamic schedule

Pseudocode for Best-DC

```
1. input STPPU P;
2. compute $\alpha_{min}$;
3. STPU $Q^{\alpha_{min}} \leftarrow$ Chop(P,$\alpha_{min}$);
4. if (DynamicallyControllable?($Q^{\alpha_{min}}$) inconsistent) write "not $\alpha_{min}$-DC" and stop;
5. else {
6.   STP $P^{\alpha_{min}} \leftarrow$ DynamicallyControllable?($Q^{\alpha_{min}}$);
7.   preference $\beta \leftarrow \alpha_{min} + 1$;
8.   bool ODC $\leftarrow$ false, bool $\alpha$-DC $\leftarrow$ false;
9.   do {
10.     STPU $Q^{\beta} \leftarrow$ Chop(P,$\beta$);
11.     if (PC-2($Q^{\beta}$) inconsistent) ODC $\leftarrow$ true;
12.     else {
13.      if (DynamicallyControllable?(PC-2($Q^{\beta}$)) inconsistent) $\alpha$-DC $\leftarrow$ true;
14.      else {
15.        STPU $T^{\beta} \leftarrow$ DynamicallyControllable?(PC-2($Q^{\beta}$));
16.        STPU $P^{\beta} \leftarrow$Merge($P^{\beta-1}$, $T^{\beta}$);
17.        if ($P^{\beta}$ inconsistent ) { $\alpha$-DC $\leftarrow$ true }
18.        else {
19.          $\beta \leftarrow \beta + 1$;
20.          };
21.        };
22.      };
23.   }while (ODC=false and $\alpha$-DC=false);
24.  if (ODC=true) write "P is ODC";
25.  if ($\alpha$-DC=true) write "P is" $(\beta - 1)$ "-DC";
26.  return STPPU $F^{\beta-1} \leftarrow$ resulting_STPPU(P,$P^{\beta-1}$);
27.  };
```

Figure 3.17: Algorithm that tests if an STPPU is ODC and, if not, finds the highest γ such that STPPU P is γ-DC returning the minimal network.

[MMV01]. In addition, since we have preferences, the elements of the requirement intervals, in addition to belonging to at least a dynamic schedule, are part of optimal schedules for scenarios which have a projection with optimal preference equal to α_{min}[10].

In line 7 the preference level is updated to the next value in the ordering to be considered (according to the given preference granularity). In line 8 two Boolean flags, ODC and α-DC are defined. Setting flag ODC to $true$ will signal that the algorithm has established that the problem is ODC, while setting flag α-DC to $true$ will signal that the algorithm has found the highest preference level at which the STPPU is α-DC.

Lines 9-25 contain the main loop of the algorithm. In short, each time the loop is executed, it chops P at the current preference level, looks if the chopping has produced a path consistent STPU (seen as an STP). If so, it checks if the path consistent version of the STPU is also dynamically controllable and, if also this test is passed, then a new STPU is created "merging" the current results with those of previous levels.

We will now go through each step in detail.

Line 10 chops P at the current preference level β. In line 11 the consistency of the STPU Q^β is tested applying algorithm PC-2. If PC-2 returns an inconsistency then we can conclude that P has no schedule with preference β (or greater). The algorithm sets the ODC flag to true, and the $while$ cycle will end (line 24).

The next step is to check if STPU PC-2(Q^β) is dynamically controllable. Notice that this is required for all preference levels up to the optimal one in order for P to be ODC and up to γ in order for P to be γ-DC (Theorem 3.12). If applying algorithm DynamicallyControllable? detects that PC-2(Q^β) is not dynamically controllable, then the algorithm sets flag α-DC to true and this will end the *while* cycle. If, instead, PC-2(Q^β) is dynamically controllable the resulting minimal STPU is saved and denoted T^β (line 15).

[10]In fact, they all have preference at least α_{min} by definition.

In line 16 a new STPU P^{β} is obtained by merging the results up to preference $\beta - 1$ with those at preference β. This is achieved by applying procedure Merge to the STPU obtained at the end of the previous $while$ iteration, $P^{\beta-1}$, and STPU T^{β}. The pseudocode for Merge is shown in Figure 3.18, and we will describe it in details shortly. First, however, we will complete the description of Best-DC.

In line 17, if the execution of Merge has failed, then the algorithm sets label $\alpha\text{-}DC$ to $true$ and this will cause the algorithm to exit the $while$ loop. If instead no inconsistency is found a new preference level is considered (line 19).

Lines 24-27 take care of the output. Lines 24 and 25 will write in output if P is ODC or, if not, the highest γ at which it is γ-DC. In line 27 the final STPPU, F, to be given in output, is obtained from STPU $P^{\beta-1}$, that is, the STPU obtained by the last iteration of the $while$ cycle which was completed with success (i.e., it had reached line 20). Function Resulting_STPPU restores all the preferences on all the intervals of $P^{\beta-1}$ by setting them as they are in P. We will show that the requirement constraints of F will contain only elements corresponding to dynamic schedules that are always optimal if the result is that P is ODC or are optimal for scenarios corresponding to projections with optimal preference $\leq \gamma$ if the result is that P is γ-DC.

Algorithm MERGE

We will now give more detail on how the results coming from different preference levels are synthesized by function Merge, shown in Figure 3.18.

The input of procedure Merge consists of two STPUs defined on the same set of variables. In describing how Merge works, we will assume it is given in input two STPUs, T^{β} and $T^{\beta+1}$, obtained by chopping an STPPU at preference levels β and $\beta + 1$ and applying, by hypothesis with success, PC-2 and DynamicallyControllable? (line 1 Figure 3.18). (In algorithm Best-DC, Merge takes as input (line 16 of Figure 3.17) STPU $P^{\beta-1}$, obtained at the previous iteration, and STPU T^{β} defined in line 15 of Figure 3.17.) The STPU which will be given in output, denoted as $P^{\beta+1}$, is initialized to T^{β} in line 2 of Figure 3.18. Merge

Pseudocode for Merge

```
1. input (STPU $T^\beta$, STPU $T^{\beta+1}$);
2. STPU $P^{\beta+1} \leftarrow T^\beta$;
3. for each constraint AB, A and B executables, in $P^{\beta+1}$
define interval $[p', q']$ and wait $t'$,
given { interval $[p^\beta, q^\beta]$, wait $t^\beta$ in $T^\beta$ }
and { interval $[p^{\beta+1}, q^{\beta+1}]$, wait $t^{\beta+1}$ in $T^{\beta+1}$ }, as follows:;

4. if $(([p^{\beta+1}, q^{\beta+1}] \cap [p^\beta, q^\beta] = \emptyset)$
or $((t^\beta \leq q^\beta)$ and $(t^{\beta+1} \leq q^{\beta+1})$ and $[t^\beta, q^\beta] \cap [t^{\beta+1}, q^{\beta+1}] = \emptyset)$
then return FAILED MERGE;
5. if $(t^\beta > q^\beta$ and $t^{\beta+1} > q^{\beta+1})$ (FOLLOW - FOLLOW)
6.    $p' \leftarrow min(p^\beta, p^{\beta+1}), q' \leftarrow max(q^\beta, q^{\beta+1}), t' \leftarrow q' + 1$;

7. if $(t^\beta \leq q^\beta$ and $t^{\beta+1} > q^{\beta+1})$ (UNORDERED - FOLLOW)
8.    $p' \leftarrow min(p^\beta, p^{\beta+1}), q' \leftarrow max(q^\beta, q^{\beta+1}), t' \leftarrow max(t^\beta, t^{\beta+1})$;

9. if $(t^\beta = p^\beta$ and $t^{\beta+1} = p^{\beta+1})$ (PRECEDE - PRECEDE)
10.   $p' \leftarrow max(p^\beta, p^{\beta+1}), q' \leftarrow min(q^\beta, q^{\beta+1}), t' \leftarrow max(t^\beta, t^{\beta+1})$;

11. else (ALL OTHER CASES)
12.   $p' \leftarrow min(p^\beta, p^{\beta+1}), q' \leftarrow min(q^\beta, q^{\beta+1}), t' \leftarrow max(t^\beta, t^{\beta+1})$;
13. output $P^{\beta+1}$.
```

Figure 3.18: Algorithm Merge.

considers every requirement constraint defined on any two executables, say A and B, respectively in T^{β} and $T^{\beta+1}$. Recall that we are assuming that algorithm DynamicallyControllable? has been applied to both STPUs. Thus, there can be some waits on the intervals. Figure 3.8 illustrates the three cases in which the interval on AB can be. If the wait expires after the upper bound of the interval (Figure 3.8 (a)), then the execution of B must follow the execution of every contingent time point (*Follow case*). If the wait coincides with the lower bound of the interval (Figure 3.8 (b)), then the execution of B must precede that of any contingent time point (*Precede case*). Finally, as shown in Figure 3.8 (c), if the wait is within the interval, then B is in the *Unordered case* with at least a contingent time point, say C. Merge considers in which case the corresponding intervals are in T^{β} and in $T^{\beta+1}$ (line 3). Such intervals are resp. indicated as $[p^{\beta}, q^{\beta}]$, with wait t^{β}, and $[p^{\beta+1}, q^{\beta+1}]$, with wait $t^{\beta+1}$. Merge obtains a new interval $[p', q']$ and new wait t', which will replace the old one in $P^{\beta+1}$. In line 4, the two intersections $[p^{\beta}, q^{\beta}] \cap [p^{\beta+1}, q^{\beta+1}]$, and $[t^{\beta}, q^{\beta}] \cap [t^{\beta+1}, q^{\beta+1}]$, are tested in order to detect their emptiness. Notice that, the emptiness of $[t^{\beta}, q^{\beta}] \cap [t^{\beta+1}, q^{\beta+1}]$, is only considered if neither one of the two constraints is in a $\texttt{Follow}$ case. In fact, in such case, we have that either $q^{\beta} < t^{\beta}$ or $q^{\beta+1} < t^{\beta+1}$ or both inequalities hold due to the notation we have adopted. However, in such case, the algorithm should stop in line 4 only if $[p^{\beta}, q^{\beta}] \cap [p^{\beta+1}, q^{\beta+1}] = \emptyset$, that is, if the intersection of the two intervals is empty.

In Theorem 3.14 and in Corollary 3.3 we will show that, if any of the intersections is empty then there is no dynamic strategy which can be optimal at the same time for projections with optimal preference β and for projections with optimal value $\beta + 1$.

In lines 5 and 6 the merging procedure for the case in which AB is in the Follow case in both T^{β} and $T^{\beta+1}$ is given. This scenario id depicted in Figure 3.19.

In both STPUs B must follow the execution of any contingent event C. This means that interval $[p^{\beta}, q^{\beta}]$ (resp. $[p^{\beta+1}, q^{\beta+1}]$), contains all and only assignments of B consistent with some past value for C mapped into a preference $\geq \beta$ (resp.

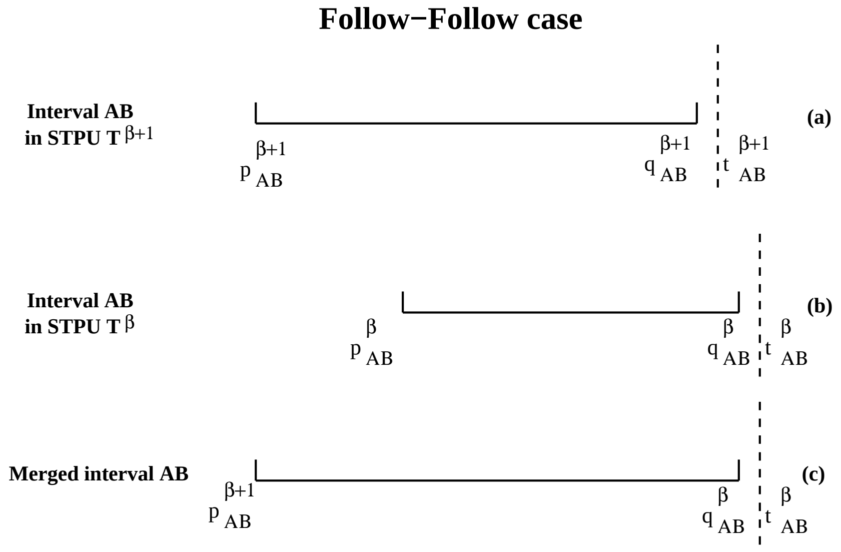


Figure 3.19: Merging two intervals, both in the *follow* case.

$\geq \beta + 1$). From an execution point of view, when C happens, this information will be propagated and such a propagation will leave in the intervals only values for B consistent with that value assigned to C (see Definition 3.23). It is thus reasonable to take the union of the intervals since the propagation will prune the elements with preference lower than that induced by the assignment to C.

In lines 7 and 8 the case in which AB is in the Unordered case in T^{β} and in a Follow case in $T^{\beta+1}$ is considered. This case is shown in Figure 3.20. In such a case, B must wait for the occurrence of some contingent time point C until t^{β}, while it must occur after all contingent time points in $T^{\beta+1}$. The merging is done so that the longest wait is respected and all the values preceding such wait are kept.

In lines 9 and 10 the case in which AB is in a Precede case in both STPUs is examined. This scenario is shown in Figure 3.21. In such case B will always occur before any contingent time point. The values in the $[p^{\beta}, q^{\beta}]$ (resp. $[p^{\beta+1}, q^{\beta+1}]$) are assignments for B that will be consistent with any future occurrence of C mapped into a preference $\geq \beta$ (resp. $\geq \beta + 1$). Clearly the intersection should be taken in order not to lower the preference if C occurs with preference $\geq \beta + 1$.

In lines 12 and 13 two scenarios are considered: when AB is in the Unordered

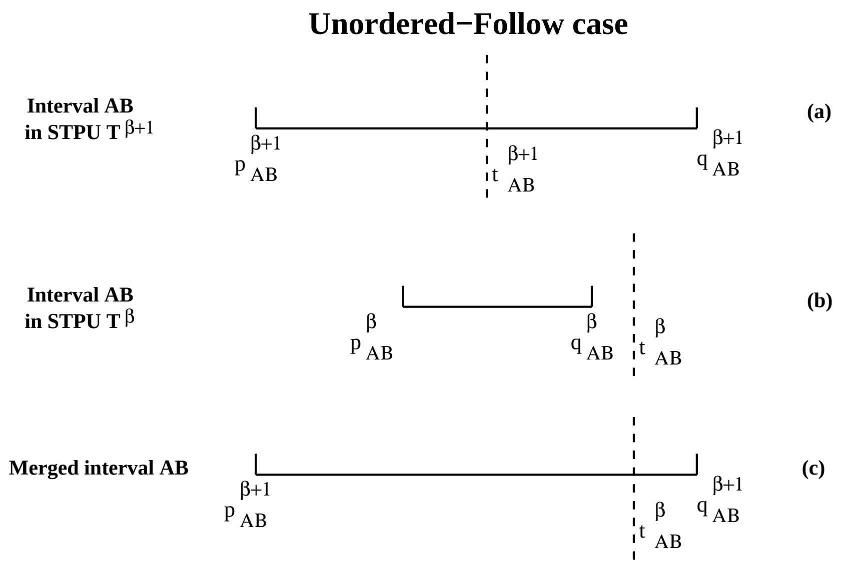


Figure 3.20: Merging an *unordered* and a *follow* interval.

case in T^{β} and in the Precede case in $T^{\beta+1}$ and when AB is in the Unordered case in both STPUs. Figure 3.22 shows the second case. Notice that, due to the semiconvexity of the preference functions it cannot be the case that:

- AB is a Follow or a Precede case in T^{β} and an Unordered case in $T^{\beta+1}$;
- AB is a Follow case in T^{β} and a Precede case in $T^{\beta+1}$;
- AB is a Precede case in T^{β} and a Follow case in $T^{\beta+1}$;

Merge takes the union of the parts of the intervals preceding the wait and the intersection of the parts following the wait. The intuition underlying this is that, any execution of B identifying an element of either $[p^{\beta}, t^{\beta}[$ or $[p^{\beta+1}, t^{\beta+1}[$ will be preceded by the execution of all the contingent time points for which it has to wait. This means that when B is executed, for any such contingent time point C, both the time at which C has been executed, say t_C, and the associated preference, say $f_{AC}(t_C)$ on constraint AC in STPPU P will be known. The propagation of this information will allow to identify those elements of $[p^{\beta}, t^{\beta}[$ (resp. $[p^{\beta+1}, t^{\beta+1}[$) that have a preference $\geq f_{AC}(t_C)$ and thus an optimal assignment for B. This means that all the elements in both interval $[p^{\beta}, t^{\beta}[$ and interval $[p^{\beta+1}, t^{\beta+1}[$ are eligible

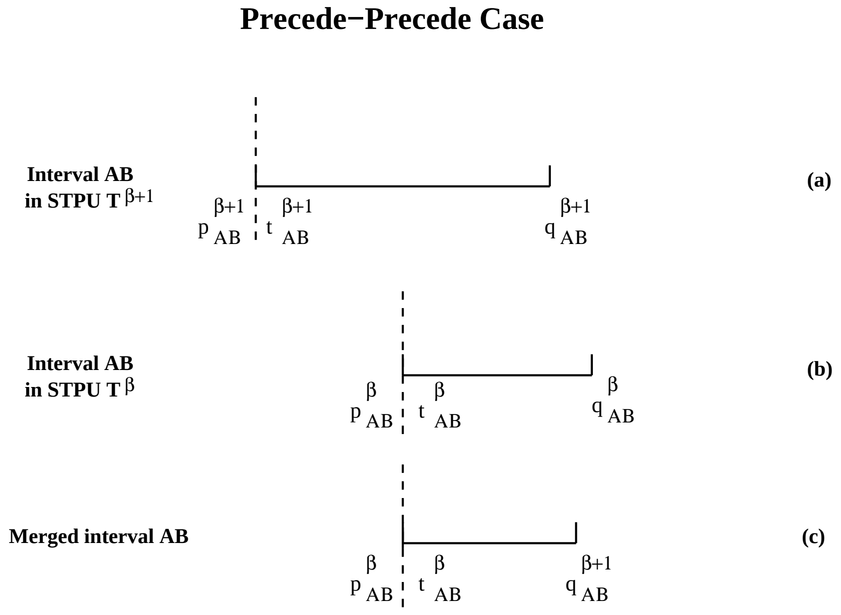


Figure 3.21: Merging two intervals in the *precede* case.

to be chosen. For example, if $f_{AC}(t_C) = \beta$ there might be values for B with preference equal to β that are optimal in this case but would not if C occurred at a time such that $f_{AC}(t_C) > \beta$. But since in any case we know when and with what preference C has occurred, it will be the propagation step that will prune non optimal choices for B. In short, leaving all elements allows more flexibility in the propagation step. If, instead we consider elements of interval $[t^\beta, q^\beta]$ we know that they identify assignments for B that can be executed regardless of when C will happen (however we know it will happen with a preference greater $\geq \beta$). This means that we must take the intersection of this part with the corresponding one, $[t^{\beta+1}, q^{\beta+1}]$, in order to guarantee consistency and optimality also when C occurs at any time with preference $= \beta + 1$. An easy way to see this, is that interval $[t^\beta, q^\beta]$ may very well contain elements that in P are mapped into preference exactly β. These elements can be optimal in scenarios in which C happens at a time associated with a preference $= \beta$ in the AC constraint, however they cannot be optimal in scenarios with C occurring at a time with preference $= \beta + 1$.

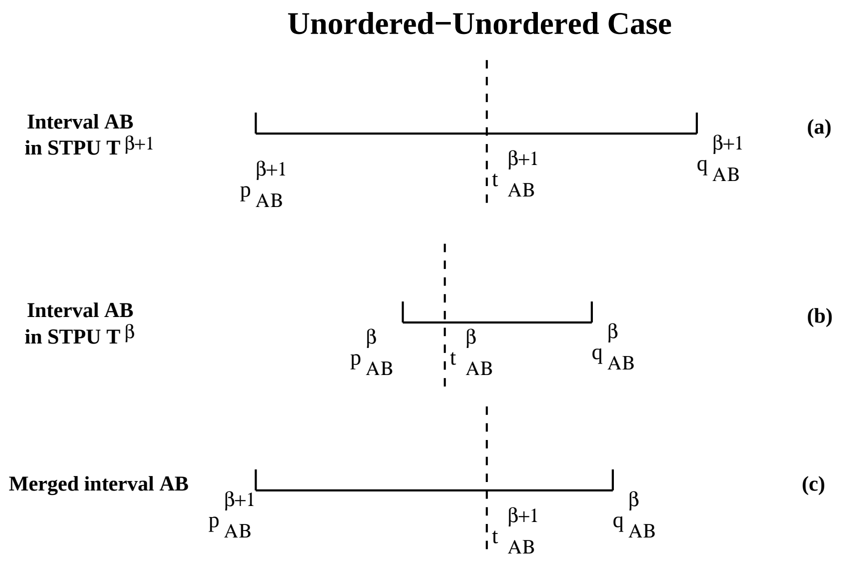


Figure 3.22: Merging two intervals in the *unordered* case.

Soundness and completeness

Having described algorithms Best-DC and Merge, we will now prove that Best-DC is a sound and complete algorithm for testing ODC and for finding the highest preference level at which an STPPU is α-DC. All the results that follow rely on the tractability assumptions requiring semiconvex preference functions and a semiring with an idempotent multiplicative operator. For clarity, we will right STPPU meaning a tractable STPPU.

We start by giving a result on algorithm DynamicallyControllable? for STPUs.

Theorem 3.13 *Consider an STPU* Q *on which DynamicallyControllable? has reported success. Consider now the viable dynamic strategy S obtained by applying the execution algorithm DC-Execute. Consider any constraint AB, where A and B are executables and the execution of A always precedes that of B. Let* t_{max} *be the longest wait B must satisfy imposed by any contingent time point C on constraint AB. Then,* $\forall Q_i \in \text{Proj}(Q)$, $[S(Q_i)]_A - [S(Q_i)]_B \leq t_{max}$.

Proof: Consider algorithm DC-Execute shown in Figure 3.10, Section 3.2. In line 5 it is stated that an executable B can be executed as soon as, at the current time, the three following conditions are all satisfied: (1) B is $live$, i.e. the current time must lie between its lower and upper bounds, (2) B is $enabled$, i.e. all the variables which must precede B have been executed, and (3) all waits on B have been satisfied. Let us translate this in terms of the execution strategy S and its projections on any ABC triangle as the one shown in Figure 3.5, having interval $[p, q]$ on AB. Let us denote the current time as T, and assume B is $live$ and $enabled$ at T. Thus, $[S(Q_i)]_A - ([S(Q_i)]_B = T) \in [p, q], \forall Q_i \in \mathrm{Proj}(Q)$. The third requirement is satisfied at T only in one of the two following scenarios: either the last contingent time point for which B had to wait has just occurred and thus B can be executed immediately, or the waits for the contingent time points, among those for which B had to wait, which have not yet occurred have expired at T. In both cases it must be that $T \leq t_{max} + [S(Q_i)_A]$. Thus, $([S(Q_i)]_B = T) - [S(Q_i)]_A \leq t_{max}$.

We will now use the result presented in Theorem 3.13 to show that if Merge, applied to two path consistent and dynamically controllable STPUs obtained by chopping a given STPPU at two different preference levels, does not fail, then there is a viable dynamic strategy for the STPU obtained by Merge which is optimal for projections having optimal preference equal to either of the chopping levels.

Theorem 3.14 *Consider STPPU P and STPUs, T^{β} and $T^{\beta+1}$, obtained by chopping P respectively at level β and $\beta+1$ and applying PC-2, without finding inconsistencies, and DynamicallyControllable? with success. Consider STPU $P^{\beta+1} = Merge(T^{\beta}, T^{\beta+1})$ and assume Merge has not failed. Then, $P^{\beta+1}$ is dynamically controllable and there is a viable dynamic strategy S such that for every projection P_i of $P^{\beta+1}$, if $\mathrm{opt}(P_i) = \beta$ or $\mathrm{opt}(P_i) = \beta + 1$, $S(P_i)$ is an optimal solution of P_i, otherwise $\mathrm{pref}(S(P_i)) \geq \beta + 1$.*

Proof: The following is a constructive proof in which a strategy S satisfying the requirements of the theorem is defined.

We first recall that procedure Merge, when applied to T^{β} and $T^{\beta+1}$, considers every requirement constraint on executables A and B, of T^{β} and only replaces it with another interval obtained from the original one in T^{β} and the corresponding one in $T^{\beta+1}$. This implies that $\mathrm{Proj}(P^{\beta+1}) = \mathrm{Proj}(T^{\beta})$.

Since T^{β} is dynamically controllable, the dynamic strategy S' produced by running algorithm DC-execute is such that $\forall P_1, P_2$ in $\mathrm{Proj}(T^{\beta})$ and for any executable timepoint B: if $[S'(P_1)]_{<B} = [S'(P_2)]_{<B}$ then $[S'(P_1)]_B = [S'(P_2)]_B$ and $S'(P_1)$ is a consistent complete assignment for P_1 and $S'(P_2)$ is a consistent complete assignment for P_2. The same holds for $T^{\beta+1}$ and we will indicate with S'' the viable dynamic strategy for $T^{\beta+1}$ produced by DC-Execute. Now since $\mathrm{Proj}(T^{\beta+1}) \subseteq \mathrm{Proj}(T^{\beta})$, the projections of T^{β} will be mapped into two, possibly different, schedules: one by S' and on by S''. For every projection $P_i \in \mathrm{Proj}(P^{\beta+1})$ and for every executable B, notice that if $S''[P_i]_{<B}$ exists then it is equal to $S'[P_i]_{<B}$. We can thus define the history of B (which we recall is the set of durations of all contingent events which have finished prior to B) in the new strategy S: $S[P_i]_{<B} = S'[P_i]_{<B}$ for every projection $P_i \in \mathrm{Proj}(P^{\beta+1})$. Notice that $S''[P_i]_{<B}$ is not defined if the history of B in P_i contains a duration which is mapped into a preference exactly β and thus P_i cannot be a projection of $T^{\beta+1}$. On the other side if $S''[P_i]_{<B}$ exists the history of B contains durations mapped into preferences $\geq \beta + 1$ and is thus compatible with P_i being a projection of $T^{\beta+1}$, in addition to being one of T^{β}.

We must study all the possible cases in which the AB constraint is in T^{β} and in $T^{\beta+1}$.

- AB is Unordered in T^{β} and Unordered in $T^{\beta+1}$. The result of applying Merge is interval $[p', q']$, where $p' = min(p^{\beta}, p^{\beta+1})$, $q' = min(q^{\beta}, q^{\beta+1})$ a wait $t' = max(t^{\beta}, t^{\beta+1})$. Note that t^{β} and $t^{\beta+1}$ are the maximum wait imposed on AB by any contingent time point C resp. in T^{β} and $T^{\beta+1}$.

  In this case we will define strategy S as follows. For any pair of projections $P_i, P_j \in \mathrm{Proj}(P^{\beta+1})$, if $[S(P_i)]_{<B}$ =$[S(P_j)]_{<B}$ then $[S(P_i)]_B = [S(P_j)]_B$ =$max([S''(P_i)]_B, [S'(P_i)]_B)$ whenever $[S''(P_i)]_B$ is defined. Otherwise $[S(P_i)]_B$ =$[S(P_j)]_B$ = $[S'(P_i)]_B$. From Theorem 3.13 we have that $max([S''(P_i)]_B, [S'(P_i)]_B) \leq t'$, hence $[S(P_i)]_B = ([S(P_j)]_B) \in [p', q']$.

Let us now consider the preferences induced on the constraints by this assignment. First let us consider the case when $max([S''(P_i)\ ]_B, [S'(P_i)]_B) = [S''(P_i)]_B$. Since S'' is the dynamic strategy in $T^{\beta+1}$ all its assignment identify projections with preference $\geq \beta + 1$. If instead $max([S''(P_i)]_B, [S'(P_i)]_B) = [S'(P_i)]_B$, then it must be that $[S'(P_i)]_B > [S''(P_i)]_B$. However we know, from Theorem 3.13 that $[S''(P_i)]_B \leq t^{\beta+1} \leq t'$. This implies that $[S'(P_i)]_B \in [p^{\beta+1}, t']$ and thus it is an assignment with preference $\geq \beta + 1$. Finally, if $[S''(P_i)]_B$ is not defined, as noted above, then $P_i \not\in \mathtt{Proj}(T^{\beta+1})$ and thus $\mathtt{opt}(P_i) \leq \beta$. Thus, $[S(P_i)]_B$ =$[S(P_j)]_B = [S'(P_i)]_B$, which, being an assignment in T^β, identifies preferences $\geq \beta$, will never cause $\mathtt{pref}(S(P_i)) < \mathtt{opt}(P_i)$ for projections with optimal preference $\leq \beta$.

- constraint AB is a Follow case in T^β and in $T^{\beta+1}$. The execution of B will always follow that of any contingent time point C. Looking at Figure 3.8 we can see that the interval left by the DynamicallyControllable? algorithm in the Follow case on the AB constraint can be seen as an interval with a wait that expires after the end of the interval itself. This is why the new merged interval must be the union of the two intervals. This means that $[p^\beta, q^\beta] \subseteq [p', q']$ and also $[p^{\beta+1}, q^{\beta+1}] \subseteq [p', q']$. It is, hence, sufficient to define the new strategy S as follows: on all projections, $P_i, P_j \in \mathtt{Proj}(P^{\beta+1})$ such that $[S(P_i)]_{<B} = [S(P_j)]_{<B}$ then $[S(P_i)]_B = [S(P_j)]_B = [S''(P_i)]_B$ if $[S''(P_i)]_B$, otherwise $[S(P_i)]_B = [S(P_j)]_B = [S'(P_i)]_B$. This assignment guarantees to identify projections on constraints mapped into preferences $\geq \beta+1$ if $[S''(P_i)]_B$ exists and thus $P_i \in \mathtt{Proj}(T^{\beta+1})$, otherwise $\geq \beta$ for those projections in $\mathtt{Proj}(T^\beta)$ but not in $\mathtt{Proj}(T^{\beta+1})$.

- constraint AB is an Unordered case in T^β and a Follow case in $T^{\beta+1}$. The result of Merge is the union of the two intervals. In fact, $[p', q']$ is defined as $p' = min(p^{\beta+1}, p^{\beta+1})$, $q' = max(q^\beta, q^{\beta+1})$ and $t' = max(t^\beta, t^{\beta+1})$. We can thus define S as in the previous case: on all projections, $P_i, P_j \in \mathtt{Proj}(P^{\beta+1})$ such that $[S(P_i)]_{<B} = [S(P_j)]_{<B}$ then $[S(P_i)]_B = [S(P_j)]_B = [S''(P_i)$

$]_B$ if $[S''(P_i)]_B$ is defined, otherwise $[S(P_i)]_B = [S(P_j)]_B = [S'(P_i)]_B$. Notice, in fact, that since B must follow the execution of all contingent time points in $T^{\beta+1}$, then, whenever B is executed by S'', it cannot violate any wait in T^{β}.

- Constraint AB is a Precede case in T^{β} and in $T^{\beta+1}$. B must precede any contingent time point C. Interval $[p', q']$ is obtained intersecting the two intervals and any assignment to B in interval $[p', q']$, which must be not empty since Merge has not failed, will be consistent with any assignment of C. Moreover, it will be an assignment in $T^{\beta+1}$ hence with preference $\geq \beta + 1$. We can, thus, define S as follows: on any pair of projections $P_i, P_j \in \mathtt{Proj}(P^{\beta+1})$ if $[S(P_i)]_{<B} = [S(P_j)]_{<B}$ then $[S(P_i)]_B (= [S(P_j)]_B) = p'$.

- AB is an Unordered case in T^{β} and a Precede in $T^{\beta+1}$. This means that, in T^{β}, B can either precede or follow some contingent time point C while in $T^{\beta+1}$ it must precede all the contingent time points. This means that interval $[p^{\beta+1}, q^{\beta+1}]$ contains only values that are consistent with any possible future assignment of C in $T^{\beta+1}$. As shown in Figure 3.8, $t^{\beta+1} = p^{\beta+1}$. This means that, when it is defined, $S''(P_i)_B = t^{\beta+1}$. We can define S as in the case when AB is Unordered both in T^{β} and in $T^{\beta+1}$. For any pair of projections $P_i, P_j \in \mathtt{Proj}(P^{\beta+1})$, if $[S(P_i)]_{<B}$ =$[S(P_j)]_{<B}$ then $[S(P_i)]_B$ =$[S(P_j)]_B$ = $max([S''(P_i)]_B, [S'(P_i)]_B)$ whenever $[S''(P_i)]_B$ is defined. Otherwise, $[S(P_i)]_B$ =$[S(P_j)]_B = [S'(P_i)]_B$. Let us now consider the preferences induced on the constraints by this assignment. First let us consider the case when $max([S''(P_i)]_B, [S'(P_i)]_B) = [S''(P_i)]_B$. It must be that none of the contingent time points have occurred yet, otherwise S'' would not be defined. Moreover, since the $max([S''(P_i)]_B, [S'(P_i)]_B) = [S''(P_i)]_B$ no wait in T^{β} can be violated. We can conclude that this assignment identifies projections with preference $\geq \beta + 1$. If, instead, $max([S''(P_i)]_B, [S'(P_i)]_B) = [S'(P_i)]_B$, then it must be that $[S'(P_i)]_B > [S''(P_i)]_B$ and the assignment induces preferences $\geq \beta + 1$. Finally, if $[S''(P_i)]_B$ is not defined then some contingent time point has occurred before $t^{\beta+1}$ and, thus, $P_i \notin \mathtt{Proj}(T^{\beta+1})$ and $\mathtt{opt}(P_i) \leq \beta$. As before,

$[S(P_i)]_B$ $=[S(P_j)]_B = [S'(P_i)]_B$, which, being an assignment in T^β, identifies preferences $\geq \beta$ which will never cause $\mathtt{pref}(S(P_i)) < \mathtt{opt}(P_i)$.

We will now give three corollaries of Theorem 3.14 that will be useful for proving the soundness and completeness of Best-DC.

Corollary 3.1 *Consider strategies S', S'' and S as defined in Theorem 3.14. Then*

1. *for any projection of $P^{\beta+1}$, P_i, $\mathtt{pref}(S(P_i)) \geq \mathtt{pref}(S'(P_i))$ and for every projection, P_z, of $T^{\beta+1}$, $\mathtt{pref}(S(P_z)) \geq \beta+1$;*
2. *for any constraint AB, $[S(P_i)]_B \geq t'$.*

Proof:

1. Obvious, since in all cases either $[S(P_i)]_B = [S'(P_i)]_B$ or $[S(P_i)]_B = [S''(P_i)$ $]_B$ and $\mathtt{pref}(S''(P_i)) \geq \mathtt{pref}(S'(P_i))$ since for every executable B $[S''(P_i)]_B$ $\in T^{\beta+1}$. Moreover, for every projection P_z of $T^{\beta+1}$, for every executable B, $[S(P_z)]_B = [S''(P_z)]_B$.
2. Derives directly from the fact that either $[S(P_i)]_B = [S'(P_i)]_B$ or $[S(P_i)]_B = [S''(P_i)]_B$ and Theorem 3.13.

Corollary 3.2 *Consider STPPU P and for every preference level, α, define T^α as the STPU obtained by chopping P at α, then applying PC-2 and then DynamicallyControllable?. Assume that $\forall \alpha \leq \beta$, T^α is DC. Consider STPU P^β:*

$$P^\beta = \mathit{Merge}(\mathit{Merge}(\ldots$$

$$\mathit{Merge}(\mathit{Merge}(T\alpha_{min}, T\alpha_{min}+1), T^{\alpha_{min}+2}), \ldots), T^\beta)$$

with α_{min} the minimum preference on any constraint in P. Assume that, when applied, Merge never failed. Then there is a viable dynamic strategy S, such that $\forall P_i \in \mathrm{Proj}(P)$, if $\mathtt{opt}(P_i) \leq \beta$ then $S(P_i)$ is an optimal solution of P_i otherwise $\mathtt{pref}(S(P_i)) \geq \beta+1$.

Proof: We will prove the corollary by induction. First, notice that, by construction $\text{Proj}\ (T^{\alpha_{min}}) = \text{Proj}(P)$. This allows us to conclude that $\text{Proj}(P^{\beta}) = \text{Proj}(P)$, since, every time Merge is applied, the new STPU has the same contingent constraints as the STPU given as first argument.

Now, since $T^{\alpha_{min}}$ is dynamically controllable any of its viable dynamic strategies, say $S^{\alpha_{min}}$ will be such that $S^{\alpha_{min}}(P_i)$ is optimal if $\text{opt}(P_i) = \alpha_{min}$ and, otherwise, $\text{pref}(S(P_i)) \geq \alpha_{min}$. Consider now $P^{\alpha_{min}+1}$=Merge $(T^{\alpha_{min}}, T^{\alpha_{min}+1})$. Then by Theorem 3.14, there is a strategy, $S^{\alpha_{min}+1}$, such that $S^{\alpha_{min}+1}(P_i)$ is an optimal solution of P_i if $\text{opt}(P_i) \leq \alpha_{min} + 1$ and $\text{pref}(S(P_I)) \geq \alpha_{min} + 1$ otherwise.

Let us assume that STPU $P^{\alpha_{min}+k}$, as defined in the hypothesis, satisfies the thesis and that $P^{\alpha_{min}+k+1}$, as defined in the hypothesis, where $\alpha_{min} + k + 1 \leq \beta$, does not. Notice that this implies that there is a strategy, $S^{\alpha_{min}+k}$, such that $S^{\alpha_{min}+k}(P_i)$ is an optimal solution of P_i if $\text{opt}(P_i) \leq \alpha_{min} + k$ and $\text{pref}(S(P_i) \geq \alpha_{min} + k$ for all other projections. Since $\alpha_{min} + k + 1 \leq \beta$, then, by hypothesis we also have that $T^{\alpha_{min}+k+1}$ is DC. Moreover, by construction, $P^{\alpha_{min}+k+1}$=Merge $(P^{\alpha_{min}+k}, T^{\alpha_{min}+k+1})$, since Merge doesn't fail. Thus, using Theorem 3.14 and using strategy $S^{\alpha_{min}+k}$ for $P^{\alpha_{min}+k}$ in the construction of Theorem 3.14, by Corollary 3.1, we will obtain a dynamic strategy, $S^{\alpha_{min}+k+1}$, such that for every projection P_i, $\text{pref}(S^{\alpha_{min}+k+1}(P_i)) \geq \text{pref}(S^{\alpha_{min}+k}(P_i))$ and such that $S^{\alpha_{min}+k+1}\ (P_j)$ is an optimal solution for all projections P_j such that $\text{opt}(P_j) = \alpha_{min} + k + 1$ and $\text{pref}(S(P_j)) \geq \alpha_{min} + k + 1$ on all other projections. This allows us to conclude that $S^{\alpha_{min}+k+1}(P_h)$ is an optimal solution for all projections P_h such that $\text{opt}(P_h) \leq \alpha_{min} + k + 1$. This is contradiction with the assumption that $P^{\alpha_{min}+k+1}$ doesn't satisfy the thesis of the corollary.

Corollary 3.3 *Consider STPPU* P *and STPUs* T^{β}, $T^{\beta+1}$ *and* $P^{\beta+1}$ *as defined in Theorem 3.14. If* Merge *fails then there is no viable strategy such that for every pair of projections* P_i, P_j *and for every executable B, if* $[S(P_i)]_{<B} = [S(P_j)]_{<B}$ *then*$[S(P_i)]_B = [S(P_j)]_B$ *and* $S(P_i)$ *is an optimal solution of* P_i *and* $S(P_j)$ *is an optimal solution of* P_j.

Proof: Assume Merge fails on some constraint AB because $[p^{\beta}, q^{\beta}] \cap [p^{\beta+1}, q^{\beta+1}] = \emptyset$. As proven in [MMV01], the projection on AB of any viable dynamic strategy for T^{β} is in $[p^{\beta}, q^{\beta}]$ and the projection on AB of any viable dynamic strategy for $T^{\beta+1}$ is in $[p^{\beta+1}, q^{\beta+1}]$. The dynamic viable strategies of T^{β} give optimal solutions for projections with optimal preference equal to β. The dynamic viable strategies of the $T^{\beta+1}$ give optimal solutions for projections with optimal preference equal to $\beta + 1$. Since the projections of $T^{\beta+1}$ are a subset of those in T^{β}, if $[p^{\beta}, q^{\beta}] \cap [p^{\beta+1}, q^{\beta+1}] = \emptyset$ then a strategy either is optimal for projection in T^{β} but not for those in $T^{\beta+1}$ or vice-versa.

Assume now that Merge fails on some constraint AB because $[t^{\beta}, q^{\beta}] \cap [t^{\beta+1}, q^{\beta+1}] = \emptyset$. It must be that either $q^{\beta+1} < t^{\beta}$ or $q^{\beta} < t^{\beta+1}$. If the upper bound of the interval on AB is $q^{\beta+1}$ the there must be at at least a contingent time point C such that executing B more than $q^{\beta+1}$ after A is either inconsistent with some assignment of C or it gives a preference lower than $\beta + 1$. On the other side, if the wait on constraint AB in T^{β} is t^{β} there must be at least a contingent time point C' such that executing B before t^{β} is either inconsistent or not optimal with some future occurrences of C'. Again there is no way to define a viable dynamic strategy that is simultaneously optimal for projections with optimal value equal to β and for those with optimal value $\beta + 1$.

We will now use these results for proving that Best-DC is a sound and complete algorithm for testing if an STPPU is ODC and for finding the highest preference at which it is α-DC.

Let us start enumerating the conditions at which Best-DC terminates.

- (line 4) Best-DC stops because the STPU obtained at level α_{min} is not DC;
- (line 11) Best-DC exits because it has reached a preference level β at which the STPU (seen as an STP) is not path consistent;
- (line 13) Best-DC stops because it has reached a preference level β at which the STPU is path consistent but not dynamically controllable;
- (line 17) Best-DC stops because procedure Merge has found an inconsistency.

We will first prove that the algorithm terminates.

Theorem 3.15 *Given an STPPU P, the execution of algorithm* Best-DC *on* P *will terminate.*

Proof: In this proof we assume that the preference set is discretized and that there are a finite number of different preferences. Best-DC starts from the lowest preference and chops at each level P. Assume that, as it moves up in the preference ordering, it always finds path consistent STPUs that are dynamically controllable as well. However at a certain point the chopping level will be higher than the maximum on some preference function (or it will be outside of the preference set) in which case chopping the problem will give an inconsistent STP.

We are now ready to prove the soundness and completeness of Best-DC. First we consider the case in which the underlying STPU obtained from P, ignoring the preferences, is not DC.

Theorem 3.16 *Given an STPPU* P *as input, then* Best-DC *will terminate in line 4 iff* $\nexists \alpha \geq 0$ *such that P is* α*-DC.*

Proof: $\Rightarrow$. Assume Best-DC terminates in line 4. Then, the STPU obtained by chopping P at the minimum preference, α_{min}, on any constraint is not DC. However chopping at the minimum preference on any constraint or at preference level 0 gives the same STPU. By Theorem 3.12 we can conclude that P is not α-DC $\forall \alpha \geq 0$ and, thus, not ODC.

$\Leftarrow$. Assume P is not α-DC for all preferences $\alpha \geq 0$. Then chopping P at the minimum preference α_{min} cannot give a dynamically controllable problem, otherwise, P would be α_{min}-DC. Hence, Best-DC will exit in line 4.

Now we will focus on showing that Best-DC is a sound and complete algorithm for detecting ODC.

Theorem 3.17 *Given an STPPU* P *as input, then* Best-DC *will terminate in line 11 iff P is ODC.*

Proof: $\Rightarrow$. Assume Best-DC terminates in line 11 when considering preference level β. Then, STPU Q^{β} obtained by chopping STPPU P at level β is not path consistent. From this we can immediately conclude that there is no projection $P_i \in \mathrm{Proj}(P_i)$ such that $\mathrm{opt}(P_i) \geq \beta$.

Since Best-DC did not terminate before, we must assume that up to preference $\beta - 1$, all the tests (path consistency, dynamic controllability, and Merge) were successful.

Now consider the STPU $P^{\beta-1}$ obtained at the end of the iteration corresponding to preference level $\beta - 1$. It is easy to see that $P^{\beta-1}$ satisfies the hypothesis of Corollary 3.2. This allows us to conclude that there is a viable dynamic strategy S such that for every projection P_i, such that $\mathrm{opt}(P_i) \leq \beta - 1$, $S(P_i)$ is an optimal solution of P_i. However since we know that all projections of P are such that $\mathrm{opt}(P_i) < \beta$, this allows us to conclude that P is ODC.

$\Leftarrow$. If P is ODC then there is a viable strategy S such that for every pair of projections, $P_i, P_j \in \mathrm{Proj}(P)$, and for very executable B, if $[S(P_i)]_{<B} = [S(P_j)]_{<B}$ then $[S(P_i)]_B = [S(P_j)]_B$ and $S(P_i)$ is an optimal solution of P_i and $S(P_j)$ is an optimal solution of P_j.

By Theorem 3.16 we know that Best-DC cannot stop in line 4.

Let us now consider line 13 and show that if Best-DC sets α-DC to $true$ in that line then P cannot be ODC. In fact the condition of setting α-DC to $true$ in line 13 is that the STPU obtained by chopping P at preference level β is path consistent but not dynamically controllable. This means that there are projections, e.g. P_j, of P such that $\mathrm{opt}(P_j) = \beta$. However, there is no dynamic strategy for the set of those projections. Thus, P cannot be ODC.

Let us now consider line 17, and show that, if P is ODC Best-DC cannot set α-DC to $true$. If Best-DC sets α-DC to $true$ then Merge failed. Using Theorem 3.14 and Corollary 3.3, we can conclude that there is no dynamic viable strategy S such that for every projection of P, P_i, (remember that $\mathrm{Proj}(P^{\beta-1}) = \mathrm{Proj}(P)$) $S(P_i)$ is an optimal solution if $\mathrm{opt}(P_i) \leq \beta$. However, we know there are projections of

P with optimal preference equal to β (since we are assuming Best-DC is stopping at line 17 and not 11). Thus, P cannot be ODC.

We will now show that Best-DC stops at line 13 or 17 when considering preference level β iff it is $(\beta - 1)$-DC.

Theorem 3.18 *Given STPPU* P *in input,* Best-DC *stops at lines 13 or 17 at preference level* β *iff* P *is* $(\beta - 1)$*-DC and not ODC.*

Proof: $\Rightarrow$. Assume that Best-DC sets α-DC to $\mathtt{true}$ in line 13, when considering preference level β. Thus, the STPU obtained by chopping P at level β is path consistent but not DC. However since β must be the first preference level at which this happens, otherwise the Best-DC would have stopped sooner, we can conclude that the iteration at preference level $\beta - 1$ was successful. Considering $P^{\beta-1}$ and using Corollary 3.2 we can conclude that there is a viable dynamic strategy S such that, for every projection of P, P_i, if $\mathtt{opt}(P_i) \leq \beta - 1$ then $S(P_i)$ is an optimal solution of P_i and $\mathtt{pref}(S(P_i)) \geq \beta - 1$ otherwise. But this is the definition of $\beta - 1$-dynamic controllability.

If Best-DC terminates in line 17, by Corollaries 3.2 and 3.3 we can conclude that, while there is a viable dynamic strategy S such that for every projection of P, P_i, if $\mathtt{opt}(P_i) \leq \beta - 1$ then $S(P_i)$ is an optimal solution of P_i and $\mathtt{pref}(S(P_i)) \geq \beta - 1$ otherwise, there is no such strategy guaranteeing optimality also for projections with optimal preference β. Again, P is $\beta - 1$-DC.

$\Leftarrow$. If P is α-DC, for some $\alpha \geq 0$ then by Theorem 3.16, Best-DC does not stop in line 4. If P is α-DC, but not ODC, for some $\alpha \geq 0$ then by Theorem 3.17, Best-DC does not stop in line 11. By Theorem 3.15, Best-DC always terminates, so it must stop at line 13 or 17.

Complexity of Best-DC

We will now consider the complexity of Best-DC. Also in what follows we will be assuming the usual tractability conditions. When considering complexity the issue of precision arises. Up to now, we have described a soft temporal constraint as an interval containing the allowed durations or interleaving times and a preference function mapping the durations into preferences. Since all the practical applications can be modeled using finite intervals, this is a reasonable condition to impose.

Moreover, time is a quantity that is well suited for discretization (10 minutes, 1 hour....). Without loss of generality, we can assume a minimum unit of time, δ, that represents the precision with which we discretize the intervals. This, of course, induces a discretization of the preferences as well, as we have already assumed in Theorem 3.15 in order to prove that Best-DC always terminates. Thus, we will denote with l the finite number of different preference levels, and with r the maximum number of points in an interval.

Given this notation, in [MMV01], it is proven that checking DC of an STPU can be done in $O(n^2 r)$, where n is the number of variables. Since the complexity depends on the granularity of time, it is said to be deterministic polynomial. We will know prove formally that Best-DC determines the ODC or the highest preference α such that a given STPPU is α-DC in deterministic polynomial time.

Theorem 3.19 *The complexity of determining ODC or the highest preference level* α *of* α*-DC of an STPPU with* n *variables, a bounded number of preference levels* l *and intervals of bounded maximum size* r *is* $O(n^3 l r)$.

Proof: Consider the pseudocode of algorithm Best-DC in Figure 3.17.

The complexity of $\mathrm{Chop}(P, \alpha_{min})$ in line 3 is $O(n^2)$, since every constraint must be considered, an there are up to $O(n^2)$ constraints, and for each constraint the time for finding the interval of elements mapped into preference $\geq \alpha_{min}$ is constant. The complexity of checking if the STPU obtained is DC is $O(n^2 r)$. Thus, lines 3 and 4, which are always performed, have an overall complexity of $O(n^2 r)$.

Lines 7 and 8, clearly, take constant time.

Let us now consider a fixed preference level β and compute the cost of a complete *while* iteration on β.

- (line 10) the complexity of $\mathrm{Chop}(P, \beta)$ is $O(n^2)$;

- (line 11) the complexity of applying PC-2 for testing path consistency is $O(n^3r)$ (see Chapter 2 Section 2.2.1, [DMP91]);
- (line 13) the complexity of testing DC using DynamicallyControllable? is $O(n^2r)$, cite (see Section 3.2, [MMV01]);
- (line 15) constant time;
- (line 16-17) the complexity of Merge is $O(n^2)$, since at most $O(n^2)$ constraints must be considered and for each constraint merging the two intervals has constant cost;
- (line 19) constant time.

We can conclude that the complexity of a complete iteration at any given preference level is $O(n^3r)$.

In the worst case, the $while$ cycle is performed l times. We can, thus, conclude that the total complexity of Best-DC is $O(n^3lr)$ since the complexity of the operations performed in lines 24-27 is constant.

The complexity result given in Theorem 3.19 is unexpectedly good. In fact, it shows that the cost of adding a considerable expressiveness power through preferences to STPUs is a factor equal to the number of different preference levels. This implies that solving the optimization problem and the controllability problem simultaneously remains in P, assuming that the range of the intervals and the number of different preference levels are bounded.

The Execution algorithm

The execution algorithm we propose is very similar to that for STPUs presented in [MMV01] which we have described in Section 3.2 and is shown in Figure 3.10. Of course the execution algorithm for STPPUs will take in input an STPPU to which Best-DC has been successfully applied. In line 2 of Figure 3.10 the algorithm performs the initial propagation from the starting point. The main difference between our execution algorithm and the one in [MMV01] is the definition of "propagation". In fact, in our case it involves also preferences.

Definition 3.23 (soft temporal propagation) *Consider an STPPU* P *and a variable* $Y \in P$ *and a value* $v_Y \in D(Y)$. *Then propagating the assignment* $Y = v_Y$ *in* P, *means:*

- *for all constraints,* c_{XY} *involving* Y *such that* X *is already assigned value* $v_X \in D(X)$*: replace the interval on* c_{XY} *with interval* $\langle [v_Y - v_X, v_Y - v_X] \rangle$*;*
- *chop* P *at preference level* $\min_X \{f_{c_{XY}}(v_Y - v_X)\}$.

We will call ODC-Execute algorithm DC-Execute where propagation is defined as in Definition 3.23. Assume to apply ODC-Execute to an ODC STPPU P to which Best-DC has been applied. If up to a given time T the preference of the partial schedule was β, then we know that if P was ODC or α-DC with $\alpha \geq \beta$, by Theorem 3.14 and its corollaries the execution algorithm has been assigning values in $T^{\beta+1}$. Assume now that a contingent time point occurs and lowers the preference to $\beta - 2$. This will be propagated and the STPPU will be chopped at preference level $\beta - 2$. From now on, the execution algorithm will assign values in $T^{\beta-2}$ and, by Theorem 3.14 and its corollaries, the new waits imposed will be such that the assignments for the executables will be optimal in any situation where the optimal preference is $\leq \beta - 2$. In all other situations such assignments guarantee a preference of at least $\beta - 2$.

3.8 Using the Algorithms

In Section 3.4.4 we have described the relations between the new notions of controllability we have introduced. As a general strategy, given an STPPU, the first property to consider is OSC. If it holds, the solution obtained is valid and optimal in all possible scenarios, but OSC is a strong property and holds infrequently. If the STPPU is not OSC, Best-SC will find the best solution that is consistent with all possible future situations.

Most commonly, dynamic controllability will be more useful. If the control sequence needs not be known before execution begins, ODC is ideal. Notice that, from the results in Section 3.4.4, an STPPU may be not OSC and still be ODC. If, however, the STPPU is not even ODC, then Best-DC will give a dynamic solution with the highest preference. Let us recall, as we have shown in Section 3.4.4, that for any given preference level α, α-SC implies α-DC but not viceversa. Thus, it may be that a given STPPU is β-SC and γ-DC with $\gamma > \beta$. Being β-SC means that there is a fixed way to assign values to the executables such that it will be optimal only in situations with optimal preference $\leq \beta$ and will give a preference at least β in all other cases. On the other hand, γ-DC implies that a solution obtained dynamically, by the ODC-Execute algorithm, will be optimal for all those situations where the best solution has preference $\leq \gamma$ and will yield a preference $\geq \gamma$ in all other cases. Thus if, as we are assuming, $\gamma > \beta$, using the dynamic strategy will guarantee optimality in more situations and a higher preference in all others.

The last possibility is to check OWC. This will at least allow the agent to know in advance if there is some situation that has no solution. Moreover, if the situation is revealed just before the execution begins, using any of the solvers for STPPs presented in Section 2.4 will allow us to find an optimal assignment for that scenario.

3.9 Related Work

We will now consider some work which we regard as closely related to ours. Temporal uncertainty has been studied before but it has been defined in different ways according to the different contexts where it has been used.

We start considering the work proposed by Vila and Godo in [VG94] which we have already considered in Section 2.7 relatively to STPPs. They propose **Fuzzy Constraint Networks**, which are STPs where the interval in each constraint is mapped into a possibility distribution. In fact, they handle temporal uncertainty using possibility theory. They use the term uncertainty to describe vagueness in the temporal information available. Their aim, in fact, is to model statements as ”He called me more less an hour ago”, where the uncertainty is the lack of precise information on a temporal event. Their goal thus is completely different from ours. In fact, we are in a scenario where an agent must execute some activities at certain times, and such activities are constrained by temporal relations with uncertain events. Our goal is to find a way to execute what is in the agents control in a way that will be consistent whatever nature decides in the future. In [VG94], they assume to have imprecise temporal information on events happened in the past. Their aim is to check if such information is consistent, that is, if there are no contradictions implied and to study what is entailed by the set of constraints. In order to model such imprecise knowledge, as mentioned before, they use possibilities [Zad75]. Every element of an interval is mapped into a value that indicates how possible that event is or how certain it is. Thus, another major difference with their approach is that they don’t allow preferences, but just possibilities. On the other hand, in the work presented in this chapter we do not allow to express information on how possible or probable a value is for a contingent time point. This is one of the lines of research we want to pursue in the future. Moreover, in [VG94], they are concerned with the classical notion of consistency (consistency level) rather than with controllability, although their definition of consistency is somehow comparable to weak controllability.

Another work that in addition to being related to what we have presented in the previous chapter is also related to the way we handle uncertainty is that of Giacomin and Badaloni presented in [BG00]. As we have said in Section 2.7, they introduce **Flexible Temporal Constraints** where soft constraints are used to express preferences among feasible solutions and prioritized constraints are used to express the degree of necessity of the constraints satisfaction. At first sight this work seems to have several aspects in common with what we have presented in this chapter, since they also have preferences and uncertainty in a temporal environment. However, the way they handle preferences is different in the sense we have described in Section 2.7. In particular, they consider qualitative ”a la Allen” temporal relations and they associate each such relation to a preference. The uncertainty they deal with is also very different from ours. In fact the uncertainty is not on the time of occurrence of an event but is on whether a constraint belongs or not to the constraint problem. In other words they have a set of soft constraints, C, which, as we have said in Section 2.7, are qualitative interval constraints where a preference is attached to every relation in the constraint. Then they have a possibility distribution associating to each constraint a degree expressing the "plausibility" of it belonging to the set of constraints. They then transform these soft constraints with plausibilities into prioritized constraints. In [BG00] a prioritized constraint is again a soft qualitative interval constraint with a priority degree, p, attached expressing the degree of necessity of its satisfaction. Any prioritized constraint is then transformed into a soft constraint by associating to each relation a new preference which is the maximum between the old preference and $1 - p$. In this way they obtain a set of soft constraints where preferences and uncertainty coexist. Since they have qualitative relations they use a branch and bound technique which they customize for their constraints. In addition, to the different concept of uncertainty they handle, it must be said that in their model information coming from plausibility and information coming from preferences gets mixed and is not distinguishable by the solver. In other words, it is not possible to say whether a solution is bad due to its poor preference on some relation or due to it violating a constraint with a high priority. In our approach, instead, uncertainty and preferences are separated. The compatibility with an uncertain event does not change the preference of an assignment to an executable. The robustness to temporal uncertainty is handled intrinsically imposing the different degrees of controllability.

In [DHP03] the authors consider fuzziness and uncertainty in temporal reasoning by introducing **Fuzzy Allen Relations**. More precisely, they present an extension of Allen relational calculus, based on fuzzy comparators expressing linguistic tolerance. The concept of temporal uncertainty modeled [DHP03] is quite different from ours. In fact, the authors in [DHP03] want to handle the following scenarios. First, situations in which the information about dates and relative positions of intervals is complete but, for some reason, there is no interest in describing it in a precise manner. For example, when one wants to speak only in terms of "approximate equality", or proximity rather that in terms of precise equality. Secondly, they want to be able to deal with available information pervaded with imprecision, vagueness or uncertainty. This is the case when relations between dates or intervals are known with precision and certainty but some dates may be imprecisely located (i.e., the date belongs to some interval), fuzzily located (i.e., the more or less possible values of an event are restricted by a fuzzy set), or the dates are pervaded with uncertainty (i.e., when there is complete ignorance on when an event occurs). Another instance of this case is when the knowledge about the relations is imprecise or vague. In the framework we have presented in this chapter we restrict the uncertainty to when an event will occur within a range. On the other hand, we put ourselves into a "complete ignorance" position, that would be equivalent, in the context of [DHP03], to having all possibilities equal to 1. Moreover, in [DHP03] they do not allow preferences nor address controllability. Instead, they consider, similarly to [VG94], consistency and entailment. The first is checked by computing the transitive closure of the fuzzy temporal relations using inference rules appropriately defined. The second by defining several patterns of inference.

Another work, which we have already considered in Section 2.7, and which addresses also temporal uncertainty is the one presented in [DFP95] and in [DFP03]. In this work both preferences and activities with ill-know durations in the classical job-shop scheduling problem (see Section 2.7 for more details) are handled using the fuzzy framework. There are three types of constraints: precedence constraint, capacity constraints and due dates and release time constraints. In Section 2.7 we have seen how they model preferences. We will know describe how the handle activities of ill know duration. In order to model such unpredictable events they use possibility theory. As the authors mention in [DFP95], possibility distributions can be viewed as modeling uncertainty as well as preference (see [DFP93]). Everything depends on whether the variable X on which the possibility distribution is defined is controllable or not. If X is controllable then the distribution π_X is a profile expressing the preferences on the different values of X. If, instead X is not controllable, then π_X models an agent's uncertainty about the value X will eventually take. Thus the authors in [DFP95] distinguish between controllable an uncontrollable variables. However they do not allow to specify preferences on uncontrollable events. Our preference functions over contingent constraints would be interpreted as possibility distributions in their framework. In some sense, our work is complementary to theirs. We assume a constraint possibility distribution on contingent events always equal to 1 and we don't allow the representation of any further information on more or less possible values; on the other hand, we allow to specify preferences also on uncontrollable events. They, on the contrary, allow to put possibility distributions on contingent events, but not preferences. In [DFP95], when a fuzzy duration represents imprecise knowledge about the *uncontrollable* duration of a task, the satisfaction degree of the constraint is a necessity degree which represents the requirement to get a robust feasibility of the schedule. For example, assume that activity i, with start time s_i, has an uncontrollable duration t_i (notice that we would model this with an uncertain finishing time). Then the possibility distribution describing the more or less possible values for t_i corresponds to a trapezoidal fuzzy number $T(i)$. Assume that there is a precedence constraint such that activity i must precede activity k, i.e. $s_k \geq s_i + t_i$, where s_k is the start time of k. In [DFP95] they define the satisfaction degree of the precedence constraint as the necessity of the event $t_i \leq s_k - s_i$: $N(t_i \in (-\infty, s_k - s_i])$. This necessity is then expressed using the fuzzy number $T(i)$. Finally, in [DFP95], the authors show that a scheduling problem with uncertain durations can be formally expressed by the same kind of constraints as a problem involving what they call flexible durations (i.e. durations with fuzzy preferences). However the interpretation is quite different: in the case of flexible durations, the fuzzy information comes from the specifications of preferences and represents the possible values that can be assigned to a variable representing a duration. In the case of imprecisely known durations, the fuzzy information comes from uncertainty about the real value of some durations. The formal correspondence between the two constraints is so close that the authors do not distinguish among them when describing the solving procedure. Another thing that should be considered is that the problem they solve is to find the starting times of activities such that these activities take place within the global feasibility window *whatever the actual values of the unpredictable durations will be*. Clearly this is equivalent to Optimal Strong Controllability. They do not address the problem of dynamic or weak controllability with preferences.

3.10 Future Work

Recently, another approach for handling temporal uncertainty has been introduced in [Tsa02, TPR03]: Probabilistic Simple Temporal Problems (PSTPs). In this framework rather than bounding the occurrence of an uncontrollable event within an interval, as in STPUs, a probability distribution describing when the event is more likely to occur is defined on the entire set of reals. Formally, a PSTP is a tuple $\langle V_C, E, V_U, Par, P\rangle$ where: V_C is the set of controllable variables, V_U is the set of uncontrollable variables, E is a set of constraints $X - Y \leq b$, where $X, Y \in V_C \cup V_U$ and b is a real number, Par is a function $V_U \longrightarrow V_C$ that assigns to each uncontrollable a controllable parent, and P is a set of conditional probability density functions $p_X(t)$ providing the mass of probability of X occurring t time units after $Par(X)$. For example, in order to model the fact that an activity A has a duration normally distributed with mean duration of 30′ and 5′ standard deviation, they use two variables: controllable variable Y representing the beginning of A and uncontrollable variable X representing the end of A. Then they define $Par(X) = Y$ and $P_X(t) = N(30, 5)$ where N is a Gaussian normal density function with mean 30 and standard deviation 5. Given a PSTP the goal is to asses the probability that all the constraints in E will be satisfied during an execution. In other words, the problem they solve is executing a PSTP so that the probability that all constraints are satisfied is maximized. As in STPUs, the way the problem is solved depends on the assumptions made regarding the knowledge about the uncontrollable variables. In particular they define *Static Scheduling Optimization Problem* which is the equivalent to finding an execution satisfying SC in STPUs, and *Dynamic Scheduling Optimization Problem* equivalent to finding a dynamic execution strategy in the context of STPUs. The first problem is solved in [TPR03], defining $P(Success|T)$ as the probability that all constraints will be satisfied during execution if the controllables are executed according to schedule T, and then solving the static optimization problem $maxP(Success|T)$ subject to the constraints in E. To compute the equivalent of a dynamic strategy, it is necessary to iterate the process of finding a static optimal strategy whenever an observation of an uncontrollable occurs. In [Tsa02] it is shown that computing the exact probability of success of this overall dynamic strategy is difficult. However it is possible to compute lower and upper bounds on such probability by converting the PSTP into an STPU. In the above framework optimal means with the highest probability of satisfying all the constraints. Preferences are not considered in this framework. As we have added preferences to STPUs we believe it would be interesting to do as well to this approach for handling uncertainty. A first step could be that of keeping for each strategy separately its global preference and its probability of success. In this way we could use the existing frameworks for handling the two aspects. Once that for each tuple we have the preference and the probability, different semantics may order the strategies in different ways, for example by giving priority to preferences, taking in some sense a risky attitude, or, on the contrary giving priority to probabilities, in a more cautious attitude. It seems that the best way to handle such an approach would be that of using a 2-dimensional semiring, obtained by the Cartesian product of a semiring used to handle preferences, as S_{FCSP} for fuzzy preferences or S_{WCSP} in utilitarian optimization, and another semiring for probabilities, which of course would be S_{PSCP} (see Chapter 1 Section 2.2.2 for details on the semirings). It should be, however, also interesting to consider how preferences can be induced from probabilities, in a similar way to what has been done for possibilities in [DFP96].

Up to now we have focused our attention on non disjunctive temporal problems, that is, with only one interval per constraint. We would like to consider adding uncertainty to Disjoint Temporal Problems [SK00] and to consider scenarios where there are both preferences and uncertainty. Such problems are not polynomial even without preferences or uncertainty but it has been shown that the cost of adding preferences is small [PP04], so we hope that the same will hold in environments with uncertainty as well. Surprisingly uncertainty in Disjoint Temporal Problems has not been considered yet, although it is easy to see how allowing multiple intervals on a constraint is itself a form of uncontrollability. We, thus, plan to start defining DTPUs and then to merge this approach with the existing one for DTPPs.

Chapter 4

Conditional Statements vs Soft Constraints

In Chapter 2 we have considered specialized tools for handling quantitative temporal preferences. In Chapter 3 we have broadened the class of applications of our results to temporal preferences with possibly uncontrollable events. The approach we have proposed can be seen as an attempt to use soft constraints in order to introduce the expressive power of preferences into a specific area, characterized by specific reasoning tools as the temporal reasoning domain.

At this point we want to enlarge even more the target and to consider more general scenarios where preferences play a fundamental role. One of the main aspects we want to investigate in this chapter is how soft constraints cope with conditionality and how they can interact with other qualitative models to represent conditional preferences. In this chapter we are going to consider the coexistence of qualitative conditional preferences, modeled using Conditional Preference networks (CP-nets)[BBHP99, BBD$^+$04a], quantitative preferences modeled using soft constraints, and hard constraints.

First we consider approximating CP-nets using semiring based soft constraints. This means giving quantitative approximations of qualitative statements. We will show that such approximations can be obtained in polynomial time in the size of the CP-net. Moreover we will prove that the partial loss of information, due to the approximation, is well compensated by a gain in terms of complexity for dominance testing.

Secondly, we will focus on finding outcomes that are optimal both for the CP-net and for the hard and soft constraints. We will first propose a semantics that allows to find such optimals by transforming the optimization problem into a hard constraint problem. This approach is attractive in terms of complexity, since, for example, it allows us to test if an outcome is optimal in linear time in the number of generated constraints. However its drawback is in the weak ordering it induces, since it may potentially leave many pairs of outcomes incomparable. To overcome this, we have considered applying the hard constraint reduction for finding optimals w.r.t. to another, stronger, semantics, originally defined in [BBD$^+$04b]. The strength of our algorithm is that it is able to deal also with cyclic nets.

4.1 Motivations and Chapter Structure

As stated many times, representing and reasoning about preferences is an area of increasing interest in theoretical and applied AI. For example, in a product configuration problem, the producer may have hard and soft constraints, while the user has a set of conditional preferences. Moreover, until now, there has been no single formalism which allows both qualitative and quantitative information to be specified naturally and reasoned with effectively. As we have mentioned before, soft constraints [BMR97, SFV95] are one of the main methods for dealing with preferences in constraint optimization. However they represent preferences in a quantitative way.

Qualitative preferences have been widely studied in decision theoretic AI [DT99]. Of particular interest are CP-nets [BBHP99]. These model statements of qualitative and conditional preference such as “I prefer a red dress to a yellow dress”, or “If the car is convertible, I prefer a soft top to a hard top”. These are interpreted under the *ceteris paribus* (that is, “all else being equal”) assumption. Preference elicitation in such a framework is intuitive, independent of the problem constraints, and suitable for naive users. However, the Achille’s heel of CP-nets and other sophisticated qualitative preference models [Lan02] is the complexity of reasoning with them [DB02, BBD$^+$04a].

Many real life optimization problems, however, contain both qualitative conditional preferences hard and soft constraints. An interesting example recently proposed in [BDSar] is that of adapting rich-media messages to specific user devices. Rich-media messages contain multiple elements of different types, such as audio, video, text, image, and animation. Each element can be displayed in multiple ways. However, the message author has a single most preferred way for presenting her message. In addition there is the problem of feasibility of the presentation on all the devices. For example, a PDA with a black-and-white screen cannot display color. To overcome this problem, and other similar ones, the message transcoder needs to adapt the message presentation to the particular capabilities of the user device by finding a feasible presentation of the message. The message author, who does not know at composition time on what devices her message will be displayed, needs to provide information that will help the transcoder select the best feasible presentation. Using a language for annotating her message, the author could rank different presentation options for various message elements. When the message recipient requests the message, his device submits a description of device capabilities to the transcoder, which attempts to select among the feasible presentations the one which is most preferred by the author. For example, given a user with a black-and-white PDA, the video is constrained to be black and white. Problems of this type can be viewed abstractly as follows: there is a finite set of variables, each with a finite domain (e.g., display color of a particular message, display size, etc.). Various constraints are imposed on the value of these variables, such as the fact that video cannot be displayed in color on a black-and-white device, etc.. Finally, some preference relation on the possible assignments is provided. The goal is to find assignments to the set of variables that satisfy the constraints and are optimal with respect to the preferences.

The above example is only one of the many in which preferences and constraints coexists. Other examples of possible applications range from aircraft configuration optimization and aircraft design [MK90] to water supply management [Nis98].

The work presented in this chapter has previously appeared in the proceedings of the following international conferences and workshops.

- C. Domshlak, F. Rossi, K. B. Venable and T. Walsh. *Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques.* Proc. IJCAI 2003, pp. 215-220; Acapulco, Mexico; August 2003, Morgan Kauffman.
- F. Rossi, K. B. Venable and T. Walsh. *CP-networks: semantics, complexity, approximations and extensions.* Proc. CP02 Workshop: Soft'02 (Fourth International Workshop on Soft Constraints), Cornell University, Ithaca, NY, USA; Sept. 8th, 2002.
- S. Prestwich, F. Rossi, K. B. Venable and T. Walsh. *Constrained CP-nets.* Proc. Joint Annual Workshop of ERCIM/CoLogNet on Constraint Solving and Constraint Logic Programming (CSCLP'04), Lausanne, 2004.
- S. Prestwich, F. Rossi, K. B. Venable, T. Walsh. *Softly Constrained CP-nets.* Doctoral Paper in Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP-04), LNCS 3258, Springer, p. 806, Toronto, Canada, September 2004.

The chapter is structured as follows. In Section 4.2 we give the background on conditional preferential independence, ceteris paribus statements, and CP-nets. In Section 4.3 we consider a new property of a set of conditional preferences and give some complexity results on testing it. We then propose a procedure that maps a CP-net into a network of soft constraints and consider two possible approximations of the CP-net ordering obtained by choosing two different semirings. We show that the complexity of the mapping algorithm is polynomial in the size of the CP-net and that the preferences on the soft constraints network obtained can be chosen in a way that respects the ceteris paribus condition. In Section 4.4 we describe a new semantics for constrained CP-nets (that is, CP-nets plus hard and soft constraints) which applies also in the case multiple agents. For this semantics, we highlight the attractive complexity features and the drawbacks in the ordering induced. In Section 4.4.4 we consider the semantics for constrained CP-nets with hard constraints proposed in [BBD$^+$04a] and we use the techniques proposed in the first part of the section giving an algorithm that finds all (or just one) of the optimals according to that semantics. In Section 4.5 we consider the related work in the literature.

4.2 Background

In this section we give the background on conditional and unconditional preferences. In Section 4.2.1 we give basic notions of decision theory and multi-attribute utility theory highlighting the different forms of utility independence which have been proposed. In Section 4.2.2 we describe ceteris paribus preference statements and in 4.2.3 we consider CP-nets, a graphical representation of sets of conditional statements which have been inspired by Bayesian networks [BBHP99, BBD$^+$04a].

4.2.1 Decision theory and multiattribute utility theory

The field of decision theory deals with the making of decisions. As developed by philosophers, economist, and mathematicians over three hundred years, this discipline has developed many powerful ideas and techniques, which exert major influence virtually all over the cognitive and social sciences. Recently, decision theory has been successfully applied in the context of artificial intelligence. Many contributions have been given in qualitative decision theory [Doy80, BG96, DT99], multi-attribute utility theory [Fis69, KR76], and graphical models [BG95, Sho97, BBHP99, BBD$^+$04b].

In decision theory, a *decision* is defined as *a choice made by some entity of an action* *from some set of alternatives* [DT99]. Decision theory is not concerned about the actions, their nature or with how a set of them becomes available. The main focus is on the decision itself. Different branches of decision theory are specialized for studying respectively individual decisions and collective decisions. In this chapter we will focus on the first scenario: that is, where there is a single actor in the process. In Chapter 5, instead, we will consider the case in which there are several agents involved in the decision process.

In general, a decision is regarded as *good* if the decision maker beliefs will prove at least as good as the other alternatives. This is in some sense the definition of the ideal decision. In some cases a decision will be regarded as satisfactory if it is *good enough* for the decision maker, although there might be better ones. The formalization of how to evaluate decisions is characterized as alternatives which maximize *expected utility*, which is a notion involving both belief and goodness. More in detail, given a set of alternatives A, each alternative is associated with the outcome it represents. Let O stand for the set of all outcomes identified with any alternative. A measure u is defined on the set of outcomes O and it maps each outcome, say o, with its *utility*, $u(o)$. In *multi-attribute* utility theory [Fis69, KR76], a set of n attributes is given, $\mathbf{V} = \{X_1, \dots, X_n\}$ with corresponding domains $D(X_1), \dots, D(X_n)$. The set of outcomes is, then, defined as $O = D(X_1) \times \cdots \times D(X_n)$, that is, the Cartesian product of the domains. The size of the domain of the utility function is, thus exponential and it can be expensive to describe. However there are some conditions under which the representation can be more compact. In general, one specifies some *independence conditions* and then formulates a *representation theory* that allows, when the conditions are satisfied, for the utility functions to be in a compact form. We will now consider one of such conditions and the corresponding representation which has attracted a lot of attention in the AI community. The following definitions follow those given in [Sho97].

Given a utility function u defined on the set of outcomes O, it induces an ordering on the set of probability distributions defined on O in the following way:

$$p_1 \succ p_2 \Longleftrightarrow \sum_{o \in O} p_1(o)u(o) \geq \sum_{o \in O} p_2(o)u(o).$$

Such ordering can be made *conditional* if it depends on the value that is assigned to a certain attribute X:

$$p_1 \succ p_2 | X = x \Longleftrightarrow \sum_{o \in O} p_1(o|X = x)u(o) \geq \sum_{o \in O} p_2(o|X = x)u(o).$$

We will indicate with $\mathbf{X}$ a subset of attributes and with $\mathbf{x}$ an assignment to the attributes in $\mathbf{X}$. We will write $\mathbf{xy}$ to indicate an assignment to $\mathbf{X} \cup \mathbf{Y}$. Consider $\mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ such that $\mathbf{Z} = \mathbf{V} - \mathbf{Y}$.

Definition 4.1 (general utility independence) $\mathbf{Y}$ *is* utility independent *of* $\mathbf{Z}$ *if the preferences on probability distributions on* $\mathbf{Y}$ *are not conditional on* $\mathbf{Z}$*.*

We now define additive utility independence, a condition which has been widely used due to the tractability of its representation.

Definition 4.2 (additive utility independence) *The attributes* $\mathbf{V} = \{X_1, \ldots, X_n\}$ *are* additive utility independent *if preferences over probability distributions on* $\mathbf{V}$ *depend only on the marginal probability of each variable and not on their joint probability distributions.*

The representation theorem, corresponding to this condition, gives a complexity of n simple utility functions, n constants and n additions and multiplications.

Definition 4.2 can be extended to a disjoint partition of $\mathbf{V}$, $\mathbf{Z}_i \ldots \mathbf{Z}_k$, meaning that $\mathbf{Z}_1 \ldots \mathbf{Z}_k$ are additively independent if for any two probability distributions p_1 and p_2 on $\mathbf{V}$, with the same marginals on $\mathbf{Z}_1, \ldots \mathbf{Z}_k$ then, $p_1 \succeq p_2$ and $p_2 \succeq p_1$ both hold.

From the above definition, the additive form of a utility function is obtained:

$$u(\mathbf{V}) = \sum_{i=1}^{k} u_i(\mathbf{Z}_i).$$

There is also a *conditional additive utility independence*. It requires that additive independence holds whenever some subset of variables are held fixed.

Definition 4.3 *Let* $\mathbf{X}$, $\mathbf{Y}$, *and* $\mathbf{Z}$ *be a disjoint partition of* $\mathbf{V}$. $\mathbf{X}$ *and* $\mathbf{Y}$ *are* conditionally additively independent *given* $\mathbf{Z}$ *iff, for any assignment* $\mathbf{z}$ *to* $\mathbf{Z}$, $\mathbf{X}$ *and* $\mathbf{Y}$ *are additively independent in the preference structure* $\succ_{\mathbf{z}}$ *over* $\mathbf{X} \cup \mathbf{Y}$, *obtained from* $\succeq$ *by fixing the value of* $\mathbf{Z}$ *to* $\mathbf{z}$.

In [BG95] it is shown that a multi-attribute utility function u is additively independent over disjoint $\mathbf{X}$ and $\mathbf{Y}$ given $\mathbf{Z}$ ($\mathbf{Z} = \mathbf{V} - (\mathbf{X} \cup \mathbf{Y})$) iff there exist some functions u_1 and u_2, such that:

$$u(\mathbf{V}) = u_1(\mathbf{X}, \mathbf{Z}) + u_2(\mathbf{Y}, \mathbf{Z}).$$

It should be noticed that conditional additive independence is not a strong requirement as additive independence. Moreover, in [BG95] the authors have shown that the decomposition yielded by conditional additive independence can significantly improve the efficiency of computing expected utility, and that it is useful when using graphical modeling. In [BG95], a graphical model in which the representation of conditional independence is obtained by analogy with Bayesian networks [AP91], is proposed. In particular, they show that every utility function can be mapped into an undirected graph G, where the nodes are the attributes, and such that $\mathbf{X}$ and $\mathbf{Y}$ are conditionally additively independent given $\mathbf{Z}$, iff $\mathbf{Z}$ *separates* $\mathbf{X}$ from $\mathbf{Y}$, i.e. every path from a node in $\mathbf{X}$ to a node in $\mathbf{Y}$ passes through a node in $\mathbf{Z}$. The main difference with Bayesian networks is that the graph is undirected. Definitely, the work in [BG95] represents the seminal paper that put forward the research on graphical modeling of preferential structure. Many have followed this line of study, see for example [Sho97, MS99].

4.2.2 Ceteris paribus preferences

Sometimes, however, utility functions cannot be explicitly specified. In this case one should resort to their, less quantitative, or even non-quantitative forms of preference representation. Ideally, such information should be obtained from the user with minimum effort. In order to do so, the information should be elicited in the form of natural and relatively simple statements in a process that can be automated. Philosophers have speculated on qualitative preferential statements which could be formulated in intuitive ways since 1957 when the pioneering work by Hallden [Hal57] appeared. In the following years, many pursued this investigation, among which [Cas58, KM75, Han01] and especially the work by von Wright [vW63, vW72]. In these papers, the author defined *ceteris paribus preference* of some outcome p to q as the fact that "any given world which contains p but not q is preferred to a total state of the world which differs from the first in that it contains q but not p, but otherwise is identical with it" [vW63].

Surprisingly, only much later, ceteris paribus preferences have picked up interest by scientist of the AI community. In [DW94] the authors adopted the framework proposed by von Wright as a foundation of their work on ceteris paribus comparatives. However, to the best of our knowledge, the first work proposing a structured model of a subset of ceteris paribus preferential statements is [BBHP99]. It is following this paper that we describe the fundamentals of ceteris paribus preferential independence and Conditional Preference networks. As in multi-attribute utility theory, in [BBHP99] the authors assume a set of variables $\mathbf{V} = \{X_1, ..., X_n\}$ (the attributes) with finite domains $D(X_1), ..., D(X_n)$. The goal is to allow the agent making decisions to specify a preference ranking over complete assignments on $\mathbf{V}$. Each such assignment is referred to as an *outcome*, o, and the set of all outcomes is denoted as O.

Definition 4.4 (preference ranking $\succeq$) *A preference ranking is a total preorder (i.e., a reflexive, transitive and complete binary relation) over the set of outcomes. $o_1 \succeq o_2$ means that outcome o_1 is equally as good as or preferred to o_2. $o_1 \succ o_2$, which holds if $o_1 \succeq o_2$ and $o_2 \not\succeq o_1$, denotes the fact that outcome o_1 is strictly preferred to o_2.*

As we have mentioned above, the direct assessment of a preference relation is generally infeasible due to the exponential size of O and, in this respect, independence assumptions play a key role in such specifications.

In [BBHP99] the authors adapt the standard notions of independence in multi-attribute utility theory which we have briefly described in Section 4.2.1, by providing qualitative variants of (arguably more familiar) quantitative notions of utility independence.

Definition 4.5 (preferential independence) *A subset of variables* $\mathbf{X}$ *is* preferentially independent *of its complement* $\mathbf{Y} = \mathbf{V} - \mathbf{X}$ *iff, for all* $\mathbf{x}_1, \mathbf{x}_2 \in D(\mathbf{X})$*,* $\mathbf{y}_1, \mathbf{y}_2 \in D(\mathbf{Y})$*, we have:*

$$\mathbf{x}_1\mathbf{y}_1 \succeq \mathbf{x}_2\mathbf{y}_1 \Leftrightarrow \mathbf{x}_1\mathbf{y}_2 \succeq \mathbf{x}_2\mathbf{y}_2.$$

If this relation holds, x_1 it is said to be *preferred to* x_2 *ceteris paribus* (all else being equal). This implies that the preferences for different values of X do not depend on the values assigned to other variables given that these remain fixed. The analogous notion of conditional preferential independence is defined as follows.

Definition 4.6 (conditional preferential independence) *Let* $\mathbf{X}, \mathbf{Y}$ *and* $\mathbf{Z}$ *be a disjoint partition of* $\mathbf{V}$*.* $\mathbf{X}$ *is* conditionally preferentially independent *of* $\mathbf{Y}$ *given* $\mathbf{z} \in D(\mathbf{Z})$ *iff, for all* $\mathbf{x}_1, \mathbf{x}_2 \in D(\mathbf{X})$*,* $\mathbf{y}_1, \mathbf{y}_2 \in D(\mathbf{Y})$*, we have:*

$$\mathbf{x}_1\mathbf{y}_1\mathbf{z} \succeq \mathbf{x}_2\mathbf{y}_1\mathbf{z} \Leftrightarrow \mathbf{x}_1\mathbf{y}_2\mathbf{z} \succeq \mathbf{x}_2\mathbf{y}_2\mathbf{z}.$$

In other words, the condition under which **X** is preferentially independent of **Y** is that **Z** is assigned value **z**. If such a relation holds for all assignments on **Z**, **X** is said to be conditionally preferentially independent of **Y** given **Z**. In what follows if this condition holds, we will write:

$$\mathbf{z} : \mathbf{x}_1 \succ \mathbf{x}_2$$

and we will refer to this as the standard form of a ceteris paribus conditional preference statement.

4.2.3 Conditional Preference networks (CP-nets)

CP-nets, introduced in [BBHP99], are a tool for compactly representing ceteris paribus qualitative preference relations. This graphical model exploits conditional preferential independence in structuring a decision maker's preferences under a ceteris paribus assumption. CP-nets are the first graphical model based on the notions of purely qualitative preferential independence captured by the ceteris paribus assumption, and bear a superficial similarity to Bayesian networks [AP91]. However, as pointed out in [BBHP99], the nature of the relationship between nodes within a CP-net is generally quite weak compared with the probabilistic relations in Bayesian nets.

In the context of CP-nets, during preference elicitation, the agent must specify, for each variable X, the set of *parent variables* $Pa(X)$ that can affect her preferences over the values of attribute X. Then for each possible assignment to $Pa(X)$, the agent must give a preference ordering for the values of X, ceteris paribus. The above requirements imply that, for any attribute X, it is conditionally preferentially independent of $\mathbf{V} - X$, given $Pa(X)$. This information is used to create the graph of the CP-net in which each node X has each variable in $Pa(X)$ as its immediate predecessors, that is, there is an edge from each variable in $Pa(X)$ to X. The conditional preference ranking over the values of X is captured by a *Conditional Preference Table (CPT)* which annotates the node X in the CP-net. That is, for each assignment to $Pa(X)$, CPT(X) specifies a total order over $D(X)$, such that for any two values $x_i, x_j \in D(X)$, either $x_i \succ x_j$ or $x_j \succ x_i$. We now will write the formal definition of a CP-net given in [BBHP99].

Definition 4.7 (CP-net) *A* CP-net *over variables* $\mathbf{V} = X_1, \ldots, X_n$ *is a directed graph* G *over* $X_1, ..., X_n$ *where each node,* X_i*, is annotated with a conditional preference table, CPT(*X_i*). Each conditional preference table CPT(*X_i*) associates a total order* $\succ^i_{\mathbf{u}}$ *with each instantiation* $\mathbf{u}$ *of* $Pa(X_i) = \mathbf{U}$.

Let us now consider an example taken from [BBD+04b].

Example 4.1 Assume the agent has the following preference on which evening dress to wear at a party. He must decide which pants, jacket and shirt to wear. He prefers black to white as a color for both the jacket and the pants, while his preference between red and white shirts is conditioned on the combination of jacket and pants: if they are the same color, then a white shirt will make the outfit too colorless, thus he prefers a red shirt. On the other hand, if the jacket and pants are different colors, then a red shirt will probably appear too flashy, thus he prefers a white shirt. It is easy to see that this example can be modeled using three variables: J (jacket), S (shirt) and P (pants). The domain of variables J and S are the same $D(J) = D(P) = \{b, w\}$, where b stands for $black$ and w stands for $white$. The domain of variable S is $D(S) = \{r, w\}$ where r stands for red. The set of ceteris paribus statements corresponding to the agent's preferences:

$$J = b \succ J = w$$

$$P = b \succ P = w$$

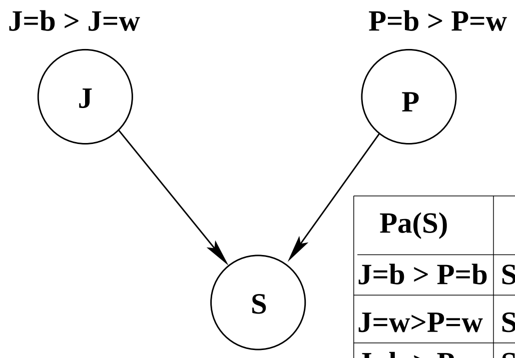
$$(J = b \wedge P = b) \vee (J = w \wedge P = w) : S = r \succ S = w$$

$$(J = b \wedge P = w) \vee (J = w \wedge P = b) : S = w \succ S = r.$$

The CP-net corresponding to Example 4.1 is depicted in Figure 4.1.

Recently, the effects of relaxing the total order assumption have been considered. In [BBD$^+$04b] the authors study also CP-nets with incompletely specified preferences and indifference. In Chapter 5 we will describe a similar extension of CP-nets, namely *partial CP-nets*, which are used in a multiple agent scenario.

In [BBD$^+$04a] the semantics of CP-nets is given in terms of the set of preference rankings that are consistent with the set of ordering constraints imposed by the conditional preference tables. The preference information captured by a CP-net N, induces an ordering over the complete assignments to variables in the



Pa(S)	S
J=b > P=b	S=r > S=w
J=w>P=w	S=r > S=w
J=b > P=w	S=w > S=r
J=w>P=b	S=w > S=r

Figure 4.1: CP-net corresponding to Example 4.1.

network. These statements are generally not complete, that is, they do not determine a unique preference ordering. Those orderings consistent with N can be viewed as possible models of the user's preferences, and any preference assertion that holds in all such models can be viewed as a consequence of the CP-net.

Definition 4.8 (satisfiability) *Let* N *be a CP-net over variables* $\mathbf{V}$, $X \in \mathbf{V}$ *be some variable, and* $\mathbf{U} \subset \mathbf{V}$ *be parents of* X *in* N. *Let* $\mathbf{Y} = \mathbf{V} - (\mathbf{U} \cup \{X\})$. *Let* $\succ_{\mathbf{u}}$ *be the ordering over* $D(X)$ *in* $\mathsf{CPT}(X)$ *for instantiation* $\mathbf{u} \in D(\mathbf{U})$ *of parents of* X. *Finally let* $\succ$ *be a preference ranking over* $D(\mathbf{V})$. *Then,*

1. *A preference ranking* $\succ$ *satisfies* $\succ_{\mathbf{u}}$ *iff* $\mathbf{yu}x_i \succ \mathbf{yu}x_j$, *for all* $\mathbf{y} \in D(\mathbf{Y})$, *whenever* $x_i \succ_{\mathbf{u}} x_j$.

2. *A preference ranking* $\succ$ *satisfies* $\mathsf{CPT}(X)$ *iff it satisfies* $\succ_{\mathbf{u}}$ *for each* $\mathbf{u} \in D(\mathbf{U})$.

3. *A preference ranking* $\succ$ *satisfies CP-network* N *iff it satisfies* $\mathsf{CPT}(X)$ *for each variable* X.

4. *A CP-net is satisfiable if there is a preference ranking that satisfies it.*

Thus, a CP-net N is satisfied by $\succ$ if $\succ$ satisfies each of the conditional preferences expressed in the statements under the ceteris paribus condition. In [BBD+04a] the authors show that acyclicity is a sufficient condition for satisfiability.

Theorem 4.1 *Every acyclic CP-net is satisfiable [BBD*$^+$*04a].*

The above definition allows one to define when a preference is *entailed* by a CP-net.

Definition 4.9 (entailement) *The preference* $o \succ o'$ *is a consequence of a CP-net* N*, written* $N \vdash o \succ o'$ *, iff* $o \succ o$ *holds in all preference orderings consistent with the ceteris paribus preference statements encoded by the CPTs of* N*.*

The set of consequences $o \succ o'$ of an acyclic CP-net constitutes a partial order over the outcomes: o is preferred to o' in this ordering iff $N \vdash o \succ o'$. This partial order can be represented by an acyclic directed graph, referred to in [BBD$^+$04a] as the *induced preference graph*.

Definition 4.10 (induced preference graph) *Given a CP-net* N*, the corresponding* induced preference graph *is a graph such that:*

- *the nodes correspond to the complete assignments to the variables of* N*;*
- *there is an edge from node* o *to node* o' *iff assignment* o *differs from assignment* o' *only in the value of a single variable* X *and, given the values assigned by* o *and* o' *to* $Pa(X)$*, the value assigned by* o *to* X *is preferred to the value assigned by* o' *to* X*.*

Moreover, the transitive closure of this graph specifies the (asymmetric) partial order over outcomes induced by the CP-net. Thus, $N \vdash o \succ o'$ iff there is a directed path from o to o' in the induced preference graph for N.

Figure 4.2 shows the induced preference ordering corresponding to the CP-net of Example 4.1.

Given the ordering induced by the CP-net, it is possible to define the notion of consistency [BBHP99].

Definition 4.11 (consistency) *A CP-net* N *is consistent iff for every two outcomes* $o_1, o_2 \in O$*, either* $o_1 \succ o_2$ *or* $o_2 \succ o_1$*.*

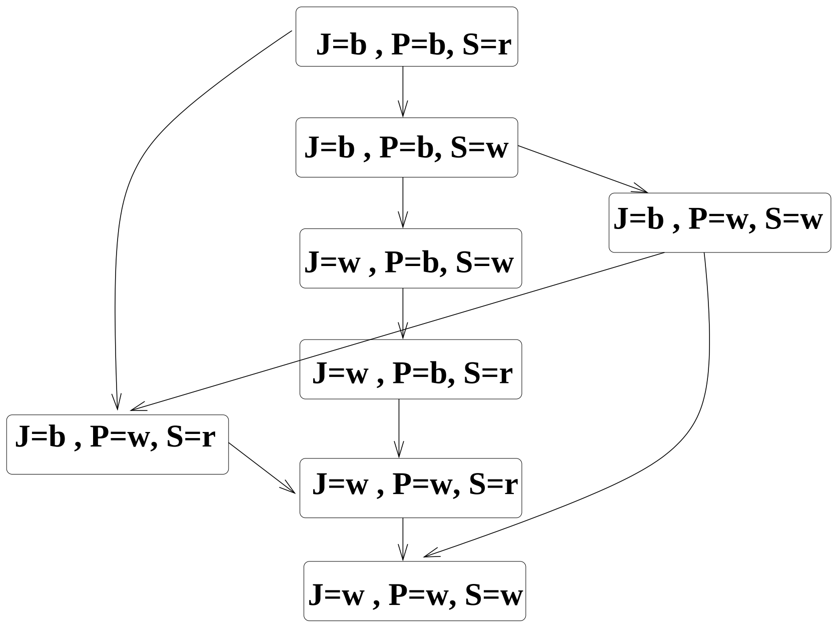


Figure 4.2: Induced preference graph of the CP-net of Example 4.1.

From Theorem 4.1 it follows that every acyclic CP-net is consistent. In fact, given two outcomes o_1 and o_2, $o_1 \succ o_2$ in the CP-net N if N entails $o_1 > o_2$, that is, $o_1 > o_2$ in all its satisfying orderings. Thus, in order for it to be inconsistent, it must be that $o_1 > o_2$ and $o_2 > o_1$ in all its orderings. Hence, there must be a cycle in each of them. However this is not true if the CP-net is satisfiable since there must be at least an ordering consistent with it.

As noted in [BBD$^+$04b], cyclic CP-nets are of potential interest in some domains, even if their applicability is limited by the fact that asymmetry of the induced preference relation is not always guaranteed in the presence of cycles. The problem of checking the consistency of a cyclic CP-net is believed to be NP-hard.

One of the main features of a preference reasoning system should be the ability of answering queries about the preferences elicited from the agent. In [BBHP99] the authors identify two fundamental types of queries:

- *Outcome optimization*, that is, determining the best outcome according to the information in the CP-net.

- *Outcome comparison*, that is, comparing outcomes with respect to the preferences in the CP-net.

General procedures for both of these tasks with respect to a CP-net have been introduced in [BBHP99, BBD+04a]. However, while outcome optimization with respect to an acyclic CP-net has been shown to be computationally easy, outcome comparison is NP-hard.

We will now give a brief overview of the relevant results on reasoning about preferences modeled by a CP-net [BBD+04a].

Outcome optimization queries

Given an acyclic CP-net N , the authors in [BBHP99] show that the best outcome among those preference rankings that satisfy N can be easily determined. The intuition is to, simply, sweep through the network from top to bottom (i.e., from ancestors to descendants) setting each variable to its most preferred value given the instantiation of its parents. In fact, in [BBD+04a] it is shown that, while the network does not usually determine a unique ranking, it does determine a unique best outcome. Formally, outcome optimization queries can be answered using the following Forward-sweep procedure, which is linear in the size of the network.

Pseudocode of Forward-sweep

```
1. input acyclic CP-net $N$;
2. for all variables of $N$ in any topological order, $X_1, \dots, X_n$, from the roots to
   the leafs:
3. assign to $X_i$ the most preferred value given the assignments to $X_1, \dots, X_{i-1}$;
```

Figure 4.3: Algorithm Forward-sweep proposed in [BBHP99] to find the best outcome of a CP-net.

In [BBD+04a] it is proven that this procedure, exploiting the considerable power of the ceteris paribus semantics, always finds an optimal outcome in linear time given an acyclic CP-net. As mentioned above, in [BBD+04a] the authors consider CP-nets with partially specified preferences and, thus, with more than one optimal outcome. They show that computing the set of all undominated outcomes with respect to such acyclic CP-nets is linear in the size of the output.

In Chapter 5 we will consider the case of a multiagent scenario, where the CP-net of an agent might be connected to that of another agent. We will show that also in this case partial orders are desirable and that similar results for optimization queries hold.

Outcome comparison queries

Another basic query which should be supported by a preferential reasoning system is comparison between outcomes. Two outcomes o and o' can be in one of three possible relations with respect to a CP-net N :(1) $N \vdash o \succ o'$; (2) $N \vdash o' \succ o$; (3) $N \not\vdash o \succ o'$ and $N \not\vdash o' \succ o$, in which case o and o' are said to be incomparable ($o \bowtie o'$). In [BBD+04a] the authors suggest two distinct ways to compare two outcomes using a CP-net:

1. *Dominance queries*: given a CP-net N and a pair of outcomes o and o', ask whether $N \vdash o \succ o'$. If this relation holds, o is preferred to o', and o is said to *dominate* o' with respect to N.

2. *Ordering queries*: given a CP-net N and a pair of outcomes o and o', ask if $N \not\vdash o' \succ o$. If this relation holds, there exists a preference ordering consistent with N in which $o \succ o'$.

Answering an ordering query allows to establish whether it is consistent with the knowledge expressed by N to order o "above" o' i.e., to assert that o is preferred to o'. In such a case, o is said to be *consistently orderable* over o' with respect to N.

As noted by [BBD+04a], ordering queries are a weak version of dominance queries, but they are sufficient in many applications where, given a pair of outcomes o and o' , one can be satisfied with knowing that o is not worse than o'.

Ordering queries with respect to acyclic CP-nets can be answered in time linear in the number of variables [BBD+04a]. Informally, ordering a pair of outcomes o and o' is based on a procedure that, given a query $N \vdash o \not\succ o'$, returns, in linear time, either "yes" or "don't know". However, if $o \neq o'$, then this procedure returns "yes" for at least one of the two queries $N \vdash o \not\succ o'$ and $N \vdash o' \not\succ o$.

Dominance queries are an important part of the research on CP-nets [DB02]. Comparisons between outcomes that differ in the value of a single variable X only are easy, since such information is provided directly by the CPT of X. One can view the improved outcome as a product of a single *improving "flip"* in the value of X. The only current technique for solving more general dominance queries is based on the idea of a *flipping sequence* [BBHP99]. An *improving* flipping sequence is a sequence of progressively better outcomes, each obtained from its predecessor via a single value flip. In [BBHP99] this idea is formalized as follows.

Definition 4.12 (improving flip) *Let* N *be a CP-net over variables* $\mathbf{V}$, *with* $X \in \mathbf{V}$, $\mathbf{U}$ *the parents of* X, *and* $\mathbf{Y} = \mathbf{V} - (\mathbf{U} \cup \{X\})$. *Let* $\mathbf{u}x'\mathbf{y} \in D(\mathbf{V})$ *be any outcome. An* improving flip *of outcome* $\mathbf{u}x'\mathbf{y}$ *with respect to variable* X *is any outcome* $\mathbf{u}x\mathbf{y}$ *such that* $x \succ x'$ *given* $\mathbf{u}$ *in network* N.

Note that an improving flip with respect to X may not exist if x is the most preferred value of X given $\mathbf{u}$.

Definition 4.13 (improving flipping sequence) *An* improving flipping sequence *with respect to a CP-net* N *is any sequence of outcomes* $o_1, \ldots, o_k$ *such that* o_{i+1} *is an improving flip of* o_i *with respect to some variable (for each* $i < k$*).*

Example 4.2 Consider again the scenario in Example 4.1. Passing from $(J = b, P = b, S = w)$ to $(J = b, P = b, S = r)$ is an improving flip, while $(J = b, P = w, S = r) \prec (J = b, P = w, S = w) \prec (J = b, P = b, S = w) \prec (J = b, P = b, S = r)$ is an improving flipping sequence from outcome $(J = b, P = w, S = r)$ to $(J = b, P = b, S = r)$.

Worsening flips and worsening flipping sequences are defined analogously. From the above definition it follows that an improving flipping sequence for a pair of outcomes o and o' is any improving sequence $o_1, \ldots, o_k$ with $o = o_1$ and $o' = o_k$. Informally, each improving flipping sequence from an outcome o to another o' corresponds to a directed path from the node o to the node o' in the preference graph induced by N. In [BBHP99] is shown that the existence of an improving flipping sequence from o to o' implies that all the total orderings of the outcomes that are consistent with N must prefer o to o'.

Moreover, the lack of a flipping sequence from o to o' implies that there is no total ordering in which o is preferred to o' satisfying N. Thus, the following theorem holds.

Theorem 4.2 *Given a CP-net N and a pair of outcomes o and o', $N \vdash o \succ o'$ iff there is an improving flipping sequence with respect to N from o to o' [BBHP99].*

Improving and worsening flipping sequences are used in a search procedure for answering dominance queries. For example, in [BBHP99] an *improving search* is defined as follows. Given any CP-network N, and a query $N \vdash o \succ o'$, *the improving search tree* is rooted at $o' = y_1 y_2 \cdots y_n$ and the children of any node $\bar{o} = z_1 z_2 \cdots z_n$ in the search tree are those outcomes that can be reached by changing one feature value z_i to z_i' such that $z_i' \succ z_i$ given the values z_j, $j < i$. It is shown that $N \vdash o \succ o'$ iff there exists a path from o' to o in the improving search tree. Thus, any complete search procedure, i.e. a procedure guaranteed to examine every branch of the search tree, will be a sound and complete query answering procedure. In [BBHP99] the analogous definition of *worsening search tree* is given.

Furthermore, [BBHP99] provides simple rules that allow one to make deterministic moves in search space (i.e., to choose flips that need not be backtracked over, or reconsidered) without impacting completeness of the search procedure.

In addition, in [BBHP99] several heuristics for exploring the search tree are proposed. For example, the *rightmost heuristic* requires that the variable whose value one flips when deciding which child to move to is the rightmost variable that can legally be flipped. In multivalued domains, another useful heuristic is the *least improving heuristic* (or in worsening searches, the least worsening heuristic): when the rightmost value can be flipped to several improving values given its parents, the improving value that is least preferred is adopted.

Despite these efforts, it turns out that answering dominance queries even with respect to CP-nets with only Boolean variables is hard in general. In particular, in [BBD$^+$04a] it is proved that:

- When the binary CP-net forms a directed tree, the complexity of dominance testing is quadratic in the number of variables.
- When the binary CP-net forms a polytree (i.e., a singly connected graph with no undirected cycles), dominance testing is polynomial in the size of the CP-net description.
- When the binary CP-net is directed-path singly connected (i.e., there is at most one directed path between any pair of nodes), dominance testing is NP-complete. The problem remains hard even if the node in-degree in the network is bounded by a small constant.
- Dominance testing remains in NP-complete if the number of alternative paths between any pair of nodes in the CP-net is polynomially bounded.

4.3 Approximating CP-nets via Soft Constraints

This section provides a first connection between the CP-nets and soft constraints machinery. In particular we describe a framework to reason simultaneously about qualitative conditional preference statements and hard and soft constraints. It is not difficult to understand that such a coexistence is not rare in real life problems. For example, in product configuration, the producer has hard (e.g., component compatibility) and soft (e.g., supply time) constraints, while the customer has preferences over the product features [SW98]. We first investigate the complexity of reasoning about qualitative preference statements, addressing in particular preferential consistency (Section 4.3.1). To tackle the complexity of preference reasoning, we then introduce two approximation schemes based on soft constraints (Section 4.3.2). Finally, we compare the two approximations in terms of both expressivity and complexity (Section 4.3.2).

We start by giving the following result that states the fundamental incomparability of CP-nets and semiring-based Soft constraints in terms of the orderings induced on the outcomes. Consider the following definition.

Definition 4.14 (equivalence) *Consider a CP-net* N *and soft constraint problem* P *defined on the same set of variables* V. P *and* N *are* equivalent *if and only if they induce they same ordering on the set of assignments of the variables in* V.

We will now show that there are CP-nets from which it is not possible to build an equivalent SCSP, and vice-versa. In other words, the two formalisms are incomparable with respect to the above notion of equivalence.

Theorem 4.3 *There are CP-nets for which it is not possible to build an equivalent SCSP. Conversely, there are SCSPs for which it is not possible to build an equivalent CP-net.*

Proof: To see that there are CP-nets for which it is not possible to build an equivalent SCSP, it is enough to consider any CP-net whose induced ordering is a preorder but not a partial order. The CP-net of Figure 4.14, whose induced ordering can be seen in Figure 4.15, is one example.

To see that there are SCSPs for which it is not possible to build an equivalent CP-net, it is enough to consider the fuzzy SCSP with three binary-valued variables, say A, B, and C, with the following soft constraints:

- a constraint connecting all three variables, and stating that $\bar{a}b\bar{c}$ has preference 0.9, $a\bar{b}c$ has preference 0.8, $ab\bar{c}$ has preference 0.7, and all other assignments have preference 1;

- a constraint over variables A and B such that ab has preference 0.9 and all other assignments have preference 1.

The induced ordering is total and has $a\bar{b}c$ with preference 0.8, $ab\bar{c}$ with preference 0.7, and all other assignments are above in the ordering (they have preference 0.9 or 1). Let us now assume that there is an equivalent CP-net, that is a CP-net with this ordering as induced ordering. Then, there must be a worsening path from $a\bar{b}c$ to $ab\bar{c}$. However, since the two assignments differ for more than one flip, this is possible only if there is at least another assignment which is strictly worse than $a\bar{b}c$ and strictly better than $ab\bar{c}$. This is not true given the ordering, so there cannot be such a CP-net.

One could say that preorders and partial orders do not really encode different information, because any preorder can be directly mapped onto a partial order simply by transforming each cycle into tied elements of the partial order. Thus we could consider a more tolerant notion of equivalence, where a CP-net inducing a preorder O and an SCSP inducing a partial order O' are considered equivalent if O' can be obtained from O as stated above. Let us call this notion *pre-equivalence*.

One could think that, if we allow this notion of equivalence, the SCSP framework is more expressive than CP-nets. In fact, SCSPs can induce any partial order, while (as noted in the proof of the above theorem) CP-nets do not. But this is true only if we allow for exponential-time mappings from CP-nets to SCSPs. If we restrict ourselves to polynomial time mappings then the following theorem holds [MRSV04].

Theorem 4.4 *Assuming $P \neq NP$, there are CP-nets for which it is not possible to build in polynomial time a pre-equivalent SCSP, and there are SCSPs for which it is not possible to build an pre-equivalent CP-net.*

Proof: Assume that, for all CP-nets, it is possible to build in polynomial time a pre-equivalent SCSP. Then, dominance testing in the given CP-net would become a polynomial problem since it would be sufficient to map in polynomial time the CP-net into the SCSP and then to perform the dominance test in the SCSP, which can be done in polynomial time. Since dominance testing in CP-nets is known to be NP-hard [BBD+04a], this would contradict the hypothesis that $P \neq NP$. The other direction of the theorem follows directly from Theorem 4.3. In fact, in this direction we never start from a preorder but always from a partial or total order.

We have just proved that CP-nets and soft constraints are incomparable with respect to the induced ordering. This means that, if we want the soft constraint machinery to reason about CP-net preferences efficiently, in general we must approximate the ordering of the given CP-net. In the Section 4.3.2 we will propose two such approximations.

4.3.1 Complexity results

Given a set of preference statements Ω extracted from a user, we might be interested in testing *consistency* of the induced preference relation. In general, there is no single notion of preferential consistency [Han01]. In [BBHP99], a CP-net N was considered consistent iff the preorder $\succ$ induced by N is *asymmetric*, i.e. there exist at least one total ordering of the outcomes consistent with $\succ$ (see Section 4.2.3). However, in many situations, we can ignore cycles in the preference relation, as long as these do not prevent a rational choice, i.e. there exist an outcome that is not dominated by any other outcome. In preference logic [Han01], the notions of "consistency as satisfiability" and "consistency as asymmetry" correspond to the notions of *eligibility* and *restrictable eligibility*, respectively. Thus, in what follows, we refer to the aforementioned property as *eligibility*.

Definition 4.15 (eligibility) *A set of of preference statements Ω is* eligible *if the ordering induced $\succ$ on the set of outcomes O, is such that $\exists o$ with $o' \not\succ o, \forall o' \in O$.*

In the following theorem we consider the relation between the notion of consistency and the notion of eligibility.

Theorem 4.5 *Given a CP-net, if it is consistent then it is also eligible. However the converse does not hold in general.*

Proof: $\mathtt{Consistent} \Rightarrow \mathtt{eligible}$. By the definition of consistency in [BBHP99] (Definition 4.11 in Section 4.2.3), N is consistent iff the preorder induced on the set of outcomes is asymmetric, i.e. for every two outcomes $o_1, o_2 \in O$, $o_1 \succ o_2$ and $o_2 \succ o_1$ cannot hold simultaneously. In other words, the induced ordering is a partial order. Every finite partial order has at least a maximal element, thus the CP-net N is eligible.

$\mathtt{Eligible} \not\Rightarrow \mathtt{consistent}$. It follows from the fact that not every set with a maximal element is a partial order.

We will consider two cases: when the set Ω of preference statements induces a CP-net and, more generally, when preferences can take any form (and may not induce a CP-net). When Ω defines an acyclic CP-net, the partial order induced by Ω is asymmetric [BBHP99] (see Theorem 4.1 Section 4.2.3). However, for cyclic CP-nets, asymmetry is not guaranteed. In the more general case, we are given a set Ω of conditional preference statements without any guarantee that they define a CP-net. Let the *dependence graph* of Ω be defined similarly to the graphs of CP-nets: the nodes stand for problem features, and a directed arc goes from X_i to X_j iff Ω contains a statement expressing preference on the values of X_j conditioned on the value of X_i.

Let us consider the following examples. In the first one, both the dependence graph and the induced graph are cyclic, in the second one only the induced graph is cyclic. In both cases however the set of statements is eligible.

Example 4.3 Consider the set of statements over Boolean variables: $\Omega = \{a \succ \bar{a}, \bar{a} : b \succ \overline{b}, \bar{b} : c \succ \bar{c}, \bar{c} : \bar{b} \succ b, \bar{a} : \bar{c} \succ c\}$. Clearly this set of statements does not identify a CP-net since it is not the case that for every attribute X a total order is given for every possible assignment of its parents. As it can be seen in Figure 4.4, which shows the dependence and the corresponding induced graph, the induced ordering is not asymmetric since there is a cycle. Nevertheless, there is an undominated outcome: abc.

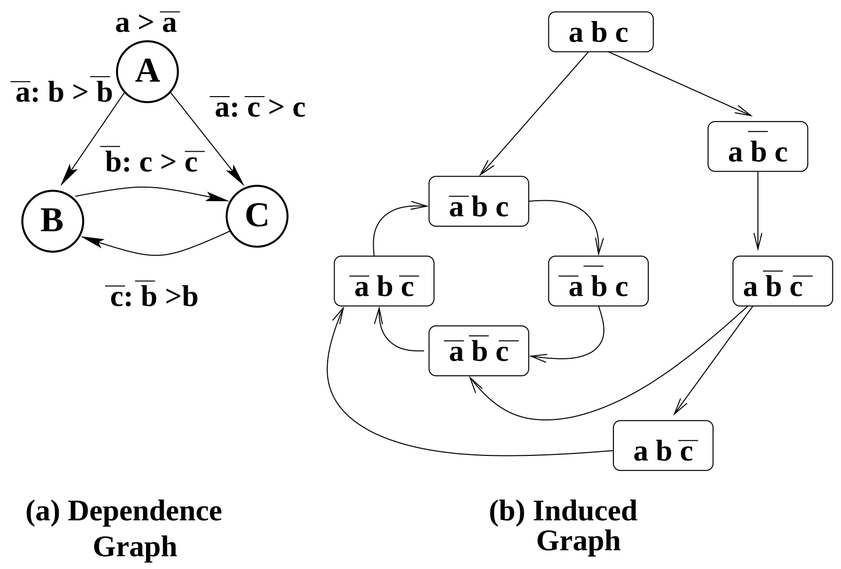


Figure 4.4: Dependence and induced graph of the set of statements in Example 4.3.

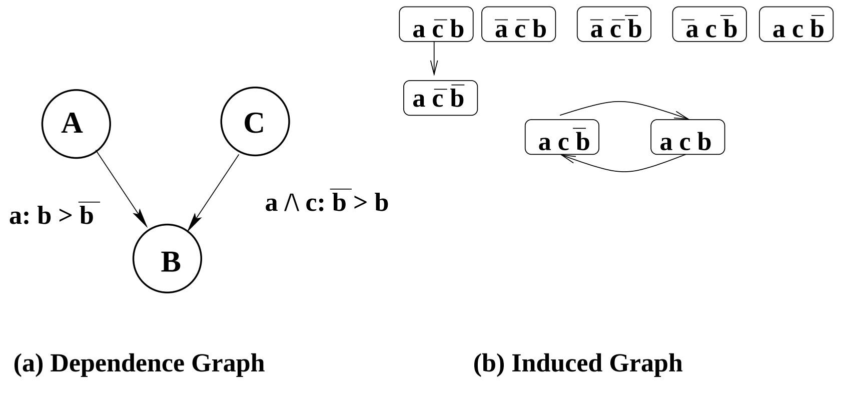


Figure 4.5: Dependence and induced graph of the set of statements in Example 4.4.

Example 4.4 Consider the set of ceteris paribus statements defined on Boolean variables: $\Omega = \{a : b \succ \overline{b}, a \wedge c : \overline{b} \succ b\}$. Clearly this set does not induce a CP-net, since the two conditionals are not mutually exclusive. The preference relation induced by Ω is not asymmetric, as shown in Figure 4.5, despite the fact that the dependence graph of Ω is acyclic. Again we can conclude that the set Ω above is not asymmetric, but it is eligible, for example the assignment $a\overline{c}b$ is undominated.

The above examples prove that a set of preference statements can be eligible even if not consistent. Assuming that the *true* preferences of the user are asymmetric, given such a set of statements, one can try to continue questioning the user, trying to eliminate the in-asymmetry that has been detected. However, even ignoring the fact that detecting asymmetry can be hard in general, in online configuration tasks that are of our interest, long interactions with the user should be prevented as much as possible.

Given an eligible set of statements, we can instead prompt the user with one of the undominated assignments without further refinement of its preference relation. Theorem 4.6 shows that, in general, determining eligibility of a set of statements is NP-complete. On the other hand, even for CP-nets, determining asymmetry is not known to be in NP [DB02].

Theorem 4.6 ELIGIBILITY *of a set of conditional preference statements Ω is* NP*-complete.*

Proof: An assignment is a polynomial-size witness in the size of the input, that is, the set of conditional preference statements. In fact, it is linear in n, where n is the number of features. Checking for non-dominance takes time linear in the size of the witness. In fact, it is sufficient to consider just the outcomes which differ from the witness only on the value of a single feature (i.e., outcomes that are one flip away from the witness).

To show hardness, we reduce 3-SAT to our problem: given a 3-cnf formula F, for each clause $(x \vee y \vee z) \in F$ we construct the conditional preference statement:

$\overline{x} \wedge \overline{y} : z \succ \overline{z}$. This set of conditional preferences is eligible iff the original formula F is satisfiable (in the SAT sense).

While testing eligibility is hard in general, Theorem 4.7 presents a wide class of statement sets that can be tested for eligibility in polynomial time.

Theorem 4.7 *A set of conditional preference statements Ω, whose dependency graph is acyclic and has bounded node in-degree can be tested for eligibility in polynomial time.*

Proof: The proof is constructive, and the algorithm is as follows: first, for each feature $X \in \mathbf{V}$, we construct a table T_X with an entry for each assignment $\pi \in \mathcal{D}(Pa(X))$, where each entry $T_X[\pi]$ contains all the values of X that are not dominated given Ω and π. Subsequently, we remove all the empty entries. For example, let A, B and C be a set of Boolean problem features, and let $\Omega = \{c \succ \overline{c}, a : b \succ \overline{b}, a \wedge c : \overline{b} \succ b\}$. The corresponding table will be as follows:

Feature	π	Values
T_A	$\emptyset$	$\{a, \overline{a}\}$
T_C	$\emptyset$	$\{c\}$
T_B	$a \wedge \overline{c}$	$\{b\}$
	$\overline{a} \wedge \overline{c}$	$\{b, \overline{b}\}$
	$\overline{a} \wedge c$	$\{b, \overline{b}\}$

Observe that the entry $T_B[a \wedge c]$ has been removed, since, given $a \wedge c$, b and $\overline{b}$ are dominated according to the statements $a \wedge c : \overline{b} > b$ and $a : b > \overline{b}$, respectively. Since the in-degree of each node X in the dependence graph of Ω is bounded by a constant k (i.e. $|Pa(X)| \leq k$), these tables take space and can be constructed in time $O(n2^k)$. Given such tables for all the features in $\mathbf{V}$, we traverse the dependence graph of Ω in a topological order of its nodes, and for each node X being processed we remove all the entries in T_X that are not "supported" by (already processed) $Pa(X)$: An entry $T_X[\pi]$ is not supported by $Pa(X)$ if there exists a feature $Y \in Pa(X)$ such that the value provided by π to Y appears in no entry of T_Y. For instance, in our example, the rows corresponding to $a \wedge \overline{c}$ and $\overline{a} \wedge \overline{c}$ will be removed, since $\overline{c}$ does not appear in the (already processed) table of C. Now, if the processing of a feature X results in $T_X = \emptyset$, then Ω is not eligible. Otherwise, any assignment to **V** consistent with the processed tables will be non-dominated with respect to Ω.

Note that, for sets of preference statements with cyclic dependence graphs, ELIGIBILITY remains hard even if the in-degree of each node is bounded by $k \geq 6$, since 3-SAT remains hard even if each variable participates in at most three clauses of the formula the proof of Theorem 4.6. However, when at most one condition is allowed in each preference statement, and the features are Boolean, then ELIGIBILITY can be reduced to 2-SAT, and thus tested in polynomial time. In Section 4.4.4 we give an algorithm which tests the eligibility of a given set of statements by constructing a set of hard constraints from such statements and then testing its consistency.

4.3.2 The mapping algorithm

In addition to testing consistency and determining preferentially optimal outcomes, we can be interested in the *preferential comparison* of two outcomes. Unfortunately, determining dominance between a pair of outcomes with respect to a set of qualitative preferential statements under the *ceteris paribus* assumption is PSPACE-complete in general [Lan02], and is NP-hard even for acyclic CP-nets [DB02] (see Section 4.2.3).

However, given a set Ω of preference statements, instead of using a preference relation $\succ$ induced by Ω, one can use an approximation $\gg$ of $\succ$, achieving tractability while sacrificing precision to some degree. Moreover it is reasonable to look for approximations since by Theorem 4.3 it is not possible to represent all CP-nets using soft constraints and vice-versa.

Clearly, different approximations $\gg$ of $\succ$ are not equally good, as they can be characterized by the precision with respect to $\succ$, the time complexity of generating $\gg$, and the time complexity of comparing outcomes with respect to $\gg$. There are, however, some conditions that any reasonable approximation must satisfy.

Definition 4.16 (information preserving) *Consider any set* S*, and an ordering* $\succ$ *defined on the elements of* S*. An approximation of* $\succ$*,* $\gg$*, is* information preserving *iff* $\forall \alpha, \beta \in S\ \alpha \succ \beta \Rightarrow \alpha \gg \beta$*.*

In addition to preserving the ordering on those pairs which are ordered by $\succ$, in our context it is also desirable that any approximation respects the ceteris paribus property.

Definition 4.17 (cp-condition) *Consider a set of ceteris paribus statements defined on a set of attributes* $\{X_1, \ldots, X_n\}$ *and let* Ω *be the corresponding space of outcomes, with induced ordering* $\succ$*. An approximation of* $\succ$*,* $\gg$*, is* ceteris paribus preserving *if for each variable* $X_i \in N$*, each assignment* $\mathbf{u}$ *to* $Pa(X)$*, and each pair of values* $x_1, x_2 \in D(X)$*, if the CP-net specifies that* $\mathbf{u} : x_1 \succ x_2$*, then we have* $x_1\mathbf{uy} \gg x_2\mathbf{uy}$*, for all assignments* $\mathbf{y}$ *of* $\mathbf{Y} = \mathbf{V} - \{\{X\} \cup Pa(X)\}$*.*

Here we propose to approximate CP-nets via soft constraints (SCSPs). This allows us to use the rich machinery underlying SCSPs to answer comparison queries in linear time. Moreover, this provides us a uniform framework to combine user preferences with both hard and soft constraints. Given an acyclic CP-net, we construct a corresponding SCSP in two steps. First, we build a constraint graph, which we call *SC-net*. Second, we compute the preferences and weights for the constraints in the SC-net, and this computation depends on the actual semi-ring being used. Here we present and discuss two alternative semi-ring frameworks, based on *min+* and *SLO* (Soft constraint Lexicographic Ordering) semi-rings, respectively. In both cases, our computation of preferences and weights ensures information preserving and satisfies the cp-condition.

Figure 4.6 shows the pseudocode of algorithm CPnetToSCSP. Given a CP-net N (line 1), the corresponding SC-net N_c has two types of nodes: First, each feature $X \in N$ is represented in N_c by a node V_X that stands for a SCSP variable with $\mathcal{D}(V_X) = \mathcal{D}(X)$ (lines 3-4). Second, for each feature $X \in N$, such that $|Pa(X)| \geq 2$, we have a node $V_{Pa(X)} \in N_c$, with $\mathcal{D}(V_{Pa(X)}) = \Pi_{Y \in Pa(X)} \mathcal{D}(Y)$ (lines 5-6) (notice that Π stands for Cartesian product). Edges in N_c correspond to hard

Pseudocode for CPnetToSCSP

```
1. input CP-net $N$;
2. Initialize SCSP $N_c$ as follows:
3. for every feature $X$ of $N$ add variable $V_X$ to $N_c$;
4. for every variable $V_X \in N_c$, $D(V_X) \leftarrow D(X)$;
5. for every feature $X$ of $N$ such that $|Pa(X)| \geq 2$, add variable $V_{Pa(X)}$ to $N_c$;
6. for every variable $V_{Pa(X)} \in N_c$, $D(V_{Pa(X)}) \leftarrow \prod_{Y \in Pa(X)} D(Y)$;
7. for every variable $V_X \in N_c$:
8.    if ($|Pa(X)| = 0$) add a soft unary constraint on $V_X$;
9.    if ($|Pa(X)| = 1$) add a soft binary constraint on $V_{P(X)}$ and $V_X$ in $N_c$;
10.   if ($|Pa(X)| \geq 2$):
11.      for each $Y \in Pa(X)$ add a hard binary constraint on $V_Y$ and $V_{Pa(X)}$;
12.      add a soft binary constraint on $V_{Pa(X)}$ and $V_X$;
13. output SCSP $N_c$;
```

Figure 4.6: Algorithm CPnetToSCSP: given in input a CP-net produces in output a generic SCSP.

and soft constraints, where the latter are annotated with weights. Notice that in the SCSP framework (see Chapter 2 Section 2.2.2) weights on constraints are not allowed. The weights we impose here are merely a working tool for ensuring that the preferences on soft constraints will satisfy information preserving and the cp-condition (see Theorem 4.8). Once the semiring is chosen, the weights will be combined (not necessarily using the semiring multiplicative operator) with initial preferences, allowing to obtain the final preferences in the SCSP (see the following paragraph). Each node V_X corresponding to an "independent feature" $X \in N$ has an incoming (source-less) soft constraint edge (line 8). For each node V_X corresponding to a "single-parent" feature $X \in N$ with $Pa(X) = \{Y\}$, we have a soft constraint edge between X and Y (line 9). Finally, for each node V_X such that $|Pa(X)| \geq 2$, we have (i) hard constraint edges between $V_{Pa(X)}$ and each $Y \in Pa(X)$ to ensure consistency and (ii) a soft constraint edge between $V_{Pa(X)}$ and V_X (lines 10-12). The output is a generic (i.e. not instantiated to any semiring) SCSP N_c.

To assign preferences to variable assignments in each soft constraint, each soft constraint c (between $V_{Pa(X)}$ and V_X) is associated with two items: w_c, a real number which can be interpreted as a weight (will be defined in the next section), and $P_c = \{p_1, ..., p_{|\mathcal{D}(V_X)|}\}$, a set of reals which can be interpreted as "quantitative levels of preference". We will see in the next section how to generate the preference for each assignment to the variables of c, depending on the chosen semiring. We will now illustrate the construction of the SCSP using an example.

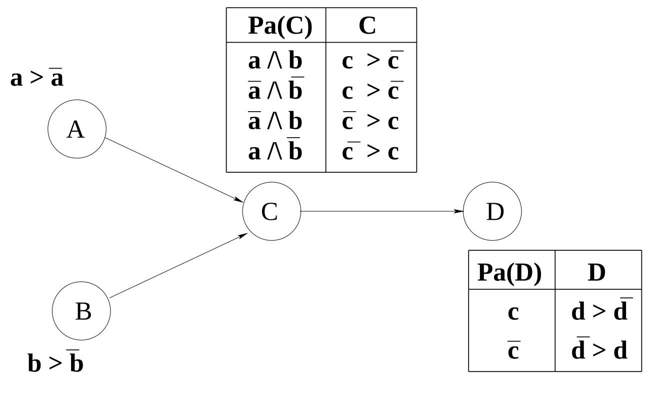

$a > \bar{a}$
A
B
$b > \bar{b}$
C
D
Pa(C) | C
$a \wedge b$ | $c > \bar{c}$
$\bar{a} \wedge \bar{b}$ | $c > \bar{c}$
$\bar{a} \wedge b$ | $\bar{c} > c$
$a \wedge \bar{b}$ | $\bar{c} > c$
Pa(D) | D
c | $d > \bar{d}$
$\bar{c}$ | $\bar{d} > d$

Figure 4.7: The CP-net corresponding to Example 4.5.

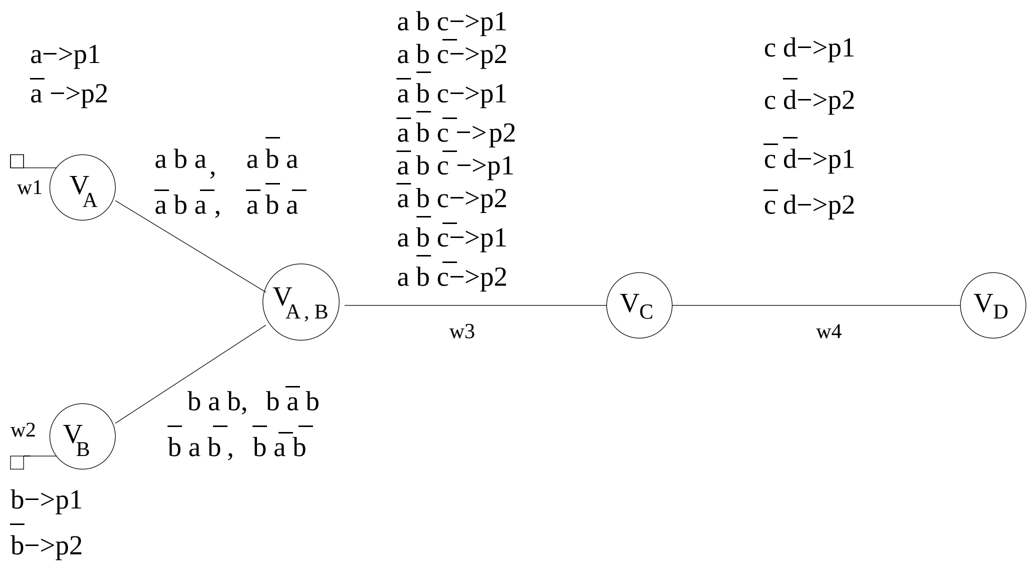


Figure 4.8: The SCSP corresponding to Example 4.5.

Example 4.5 Consider the following set of statements $\Omega = \{a \succ \overline{a}, b \succ \overline{b}, (a \wedge b) \vee (\overline{a} \wedge \overline{b}) : c \succ \overline{c}, (a \wedge \overline{b}) \vee (\overline{a} \wedge b) : \overline{c} \succ c, c : d \succ \overline{d}, \overline{c} : \overline{d} \succ d\}$. The corresponding CP-net, N, is shown in Figure 4.7. Let us consider the SCSP, N_c obtained applying CPnetToSCSP, shown in Figure 4.8. As it can be seen, there is a variable for every feature of N: V_A, V_B, V_C, and V_D. Since the features in N are Boolean, so are the corresponding variables in N_c. In addition, there is variable, $V_{A,B}$ for the parents of C which is the only feature in N with more than one parent, and its domain is $D(A) \times D(B)$. Features A and B in N are independent (i.e. $Pa(A) = Pa(B) = \emptyset$. This means that the preference on their values do not depend on any assignment to other variables. Thus, a unary soft constraint is defined over their corresponding variables in N_c, assigning to each value of their domain a preference. If, instead we consider feature C, the preferences on its domain depend on the assignments of its parents, A and B. Thus in N_c there is a soft constraint between the variable representing the parents, $V_{A,B}$, and V_C, that assigns to each possible triple of values $v_A v_B v_C$ its preference. However, additional hard constraints are needed between variables V_A and V_B and $V_{A,B}$. Their role is to ensure that if V_A is assigned a value, e.g. a, and V_B is assigned another value, e.g. $\bar{b}$, then $V_{A,B}$ can only be assigned pair $(a, \bar{b})$. Finally there is a soft constraint between variable V_D and its only parent V_C assigning preferences to tuples $v_C v_D$. Each soft constraint has an additional weight attached to it, the meaning of which will appear clear in the next paragraph.

Weighted soft constraints

The first approximation we propose applies in scenarios where soft constraints are used to model cost (or penalty) minimization. We will, thus, generate a weighted SCSP, based on the *min*+ semi-ring $S_{WCSP} = \langle R_+, min, +, +\infty, 0\rangle$ (see Chapter 2 Section 2.2.2). We assign preferences using real positive numbers (or penalties) and prefer assignments with smaller total penalty (i.e. the sum of all local penalties). In a soft constraint c on $V_{Pa(X)}$ and V_X, there are $|\mathcal{D}(V_X)|$ penalties, intuitively each corresponding to a position in a total order over the values of feature X. Without loss of generality, we assume they range between 0 and $|\mathcal{D}(V_X)| - 1$, that is, $p_1 = 0, ..., p_{\mathcal{D}(V_X)|} = |\mathcal{D}(V_X)| - 1$. In our example, since all variables are binary, there are only two penalties i.e., $p1 = 0$ and $p2 = 1$, in all the constraints, and $p1 >_{min+} p2$.

```
Pseudocode of Min+weights
1.Input CP-net $N$ and SCSP $N_c$=CPnetToSCSP($N$)
2. Order variables of $N$ in a reverse topological ordering
3. foreach $X \in N$ do
4.   if $X$ has no successors in $N$ then
5.     $w(V_X) \leftarrow 1$
6.   else
7.     $w(V_X) \leftarrow \sum_{Y \text{ s.t. } X \in Pa(Y)} w(V_Y) \cdot |\mathcal{D}(V_Y)|$
8. return $N_c$
```

Figure 4.9: Algorithm Min+weights: computes weights on constraints when the semiring is S_{WCSP}.

The way we ensure that the cp-condition is satisfied in similar to what in proposed in [BBB01] in the context of UCP-nets. We will consider the relation between this work and what we propose in depth in Section 4.5. For now we just say that UCP-nets are as well a quantitative approximation of CP-nets in a max + scenario, that is for maximizing the sum of utilities. To ensure the cp-condition, we will impose that each variable dominates its children. We rewrite such property, defined in [BBB01], in our context.

Definition 4.18 *Consider a CP-net* N *and the corresponding SCSP* N_c*, obtained applying* CPnetToSCSP *to* N*. Consider variable* V_X *and* $V_{Pa(X)}$*in* N_c *and let* $\mathbf{Y}$ *be the variables in* $\mathbf{V} - \{\{X\} \cup Pa(X)\}$ *in* N*. Denote the children of* X *by* $\mathcal{B} = \{V_{B_1}, \cdots, V_{B_h}\}$*.* V_X dominates its children *iff for any two values* $x_1, x_2 \in D(X)$ *such that* $x_1\mathbf{u} \succ x_2\mathbf{u}$*, then for any assignment* $\mathbf{y}$ *to* $\mathcal{B}$*,*

$$p'((x_1\mathbf{uy})_{|c}) - p'((x_2\mathbf{uy})_{|c}) < \sum_{t_i \in T} p'((x_1\mathbf{uy})_{|t_i}) + \sum_{t_i \in T} p'((x_2\mathbf{uy})_{|t_i})$$

where T *is the set of soft constraints of* N_c *and notation* $(x_1\mathbf{uy})_{|s}$ *stands for the projection on the outcome on constraint* s*, constraint* c *is that on* $V_{Pa(X)}$ *and* V_X*, and constraints* $t_i \in T$ *are on* $V_{Pa(B_i)}$ *and* V_{B_i} *such that* $X \in Pa(B_i)$*.*

We will show that, as in [BBB01], also in our context this property is sufficient for the cp-condition (Theorem 4.8).

First we describe an algorithm which sets the weights on N_C in way such that every variable will dominate its children. This is achieved by setting the minimum penalty on a variable, $\neq 0$, to be greater than the sum of the maximum penalties of the children. In Figure 4.9 we show the pseudocode for algorithm Min+weights that computes such weights. In this code, $w(V_X)$ represents the weight of the soft constraint c between $V_{Pa(X)}$ and V_X. As shown by Figure 4.9 the algorithm considers the features of N in reversed topological order. Hence a successor of X in Min+weights is a parent of X in N. When it considers a feature X it sets the weight of the soft constraint defined on V_X and $V_{Pa(X)}$ in N_c. The value assigned is obtained multiplying the weight of the corresponding constraints for all the children of X in N (which is known due to the reverse topological order), by the corresponding size of the domains and adding up the quantities obtained.

Considering Example 4.5, let $\{D, C, B, A\}$ be the reverse topological ordering obtained in line 2. Notice that, in the following, $\times$ and + represent the usual arithmetic operations. Therefore, the first soft constraint to be processed is the one between V_C and V_D. Since D has no children in N, in line 5 we assign $w(V_D)$ to 1. Next, we process the soft constraint between $V_{A,B}$ and V_C: V_D is the only child of V_C, hence $w(V_C) = w(V_D) \times \mathcal{D}(V_D) = 1 \times 2 = 2$. Subsequently, since V_C is the only child of both V_A and V_B, we assign $w(V_A) = w(V_B) = w(V_C) \times |\mathcal{D}(V_C)| = 2 \times 2 = 4$. Now, consider two outcomes $o_1 = abcd$ and $o_2 = a\bar{b}cd$. The total penalty of o_1 is $(w(V_A) \times p_1) + (w(V_B) \times p_1) + (w(V_C) \times p_1) + (w(V_D) \times p_1) = 0$, since $p_1 = 0$, while the total penalty of o_2 is $(w(V_A) \times p_1) + (w(V_B) \times p_2) + (w(V_C) \times p_2) + (w(V_D) \times p_1) = (4 \times 1) + (2 \times 1) = 6$ since $p_2 = 1$. Therefore, we can conclude that o_1 is better than o_2 since $min(0, 6) = 0$. Figure 4.10 shows the result of applying Min+weights and the final SCSP defined of semiring S_{WCSP}.

We now prove that our algorithm for weight computation ensures the cp-condition on the resulting set of soft constraints, and this also implies preserving the ordering information with respect to the original CP-net.

Theorem 4.8 *The SC-net based weighted SCSP N_c, generated from an acyclic CP-net N, is an approximation of N which respects the cp-condition and is information preserving, i.e. for each pair of outcomes α, β we have $\alpha \succ \beta \Rightarrow \alpha >_{min+} \beta$.*

Proof: In order to prove that $>_{min+}$ satisfies the cp-condition it is enough to show that, for each variable $X \in N$, each assignment $\mathbf{u}$ on $Pa(X)$, and each pair of values $x_1, x_2 \in \mathcal{D}(X)$, if the CP-net specifies that $u : x_1 \succ x_2$, then we have $x_1\mathbf{uy} >_{min+} x_2\mathbf{uy}$, for all assignments $\mathbf{y}$ on $\mathbf{Y} = \mathbf{V} - \{\{X\} \cup Pa(X)\}$. If we are able to prove this then $>_{min+}$ is information preserving as well, since, by Definition 4.8 of satisfiability, we have that any ordering, $\succ$, satisfies the CP-net iff for any statement $u : x_1 \succ x_2$ then, $x_1\mathbf{uy} \succ x_2\mathbf{uy}$, for all assignments $\mathbf{y}$ on $\mathbf{Y} = \mathbf{V} - \{\{X\} \cup Pa(X)\}$.

By definition, $x_1\mathbf{uy} >_{min+} x_2\mathbf{uy}$ iff $\sum_{s \in S} p'((x_1uy)_{|_s}) < \sum_{s \in S} p'((x_2uy)_{|_s})$, where S is the set of soft constraints of N_c and notation $(x_1uy)_{|_s}$stands for the projection of the outcome on constraint s. The constraints on which x_1uy differs from

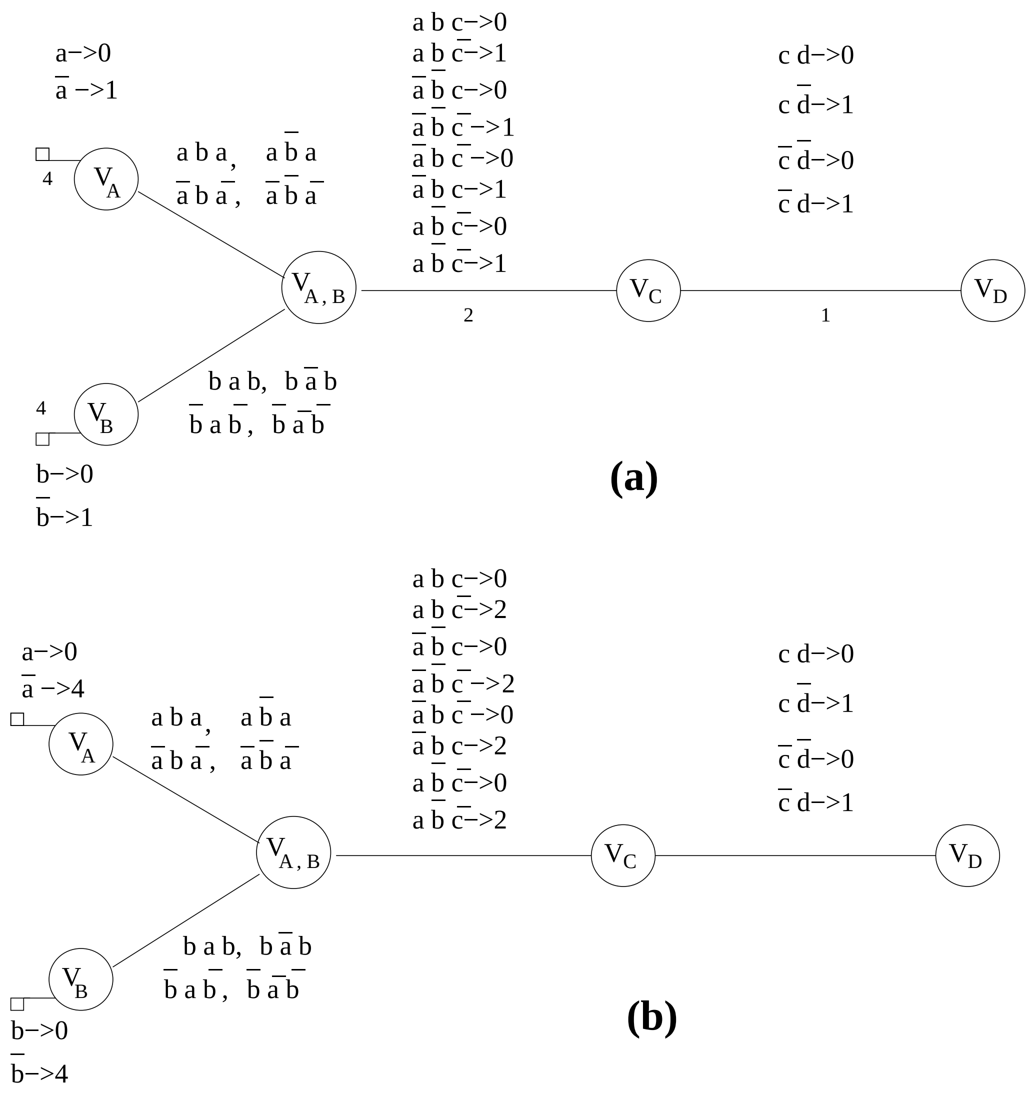

$a \rightarrow 0$
$\bar{a} \rightarrow 1$
4
V_A
$a\ b\ a$, $a\ \bar{b}\ a$
$\bar{a}\ b\ \bar{a}$, $\bar{a}\ \bar{b}\ \bar{a}$
$a\ b\ c \rightarrow 0$
$a\ b\ \bar{c} \rightarrow 1$
$\bar{a}\ \bar{b}\ c \rightarrow 0$
$\bar{a}\ \bar{b}\ \bar{c} \rightarrow 1$
$\bar{a}\ b\ \bar{c} \rightarrow 0$
$\bar{a}\ b\ c \rightarrow 1$
$a\ \bar{b}\ \bar{c} \rightarrow 0$
$a\ \bar{b}\ \bar{c} \rightarrow 1$
$V_{A,B}$
2
V_C
$c\ d \rightarrow 0$
$c\ \bar{d} \rightarrow 1$
$\bar{c}\ \bar{d} \rightarrow 0$
$\bar{c}\ d \rightarrow 1$
1
V_D
4
V_B
$b\ a\ b$, $b\ \bar{a}\ b$
$\bar{b}\ a\ \bar{b}$, $\bar{b}\ \bar{a}\ \bar{b}$
$b \rightarrow 0$
$\bar{b} \rightarrow 1$
(a)
$a \rightarrow 0$
$\bar{a} \rightarrow 4$
V_A
$a\ b\ a$, $a\ \bar{b}\ a$
$\bar{a}\ b\ \bar{a}$, $\bar{a}\ \bar{b}\ \bar{a}$
$a\ b\ c \rightarrow 0$
$a\ b\ \bar{c} \rightarrow 2$
$\bar{a}\ \bar{b}\ c \rightarrow 0$
$\bar{a}\ \bar{b}\ \bar{c} \rightarrow 2$
$\bar{a}\ b\ \bar{c} \rightarrow 0$
$\bar{a}\ b\ c \rightarrow 2$
$a\ \bar{b}\ \bar{c} \rightarrow 0$
$a\ \bar{b}\ \bar{c} \rightarrow 2$
$V_{A,B}$
V_C
$c\ d \rightarrow 0$
$c\ \bar{d} \rightarrow 1$
$\bar{c}\ \bar{d} \rightarrow 0$
$\bar{c}\ d \rightarrow 1$
V_D
V_B
$b\ a\ b$, $b\ \bar{a}\ b$
$\bar{b}\ a\ \bar{b}$, $\bar{b}\ \bar{a}\ \bar{b}$
$b \rightarrow 0$
$\bar{b} \rightarrow 4$
(b)

Figure 4.10: The Figure shows: (a) the network obtained applying Min+weights to Example 4.5 and (b) the final SCSP.

x_2uy are: constraint c on $V_{Pa(X)}$ and V_X, and all the constraints $t_i \in T$ on $V_{Pa(B_i)}$ and V_{B_i} such that $X \in Pa(B_i)$ (in what follows, we denote the children of X by $\mathcal{B} = \{V_{B_1}, \cdots, V_{B_h}\}$). Thus, we can rewrite the above inequality as $p'((x_1uy)_{|_c}) + \sum_{t_i \in T} p'((x_1uy)_{|_{t_i}}) < p'((x_2uy)_{|_c}) + \sum_{t_i \in T} p'((x_2uy)_{|_{t_i}})$ By construction of N_c we have $p'((x_1uy)_{|_c}) = w_c \times p(x_1u) < p'((x_2uy)_{|_c}) = w_c \times p(x_2u)$ and thus $x_1\mathbf{uy} >_{min+} x_2\mathbf{uy}$ iff $w_c p(x_2u) - w_c p(x_1u) > \sum_{t_i \in T} p'((x_1uy)_{|_{t_i}}) - \sum_{t_i \in T} p'((x_2uy)_{|_{t_i}})$. In particular, this will hold if

$$w_c(min_{x,x' \in \mathcal{D}(X)}|p(xu) - p(x'u)|) > \sum_{t_i \in T} w_{t_i}(max_{x,x',z,b}|p(x'zb) - p(xzb)|)$$

where z is the assignment to all parents of $\mathcal{B}$ other than X. Observe , that the maximum in the right term is obtained when $p(x'zb) = |\mathcal{D}(\mathcal{B})| - 1$ and $p(xzb) = 0$. On the other hand, $min_{x,x' \in \mathcal{D}(X)}|p(x'u) - p(xu)| = 1$. In other words: $w_c > \sum_{t_i \in T} w_{t_i}(|\mathcal{D}(B_i)| - 1)$ must hold. But this is ensured by the algorithm, setting (in line 7) $w_c = \sum_{t_i \in T} w_{t_i}(|\mathcal{D}(B_i)|)|$.

We will now prove that the complexity of the mapping we propose is polynomial in the size of the CP-net which is the number features, n.

Theorem 4.9 (Complexity) *Given an acyclic CP-net* N *with the node in-degree bounded by a constant, the construction of the corresponding SC-net based weighted SCSP* N_c *is polynomial in the size of* N.

Proof: If the CP-net has n nodes then the number of vertices V of the derived SC-net is at most $2n$. In fact, in the SC-net a node representing a feature appears at most once and there is at most one node representing its parents. If the number of edges of the CP-net is e, then the number of edges E in the SC-net (including hard and soft edges) is at most $e+n$, since each edge in the CP-net corresponds to at most one constraint, and each feature in the CP-net generates at most one new soft constraints. Topological sort can be performed in $O(V + E)$, that is, $O(2n + e + n) = O(e + n)$. Then, for each node, that is, $O(V)$ times, at most V children must be checked to compute the new weight value, leading to a number of checks which is $O(V^2) = O(n^2)$. Each check involves checking a number of assignments which is exponential in the number of parents of a node. Since we assume that

CP-nets $\Rightarrow$ min+	
$\prec$	$<$
$\succ$	$>$
$\sim$	$<, >, =$

Table 4.1: Relation between the ordering induced by the CP-net semantics and that induced by the Weighted semiring based soft constraints.

the number of parents of a node is limited by a constant, this exponential is still a constant. Thus the total time complexity is $O(V^2)$ (or $O(n^2)$ if we consider the size of the CP-net).

Let us compare in more details the original preference relation induced by the CP-net and this induced by its min+ semi-ring based SC-net. The comparison is summarized in Table 4.1, where $\sim$ denotes incomparability.

By Theorem 4.8 we have that the pairs that are ordered in some way by $\succ$ remain ordered in the same way by $>_{min+}$. Clearly, what is incomparable in the CP-net semantics is ordered in min+ since the ordering induced is total. If we consider Example 4.5, outcome $o_1 = a\bar{b}cd$ is incomparable to outcome $o_2 = \bar{a}bcd$. However since they both have penalty 6, $o_1 =_{min+} o_2$. This is an example of a pair of outcomes that are incomparable in the CP-net ordering and become equally ranked in min+. Instead, outcomes $o_3 = ab\bar{c}d$ and $o_4 = \bar{a}b\bar{c}\bar{d}$ are incomparable in the CP-net but are ordered by min+. In fact, $o_3 >_{min+} o_4$ since the cost associated to o_3 is 3 while that associated to o_4 is 4. Notice that these examples, together with Theorem 4.8 prove the first two implications of the right side of table 4.1. It is easy to see that whatever pair is equally ranked by min+ cannot be (strictly) ordered in the CP-net semantics. In fact consider a CP-net N and its corresponding SCSP N_c with underlying semiring S_{WCSP}. Assume that two outcomes, $o_1, o_2 \in O$ are such that $o_1 =_{min+} o_2$. On the contrary, assume that that $o_1 \succ o_2$. Then by Theorem 4.8 it must be that $o_1 >_{min+} o_2$. But this contradicts the fact that $o_1 =_{min+} o_2$.

We conclude this paragraph by pointing out that a clear drawback of the mapping we have proposed is the introduction of new information by the linearization. In other words, we are forcing the agent to take a position on pairs of outcomes she regarded as incomparable. On the other hand the gain in complexity is considerable. In fact, preferential comparison is now achievable in linear time. Given any two outcomes, it is sufficient to compute their penalties and then compare them.

SLO soft constraints

We also consider a different semi-ring to approximate CP-nets via soft constraints.

Definition 4.19 *Consider the algebraic structure defined as follows:* $S_{SLO} = \langle A , max_s, min_s, \mathbf{MAX}, \mathbf{0} \rangle$*, where* A *is the set of sequences of* n *integers from 0 to MAX,* **MAX** *is the sequence of* n *elements all equal to MAX, and* **0** *is the sequence of* n *elements all equal to 0. The additive operator,* max_s *and the multiplicative operator,* min_s *are defined as follows:*
given $s = s_1 \cdots s_n$ *and* $t = t_1 \cdots t_n$, $s_i = t_i, i = 1 \leq k$ *and* $s_{k+1} \neq t_{k+1}$, *then*

$$max_s(s,t) = s \textit{ if } s_{k+1} \succ t_{k+1} \textit{ else } max_s(s,t) = t;$$

and

$$min_s(s,t) = s \textit{ if } s_{k+1} \prec t_{k+1} \textit{ else } min_s(s,t) = t.$$

In the following theorem we prove that the algebraic structure defined above is a c-semiring (see Chapter 2 Section 2.2.2).

Theorem 4.10 $S_{SLO} = \langle A, max_s, min_s, \mathbf{MAX}, \mathbf{0} \rangle$ *is a c-semiring.*

Proof: The result of applying max_s to two sequences of A is univocally determined by the result of applying max on the values contained in the first component on which the two sequences differ. This allow us to derive that max_s is commutative, associative and idempotent since max satisfies all these properties on the set of integers. It is also easy to see that the **0** is the unit element of max_s, since being defined as the string of n elements all equal to 0, given any other string $s = s_1 \cdots s_n$ since $s_i \in [0, MAX]$, $s_i \leq 0$ $\forall i$. The same reasoning can be applied to min_s, since its result is univocally defined by that of min. Thus, min_s is associative and commutative. Moreover, **0** is the absorbing element of min_s since, as said before, all other strings contain all elements which greater or equal to 0. Since we know that for any string $s = s_1 \cdots s_n$, $s_i \in [0, MAX]$, then applying min_s to s and **MAX** will always returns s as a result, thus **MAX** is the unit element of min_s. For the same reason **MAX** is the absorbing element of max_s. Finally, we have that min_s distributes over max_s since min distributes over max.

The ordering induced by max_s on A is lexicographic ordering [FLP93]. To model a CP-net as a soft constraint problem based on S_{SLO}, we set MAX equal to the cardinality of the largest domain - 1, and n equal to the number of soft constraints of the SC net. All the weights of the edges are set to 1. A preference will be a string of n integers, one for each constraint. Moreover the constraints are represented in the string in an order induced by a topological order of the variables in N. In detail, given string $s = (s_1 \cdots s_n)$, assume s_i is associated to the soft constraint defined on $Pa(V_X)$ and V_X and s_j, with $j > i$, is associated to the soft constraint defined on $Pa(V_Y)$ and V_Y, then there must be a topological order such that X comes before Y. Considering the binary soft constraint on $Pa(X) = \{U_1 \ldots U_h\}$ and X, a tuple of assignments $(u_1, \ldots, u_h, x)$ will be assigned, as preference, the sequence of n integers: $(MAX, MAX, \ldots, MAX - i + 1, \ldots, MAX)$. In this sequence, each element corresponds to a soft constraint. The element corresponding to the constraint on $Pa(X)$ and X is $MAX - i + 1$, where i is the distance from the top of the total order of the value x (i.e., we have a preference statement of the form $u : x_1 \succ x_2 \succ \ldots x_i = x \succ x_{|D(X)|}$).

In the example shown in Figure 4.8, all the preferences will be lists of four integers (0 and 1), where position i corresponds to constraint with weight w_i. For example, in constraint weighted w_3, $p_1 = (1, 1, 1, 1)$ and $p_2 = (1, 1, 0, 1)$. The full SC-net in the SLO model is shown in Figure 4.11.

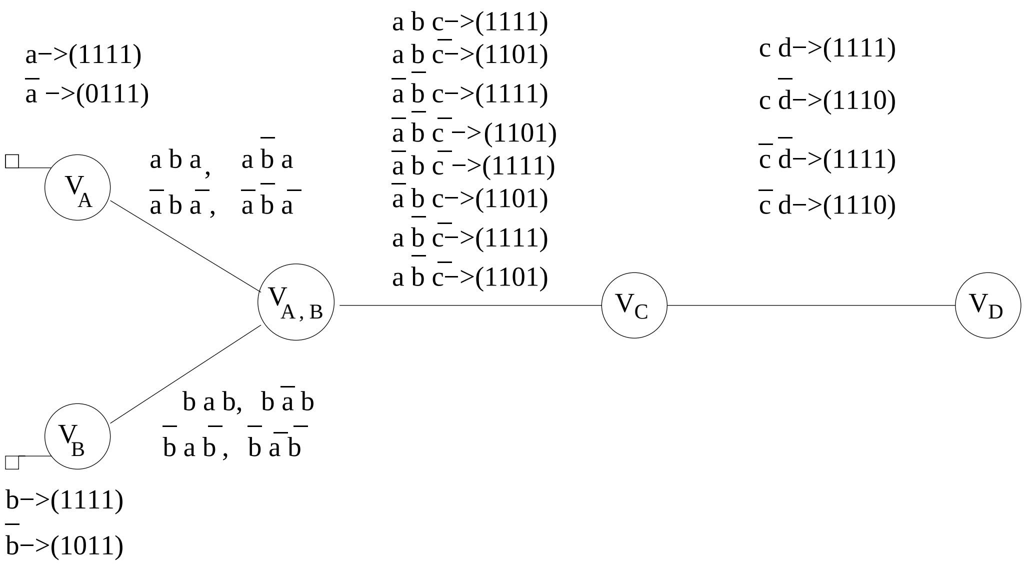

$a \rightarrow (1111)$
$\bar{a} \rightarrow (0111)$
V_A
$a\,b\,a$, $a\,\bar{b}\,a$
$\bar{a}\,b\,\bar{a}$, $\bar{a}\,\bar{b}\,\bar{a}$
$V_{A,B}$
$a\,b\,c \rightarrow (1111)$
$a\,b\,\bar{c} \rightarrow (1101)$
$\bar{a}\,\bar{b}\,c \rightarrow (1111)$
$\bar{a}\,\bar{b}\,\bar{c} \rightarrow (1101)$
$\bar{a}\,b\,\bar{c} \rightarrow (1111)$
$\bar{a}\,b\,c \rightarrow (1101)$
$a\,\bar{b}\,\bar{c} \rightarrow (1111)$
$a\,\bar{b}\,\bar{c} \rightarrow (1101)$
V_C
$c\,d \rightarrow (1111)$
$c\,\bar{d} \rightarrow (1110)$
$\bar{c}\,\bar{d} \rightarrow (1111)$
$\bar{c}\,d \rightarrow (1110)$
V_D
V_B
$b\,a\,b$, $b\,\bar{a}\,b$
$\bar{b}\,a\,\bar{b}$, $\bar{b}\,\bar{a}\,\bar{b}$
$b \rightarrow (1111)$
$\bar{b} \rightarrow (1011)$

Figure 4.11: The Figure shows the final SCSP in the SLO model corresponding to example 4.5.

Given the pair of outcomes $o_1 = abcd$ and $o_2 = a\bar{b}cd$, the global preference associated with o_1 is $(1,1,1,1)$, since it does not violate any constraint, while the preference associated with o_2 is $min_S\{(1,1,1,1),(1,0,1,1),(1,1,0,1),(1,1,1,1)\} = (1,0,1,1)$. We can conclude that o_1 is better than o_2.

In the following theorem we prove that the SLO model both preserves information and ensures the cp-condition.

Theorem 4.11 *The SC-net based SLO SCSP N_c, generated from an acyclic CP-net N, is an approximation of N which respects the cp-condition and is information preserving, i.e., for each pair of outcomes α, β we have $\alpha \succ \beta \Rightarrow \alpha >_{SLO} \beta$.*

Proof: As before it is enough to prove that $>_{SLO}$ satisfies the cp-condition that is, for each variable $X \in N$, each assignment $\mathbf{u}$ on $Pa(X)$, and each pair of values $x_1, x_2 \in \mathcal{D}(X)$, if the CP-net specifies that $u : x_1 \succ x_2$, then we have $x_1\mathbf{uy} >_{SLO} x_2\mathbf{uy}$, for all assignments $\mathbf{y}$ on $\mathbf{Y} = \mathbf{V} - \{\{X\} \cup Pa(X)\}$. In the SLO model the preference associated to $x_1\mathbf{uy}$ and that associated to $x_2\mathbf{uy}$ are strings of positive integers each corresponding to a constraint. The constraints on which the two outcomes differ on are: constraint c on $V_{Pa(X)}$ and V_X, and all the constraints

CP-nets $\Rightarrow$ SLO	
$\prec$	$<$
$\succ$	$>$
$\sim$	$<,>$

Table 4.2: Relation between the ordering induced by the CP-net semantics and that induced by the SLO semiring based soft constraints.

$t_i \in T$ on $V_{Pa(B_i)}$ and V_{B_i} such that $X \in Pa(B_i)$. By construction we have that since $u : x_1 \succ x_2$, the preference associated to the projection of x_1**uy** on constraint c is (MAX,MAX,$\ldots h \ldots$,MAX,MAX) while for outcome x_2**uy** is (MAX,MAX, $\ldots k \ldots$,MAX,MAX) with $k < h$. Since, by definition the position of constraint c precedes that of any constraint defined on the children of V_X, the first component on which the global preference of the outcomes will differ is that corresponding to c. Thus, applying max_s will return as a result that x_1**uy** $>_{SLO}$ x_2**uy** since $h > k$.

As in the previous case, we can easily see that the complexity of the mapping is polynomial in the size of the CP-net.

Theorem 4.12 (Complexity) *Given an acyclic CP-net N with the size of the largest domain bounded by a constant d, the construction of the corresponding SC-net based on the SLO SCSP, N_c, is polynomial in the size of N.*

Proof: As in Theorem 4.12 we know that the SC-net has at most $2n$ nodes and $E = e + n$ edges, if e is the number of edges of CP-net. Topological sort, can be performed in $O(V + E)$, that is, $O(2n + e + n) = O(e + n)$. Once each constraint is assigned a component in the string which is long at most $e + n$, on each constraint at most $O(d)$ different preferences can be assigned to its tuples. Thus, the complexity is $O(d(e + n)) = O(dn^2)$.

Similar to the comparison performed for min+ semi-ring, Table 4.2 compares the preference relation induced by the SLO semiring and that induced by the CP-net.

Let us consider, in detail, the results shown in the table (starting from the left side).

- From Theorem 4.11, we have that $\succ$ implies $>_{SLO}$ and the symmetric result.
- Obviously two outcomes which coincide have the same string as SLO preference.
- If two outcomes are incomparable in the CP-net they are ordered in the SLO, since it induces a total order. However, since they differ at least on the value assigned to one feature, say X, they also must have two different strings as SLO preference which differ on the component corresponding to the constraint defined on $Pa(V_X)$ and V_X. Going back to Example 4.5 outcome $o_1 = ab\bar{c}d$ is incomparable to outcome $o_2 = \bar{a}b\bar{c}\bar{d}$ in the CP-net, but $o_1 >_{SLO} o_2$ since it wins on the first constraint defined on V_A.
- The above example shows that what is ordered in the SLO model can be incomparable in the CP-net. However if it is ordered in the CP-net it must be in the same way due to Theorem 4.11.

The SLO model, like the weighted model, is very useful to answer dominance queries as it inherits the linear complexity from its semi-ring structure. In addition, the sequences of integers show directly the "goodness" of an assignment, i.e., where it actually satisfies the preference and where it violates it.

Comparing the two approximations Given an acyclic CP-net N, let N_c^{min+} and N_c^{SLO} stand for the corresponding min+ and SLO based SC-nets respectively. From the results in the previous section, we can see that pairs of outcomes ordered by N remain ordered the same way by both N_c^{min+} and N_c^{SLO}. On the other hand, pairs of outcomes incomparable in N are distributed among the three possibilities (equal or ordered in one the two ways) in N_c^{min+}, while being strictly ordered by N_c^{SLO}. Therefore, the (total) preference relation induced by N_c^{min+} is a less brutal linearization of the partial preference relation induced by N, compared to that induced by N_c^{SLO}. Mapping incomparability onto equality might seem more reasonable than mapping it onto an arbitrary strict ordering, since the choice is still left to the user. We might conclude that the min+ model is to be preferred to the SLO model, as far as approximation is concerned. However, maximizing the minimum reward, as in any fuzzy framework [Sch92], has proved its usefulness in problem representation. The user may therefore need to balance the linearization of the order and the suitability of the representation provided.

4.4 Constrained CP-nets

In the previous section we have proposed to handle scenarios where there are both constraints and conditional preferences by mapping the CP-net into soft constraints. In particular we have shown that this gives an advantage in complexity when considering dominance testing. In this section we focus on the other important aspect of preference reasoning: outcome optimization. In particular we are going to consider finding optimal outcomes when there are both preferences and hard or soft constraint. We introduce here the notion of constrained CP-net, that is a CP-net plus a set of hard and soft constraint. We then consider two different semantics that induce different orderings on the space of outcomes. In Section 4.4.1 we consider a semantics which has attractive complexity features, but introduces a lot of incomparability. In Section 4.4.2 we extend this semantics to the case of soft constraints and in Section 4.4.3 we consider the case in which there are multiple agents with CP-nets and constraints. In Section 4.4.4 we instead consider a semantics which is more intuitive, by the point of view of the ordering, but may require dominance testing. The main idea presented in this section is that it is possible to write a set of hard constraints such that the outcomes satisfying these constraints are the optimal outcomes of the CP-net. In Sections 4.4.1, 4.4.2 and 4.4.3 a unique set of constraints is created by replacing the original constraints and the CP-net, in Section 4.4.4 the two set of constraints are kept separated.

4.4.1 CP-nets and hard constraints

We now define a structure which is a CP-net plus a set of hard constraints. In later sections we will relax this concept by allowing soft constraints rather than hard constraints.

Definition 4.20 (constrained CP-net) *A* constrained *CP-net is a CP-net plus some constraints on subsets of its variables. We will thus write a constrained CP-net as a pair* $\langle N, C\rangle$*, where* N *is a set of conditional preference statements defining a CP-net and* C *is a set of constraints.*

The hard constraints can be expressed by generic relations on partial assignments or, in the case of binary features, by a set of Boolean clauses. As with CP-nets, the basis of the semantics of constrained CP-nets is the preference ordering, $\succ$, which is defined by means of the notion of a worsening flip. A worsening flip is defined very similarly to how it is defined in a regular (unconstrained) CP-net.

Definition 4.21 ($o_1 \succ o_2$) *Given a constrained CP-net* $\langle N, C\rangle$*, outcome* o_1 *is* better *than outcome* o_2 *(written* $o_1 \succ o_2$*) iff there is a chain of flips from* o_1 *to* o_2*, where each flip is worsening for* N *and each outcome in the chain satisfies* C*.*

The only difference with the semantics of (unconstrained) CP-nets is that we now restrict ourselves to chains of *feasible* outcomes. As we show shortly, this simple change has some very beneficial effects. First, we observe that the $\succ$ relation remains a strict partial ordering as it was for CP-nets [DB02, BBHP99]. Second, it is easy to see that checking if an outcome is optimal is linear (we merely need to check if it is feasible and any flip is worsening). Third, if a set of hard constraints are satisfiable and a CP-net is acyclic, then the constrained CP-net formed from putting the hard constraints and the CP-net together must have at least one feasible and undominated outcome. In other words, adding constraints to an acyclic CP-net does not eliminate all the optimal outcomes (unless it eliminates all outcomes). Compare this to [BBD^{+}04b] where adding constraints to a CP-net may make all the undominated outcomes infeasible while not allowing any new outcomes to be optimal. For example, if we have $o_1 \succ o_2 \succ O_3$ in a CP-net, and the hard constraints make o_1 infeasible, then according to our semantics o_2 is optimal, while according to the semantics in [BBD+04b] no feasible outcome is optimal.

Theorem 4.13 *A constrained and acyclic CP-net either has no feasible outcomes or has at least one feasible and undominated outcome.*

Proof: Take an acyclic constrained CP-net $\langle N, C \rangle$. N induces a preference ordering that contains no cycles and has exactly one most preferred outcome, say o. If o is feasible, it is optimal for $\langle N, C \rangle$. If o is infeasible, we move down the preference ordering until at some point we hit the first feasible outcome. This is optimal for $\langle N, C \rangle$.

We will illustrate constrained CP-nets by means of a simple example. This example illustrates that adding constraints can eliminate cycles in the preference ordering defined by the CP-net. This is not true for the semantics of [BBD+04b], where adding hard constraints cannot break cycles.

Example 4.6 Suppose I want to fly to Australia. I can fly with British Airways (BA) or Singapore Airlines, and I can choose between business or economy. If I fly Singapore, then I prefer to save money and fly economy rather than business as there is good leg room even in economy. However, if I fly BA, I prefer business to economy as there is insufficient leg room in their economy cabin. If I fly business, then I prefer Singapore to BA as Singapore's inflight service is much better. Finally, if I have to fly economy, then I prefer BA to Singapore as I collect BA's airmiles. If we use a for British Airways, $\overline{a}$ for Singapore Airlines, b for business, and $\overline{b}$ for economy then we have: $a : b \succ \overline{b}$, $\overline{a} : \overline{b} \succ b$, $b : \overline{a} \succ a$, and $\overline{b} : a \succ \overline{a}$.

This CP-net has chains of worsening flips which contain cycles. For instance, $ab \succ a\overline{b} \succ \overline{a}\overline{b} \succ \overline{a}b \succ ab$. That is, I prefer to fly BA in business (ab) than BA in economy ($a\overline{b}$) for the leg room, which I prefer to Singapore in economy ($\overline{a}\overline{b}$) for the airmiles, which I prefer to Singapore in business ($\overline{a}b$) to save money, which I prefer to BA in business (ab) for the inflight service. According to the semantics of CP-nets, none of the outcomes in the cycle is optimal, since there is always another outcome which is better.

Suppose now that my travel budget is limited, and that whilst Singapore offers no discounts on their business fares, I have enough airmiles with BA to upgrade from economy. I therefore add the constraint that, whilst BA in business is feasible, Singapore in business is not. That is, $\overline{a}b$ is not feasible. In this constrained CP-net, according to our new semantics, there is no cycle of worsening flips as the hard constraints break the chain by making $\overline{a}b$ infeasible. There is one feasible outcome that is undominated, that is, ab. I fly BA in business using my airmiles to get the upgrade. I am certainly happy with this outcome.

Notice that the notion of optimality introduced here gives priority to the constraints with respect to the CP-net. In fact, an outcome is optimal if it is feasible and it is undominated in the constrained CP-net ordering. Therefore, while it is not possible for an infeasible outcome to be optimal, it is possible for an outcome which is dominated in the CP-net ordering to be optimal in the constrained CP-net.

Finding optimal outcomes. We now show how to map any constrained CP-net onto an equivalent constraint satisfaction problem containing just hard constraints, such that the solutions of these hard constraints correspond to the optimal outcomes of the constrained CP-net. The basic idea is that each conditional preference statement of the given CP-net maps onto a conditional hard constraint. For simplicity, we will first describe the construction for Boolean variables. In the next section, we will pass to the more general case of variables with more than two elements in their domain.

Consider a constrained CP-net $\langle N, C\rangle$. Since we are dealing with Boolean variables, the constraints in C can be seen as a set of Boolean clauses, which we will assume are in conjunctive normal form. We now define the **optimality constraints** for $\langle N, C\rangle$, written as $N \oplus_b C$ where the subscript b stands for Boolean variables, as $C \cup \{opt_C(p) \mid p \in N\}$. The function opt maps the conditional preference statement $\varphi : a \succ \overline{a}$ onto the hard constraint:

$$(\varphi \wedge \bigwedge_{\psi \in C, \overline{a} \in \psi} \psi|_{a=true}) \rightarrow a$$

where $\psi|_{a=true}$ is the clause ψ where we have deleted $\overline{a}$. The purpose of $\psi|_{a=true}$ is to model what has to be true so that we can safely assign a to true, its more preferred value. Optimality constraints only mention the most-preferred value for each feature. So, if we're only asking for undominated outcomes, during preference elicitation we only need to ask for those values for each combination of parent values: the rest of the total order is irrelevant. Moreover, since they incorporate the hard constraints of the problem, they enforce feasibility as well.

To return to our flying example, the hard constraints forbid b and $\overline{a}$ to be simultaneously true. This can be written as the clause $a \vee \overline{b}$. Hence, we have the constrained CP-net $\langle N, C \rangle$ where $N = \{a : b \succ \overline{b}, \overline{a} : \overline{b} \succ b, \overline{b} : a \succ \overline{a}. b : \overline{a} \succ a\}$ and $C = \{a \vee \overline{b}\}$. The optimality constraints corresponding to the given constrained CP-net are therefore $a \vee \overline{b}$ plus the following clauses:

$$\begin{array}{cc} (a \wedge a) \rightarrow b & (b \wedge \overline{b}) \rightarrow \overline{a} \\ \overline{a} \rightarrow \overline{b} & \overline{b} \rightarrow a \end{array}$$

The only satisfying assignment for these constraints is ab. This is also the only optimal outcome in the constrained CP-net. In general, the satisfying assignments of the optimality constraints are exactly the feasible and undominated outcomes of the constrained CP-net.

Theorem 4.14 *Given a constrained CP-net $\langle N, C \rangle$ over Boolean variables, an outcome is optimal for $\langle N, C \rangle$ iff it is a satisfying assignment of the optimality constraints $N \oplus_b C$.*

Proof: ($\Rightarrow$) Consider any outcome o that is optimal. Suppose that o does not satisfy $N \oplus C$. Clearly o satisfies C, since to be optimal it must be feasible (and undominated). Therefore o must not satisfy some $opt_C(p)$ where $p \in N$. The only way an implication is not satisfied is when the hypothesis is *true* and the conclusion is *false*. That is, $o \vdash \varphi$, $o \vdash \psi|_{a=true}$ and $o \vdash \overline{a}$ where $p = \varphi : a \succ \overline{a}$. In this situation, flipping from $\overline{a}$ to a would give us a new outcome o' such that $o' \vdash a$ and this would be an improvement according to p. However, by doing so, we have to make sure that the clauses in C containing $\overline{a}$ may now not be satisfied, since now $\overline{a}$ is false. However, we also have that $o \vdash \psi|_{a=true}$, meaning that if $\overline{a}$ is false these clauses are satisfied. Hence, there is an improving flip to another feasible outcome o'. But o was supposed to be undominated. Thus it is not possible that o does not satisfy $N \oplus_b C$. Therefore o satisfies all $\mathtt{opt}_C(p)$ where $p \in N$. Since it is also feasible, o is a satisfying assignment of $N \oplus_b C$.

($\Leftarrow$) Consider any assignment o which satisfies $N \oplus_b C$. Clearly it is feasible as $N \oplus_b C$ includes C. Suppose we perform an improving flip in O. Without loss of generality, consider the improving flip from $\overline{a}$ to a. There are two cases. Suppose that this new outcome is not feasible. Then this new outcome does not dominate the old one in our semantics. Thus o is optimal. Suppose, on the other hand, that this new outcome is feasible. If this is an improving flip, there must exist a statement $\varphi : a \succ \overline{a}$ in N such that $o \vdash \varphi$. By assumption, o is a satisfying assignment of $N \oplus_b C$. Therefore $o \vdash \mathtt{opt}(\varphi : a \succ \overline{a})$. Since $o \vdash \varphi$ and $o \vdash \overline{a}$, and *true* is not allowed to imply *false*, at least one $\psi|_{a=true}$ is not implied by o where $\psi \in C$ and $\overline{a} \in \psi$. However, as the new outcome is feasible, ψ has to be satisfied independent of how we set a. Hence, $o \vdash \psi|_{a=true}$. As this is a contradiction, this cannot be an improving flip. The satisfying assignment is therefore feasible and undominated.

It immediately follows that we can test for feasible and undominated outcomes in linear time in the size of $\langle N, C \rangle$: we just need to test the satisfiability of the optimality constraints, which are as many as the constraints in C and the conditional statements in N. Notice that this construction works also for regular CP-nets without any hard constraints. In this case, the optimality constraints are of the form $\varphi \rightarrow a$ for each conditional preference statement $\varphi : a \succ \overline{a}$.

It was already known that optimality testing in acyclic CP-nets is linear [DB02].

However, our construction also works with cyclic CP-nets. Therefore optimality testing for cyclic CP-nets has now become an easy problem, even if the CP-nets are not constrained. On the other hand, determining if a constrained CP-net has any feasible and undominated outcomes is NP-complete (to show completeness, we map any SAT problem directly onto a constrained CP-net with no preferences). Notice that this holds also for acyclic CP-nets to which constraints are added, and finding an optimal outcome in an acyclic constrained CP-net is NP-hard.

Non-Boolean variables. The construction in the previous section can be extended to handle variables whose domain contains more than 2 values. Notice that in this case the constraints are no longer clauses but regular hard constraints over a set of variables with a certain domain. Given a constrained CP-net $\langle N, C\rangle$, consider any conditional preference statement p for feature X in N of the form $\varphi : a_1 \succ a_2 \succ a_3$. For simplicity, we consider just 3 values. However, all the constructions and arguments extend easily to more values. The optimality constraints corresponding to this preference statement (let us call them $\mathtt{opt}_C(p)$) are:

$$\varphi \wedge (C_X \wedge X = a_1) \downarrow_{var(C_X)-\{X\}} \rightarrow X = a_1$$

$$\varphi \wedge (C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}} \rightarrow X = a_1 \vee X = a_2$$

where C_X is the subset of constraints in C which involve variable X [1], and $\downarrow \mathbf{Y}$ projects onto the variables in $\mathbf{Y}$. The optimality constraints corresponding to $\langle N, C\rangle$ are again $N \oplus C = C \cup \{\mathtt{opt}_C(p) \mid p \in N\}$. We can again show that this construction gives a new problem whose solutions are all the optimal outcomes of the constrained CP-net.

Theorem 4.15 *Given a constrained CP-net $\langle N, C\rangle$, an outcome is optimal for $\langle N, C\rangle$ iff it is a satisfying assignment of the optimality constraints $N \oplus C$.*

[1] More precisely, $C_X = \{c \in C | X \in \mathtt{con}_c\}$.

Proof: ($\Rightarrow$) Consider any outcome o that is optimal. Suppose that o does not satisfy $N \oplus C$. Clearly o satisfies C, since to be optimal it must be feasible (and undominated). Therefore o must not satisfy some $\mathtt{opt}_C(p)$ where p preference statement in N. Without loss of generality, let us consider the optimality constraints $\varphi \wedge (C_X \wedge X = a_1) \downarrow_{var(C_X)-\{X\}} \rightarrow X = a_1$ and $\varphi \wedge (C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}} \rightarrow X = a_1 \vee X = a_2$ corresponding to the preference statement $\varphi : a_1 \succ a_2 \succ a_3$. The only way an implication is not satisfied is when the hypothesis is *true* and the conclusion is *false*. Let us take the first implication: $o \vdash \varphi$, $o \vdash (C_X \wedge X = a_1) \downarrow_{var(C_X)-\{X\}}$ and $o \vdash (X = a_2 \vee X = a_3)$. In this situation, flipping from $(X = a_2 \vee X = a_3)$ to $X = a_1$ would give us a new outcome o' such that $o' \vdash X = a_1$ and this would be an improvement according to p. However, by doing so, we have to make sure that the constraints in C containing $X = a_2$ or $X = a_3$ may now not be satisfied, since now $(X = a_2 \vee X = a_3)$ is false. However, we also have that $o \vdash (C_X \wedge X = a_1) \downarrow_{var(C_X)-\{X\}}$, meaning that if $X = a_1$ these constraints are satisfied. Hence, there is an improving flip to another feasible outcome o'. But o was supposed to be undominated. Therefore o satisfies the first of the two implications above.

Let us now consider the second implication: $o \vdash \varphi$, $o \vdash (C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}}$ and $o \vdash X = a_3$. In this situation, flipping from $X = a_3$ to $X = a_2$ would give us a new outcome o' such that $o' \vdash X = a_2$ and this would be an improvement according to p. However, by doing so, we have to make sure that the constraints in C containing $X = a_3$ may now not be satisfied, since now $X = a_3$ is false. However, we also have that $o \vdash (C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}}$, meaning that if $X = a_2$ these constraints are satisfied.

Hence, there is an improving flip to another feasible outcome o'. But o was supposed to be undominated. Therefore o satisfies the second implication above. Thus o must satisfy all constraints $\mathtt{opt}_C(p)$ where $p \in N$. Since it is also feasible, o is a satisfying assignment of $N \oplus C$.

($\Leftarrow$) Consider any assignment o which satisfies $N \oplus C$. Clearly it is feasible as $N \oplus C$ includes C. Suppose we perform an improving flip in o. There are two cases. Suppose that the outcomes obtained by performing any improving flip are not feasible. Then such new outcomes do not dominate the old one in our semantics. Thus o is optimal.

Suppose, on the other hand, that there is at least one new outcome, obtained via an improving flip, which is feasible. Assume the flips passes from $X = a_3$ to $X = a_2$. If this is an improving flip, without loss of generality, there must exist a statement $\varphi : \ldots \succ X = a_2 \succ X = a_3 \succ \ldots$ in N such that $o \vdash \varphi$. By hypothesis, o is a satisfying assignment of $N \oplus C$. Therefore $o \vdash \mathtt{opt}(\varphi : \ldots \succ X = a_2 \succ \ldots \succ X = a_3 \succ \ldots) = \varphi \wedge (C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}} \rightarrow \ldots \vee X = a_2$. Since $o \vdash \varphi$ and $o \vdash x = a_3$, and *true* is not allowed to imply *false*, o cannot satisfy $(C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}}$. But, as the new outcome, which contains $X = a_2$, is feasible, such constraints have to be satisfied independent of how we set X. Hence, $o \vdash (C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}}$. As this is a contradiction, this cannot be an improving flip to a feasible outcome. The satisfying assignment is therefore feasible and undominated.

Notice that the construction $N \oplus C$ for variables with more than two values in their domains is a generalization of the one for Boolean variables. That is, $N \oplus C = N \oplus_b C$ if N and C are over Boolean variables. Similar complexity results hold also now. However, while for Boolean variables one constraint is generated for each preference statement, now we generate as many constraints as the size of the domain minus 1. Therefore the optimality constraints corresponding to a constrained CP-net $\langle N, C\rangle$ are $|C| + |N| \times |D|$, where D is the domain of the variables. Testing optimality is still linear in the size of $\langle N, C\rangle$, if we assume D bounded. Finding an optimal outcome as usual requires us to find a solution of the constraints in $N \oplus C$, which is NP-hard.

4.4.2 CP-nets and soft constraints

It may be that we have soft and not hard constraints to add to our CP-net. For example, we may have soft constraints representing quantitative preferences, while we use the CP-nets to represent qualitative preferences. In the rest of this section, a constrained CP-net will be a pair $\langle N, C\rangle$, where N is a CP-net and C is a set of soft constraints. Notice that this definition generalizes the one given in Section 4.4.1 since hard constraints can be seen as a special case of soft constraints.

The construction of the optimality constraints for constrained CP-nets can be adapted to work with soft constraints. To be as general as possible, variables can again have more than two values in their domains. The constraints we obtain are very similar to those of the previous sections, except that now we have to reason about optimization as soft constraints define an optimization problem rather than a satisfaction problem.

Consider any CP statement p of the form $\varphi : X = a_1 \succ X = a_2 \succ X = a_3$. For simplicity, we again consider just 3 values. However, all the constructions and arguments extend easily to more values. The optimality constraints corresponding to p, called $\mathrm{opt}_{soft}(p)$, are the following hard constraints:

$$\varphi \wedge \mathrm{cut}_{best(C)}((\varphi \wedge C_X \wedge X = a_1) \Downarrow_{var(C_X)-\{X\}}) \rightarrow X = a_1$$

$$\varphi \wedge \mathrm{cut}_{best(C)}((\varphi \wedge C_X \wedge X = a_2) \Downarrow_{var(C_X)-\{X\}}) \rightarrow X = a_1 \vee X = a_2$$

where C_X is the subset of soft constraints in C which involve variable X, $best(S)$ is the highest preference value for a complete assignment of the variables in the set of soft constraints S, and $\mathrm{cut}_\alpha S$ is a hard constraint obtained from the soft constraint S by forbidding all tuples which have preference value less than α in S. The optimality constraints corresponding to $\langle N, C\rangle$ are $C_{opt}(\langle N, C\rangle) = \{\mathrm{opt}_{soft}(p) \mid p \in N\}$.

Consider a CP-net with two features, X and Y, such that the domain of Y contains y_1 and y_2, while the domain of X contains x_1, x_2, and x_3. Moreover, we have the following CP-net preference statements: $y_1 \succ y_2$, $y_1 : x_1 \succ x_2 \succ x_3$, $y_2 : x_2 \succ x_1 \succ x_3$. We also have a soft (fuzzy) unary constraint over X, which gives the following preferences over the domain of X: 0.1 to x_1, 0.9 to x_2, and 0.5 to x_3. By looking at the CP-net alone, the ordering over the outcomes is given by $y_1x_1 \succ y_1x_2 \succ y_1x_3 \succ y_2x_3$ and $y_1x_2 \succ y_2x_2 \succ y_2x_1 \succ y_2x_3$. Thus, y_1x_1 is the only optimal outcome of the CP-net. On the other hand, by taking the soft constraint alone, the optimal outcomes are all those with $X = x_2$ (thus y_1x_2 and y_2x_2).

Let us now consider the CP-net and the soft constraints together. To generate the optimality constraints, we first compute $\mathtt{best}(C)$, which is 0.9. Then, we have:

- for statement $y_1 \succ y_2$: $Y = y_1$;
- for statement $y_1 : x_1 \succ x_2 \succ x_3$: we generate the constraints $Y = y_1 \wedge \mathtt{false} \rightarrow X = x_1$ and $Y = y_1 \rightarrow X = x_1 \vee X = x_2$. Notice that we have false in the condition of the first implication because $\mathtt{cut}_{0.9}(Y = y_1 \wedge C_x \wedge X = x_1) \downarrow_Y = \mathtt{false}$. On the other hand, in the condition of the second implication we have $\mathtt{cut}_{0.9}(Y = y_1 \wedge C_x \wedge X = x_2) \downarrow_Y = (Y = y_1)$. Thus, by removing false, we have just one constraint: $Y = y_1 \rightarrow X = x_1 \vee X = x_2$;
- for statement $y_2 : x_2 \succ x_1 \succ x_3$: similarly to above, we have the constraint $Y = y_2 \rightarrow X = x_2$.

Let us now compute the optimal solutions of the soft constraint over X which are also feasible for the following set of constraints: $Y = y_1$, $Y = y_1 \rightarrow X = x_1 \vee X = x_2$, $Y = y_2 \rightarrow X = x_2$. The only solution which is optimal for the soft constraints and feasible for the optimality constraints is y_1x_2. Thus this solution is optimal for the constrained CP-net.

Notice that the optimal outcome for the constrained CP-net of the above example is not optimal for the CP-net alone. In general, an optimal outcome for a constrained CP-net has to be optimal for the soft constraints, and such that there is no other outcome which can be reached from it in the ordering of the CP-net with an improving chain of optimal outcomes. Thus, in the case of CP-nets constrained by soft constraints, Definition 4.21 is replaced by the following one:

Definition 4.22 ($o_1 \succ_{soft} o_2$) *Given a constrained CP-net $\langle N, C \rangle$, where C is a set of soft constraints, outcome o_1 is* better *than outcome o_2 (written $o_1 \succ_{soft} o_2$) iff there is a chain of flips from o_1 to o_2, where each flip is worsening for N and each outcome in the chain is optimal for C.*

Notice that this definition is just a generalization of Def. 4.21, since optimality in hard constraints is simply feasibility. Thus $\succ = \succ_{soft}$ when C is a set of hard constraints.

Consider the same CP-net as in the previous example, and a binary fuzzy constraint over X and Y which gives preference 0.9 to x_2y_1 and x_1y_2, and preference 0.1 to all other pairs. According to the above definition, both x_2y_1 and x_1y_2 are optimal for the constrained CP-net, since they are optimal for the soft constraints and there are no improving paths of optimal outcomes between them in the CP-net ordering. Let us check that the construction of the optimality constraints obtains the same result:

- for $y_1 \succ y_2$ we get $cut_{0.9}(C_y \wedge Y = y_1) \downarrow_X \rightarrow Y = y_1$. Since $cut_{0.9}(C_y \wedge Y = y_1) \downarrow_X = (X = x_2)$, we get $X = x_2 \rightarrow Y = y_1$.

- for statement $y_1 : x_1 \succ x_2 \succ x_3$: $Y = y_1 \wedge cut_{0.9}(Y = y_1 \wedge C_x \wedge X = x_1) \downarrow_Y \rightarrow X = x_1$. Since $cut_{0.9}(Y = y_1 \wedge C_x \wedge X = x_1) \downarrow_Y = \mathit{false}$, we get a constraint which is always true. Also, we have the constraint $Y = y_1 \wedge cut_{0.9}(Y = y_1 \wedge C_x \wedge X = x_2) \downarrow_Y \rightarrow X = x_1 \vee X = x_2$. Since $cut_{0.9}(Y = y_1 \wedge C_x \wedge X = x_2) \downarrow_Y = (Y - y_1)$, we get $Y = y_1 \wedge Y = y_1 \rightarrow X = x_1 \vee X = x_2$.

- for statement $y_2 : x_2 \succ x_1 \succ x_3$: similarly to above, we have the constraint $Y = y_2 \rightarrow X = x_2 \vee X = x_1$.

Thus the set of optimality constraints is the following one: $X = x_2 \rightarrow Y = y_1$, $Y = y_1 \rightarrow X = x_1 \vee X = x_2$, and $Y = y_2 \rightarrow X = x_2 \vee X = x_1$. The feasible solutions of this set of constraints are x_2y_1, x_1y_1, and x_1y_2. Of these solutions, the optimal outcomes for the soft constraints are x_2y_1 and x_1y_2. Notice that, in the ordering induced by the CP-net over the outcomes, these two outcomes are not linked by a path of improving flips through optimal outcomes for the soft constraints. Thus they are both optimal for the constrained CP-net.

Theorem 4.16 *Given a constrained CP-net $\langle N, C\rangle$, where C is a set of soft constraints, an outcome is optimal for $\langle N, C\rangle$ iff it is an optimal assignment for C and if it satisfies $C_{opt}(\langle N, C\rangle)$.*

Proof: ($\Rightarrow$) Consider an outcome o that is optimal for $\langle N, C\rangle$. Then by definition it must be optimal for C. Suppose the outcome does not satisfy C_{opt}. Therefore o must not satisfy some constraint $opt_C(p)$ where p preference statement in N. Without loss of generality, let us consider the optimality constraints

$$\varphi \wedge cut_{best(C)}((\varphi \wedge C_X \wedge X = a_1) \downarrow_{var(C_X)-\{X\}}) \to X = a_1$$

$$\varphi \wedge cut_{best(C)}((\varphi \wedge C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}}) \to X = a_1 \vee X = a_2$$

corresponding to the preference statement $\varphi : a_1 \succ a_2 \succ a_3$.

The only way an implication is not satisfied is when the hypothesis is *true* and the conclusion is *false*. Let us take the first implication: $o \vdash \varphi$, $o \vdash cut_{best(C)}((\varphi \wedge C_X \wedge X = a_1) \downarrow_{var(C_X)-\{X\}})$ and $o \vdash (X = a_2 \vee X = a_3)$. In this situation, flipping from $(X = a_2 \vee X = a_3)$ to $X = a_1$ would give us a new outcome o' such that $o' \vdash X = a_1$ and this would be an improvement according to p. However, by doing so, we have to make sure that the soft constraints in C containing $X = a_2$ or $X = a_3$ may now still be satisfied optimally, since now $(X = a_2 \vee X = a_3)$ is false. We also have that $o \vdash cut_{best(C)}((\varphi \wedge C_X \wedge X = a_1) \downarrow_{var(C_X)-\{X\}})$, meaning that if $X = a_1$ these constraints are satisfied optimally. Hence, there is an improving flip to another outcome o' which is optimal for C and which satisfies C_{opt}. But o was supposed to be undominated. Therefore o satisfies the first of the two implications above.

Let us now consider the second implication: $o \vdash \varphi$, $o \vdash cut_{best(C)}((\varphi \wedge C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}})$, and $o \vdash X = a_3$. In this situation, flipping from $X = a_3$ to $X = a_2$ would give us a new outcome o' such that $o' \vdash X = a_2$ and this would be an improvement according to p. However, by doing so, we have to make sure that the constraints in C containing $X = a_3$ may now still be satisfied optimally, since now $X = a_3$ is false. However, we also have that $o \vdash cut_{best(C)}((\varphi \wedge C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}})$, meaning that if $X = a_2$ these constraints are satisfied optimally. Hence, there is an improving flip to another feasible outcome o'. But o was supposed to be undominated. Therefore o satisfies the second implication above. Thus o must satisfy all the optimality constraints $opt_C(p)$ where $p \in N$.

($\Leftarrow$) Consider any assignment o which is optimal for C and satisfies C_{opt}. Suppose we perform a flip on o. There are two cases. Suppose that the new outcome is not optimal for C. Then the new outcome does not dominate the old one in our semantics. Thus o is optimal. Suppose, on the other hand, that there is at least one new outcome, obtained via an improving flip, which is optimal for C and satisfies C_{opt}. Assume the flip passes from $X = a_3$ to $X = a_2$. If this is an improving flip, without loss of generality, there must exist a statement $\varphi : \ldots \succ X = a_2 \succ X = a_3 \succ \ldots$ in N such that $o \vdash \varphi$. By hypothesis, o is an optimal assignment of C and satisfies C_{opt}. Therefore $o \vdash \mathtt{opt}(\varphi : \ldots \succ X = a_2 \succ \ldots \succ X = a_3 \succ \ldots) = \varphi \wedge \mathtt{cut}_{best(C)}((\varphi \wedge C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}}) \to \ldots \vee X = a_2$.

Since $o \vdash \varphi$ and $o \vdash X = a_3$, and *true* is not allowed to imply *false*, o cannot satisfy $\mathtt{cut}_{best(C)}((\varphi \wedge C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}})$. But o', which contains $X = a_2$, is assumed to be optimal for C, so $\mathtt{cut}_{best(C)}((\varphi \wedge C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}})$ has to be satisfied independently of how we set X. Hence, $o \vdash (C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}}$. As this is a contradiction, this cannot be an improving flip to an outcome which is optimal for C and satisfies C_{opt}. Thus o is optimal for the constrained CP-net.

It is easy to see how the construction of this section can be used when a CP-net is constrained by a set of both hard and soft constraints, or by several sets of hard and soft constraints, since they can all be seen as just one set of soft constraints.

Let us now consider the complexity of constructing the optimality constraints and of testing or finding optimal outcomes, in the case of CP-nets constrained by soft constraints. First, as with hard constraints, the number of optimality constraints we generate is $|N| \times (|D|-1)$, where $|N|$ is the number of preference statements in N and D is the domain of the variables. Thus we have $|C_{opt}(\langle N, C\rangle)| = |N| \times (|D| - 1)$. To test if an outcome o is optimal, we need to check if o satisfies C_{opt} and if it is optimal for C. Checking feasibility for C_{opt} takes linear time in $|N| \times (|D| - 1)$. Then, we need to check if o is optimal for C. This is NP-hard the first time we do it, otherwise (if the optimal preference value for C is known) is linear in the size of C. To find an optimal outcome, we need to find the optimals for C which are also feasible for C_{opt}. Finding optimals for C needs exponential time in the size of C, and checking feasibility in C_{opt} is linear in the size of C_{opt}. Thus, with respect to the corresponding results for hard constraints, we only need to do more work the first time we want to test an outcome for optimality.

4.4.3 Multiple constrained CP-nets

There are situations when we need to represent the preferences of multiple agents. For example, when we are scheduling workers, each will have a set of preferences concerning the shifts. These ideas generalize to such a situation. Consider several CP-nets $N_1, \ldots, N_k$, and a set of hard or soft constraints C. We will assume for now that all the CP-nets have the same features. To begin, we will say an outcome is optimal iff it is optimal for each constrained CP-net $\langle N_i, C\rangle$. This is a specific choice but we will see later that other choices can be considered as well. We will call this notion of optimality, *All-optimal*.

Definition 4.23 (All-optimal) *Given a multiple constrained CP-net* $M = \langle(N_1,\ldots,N_k), C\rangle$*, an outcome* o *is* All-optimal *for* M *if* o *if it is optimal for each constrained CP-net* $\langle N_i, C\rangle$.

This definition, together with Theorem 4.16, implies that to find the all-optimal outcomes for M we just need to generate the optimality constraints for each constrained CP-net $\langle N_i, C\rangle$, and then take the outcomes which are optimal for C and satisfy all optimality constraints.

Theorem 4.17 *Given a multiple constrained CP-net* $M = \langle(N_1, \ldots, N_k), C\rangle$*, an outcome* o *is All-optimal for* M *iff* o *is optimal for* C *and it satisfies the optimality constraints in* $\bigcup_i C_{opt}(\langle N_i, C\rangle)$.

This semantics is one of consensus: all constrained CP-nets must agree that an outcome is optimal to declare it optimal for the multiple constrained CP-net. Choosing this semantics obviously satisfies all CP-nets. However, there could be no outcome which is optimal. In Chapter 5 we will describe a similar consensus semantics (although for multiple CP-nets, with no additional constraints) is called Pareto optimality, and it is one among several alternative to aggregate preferences expressed via several CP-nets. This semantics, adapted to our context, would be defined as follows:

Definition 4.24 (Pareto) *Given a multiple constrained CP-net* $M = \langle (N_1, \ldots, N_k), C \rangle$*, an outcome* o *is* Pareto-better *than an outcome* o' *iff it is better for each constrained CP-net. It is Pareto-optimal for* M *iff there is no other outcome which is Pareto-better.*

If an outcome is all-optimal, it is also Pareto-optimal. However, the converse is not true in general. These two semantics may seem equally reasonable. However, while all-optimality can be computed via the approach presented here, which avoids dominance testing, Pareto optimality needs such tests, and therefore it is in general much more expensive to compute. In particular, whilst optimality testing for Pareto optimality requires exponential time, for All-optimality it just needs linear time (in the sum of the sizes of the CP-nets).

Other possibilities proposed in Chapter 5 require optimals to be the best outcomes for a majority of CP-nets (this is called Majority), or for the highest number of CP-nets (called Max). Other semantics like Lex, associate each CP-net with a priority, and then declare optimal those outcomes which are optimal for the CP-nets with highest priority, in a lexicographical fashion. In principle, all these semantics can be adapted to work with multiple constrained CP-nets. However, as for Pareto optimality, whilst their definition is possible, reasoning with them would require more than just satisfying a combination of the optimality constraints, and would involve dominance testing.

Thus the main gain from our semantics (all-optimal and others that can be computed via this approach) is that dominance testing is not required. This makes optimality testing (after the first test) linear rather than exponential, although finding optimals remains difficult (as it is when we find the optimals of the soft constraints and check the feasibility of the optimality constraints).

4.4.4 Feasible Pareto optimals

The semantics we have introduced in Section 4.4.1 has the drawback of possibly making many outcomes incomparable thus obtaining a large set of optimals. The problem of constrained preferential optimization has been considered in [BBD+04b]. We will discuss this work in detail in Section 4.5, here we are interested in the semantics they introduce for CP-nets with hard constraints.

Definition 4.25 (feasible Pareto optimals [BBD+04b]) *Consider a set of hard constraints* C *and a CP-net* N*, and let* O *be the set of outcomes. An outcome is a* feasible Pareto optimal *iff it is feasible, i.e. it satisfies all the constraints in* C*, and* $\nexists o' \in O$, o' *feasible, such that* $o' \succ o$.

Here we will use the results presented in Section 4.4.1 to give an algorithm that finds optimals according to this semantics. We will start with considering the eligibility of a CP-net and showing that this property can be tested by solving a hard constraint problem.

Testing Eligibility of a CP-net

Let us first consider the semantics we have given in Definition 4.21 Section 4.4.1: given a constrained CP-net, outcome o_1 is **better** than outcome o_2 (written $o_1 \succ o_2$) iff there is a chain of flips from o_1 to o_2, where each flip is worsening for N and each outcome in the chain satisfies C. In the special case in which the set of constraints is empty, the only outcomes that are undominated according to this semantics are those which are optimal for the CP-net. Let us now consider the optimality constraints defined in Section 4.4.1, in this special case.

The optimality constraints corresponding to preference statement $\varphi : a_1 \succ a_2 \succ a_3$ were defined as:

$$\varphi \wedge (C_X \wedge X = a_1) \downarrow_{var(C_X)-\{X\}} \rightarrow X = a_1$$

$$\varphi \wedge (C_X \wedge X = a_2) \downarrow_{var(C_X)-\{X\}} \rightarrow X = a_1 \vee X = a_2$$

where C_X is the subset of constraints in C which involve variable X. If there are no constraints, just the CP-net, then the optimality constraints are:

$$\varphi \rightarrow X = a_1$$

$$\varphi \rightarrow X = a_1 \vee X = a_2.$$

Notice that the first one subsumes the second one.

Consider a CP-net and its optimality constraint, $\mathtt{opt}(N)$. We can derive from Theorem 4.15 the following result.

Theorem 4.18 *Given a CP-net* N *an outcome is optimal for* N *iff it is a satisfying assignment of constraints* $\mathtt{opt}(N)$.

Proof: Directly from Theorem 4.15 by setting $C = \emptyset$.

A similar approach, for Boolean features, is considered in [BD04]. It is important to notice that we do not require for the CP-net to be acyclic. This is a very interesting result since using this procedure we can test whether a CP-net is eligible, that is, if there is at least one undominated outcome (see Definition 4.15), using any hard constraint solver.

Theorem 4.19 *Consider a CP-net* N *and its optimality constraints* $\mathtt{opt}(N)$. *Then* N *is eligible iff* $\mathtt{opt}(N)$ *is consistent.*

Proof: $\Rightarrow$ If N is eligible then, by definition there is at least an outcome, o, such that $\not\exists o'\ o' \succ o$. This corresponds to being optimal according to semantics in Definition 4.21 in the case in which there are no constraints. Thus Theorem 4.18 allows to conclude that o is a solution of $\mathtt{opt}(N)$.

$\Leftarrow$ If $\mathtt{opt}(N)$ is consistent then it has at least a solution, say o. By Theorem 4.18 this solution is optimal (i.e. undominated in the ordering) for the CP-net. Thus the CP-net is eligible.

In Figure 4.12 we show the pseudocode of algorithm CPnet-Eligible? which summarizes the results given above.

```
Pseudocode of CPnet-Eligible?
1. Input CP-net N, CSP opt(N) = ∅
2. foreach statement of N add corresponding constraint to opt(N)
3. if (opt(N) is consistent) then return "N is eligible";
4. else return "N is not eligible";
```

Figure 4.12: Algorithm CP-net-Eligible?: checks if any CP-net (also cyclic) has an undominated outcome.

Notice that since the mapping is linear in the number of statements of the CP-net, the complexity of CPnet-Eligible? will be the complexity of the constraint solver used. In general, testing the consistency of a CSP is NP-hard.

Finding feasible Pareto optimal solutions

Let us now go back to the case of a CP-net constrained with a set of hard constraints C. In Figure 4.13 we present an algorithm that finds all feasible Pareto optimal solutions of a constrained CP-net.

Hard-Pareto takes as input a set of hard constraints C and a CP-net N (line 1). It then initializes the set of feasible Pareto optimal outcomes S_{opt} to the empty set (line 2). If the set of hard constraints is not consistent, i.e. there are no feasible outcomes, then the algorithm returns the empty set (line 3). Otherwise it puts in S_{opt} the set of solutions of the set of hard constraints $C \cup \mathtt{opt}(N)$, denoted in line 5 with $\mathtt{sol}(C \cup \mathtt{opt}(N))$. If such a set of solutions, S_{opt}, is equal either to the set of solutions of the hard constraints alone, $\mathtt{sol}(C)$, or to the set of solutions of the optimality constraints alone, $\mathtt{sol}(\mathtt{opt}(N))$, and such set is not empty, then it returns S_{opt} (lines 6 and 7). Otherwise the algorithm considers all the feasible outcomes that are dominated in the CP-net (line 8) and for each of them it tests whether the outcome is dominated in the CP-net by any other feasible outcome. It adds to S_{opt} only those outcomes which are not dominated by any other feasible one (lines 8-12). Once all the feasible dominated outcomes have been considered, it returns S_{opt} (line 13).

```
Pseudo-code of Hard-Pareto
1. **Input** CP-net N and set of hard constraints C;
2. $S_{opt} \leftarrow \emptyset$;
3. **if** (C inconsistent) **return** $S_{opt}$;
4. **else**
5.   $S_{opt} \leftarrow \mathtt{sol}(C \cup \mathtt{opt}(N))$;
6.   **if** $(S_{opt} = \mathtt{sol}(C))$ **return** $S_{opt}$;
7.   **if** $(\mathtt{sol}(\mathtt{opt}(N)) \neq \emptyset$ and $S_{opt} = \mathtt{sol}(\mathtt{opt}(N)))$ **return** $S_{opt}$;
8.   $S \leftarrow \mathtt{sol}(C) - S_{opt}$;
8.   **do** {Choose $o \in S$;
9.    **if** ( $\forall o' \in S - \{o\}, o' \not\succ o$)
10.     **then** $S_{opt} \leftarrow S_{opt} \cup \{o\}$ and $S \leftarrow S - \{o\}$;
11.     **else** $S \leftarrow S - \{o\}$;}
12. **while** ($S \neq \emptyset$);
13. **return** $S_{opt}$;
```

Figure 4.13: Algorithm Hard-Pareto: given a CP-net and a set of hard constraints, finds all feasible Pareto optimal solution.

We will now prove that Hard-Pareto is a sound an complete algorithm for finding all feasible Pareto optimal solutions of a CP-net constrained with a set of hard constraints.

Clearly, if the set of hard constraints C has no solution then there are no feasible outcomes and, thus, there are no feasible Pareto optimals. In such case the algorithm will stop in line 3.

Otherwise, if there are feasible outcomes that are also solutions of the optimality constraints of the CP-net, then such outcomes are feasible Pareto optimals as shown in the following theorem.

Theorem 4.20 *Let* $\mathtt{sol}(C)$ *be the set of solutions of* C, $\mathtt{sol}(\mathtt{opt}(N))$ *be the set of solutions of* $\mathtt{opt}(N)$*, and* $S = \mathtt{sol}(C) \cap \mathtt{sol}(\mathtt{opt}(N))$. *If* $S \neq \emptyset$ *then* S *is a subset of the set of feasible Pareto optimal solutions.*

Proof: Any $o \in S$ is also an element of $\mathtt{sol}(C)$, thus it is feasible. Moreover since it is in $\mathtt{sol}(\mathtt{opt}(N))$ as well, $\forall o' \in O$, where O is the set of all outcomes, $o' \not\succ o$. This allows us to conclude that any element in S is a feasible Pareto optimal.

Notice that such set, $\mathtt{sol}(C \cup \mathtt{opt}(N))$, is computed in line 5, and if it is not empty it allows to find one or possibly more feasible optimal solutions without doing dominance testing.

Moreover, if $\mathtt{sol}(C \cup \mathtt{opt}(N))$ is equal to the set of solutions $\mathtt{Sol}(C)$, i.e. $\mathtt{sol}(C) \subseteq \mathtt{sol}(\mathtt{opt}(N))$, then the feasible outcomes are a subset of the undominated outcomes of the CP-net. In such a situation the set of solutions of C is the set of all feasible Pareto optimals. In fact, since $\mathtt{sol}(C) \subseteq \mathtt{sol}(\mathtt{opt}(N))$ then each feasible is undominated in the CP-net and all other outcomes, even if undominated in the CP-net, are not feasible. If this is the case the algorithm will stop in line 6 and no dominance test is required to find all the feasible Pareto optimal solutions.

A similar reasoning applies when the set $\mathtt{sol}(C \cup \mathtt{opt}(N))$ is equal to the set of undominated outcomes of the CP-net, that is, $\mathtt{sol}(\mathtt{opt}(N))$ and such set is not empty. This happens iff all the undominated outcomes of the CP-net are feasible, $\mathtt{sol}(\mathtt{opt}(N)) \subseteq \mathtt{sol}(C)$. In this case we can conclude that $\mathtt{sol}(\mathtt{opt}(N))$ is the set of all feasible Pareto optimals. In fact each undominated outcome is also feasible and all other feasible outcomes are dominated by at least one of the undominated outcomes of the CP-net. In this situation the algorithm will stop in line 7, allowing to find the set of feasible Pareto optimals with no dominance test.

Notice, however, that if $\mathtt{sol}(\mathtt{opt}(N))$ is empty, that is the CP-net has no undominated outcome (e.g. when all the outcomes are in a cycle), then, trivially $\mathtt{sol}(\mathtt{opt}(N)) = \mathtt{sol}(C \cup \mathtt{opt}(N))$ but it is not possible to conclude that there are no feasible Pareto optimals. This is the case, for example, if there is a cycle of outcomes such that every other outcome (not belonging to the cycle) is dominated only by an outcome in the cycle, and all the outcomes in the cycle are unfeasible. In such case the feasible outcomes outside the cycle are feasible Pareto optimals.

If $\mathtt{sol}(\mathtt{opt}(N)) = \emptyset$ or if $\mathtt{sol}(C \cup \mathtt{opt}(N)) \subset \mathtt{sol}(C)$ and $\mathtt{sol}(C \cup \mathtt{opt}(N)) \subset \mathtt{sol}(\mathtt{opt}(N))$, in order to find all feasible Pareto optimal outcomes, other than those in $\mathtt{sol}(C \cup \mathtt{opt}(N))$, all those feasible outcomes for C that are undominated in the CP-net by other feasible outcomes must be found. In this case the algorithm stops in line 13.

Theorem 4.21 *Let $\mathtt{sol}(C)$ be the set of solutions of C, and $\mathtt{sol}(\mathtt{opt}(N))$ be the set of solutions of $\mathtt{opt}(N)$. Let $S = \mathtt{sol}(C) \cap \mathtt{sol}(\mathtt{opt}(N))$. Then if $\mathtt{sol}(C) \neq \emptyset$ and $S \subset \mathtt{sol}(C)$ and $S \subset \mathtt{sol}(\mathtt{opt}(N))$ then the set $S_{\mathtt{opt}} = S \cup \{o \in \mathtt{sol}(C) - S | \forall o' \in \mathtt{sol}(C)\ o' \not\succ o\}$ is the set of all feasible Pareto optimal solutions.*

Proof: If S is not empty we know from Theorem 4.20 that all the solutions in S are feasible Pareto optimals. However also those feasible outcomes which are dominated in the CP-net only by unfeasible outcomes are feasible Pareto optimals. This is the set $\{o \in \mathtt{sol}(C) - S | \forall o' \in \mathtt{sol}(C)\ o' \not\succ o\}$. Thus $S_{\mathtt{opt}} = S \cup \{o \in \mathtt{sol}(C) - S | \forall o' \in \mathtt{sol}(C)\ o' \not\succ o\}$ is included in the set of all feasible Pareto optimals. Consider now any feasible Pareto optimal o and assume that $o \in O - S_{\mathtt{opt}}$. Then either o does not belong to $\mathtt{sol}(C)$ and thus is not feasible, but this contradicts the fact that o is a feasible Pareto optimal, or o is in $\mathtt{sol}(C) - (S \cup \{o \in \mathtt{sol}(C) - S | \forall o' \in \mathtt{sol}(C) o' \not\succ o\})$. Thus o is dominated in the CP-net by some feasible outcome. This is, however, in contradiction with the fact that o is a feasible Pareto optimal.

The following result concerns the complexity of the algorithm we have presented.

Theorem 4.22 *The complexity of* Hard-Pareto *is exponential in the number of variables, n, of the constrained CP-net.*

Proof: Finding a solution, or determining inconsistency of the hard constraint problem C in line 3 is known to be hard in general and it takes exponential time in the number of variables, n. This is also the complexity of finding all solutions of $\mathtt{opt}(N) \cup C$ in line 5. In fact, if we use a branch and bound algorithm, for example, in order to find all solutions we must traverse all the search tree. However, this is also the worse case scenario if we want to find a single solution if it happens to be on the last branch in the search. The tests performed in lines 6 and 7, which check if $\mathtt{sol}(\mathtt{opt}(N) \cup C) = \mathtt{sol}(C)$ or $\mathtt{sol}(\mathtt{opt}(N) \cup C) = \mathtt{sol}(\mathtt{opt}(N))$, are linear in the number of solutions an thus exponential in number of variables n. In the cycle of lines 8-12 the algorithm performs dominance testing among such set of solutions, which is hard in general.

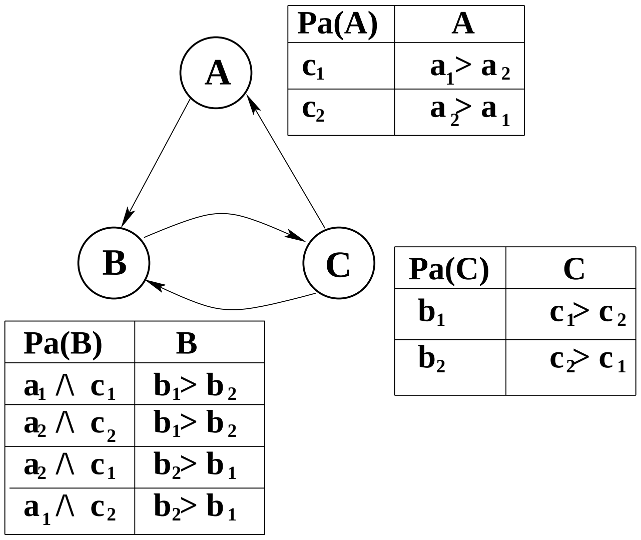
We conclude by illustrating the results of this section on an example [Dom02].

Example 4.7 Consider the following set of statements defining the CP-net shown in Figure 4.14, with three Boolean features A, B, and C. $\Omega = \{c_1 : a_1 \succ a_2, c_2 : a_2 \succ a_1, (a_1 \wedge c_1) \vee (a_2 \wedge c_2) : b_1 \succ b_2, (a_1 \wedge c_2) \vee (a_2 \wedge c_1) : b_2 \succ b_1, b_1 : c_1 \succ c_2, b_2 : c_2 \succ a_1$.

Figure 4.15 shows the induced preference graph of the CP-net in Figure 4.14. The CP-net is not consistent (Definition 4.11), since there are cycles, however it is eligible (Definition 4.15) since outcome $a_1b_1c_1$ is undominated.

The optimality constraints corresponding to the CP-net are: $c_1 \rightarrow a_1$, $c_2 \rightarrow a_2$, $(a_1 \wedge c_1) \vee (a_2 \wedge c_2) \rightarrow b_1$, $(a_1 \wedge c_2) \vee (a_2 \wedge c_1) \rightarrow b_2$, $b_1 \rightarrow$, and $b_2 \rightarrow c_2$.

Let us consider the following set of constraint $C = \{\mathtt{not}(b_2) \vee \mathtt{not}(c_2)\}$.



Pa(A)	A
c_1	$a_1 > a_2$
c_2	$a_2 > a_1$

Pa(C)	C
b_1	$c_1 > c_2$
b_2	$c_2 > c_1$

Pa(B)	B
$a_1 \wedge c_1$	$b_1 > b_2$
$a_2 \wedge c_2$	$b_1 > b_2$
$a_2 \wedge c_1$	$b_2 > b_1$
$a_1 \wedge c_2$	$b_2 > b_1$

Figure 4.14: The dependence graph and the CPT of the CP-net in Example 4.7.

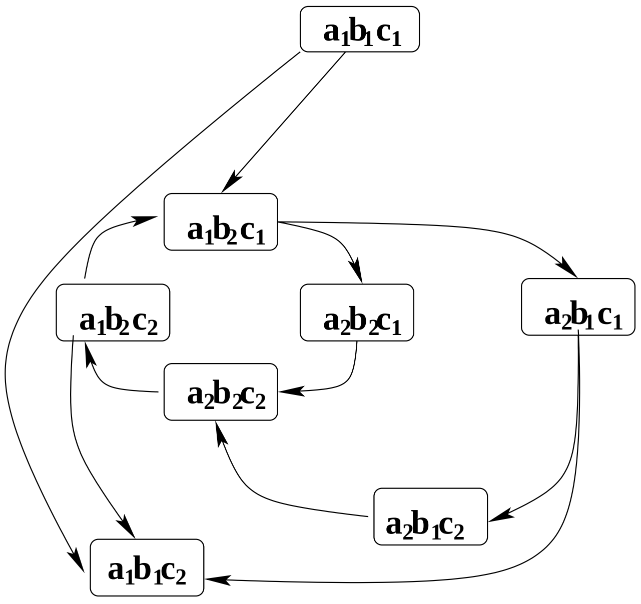


Figure 4.15: The induced preference graph and the CPT of the CP-net in Example 4.7. As it can be seen, it contains cycles.

In this case, $\mathtt{sol}(C) \cap \mathtt{sol}(\mathtt{opt}(N)) = \{a_1b_1c_1\} = \mathtt{sol}(\mathtt{opt}(N))$. Thus we can immediately conclude that $\{a_1b_1c_1\}$ is the only feasible Pareto optimal outcome without doing the dominance test against $\{a_2b_1c_1\}$. On this instance Hard-Pareto stops in line 7.

Consider now a different set of constraints $C' = \mathtt{not}(b_1)$. In this case, all the feasible outcomes, $a_1b_2c_1$, $a_2b_2c_1$, $a_2b_2c_2$, and $a_1b_2c_2$ are in a cycle. Thus there is no feasible Pareto optimal.

4.5 Related Work

We start by considering the work in [BBB01], which proposes a quantitative approximation of CP-nets. This is related to what we have presented in Section 4.3. Then we will consider two papers [BBD$^+$04b, Wil04] on CP-nets and constraints and outline their points in common and in contrast with the approach presented in Section 4.4.

4.5.1 UCP-nets

UCP-nets, introduced in [BBB01], are a directed graphical representation of utility functions combining generalized additive models and CP-nets. In such representation, the utility function is decomposed into a number of additive factors using the directionality of the arcs to reflect conditional dependence as in CP-nets. It thus represents a quantitative model respecting the ceteris paribus condition. Given a set of attributes, $\mathbf{V}$, a partition of $\mathbf{V}$, $\{X_1, \ldots, X_n\}$, and a utility function, u, they consider generalized additive independence as defined in Section 4.2 of this chapter in Definition 4.2. In short, they consider utility functions that can be decomposed in the following way: $u(\mathbf{V}) = \sum_{i=1}^{n} f_i(X_i)$. However, the definition they use does require for the partition to be disjoint. The main idea is to give an interpretation to the factors of the utility function's decomposition in terms of conditional preferential independence. In other words, if, for example, there is factor of the utility function defined on three attributes, say A, B, and C, denoted as $f_j(A, B, C)$, it is interpreted as CPI(C, {A,B},D), that is, as the conditional independence of C from D given its parents A and B. That is, $f_j(A, B, C)$ specifies the utility of C given A and B. Thus, given a utility function u defined on a set of attributes $\{X_1, \ldots, X_n\}$ and inducing a preference ordering $\succeq$, a UCP-net is defined as follows: it is a DAG over $\{X_1, \ldots, X_n\}$ and a set of factors $f_i(X_i, \mathbf{U}_i)$ expressing a quantification of the preference on X_i given its parents $\mathbf{U}_i$, such that $u(\{X_1, \ldots, X_n\}) = \sum_{i=1}^{n} f_i(\mathbf{X}_i)$ and $\succeq$ satisfies (see Definition 4.8) the DAG seen as a CP-net, i.e. satisfies all the conditional preference independence statements derived for the factors. The semantic of a UCP-net is similar to that of Weighted CSPs [SFV95], in the sense that the global preference of an outcome is the sum of the utilities and the goal is to maximize such value. This is a first difference to what we proposed in Section 4.3. The closest of the two approximations we propose to the UCP-model is clearly that using the Weighted semiring. However the semantics is slightly different since the goal is to minimize penalties. The reason for this is that the usual utility semantics, max +, does not meet the requirements of the semiring operators. In fact, in a c-semiring given any two elements of the carrier a, b then $a \times b \leq a, b$, where $\times$ is the multiplicative operator and $\leq$ is the ordering induced by the additive operator. Clearly this is not the case when using the max as the additive operator and + as the multiplicative operator. Instead, we have been inspired by the work in [BBB01] on how to give properties ensuring that the ceteris paribus condition is respected in the quantitative model. As mentioned in Section 4.3, they introduce the concept of an attribute $dominating$ its $children$, which can be summarized by saying that violating a preference on a parent feature must be worse than violating preferences on all its children. However, in contrast to what we propose, generating UCP-nets is exponential in the size of CP-net node's Markov family, i.e. the node, its parents and children, and the parents of its children, and thus in the CP-net node out-degree.

4.5.2 CP-nets and hard constraints

In [BBD+04b] the authors consider scenarios with CP-nets and hard constraints. They thus tackle the same problem we have confronted in Section 4.4. The semantics they give to a CP-net plus a set of hard constraints is that rewritten in Section 4.4.4, Definition 4.25, i.e. an outcome is optimal iff it is feasible and it is not dominated by any feasible in the CP-net ordering. Let us briefly describe the algorithm they propose: Search-CP. In takes as input an acyclic CP-net, both methods proposed in Sections 4.4.1 and 4.4.4 do not require for the CP-net to be acyclic. Their solver is recursive and at each recursive call it accepts three arguments: a subnet of the original CP-net, the subset of the original constraints containing only those defined on the variables of the subnet and an assignment to all the variables except to those in the subnet in input. The algorithm starts with an empty set of solutions and extends it adding new non-dominated solutions. At each stage, the current set is a lower bound for new candidates and each new candidate is tested against all the solutions generated up that point. It is added to the set only if no member dominates it. Let us first consider this work in terms of the semantics we proposed in Section 4.4.1. The main difference is that, if two outcomes are both feasible and there is an improving path from one to the other one in the CP-net, but they are not linked by a path of feasible outcomes, then in this previous approach only the highest one is optimal, while according to Definition 4.21, they are both optimal. For example, assume we have a CP-net with two Boolean features, A and B, and the following cp statements: $a \succ \overline{a}$, $a : b \succ \overline{b}$, $\overline{a} : \overline{b} \succ b$, and the constraint $\overline{a} \vee b$ which rules out $a\overline{b}$. Then, the CP-net ordering on outcomes is $ab \succ a\overline{b} \succ \overline{a}\overline{b} \succ \overline{a}b$. Following Definition 4.21, both ab and $\overline{a}\overline{b}$ are optimal, whilst in the previous approach only ab is optimal. Thus Definition 4.21 obtains a superset of the optimals computed in the approach [BBD+04b]. Reasoning about this superset is, however, computationally more attractive. To find the first optimal outcome, the algorithm in [BBD+04b] uses branch and bound and thus has a complexity that is comparable to solving the set of constraints. Then, to find other optimal outcomes, they need to perform dominance tests (as many as the number of optimal outcomes already computed), which are very expensive. In our approach, to find one optimal outcome we just need to solve a set of optimality constraints, which is NP-hard. Two issues that are not addressed in [BBD$^+$04b] are testing optimality efficiently and reasoning with cyclic CP-nets. To test optimality, they must run the branch and bound algorithm to find all optimals, and stop when the given outcome is generated or when all optimals are found. In our approach, we check the feasibility of the given outcome with respect to the optimality constraints. Thus it takes linear time. Our approach is based on the CP statements and not on the topology of the dependency graph. Thus it works just as well with cyclic CP-nets.

In Section 4.4.4 instead we have adopted the semantic defined in [BBD$^+$04b], thus, Search-CP and Hard-Pareto compute the same sets of optimals. Also, both Search-CP and Hard-Pareto are anytime. However, there are some differences. As noted above Hard-Pareto works also for cyclic CP-nets. Moreover, if Hard-Pareto stops in line 6 or 7, because all feasible are undominated or all undominated are feasible then the complexity to find all feasible Pareto optimal solutions is the same as finding just one and no dominance testing is required. Search-CP by comparison will require dominance testing if more than one solution is needed. On the other hand, if no undominated outcomes are feasible, then Hard-Pareto may require dominance testing just to find the first solution, while Search-CP merely performs branch and bound.

The problem of handling conditional statements and hard constraints is considered also in [Wil04]. In this work the author extends the framework of CP-nets in order to handle more general statements of the form $u : x > x'[W]$, meaning that any assignment containing u and x is to be preferred to any other assignment containing u and x', basically irrespective to the assignments to W. In the paper it is shown that these statements generalize those in CP-nets which are obtained by setting $W = \emptyset$. Given a set of such statements, Γ, as in CP-nets, the induced ordering $>_\Gamma$ is obtained by considering the transitive closure of the "local" orders induced on pairs of outcomes by the statements. As in CP-nets, the equivalent of a dependency graph is associated to each set of statements. A set of statements Γ is said to be consistent if there exists a total order $>_t$ which is a linearization of $>_\Gamma$, i.e. such that for every pair of outcomes o and o' if $o >_\Gamma o'$ then $o >_t o'$. Notice that this is a generalization of that given in [BBD$^+$04a] of consistency for CP-nets (see Section 4.2, Definition 4.11), however it is stronger than our notion of eligibility, which applies to set of statements that induce cyclic preferences as well. The main result of the paper is to show that, given a set of statements such that the dependency graph is acyclic, then it is consistent iff it is locally consistent, which means that there is a partial order consistent with $>_\Gamma$ that can be obtained by Γ imposing on the feature a generalization of a lexicographic ordering. Such generalization of the lexicographic ordering allows to have some features which are not ordered and allows the local preference on the values of the attributes to be partially ordered as well and to be conditional on assignment to previous features. In [Wil04], the author uses this result in the context of constrained optimization. In particular he shows that it is possible to modify the complete algorithm Search-CP proposed in [BBD$^+$04b] by replacing each dominance test in the true CP-net ordering, $\succ$, by a test in the partial order, $\succ_{lx}$, obtained from the statements using the lexicographic structure. Such test is easy because of the Lex-style construction of $\succ_{lx}$. In brief, to determine if $o \succ_p o'$, one must consider all the variables on which the two outcomes differ and to find all the $minimal$ features, i.e. the most important ones, among those. Then it is sufficient to check if, given the statements on those variables, the assignments in o are always preferred to those in o'. This procedure is guaranteed to compute a non empty subset of the feasible Pareto optimal solutions defined in [BBD$^+$04b], and thus a subset of the optimals w.r.t. the semantics we have proposed in Section 4.4.1. However, also this approach requires for the dependence graph to be acyclic, while we allow also sets of statements with cyclic dependency graphs.

An additional related work is described in [MD02], where a numerical value function is constructed using graph-theoretic techniques by examining the graph of the preference relation induced by a set of preference statements. Note that this framework is also computationally hard, except for some special cases.

4.6 Future Work

We will now describe some interesting directions we plan to pursue in the future. In particular, we will consider obtaining better approximations of the CP-net ordering using multidimensional semirings (Section 4.6.1), we will study further the issue of constrained CP-nets with both hard and soft constraints (Sections 4.6.2 and 4.6.3).

4.6.1 Obtaining the CP-net ordering via n-dimensional semirings

The ordering that CP-nets induce on outcomes is a preorder. The approximations we have proposed in Section 4.3 are total orderings. They are based on two important semirings the Weighted semiring and the fuzzy semiring. The reason they have been chosen is that they allow to set up the network (i.e. defining the preferences on each constraint) in a way such that the ceteris paribus property holds. To see how this is not trivial at all, consider a semiring that has the same preference set A as the one in the SLO model but a different additive operator inducing the ordering. For example if we were to consider simple dominance, where the dominant assignment must be $\geq$ on all corresponding values except for one in which it must be strictly better, the ceteris paribus condition does not hold when the array is mapped back to its corresponding sequence. A semiring that realizes this ordering will still have A as the set, but the additive operation would build the sequence taking the maximum value on all the positions, while the multiplicative operator would take the minimum. Lets see an easy example: given the preference statements: $a \succ \overline{a}$, $a : b \succ \overline{b}$, $b : \overline{c} \succ c$, $\overline{b} : c \succ \overline{c}$, consider the 2 assignments abc and $a\overline{b}c$. According to the second preference statement abc wins on $a\overline{b}c$. Lets now consider the corresponding vectors $\langle 1, 1, 0 \rangle$ and $\langle 1, 0, 1 \rangle$.

According to simple dominance they are incomparable which translates into a violation of the ceteris paribus condition.

In a sense the possibility of inducing partial orderings that the semiring framework features, has not been exploited. We hope to overcome this by using n-dimensional semirings, obtained by the Cartesian product of other semirings. For example we can consider the semiring $S_{WCSP} \times S_{SLO}$. Now each preference will be a pair $\langle a_1, a_2 \rangle$ where the first element is the preference assigned by the S_{WCSP} model and the second element, an array, is the preference assigned in the SLO model.

In the following Table we show the comparison between the Combined model, $S_{WCSP} \times S_{SLO}$, and CP-net ordering.

CP-nets ⇒ Combined		**Combined** ⇒ CP-nets	
$\prec$	$<$	$<$	$\prec, \sim$
$\succ$	$>$	$>$	$\succ, \sim$
$\sim$	$<, >, \sim$	$\sim$	$\sim$

We can see now that the incomparables in the CP-net ordering are mapped either in ordered pairs in the Combined model or in incomparables. However it is not clear how this linearization compares with the one induced the min+ model.

This example shows that reasoning on induced partial and total orderings is not trivial as it may appear at first. This makes this direction more interesting and challenging. The results so far obtained and here described give a reasonable link between qualitative and quantitative reasoning about preferences. However there is still a lot of work to be done in order to establish a unifying model equipped with the generality of qualitative frameworks and powered by the performances of quantitative representations.

Rather than randomly testing various semiring to asses the similarity of the order they induce with the CP-net one, a more general line of research could be that of investigating what is the relation between the semiring operators and the degree of similarity. That is we would like to identify a class of operators through some properties such that a semiring with operators belonging to that class induces an ordering that approximates in a certain way the CP-net ordering. In the comparison of the approximations that we have proposed in the previous sections we have seen that a good way to measure how "brutal" and approximation is, is estimating the size of the set of incomparables that are linearized into some order. If the approximation is a total order than the most auspicable event is mapping incomparability into equality (i.e. "equally preferred"). If, instead, the approximation is a partial order the more the approximation will be closer to the CP-net order the more incomparability relations remain the same in the approximation.

A way to approach the problem is to identify a "good" approximation, analyze its operators and to see if abstracting it through the framework proposed in [BCR02], a class of approximation of the same quality is obtained.

4.6.2 Refining the semantics

In Section 4.4.1 we have proposed a way to reduce the problem of finding optimal outcomes of a CP-net with constraints to that of solving a constraint problem, with hard or soft constraints. The compactness of this method and the attractive complexity it has in testing whether an outcome is optimal in the case of hard constraints does come at a price. The price is paid in the definition of the new semantics, that is, in considering chains of flips that pass only through feasible outcomes. On one side it is reasonable to question if transitivity should be allowed to flow through infeasible outcomes. Consider the following example in which we want to buy a car and we prefer to have a convertible car rather than a sedan. If the car is a convertible we prefer that it runs on gas rather than on diesel. If the car is a sedan we prefer a diesel engine to a gasoline one. The cp statements are as follows: $convertible > sedan$, $convertible$: $gas > diesel$, $sedan$: $diesel > gas$. The ordering induced is:$(conv, gas) > (conv, diesel) > (sedan, gas) > (sedan, diesel)$. There is however a hard constraint: there are no convertibles with a diesel engine! At this point our algorithm would return two optimal outcomes $(conv, gas)$ and $(sedan, diesel)$. In fact, $(conv, gas)$ does not dominate $(sedan, diesel)$ in our semantics since the path of worsening flips passes through infeasible outcome: $(conv, diesel)$. We think this is reasonable. In fact the priority of the preference on the car type (convertible or sedan) is somehow induced by how the cp statements are given. In other words since the user expresses the preference on the engine type (gasoline or diesel) depending on of which type is the car, the ceteris paribus semantics automatically gives a higher priority to the car type. However, this may not be explicitly mentioned or even intended by the user. Moreover, in order to handled such a hierarchy on features a new framework [BD02] has been introduced. We thus think that the user should be presented both outcomes.

It is unfortunately the case, however, that the more constraints we have the more optimal solutions may appear. This is in contrast with the intuitive role of constraints of "eliminating" solutions. An alternative, in addition to that presented in Section 4.4.4, to overcome this and to remain closer to the ordering induced by the CP-net without giving up completely the complexity results, is the following. After an optimal solution according to the constrained CP-net has been found, consider its neighborhoods, that is the outcomes that can be reached from it with a bounded number of flips, say between 2 and k [2]. As we build the neighborhood we consider those that are classified as optimals. If any of them dominates the center of the neighborhood we throw the center away as well as all the other optimals in the neighborhood that were dominated by it. We then start again the same procedure from the undominated outcomes remaining in the neighborhood. We think line of research may achieve an interesting compromise between complexity and representational power of preferences, and thus our intention to pursue in the near future.

Recently CP-nets have also been extended to handle importance tradeoffs between features [BD02]. This means that the framework supports a hierarchy on the features: some are more important and thus the preferences on their values

[2]We already know that all those that are one flip away are infeasible or not optimal.

should be satisfied in the best possible way regardless of the ceteris paribus semantics. We plan to study the impact of adding this new capability in a constrained environment. In other words, it is clear that constraints can have an impact on the tradeoffs just as they do on the ordering induced by the CP-nets. However the effect can change according to the priority given to the constraints. In the case of hard constraints feasibility constraints will probably override any preference on any feature, however with soft constraints the role of the constraints could be calibrated.

4.6.3 Pareto optimals in softly constrained CP-nets

In Section 4.4.4 we have given a procedure that finds outcomes that are optimal according to the semantics given in [BBD$^+$04b], that is that are feasible and undominated in the CP-net. If we consider soft constraints feasibility is usually replaced by optimality. We could thus extend this semantics given for hard constraints to the scenarios with soft constraints in the following way.

Definition 4.26 *Given a CP-net* N *and a set of soft constraints* S *then* $o \succ_{NS} o'$ *iff* $o \succ_N o'$ *and* $o \geq_S o'$

Thus, an outcome o is optimal according to this ordering iff for all outcomes o', $o' \not\succ_{NS} o$. In words, outcome o is optimal if all other outcomes either have a worst preference according to the soft constraints or are dominated in the CP-net.

We could apply the results presented in Section 4.4 to find one of such optimals. A first idea is, as in algorithm Hard-Pareto, to compute all the optimal solutions of N, using the optimality constraints and to compute the optimal preference level of S. Then we just have to compute the preference of the solutions of the optimality constraints of N, which takes linear time for each solution, and stop as soon as we find one that is associated with the optimal preference level of S. If none is associated to such preference level, then, we can keep the undominated solution of N with the highest preference. We want to give more thought on this matter which seems to be a promising direction, and to further investigate the usefulness of representing the undominated outcomes of a set of ceteris paribus statements as solutions of a CSP.

Chapter 5

Preferences in Multi-agent Scenarios

In this final chapter we want to consider even more general scenarios than those we have tackled up to here. In fact, here we are going to study preferences expressed by multiple agents. This means that we must consider both the representation of the preference of each agent, and also ways of reasoning and aggregating preferences in order to choose outcomes that satisfy all the agents up to a certain level.

In this chapter we present a new multiagent preference reasoning framework: mCP-nets. In such a framework, each agent expresses her own preferences using an acyclic partial CP-net. By partial CP-net we mean a CP-net that shares features with the CP-nets of other agents and does not necessarily specify preferences over all of them as instead required by CP-nets [BBD+04a] (see also Section 4.2). We then consider the usual queries as outcome optimization and dominance testing for partial CP-nets, giving algorithms and complexity results. Such queries are however presented in light of their usefulness for preference aggregation. In fact, the second problem we consider is that of aggregating the agent's preferences once they have been collected. In particular, inspired by social welfare theory [Kel87], we adapt the most popular aggregating criteria of such a theory to our context, studying their induced semantics and complexity. Finally, we push even further the bridge between social welfare theory and aggregation of preferences obtained using AI representations, by considering the "fairness" [Arr51] of the voting schemes we propose. The main difference from the context in which the famous Arrow's theorem [Arr51] was originally written and our scenario is that we have incomplete orders, that is, we allow incomparability. We thus extend Arrow's impossibility theorem [Arr51] and two possibility results [Sen70] to the situation in which the ordering of each agent is a partial order, and so is the order produced in the result.

5.1 Motivations and Chapter Structure

Recently, AI researchers have started to tackle problems as interaction, cooperation or negotiation in multi-agent scenarios [ZR96, San99]. In particular, a lot of attention has been devoted to both allowing compact representation of preferences which are often expressed on an exponentially large set of candidates and also for automating the process of aggregating such preferences in collective choices. Succinct representations are needed in most of AI scenarios since they are combinatorial in nature. In fact, most of the time each object (or candidate) on which the agent expresses her preferences is an assignment to many different features. On one side, such a decomposition allows for more expressiveness, since it enables conditional dependence and specification of constraints. On the other side, it generates a blow up in terms of the size of the space of possible outcomes. Secondly, the need for automating has been shown to be compelling when the preferences of many agents must be collected and synthesized into a common ordering over outcomes or a final choice of a winner.

In this chapter we will consider scenarios in which agents order the candidates using partial orders (POs). Such POs may be expressed via conditional qualitative preference statements or also using other preference representation such as soft constraints. Let us start by giving a simple example that highlights both the omnipresence of problems of preference representation and aggregation, and the non-triviality of the issue.

Example 5.1 Suppose you invite three friends for dinner. Alice prefers fish to beef. Bob, on the other hand, prefers beef to fish. Finally, Carol is like Alice and prefers fish to beef. What do you cook? Both choices are Pareto optimal. If you cook fish then changing to beef will be more preferred by Bob, but less preferred by Carol and Alice. Similarly, changing from beef to fish will be more preferred by Alice and Carol, but less preferred by Bob. However, fish is perhaps the "best" choice according to a majority ordering as it is the preferred choice for both Alice and Carol, whilst beef is the preferred choice for only Bob.

Which wine do you serve with the main course? Alice, Bob and Carol are fortunately more consistent here. If it is fish then they all prefer white wine to red. However, if it is beef, then they all prefer red wine to white. Finally, do you serve cheese or dessert? Even though you are happy to serve everyone their own choice of cheese or dessert, you must still determine and reason about your friends' preferences. For example, if his girlfriend Alice has cheese, Bob prefers cheese to dessert. However, if his girlfriend Alice has dessert, he will not be able to resist so he prefers dessert to cheese.

This example demonstrates that multiple agents may have some features in common but not all (e.g. the main dish is common to all but the choice of cheese or dessert is not), that there may no longer be a single optimal solution, that there can be several definitions of optimality (e.g. Pareto optimal *versus* a majority dominance), that preferences may be conditional (e.g. the preference for wine depends on the choice of the main course) and dependent on the features of other agents (e.g. Bob's preference for dessert or cheese depends on Alice's choice for dessert or cheese). We therefore propose a framework for representing and reasoning with such preferences.

The work described in this chapter has appeared in the proceedings of the following conferences and international workshops:

- F. Rossi, K. B. Venable, T. Walsh. *mCP-nets: representing and reasoning with preferences of multiple agents*. Proc. Nineteenth National Conference on Artificial Intelligence (AAAI-04), pp. 729-734, July 25-29, 2004, San Jose, California, USA. AAAI Press / The MIT Press 2004.

- F. Rossi, K. B. Venable and T. Walsh. *Aggregating Preferences cannot be fair* To appear: Intelligenza Artificiale, 2005.

The chapter is structured in the following way. In Section 5.2 we give some background on orderings (Section 5.2.1) and the results which are of interest of social welfare theory (Section 5.2.2). In Section 5.3 we first describe partial CP-nets (Section 5.3.1), which are obtained from CP-nets [BBD$^+$04a] relaxing some assumptions on the specification of the ordering. Then, we formally define the multiagent framework based on mCP-nets in Section Section 5.3.2. We then consider several different voting semantics, Section 5.3.3, studying the properties of the ordering they induce (Section 5.3.4) and which relations hold among them (Sections 5.3.4 and 5.3.5). In the following section, Section 5.4, we study some results of voting theory, namely Arrow's theorem (Section 5.4.1 and 5.4.2) and Sen's possibility results (Section 5.4.3) in scenarios where agents use partial orders. Finally, we consider the impact of having constraints, as in Chapter 4, when considering properties of voting rules (Section 5.6). In Section 5.5 we present the work related to what we have presented in this chapter.

5.2 Background

We start reviewing some basic notions on orders, which will be useful in what follows. Some of the following definitions have already been given in the thesis. However, we believe that a brief summary will be useful for a better understanding of this chapter. In Section 5.2.2 we describe the main notions of voting theory, those which we will consider in this chapter, and we describe briefly the fundamental result contained in Arrow's impossibility theorem and in Sen's possibility theorem.

5.2.1 Orderings

A preference ordering can be described by a binary relation on outcomes where x is preferred to y iff (x, y) is in the relation. Such relations may satisfy a number of properties. A binary relation R on a set S (that is, $R \subseteq S \times S$) is:

- *reflexive* iff for all $x \in S$, $(x, x) \in R$;
- *transitive* iff for all $x, y, z \in S$, $(x, y) \in R$ and $(y, z) \in R$ implies $(x, z) \in R$;
- *antisymmetric* iff for all $x, y \in S$, $(x, y) \in R$ and $(y, x) \in R$ implies $x = y$;
- *complete* iff for all $x, y \in S$, either $(x, y) \in R$ or $(y, x) \in R$.

A *total order* is a binary relation which is reflexive, transitive, antisymmetric, and complete. A *partial order* is a binary relation which is reflexive, transitive and antisymmetric but may be not complete. There may be pairs of elements (x, y) of S which are not in the partial order relation, that is, such that neither $(x, y) \in R$ nor $(y, x) \in R$. Such elements are *incomparable* (written $x \bowtie y$). Unlike a total order, a partial order can have several optimal and mutually incomparable elements. We say that these elements are *undominated*. Undominated elements will also be called *top* elements. The set of all top elements of a partial order o will be called $\mathtt{top}(o)$. Elements which are below or incomparable to every other element will be called *bottom* elements.

Given any relation R which is either a total or a partial order, if $(x, y) \in R$, it can be that $x = y$ or that $x \neq y$. If R is such that $(x, y) \in R$ implies $x \neq y$, then R is said to be *strict*. This means that reflexivity does not hold.

Both total and partial orders can be extended to deal with ties, that is, sets of elements which are equally positioned in the ordering. Two elements which belong to a tie will be said to be *indifferent*. To summarize, in a total order with ties, two elements can be either ordered or indifferent. On the other hand, in a partial order with ties, two elements can be either ordered, indifferent, or incomparable. Notice that, while incomparability is not transitive in general, indifference is transitive, reflexive, and symmetric.

The usual strict total order ($<_{\mathbb{Z}}$ and total order ($\leq_{\mathbb{Z}}$) defined on the set of integers $\mathbb{Z}$ are classical examples.

The following orderings are instead respectively a strict partial order and a partial order over pairs of integers. Consider the set of pairs of integers $\mathbb{Z} \times \mathbb{Z}$ and the orders defined as follows: $\langle x, y\rangle <_{\mathbb{Z}\times\mathbb{Z}} \langle z, w\rangle$ iff $x <_{\mathbb{Z}} z$ and $y <_{\mathbb{Z}} w$ and $\langle x, y\rangle \leq_{\mathbb{Z}\times\mathbb{Z}} \langle z, w\rangle$ iff $x \leq_{\mathbb{Z}} z$ and $y \leq_{\mathbb{Z}} w$. It is easy to see that $<_{\mathbb{Z}\times\mathbb{Z}}$ is not reflexive nor complete and it is transitive and antisymmetric while $\leq_{\mathbb{Z}\times\mathbb{Z}}$ is reflexive, transitive and antisymmetric but not complete. According to such orderings some elements of $\mathbb{Z} \times \mathbb{Z}$ are ordered (e.g. $\langle 1, 2\rangle <_{\mathbb{Z}\times\mathbb{Z}} \langle 3, 4\rangle$) while others are incomparable (e.g. $\langle 5, 3\rangle$ is incomparable with $\langle 2, 8\rangle$).

As an example of partial order with ties, consider the following defined on triples in $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$. Given $\langle x_1, y_1, z_1\rangle$ and $\langle x_2, y_2, z_2\rangle$ in $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$, we say $\langle x_1, y_1, z_1\rangle \leq \langle x_2, y_2, z_2\rangle$ iff $\langle y_1, z_1\rangle \leq_{\mathbb{Z}\times\mathbb{Z}} \langle y_2, z_2\rangle$. In other words we are indifferent to the value of the first component. In this case we will have elements that are respectively ordered , as $\langle 2, 3, 4\rangle < \langle 1, 5, 6\rangle$, or incomparable, as $\langle 2, 3, 4\rangle$ and $\langle 2, 1, 5\rangle$ or indifferent, as $\langle 1, 2, 3\rangle$ and $\langle 2, 2, 3\rangle$.

In the following we will sometimes need to consider partial orders with some restrictions. In particular, we will call a **rPO** a partial order where the top elements are all indifferent, or the bottom elements are all indifferent. In both POs and rPOs, ties are allowed everywhere, except when explicitly said otherwise.

5.2.2 Social welfare theory

In classical social welfare theory [Kel87, Arr51, Str80], individuals state their preferences in terms of total orders. In this section we will define the main concepts and properties in this context, that will be useful especially in Section 5.4.

Given a set of n individuals and a set of outcomes O, a *profile* is a sequence of n orderings over O, one for each individual. A *social welfare function* is a function from profiles to orderings over O. Thus social welfare functions provide a way to aggregate the preferences of the n individuals into an ordering of the outcomes.

Several properties of social welfare functions can be considered:

- *Freeness*: if the social welfare function can produce any ordering;
- *Unanimity*: if all agents agree that A is preferable to B then the resulting order must agree as well;
- *Independence to irrelevant alternatives*: if the ordering between A and B in the result depends only on the relation between A and B given by the agents;
- *Monotonicity*: if, whenever an agent moves up the position of one outcome in her ordering, then (all else being equal) such an outcome cannot move down in the result;
- *Dictatoriality*: if there is at least an agent such that, no matter what the others vote, if he says A is better than B then the resulting ordering says the same. Such an agent is then called a *dictator*.
- *Neutrality*: if, given a renaming r of the outcomes, we have that $f(o_1, \dots, o_n) = o$ iff $f(r(o_1), \dots, r(o_n)) = r(o)$, where $o, o_1, \dots, o_n$ are POs and f is a social welfare function.
- *Fairness*: if the social welfare function is free, unanimous, independent to irrelevant alternatives, and non-dictatorial.

These properties are all very reasonable and desirable also for preference aggregation. Unfortunately, a fundamental result in voting theory is Arrow's impossibility theorem [Arr51, Kel78] which shows that no social welfare function on total orders with ties can be fair. In particular, the usual proofs of this result show that, given at least two voters and three outcomes, and a social welfare function which is free, monotone, and independent of irrelevant alternatives, then there must be at least one dictator.

It is possible to prove that monotonicity and independence to irrelevant alternatives imply unanimity. Social welfare functions can be free, unanimous and independent to irrelevant alternatives but not monotonic [Sen70]. Therefore a stronger version of Arrow's result can be obtained by proving that freeness, unanimity and independence of irrelevant assumptions implies that there must be at least one dictator [Gea01].

A very reasonable social welfare function which is often used in elections is pairwise majority voting. In this voting function, for each pair of outcomes, the order between them in the result is what the majority says on this pair. This function, however, can produce an ordering which is cyclic. A sufficient condition to avoid generating cycles via a pairwise majority voting function is *triplewise value-restriction* [Sen70], which means that, for every triple of outcomes x_1, x_2, x_3, there exists $x_i \in \{x_1, x_2, x_3\}$ and $r \in \{1, 2, 3\}$ such that no agent ranks x_i as his r-th preference among x_1, x_2, x_3. A typical example where this condition is not satisfied and there are cycles in the result is one where agent 1 ranks $x_1 > x_2 > x_3$, agent 2 ranks $x_2 > x_3 > x_1$, and agent 3 ranks $x_3 > x_1 > x_2$. In this example, x_1 is ranked 1st by agent 1, 2nd by agent 3, and 3rd by agent 2, x_2 is ranked 1st by agent 2, 2nd by agent 1, and 3rd by agent 3, and x_3 is ranked 1st by agent 3, 2nd by agent 2, and 3rd by agent 1. The resulting order obtained by majority voting has the cycle $x_1 > x_2 > x_3 > x_1$.

5.3 Aggregating Preferences: mCP-nets

We will now introduce mCP-nets, an extension of the CP-net formalism (see Section 4.2) to model and handle the qualitative and conditional preferences of multiple agents. We give a number of different semantics for reasoning with mCP-nets. The semantics are all based on the idea of individual agents voting. We describe how to test optimality and preference ordering within a mCP-net, and we give complexity results for such tasks. In particular, in Section 5.3.1 we relax some of the assumptions of CP-nets allowing agents to express conditional preferences that depend on features that are of other agents, i.e. features that they, personally, don't want to rank. Then, in Section 5.3.2, we formally define mCP-nets and in Sections 5.3.3 and 5.3.4 we consider several possible semantics for aggregating preferences describing how they relate to each other.

5.3.1 Partial CP-nets

We first introduce partial CP-nets. These will be used to represent the preferences of the individual agents in a mCP-net.

Definition 5.1 (partial CP-net) *A* partial CP-net *is a CP-net (see Definition 4.7 in Chapter 4 Section 4.2) in which certain features may not be ranked. Thus, there can be features, say* X_i*, which do not depend (preferentially) on other features such that their preference tables leave all the values in the domain indifferent.*

Intuitively, this means that the agent is indifferent to the values of such features. The presence of non ranked features is needed to model the dependence of features ranked by some agent on features ranked by some other agent.

Partiality requires us to relax the original CP-net semantics, which we recall can be expressed in terms of worsening or improving flips. In particular, given a CP-net, and two outcomes o and o', we say that $o \succ o'$, iff there is sequence of improving flips from o' to o. Introducing partiality forces us to redefine the classes of $\mathtt{flips}$ in the following way.

Definition 5.2 *Given a partial CP-net* N *and an outcome* o*:*

- *A* worsening flip *is the change in the value of a variable* A *in* o*, obtaining outcome* o' *such that:*
  - *if* A *is ranked, given the assignment of* $Pa(A)$ *in* o*, the flip is worsening in the CP table of* A*;*
  - *if* A *is not ranked, it is worsening in the CP tables of all features that depend on* A*, given the values that are assigned to such a feature in* o*.*

- *An* indifferent flip *is changing the value of a non-ranked variable,* A*, in an outcome* $o = x_1 x_2 \ldots a \ldots x_n$*, obtaining outcome* $o' = x_1 x_2 \ldots a' \ldots x_n$*, such that, for each* X_i *such that* $A \in Pa(X_i)$*, in* $CPT(X_i)$*, if* $\mathbf{u}_{X_i}$ *is the assignment in* o *(and thus also in* o'*) to the (other) features in* $Pa(X_i)$*, then* $|\{x_i^j \in D(X_i) | \mathbf{u}_{X_i} a x_i^j \succ \mathbf{u}_{X_i} a x_i\}| = |\{x_i^j \in D(X_i) | \mathbf{u}_{X_i} a' x_i^j \succ \mathbf{u}_{X_i} a x_i\}|$*. Informally, an indifferent flip is a change of the value of a non ranked feature such that in all CP tables such a change leaves the outcome in the same position.*

- Incomparable flips *are all those flips which are neither worsening, nor improving, nor indifferent.*

We will now illustrate this definitions with some examples.

Example 5.2

- **Example of worsening flip.** Consider a partial CP-net defined on two features A and B such that A is not ranked and the preferences on B are defined by the following statements: $a : b \succ \overline{b}$ and $\overline{a} : \overline{b} \succ b$, Then, passing from ab to $\overline{a}b$ is a worsening flip as we go from the best ranked position to the worst in the CP table of B.

- **Example of indifferent flip.** Indifferent flips are all those flips of a non-ranked variable such that, for each CP table of the features that depend on non ranked features, they are neither improving nor worsening. Consider the following partial CP-net where A is not ranked, B has three values, and we have the CP statement $a : b_1 \succ b_2 \succ b_3$ and $\overline{a} : b_3 \succ b_2 \succ b_1$. Then, passing from ab_2 to $\overline{a}b_2$ is an indifferent flip.

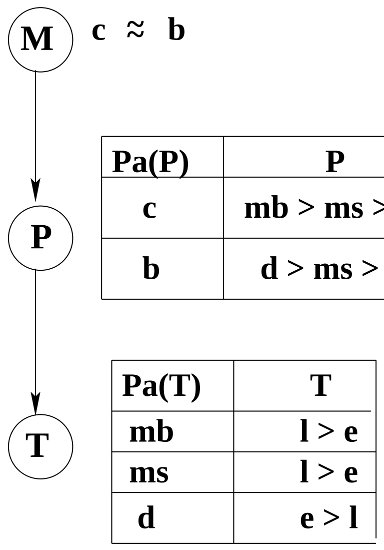
- **Example of incomparable flip.** Consider the following partial CP-net where A is not ranked, and for B we have $a : b \succ \overline{b}$ and $\overline{a} : \overline{b} \succ b$, and for C we have $\overline{a} : c \succ \overline{c}$ and $a : \overline{c} \succ c$. Then, passing from abc to $\overline{a}bc$ is an incomparable flip.

We are now able to define the ordering induced by a partial CP-net on the set of outcomes.

Definition 5.3 (partial CP-nets semantics) *Given an agent and her partial CP-net* N*:*

- *The agent* prefers *outcome α to outcome β (written $\alpha \succ \beta$) iff there is a chain of flips from α to β where each flip is worsening or indifferent, and there is at least one worsening flip.*
- *The agent is* indifferent *between outcomes α and β (written $\alpha \approx \beta$) iff at least one chain of flips between them consists only of indifferent flips. This requires the outcomes to differ only for the values of non-ranked features. If two outcomes are indifferent it is not possible for them to be ordered in the sense defined above.*
- *Similarly one outcome α is* incomparable *to another β (written $\alpha \bowtie \beta$) iff it is not the case that $\alpha \succ \beta$, $\beta \succ \alpha$ or $\alpha \approx \beta$.*

We illustrate the above definitions with the following example shown in Figure 5.1.



Pa(P)	P
c	mb > ms > d
b	d > ms > mb

Pa(T)	T
mb	l > e
ms	l > e
d	e > l

Figure 5.1: Dependence graph and the CPTs of the partial CP-net in Example 5.3.

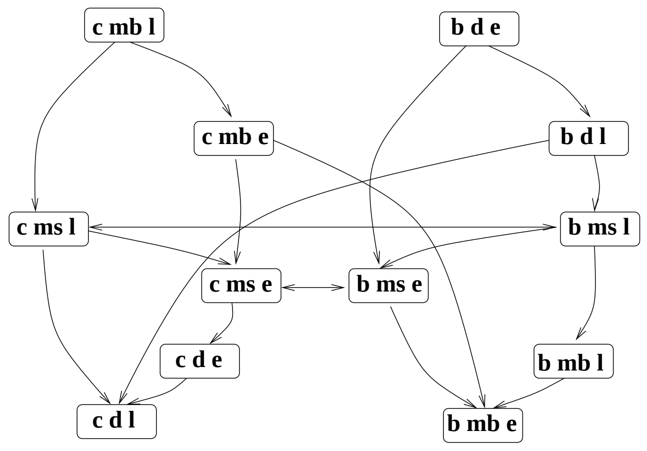


Figure 5.2: Induced preference graph of the partial CP-net in Example 5.3.

Example 5.3 Alice wants to go shopping with Carol. They can either go by bus or use the car and they can choose between the shops downtown, a mall which is closer but small and a bigger mall further away. Alice wants to ride with Carol, whatever she decides. However, if they are using the car, Alice thinks they might as well go to the bigger mall, which has more selection, or in alternative to the smaller mall but not downtown since parking there is a nightmare. If instead they are going by bus, Alice prefers to go downtown since the ride is shorter or, eventually, to the closer mall. The time at which they go is also an issue. If they go to any of the malls, then, it is ok for Alice to go late since they can eat at the food court. If they go downtown, instead, Alice prefers to go early and be back home for dinner, since the restaurants downtown are always packed.

This scenario can be modeled using a partial CP-net representing Alice's preferences. There are three features M (mean of transport) with domain $\{c, b\}$, where c stands for car and b for bus, P (shopping place), with domain $\{m_s, m_b, d\}$ where m_s is the smaller mall, m_b is the bigger mall and d stands for downtown, and finally feature T (time), with domain $\{e, l\}$, which stand for early and late.

Let us consider the ordering induced by the partial CP-net on the outcomes, shown in Figure 5.2. It is easy to see that $cm_bl \succ cde$, while $cm_se \approx bm_se$ and $cde \bowtie bm_be$.

Partial CP-nets can have more than one optimal outcome even if their dependency graphs are acyclic: there is one for each possible value of the non-ranked features. This can be seen also in Figure 5.2, where both cm_bl and bde are optimal.

As in CP-nets, the queries of main interest are finding an optimal, testing whether a given outcome is optimal, and dominance testing. We will now consider the complexity of answering these queries in the context of partial CP-nets.

Theorem 5.1 *Finding one of the optimal outcomes in an acyclic partial CP-net is linear.*

Proof: It is enough to choose any value for the non-ranked features, and to perform the forward sweep [BBHP99] (see Section 4.2) for the remaining features.

One, may instead be interested in testing the optimality of a given outcome.

Theorem 5.2 *Optimality testing in acyclic partial CP-nets is linear.*

Proof: Given a partial CP-net N, with non ranked features $Z_1, \ldots, Z_k$, consider an outcome, o, and let $z_1, \cdots, z_k$ be the values assigned to the non ranked features in o. To test if o is optimal it is sufficient to set $Z_1 = z_1, \cdots, Z_k = z_k$ and then perform the forward sweep [BBHP99] from here. If the resulting outcome is the same as the given one (which can be tested in linear time), it is optimal, otherwise it is not.

Since a CP-net is a special case of a partial CP-net, the difficulty of dominance testing is inherited by the latter.

Theorem 5.3 *Given a partial CP-net N and two outcomes o_1 and o_2, answering $o_1 \succ o_2$ is hard in general.*

However, checking if two given outcomes are indifferent is linear, as proven in the following theorem.

Theorem 5.4 *Given a partial CP-net N and two outcomes, say o_1 and o_2, checking if $o_1 \approx o_2$ takes $O(kn)$, where n is the size of the net and k is the number of non-ranked features.*

Proof: It should be noticed that we are assuming a constant time to access a CP table. For each non-ranked feature, we need to check whether flipping it from its value in o_1 to that in o_2, is indifferent for all features depending on it (there may be $O(n)$ of them).

5.3.2 mCP-nets

We now put together several partial CP-nets to represent the preferences of multiple agents.

Definition 5.4 *A* m**CP-net** *is a collection of* m *partial CP-nets which may share some features, such that every feature is ranked by at least one of the partial CP-nets.*

Each feature in a partial CP-net N of an mCP-net is shared, visible or private. Private features are ranked in N only, and are not visible to other partial CP-nets. Visible features for N are those used in the conditions of N but not ranked there. Shared features are ranked in N and also in at least one other of the partial CP-nets.

For simplicity, we assume that the partial CP-nets in a mCP-net are acyclic. However the mCP-net itself can have cycles containing edges coming from different component CP-nets. We leave it as future work to see what happens when we relax this assumption. Graphically, an mCP-net is obtained by combining the graphs of the partial CP-nets so we have one occurrence of each shared feature.

The "m" in mCP-net stands for both multiple agents and the number of agents. Thus, a 3CP-net is a mCP-net in which $m = 3$. Note that a CP-net is a 1CP-net. Hence, mCP-nets (with $m = 1$) immediately inherit all the techniques and complexity results for regular CP-nets.

An outcome for an mCP-net is an assignment to all the features in all the partial CP-nets to values in their domains. This outcome induces an outcome for each single partial CP-nets forming the overall mCP-net, by eliminating all the features which do not belong to the partial CP-net.

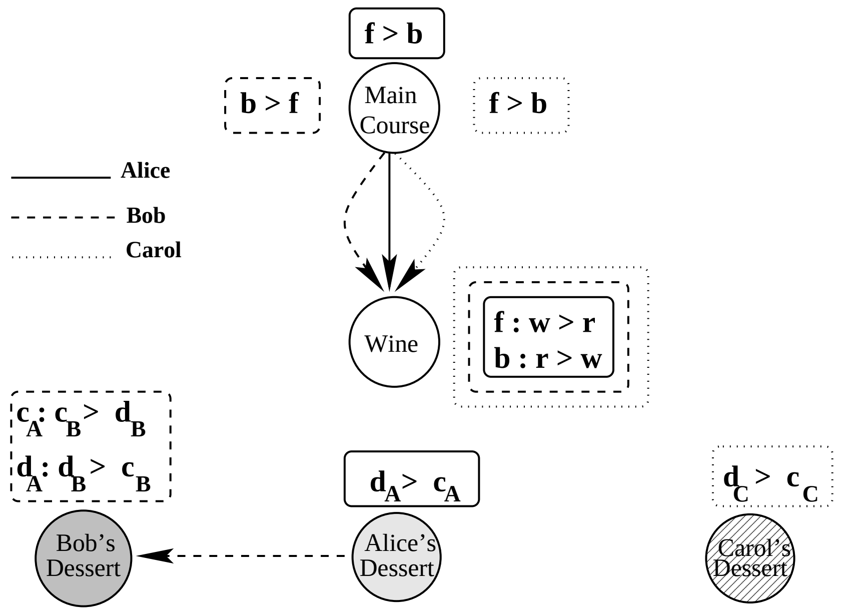


Figure 5.3: 3CP-net corresponding to Example 5.1 described in Section 5.1.

In Figure 5.3 we show the 3CP-net corresponding to Example 5.1 presented in Section 5.1. We have two shared features, $\mathit{Main\ Course}$ and Wine. Wine depends on $\mathit{Main\ Course}$ in all three partial CP-nets. Moreover, we have three private features, $\mathit{Bob's\ dessert}$ is a private feature of Bob's partial CP-net, similarly, $\mathit{Alice's\ dessert}$ in the partial CP-net corresponding to Alice and $\mathit{Carol's\ dessert}$ in the partial CP-net of Carol. Feature $\mathit{Alice's\ dessert}$ is visible in Bob's partial CP-net and it is not ranked by Bob. However, his preferences on the values of his private feature $\mathit{Bob's\ dessert}$ depend on it.

5.3.3 Voting semantics

We will reason about a mCP-net by querying each partial CP-net in turn and collecting the results. We can see this as each agent "voting" whether an outcome dominates another. We can obtain different semantics by collecting these votes together in different ways. Many of these semantics may be useful in a distributed setting, or when there are privacy issues. We do not need to know all the preferences of each partial CP-net. We may just have to ask if anyone votes against a particular outcome.

The semantics we propose are based on known concepts, like Pareto optimality, lexicographic ordering, and quantitative ranking. Therefore the value of our proposal is more in the embedding of such semantics in the context of mCP-nets, where indifference and incomparability coexist, rather than in their concept.

Given a mCP-net and two outcomes α and β, let $S_{\succ}$, $S_{\prec}$, $S_{\approx}$ and $S_{\bowtie}$ be the sets of agents who say, respectively, that $\alpha \succ \beta$, $\alpha \prec \beta$, $\alpha \approx \beta$, and $\alpha \bowtie \beta$. Note that it is not possible that $|S_{\succ}| = |S_{\prec}| = |S_{\bowtie}| = 0$ (i.e., every agent to be indifferent) since the definition of mCP-net (Definition 5.4) requires that every feature is ranked by at least one agent. Thus, given any two outcomes, some agent must order them or say they are incomparable.

Pareto. This semantics is one of consensus. We say that one outcome α is better than another β (written $\alpha \succ_p \beta$) iff every agent says that $\alpha \succ \beta$ or $\alpha \approx \beta$. That is, $\alpha \succ_p \beta$ iff $|S_{\prec}| = |S_{\bowtie}| = 0$. Two outcomes are Pareto incomparable iff they are not ordered either way. An outcome is Pareto optimal iff no other outcome is better. Consensus is a stringent test and outcomes will often be incomparable as a consequence.

Majority. An alternative criterion is just that a majority of the agents who are not indifferent vote in favor. We say that one outcome α is majority better than another β (written $\alpha \succ_{maj} \beta$) iff $|S_{\succ}| > |S_{\prec}| + |S_{\bowtie}|$. Two outcomes are majority incomparable iff they are not ordered either way. An outcome is majority optimal iff no other outcome is majority better.

Max. A weaker criterion is that more agents vote in favor than against or for incomparability. We say that one outcome α is max better than another β (written $\alpha \succ_{max} \beta$) iff $|S_{\succ}| > \max(|S_{\prec}|, |S_{\bowtie}|)$. Two outcomes are max incomparable iff they are not ordered either way. An outcome is max optimal iff no other outcome is max better.

Lex. This semantics assumes the agents are ordered in importance. If the first agent orders two outcomes then this is reflected in the final outcome. However, if they are indifferent between two outcomes, we consult the second agent, and so on. We say that one outcome α is lexicographically better than another β (written $\alpha \succ_{lex} \beta$) iff there exists some distinguished agent such that all agents higher in the order say $\alpha \approx \beta$ and the distinguished agent says $\alpha \succ \beta$. Two outcomes are lexicographically incomparable iff there exists some distinguished agent such that all agents higher in the ordered are indifferent between the two outcomes and the outcomes are incomparable to the distinguished agent. Finally, an outcome is lexicographically optimal iff no other outcome is lexicographically better.

Rank. This semantics eliminates incomparability. Each agent ranks each outcome. Given a partial CP-net, the rank of an outcome is zero if the outcome is optimal, otherwise it is the length of the shortest chain of worsening flips between one of the optimal outcomes and it. We say that one outcome α is rank better than another β (written $\alpha \succ_r \beta$) iff the sum of the ranks assigned to α is smaller than that assigned to β. Two outcomes are rank indifferent iff the sum of the ranks assigned to them are equal. Either two outcomes are rank indifferent or one must be rank better than the other. Finally, an outcome is rank optimal iff no other outcome is rank better.
In Table 5.1 we consider a scenario where there are five agents and we show some examples of how they could vote over pair of outcomes and the corresponding results in the different semantics.

5.3.4 Properties of the orders induced by the semantics

The Pareto, Lex and Rank semantics define strict orderings. By comparison, neither the Majority or Max semantics induce a strict ordering. More precisely, the following properties can be proved.

A1	**A2**	**A3**	**A4**	**A5**	**P**	**Maj**	**Max**	**Lex**
$o \succ o'$	$o \succ o'$	$o \succ o'$	$o \succ o'$	$o \approx o'$	$o \succ o'$	$o \succ o'$	$o \succ o'$	$o \succ o'$
$o \bowtie o'$	$o \succ o'$	$o \approx o'$	$o \approx o'$	$o \approx o'$	$o \bowtie o'$	$o \bowtie o'$	$o \bowtie o'$	$o \bowtie o'$
$o \succ o'$	$o \succ o'$	$o \succ o'$	$o \bowtie o'$	$o \prec o'$	$o \bowtie o'$	$o \succ o'$	$o \succ o'$	$o \succ o'$
$o \succ o'$	$o \succ o'$	$o \prec o'$	$o \bowtie o'$	$o \approx o'$	$o \bowtie o'$	$o \bowtie o'$	$o \succ o'$	$o \succ o'$

Table 5.1: Some examples of possible votes among five agents (A1, A2, A3, A4, A5) and the result of the different qualitative semantics (Pareto, Majority, Max and Lex). Note that in Lex we are assuming that the agents appear in order of importance and that rank has been omitted since it is a quantitative method.

Theorem 5.5 $\succ_p$, $\succ_{lex}$ *and* $\succ_r$ *are transitive, irreflexive, and antisymmetric. Thus they induce strict partial orders.* $\succ_{maj}$ *and* $\succ_{max}$ *are irreflexive and antisymmetric but may be not transitive.*

Proof: Let us start the proof considering Pareto. Pareto is transitive. In fact, given three outcomes o_1, o_2, and o_3 then $o_1 \succ_p o_2$ iff for all agents either $o_1 \succ o_2$ or $o_1 \approx o_2$ (notice that the two outcomes cannot be indifferent for all). Similarly for $o_2 \succ_p o_3$. Since we are assuming that each voter reasons on preference using an acyclic partial CP-net each ordering must be transitive. Thus, we can conclude that for each agent either $o_1 \succ o_3$ or $o_1 \approx o_3$. But this implies $o_1 \succ_P o_3$. Moreover, Pareto is irreflexive since the order of each agent is a strict partial order. It is antisymmetric since $o_1 \succ_P o_2$ iff $o_1 \succ o_2$ for $o_1 \approx o_2$ and $o_1 \prec_P o_2$ iff $o_1 \prec o_2$ for $o_1 \approx o_2$. But the order of each agent is antisymmetric, thus if an gent says $o_1 \prec o_2$ and $o_1 \succ o_2$ it must be that o_1 is exactly o_2. From this we can deduce that $\succ_p$ is antisymmetric as well.

We now consider $\succ_{lex}$. Given h agents in order of importance, $A_1, \cdots, A_h$ then $o_1 \succ_{lex} o_2$ iff there is an agent, A_j such that, $o_1 \approx o_2$ for all $i < j$ and $o_1 \succ o_2$ for agent A_j. Similarly $o_2 \succ_{lex} o_3$ iff there is an agent, A_k such that, $o_2 \approx o_3$ for all $i < k$ and $o_2 \succ o_3$ for agent A_k. Without loss of generality assume that $j < k$, then agent A_j must say $o_1 \succ o_3$ since there is a path from o_1 to o_3 passing through o_2

A1	**A2**	**A3**	**A4**	**A5**	**Maj**	**Max**
$o_1 \succ o_2$	$o_1 \succ o_2$	$o_1 \succ o_2$	$o_1 \bowtie o_2$	$o_1 \bowtie o_2$	$o_1 \succ o_2$	$o_1 \succ o_2$
$o_2 \bowtie o_3$	$o_2 \bowtie o_3$	$o_2 \succ o_3$	$o_2 \succ o_3$	$o_2 \bowtie o_3$	$o_2 \succ o_3$	$o_2 \succ o_3$
$o_3 \succ o_1$	$o_3 \succ o_1$	$o_1 \succ o_3$	$o_3 \succ o_1$	$o_3 \succ o_1$	$o_3 \succ o_1$	$o_3 \succ o_1$

Table 5.2: Some examples of possible votes among five agents (A1, A2, A3, A4, A5) showing that $\succ_{max}$ and $\succ_{maj}$ are not transitive.

which consists of an improving flip and an indifferent flip. However since for all $i < j$, $o_1 \approx o_2$ and $o_2 \approx o_3$, then it must be that for each of those agents $o_1 \approx o_3$. Thus the ordering of lex is that of agent A_j. Hence, $o_1 \succ_{lex} o_3$. Since for each agent we have that $o \not\succ o$, $\succ_{lex}$ is irreflexive. As far as the antisymmetric property, we know that $o_1 \succ_{lex} o_2$ iff there is an agent A_j such that $o_1 \approx o_2$ for all $i < j$ and $o_1 \succ o_2$ for agent A_j. If also $o_1 \succ_{lex} o_2$ it must be $o_1 = o_2$, due to the fact that the ordering of A_j is antisymmetric.

Rank is transitive and irreflexive and antisymmetric because strict inequality on integers has these properties.

We will now consider the properties of $\succ_{maj}$. We can conclude that it is irreflexive since $o \not\succ o$ for all agents. An agent either says $o_1 \succ o_2$ or $o_1 \prec o_2$ or $o_1 \approx o_2$ or $o_1 \bowtie o_2$. Assume that $o_1 \succ_{maj} o_2$ and $o_2 \succ_{maj} o_1$. Then it must be that $|S_\succ| > |S_\bowtie| + |S_\prec|$ but also $|S_\prec| > |S_\bowtie| + |S_\succ|$. We can conclude that at least one agent must have voted both $o_1 \succ o_2$ and $o_1 \prec o_2$. But since the agent's orderings are antisymmetric then o_1 must be o_2.

The same holds for $\succ_{max}$. In fact, as far as irreflexivity, $o \not\succ o$ for all agents allows us to conclude also for $\succ_{max}$. Assume that $o_1 \succ_{max} o_2$ and $o_2 \succ_{max} o_1$. Then it must be that $|S_\succ| > max(|S_\bowtie|, |S_\prec|)$ but also $|S_\prec| > max(|S_\bowtie|, |S_\succ|)$. We can conclude, as before, that at least one agent must have voted both $o_1 \succ o_2$ and $o_1 \prec o_2$ and thus o_1 and o_2 must be the same outcome.

It is easy to see that $\succ_{max}$ and $\succ_{maj}$ are not transitive in general. An example is shown in Table 5.2.

The five relations are closely related. As a criterion to relate them we use the notion of subsumption.

Definition 5.5 (subsumption) *A binary relation* R subsumes *another* S *iff* xRy *implies* xSy. *We say that a binary relation* R *strictly subsumes another* S *iff* R *subsumes* S *but not vice versa. Finally, we say that two binary relations are incomparable iff neither subsumes the other.*

In the following theorem we consider the semantics in terms of subsumption.

Theorem 5.6 $\succ_p$ *strictly subsumes* $\succ_{maj}$, $\succ_{lex}$. *Similarly,* $\succ_{maj}$ *strictly subsumes* $\succ_{max}$. *Any other pair of relations is incomparable.*

Proof: Let us prove the first statement: $\succ_p$ strictly subsumes $\succ_{maj}$. Two outcomes o and o' are in the following relation $o \succ_p o'$ iff for all agents either $o \succ o'$ or $o \approx o'$. This is true iff $|S_{\prec}| = 0$ and $|S_{\bowtie}| = 0$. Since we know that $|S_{\succ}| > 1$, it must be $|S_{\succ}| > |S_{\prec}| + |S_{\bowtie}|$. Thus $o \succ_{maj} o'$. The converse does not hold since it may be that $|S_{succ}| > |S_{\prec}| + |S_{\bowtie}|$ with $|S_{\prec}| > 1$ and thus the pair is ordered by Majority but is incomparable for Pareto.

Moreover, if for all agents either $o \succ o'$ or $o \approx o'$, then the first agent in the importance order of lex that strictly orders o and o' must say $o \succ o'$. Thus, $o \succ_{lex} o'$. However, if $o \succ_{lex} o'$ holds, it might be that $o \bowtie_p o'$. In fact, lex looks only at one agent, the first that orders the pair, while following agents could have ordered the pair in the opposite way, giving incomparability in Pareto.

Relation $\succ_{maj}$ strictly subsumes $\succ_{max}$ since $|S_{succ}| > |S_{\prec}| + |S_{\bowtie}|$ implies $|S_{succ}| > max(|S_{\prec}|, |S_{\bowtie}|)$ but not vice versa.

Max and Lex are incomparable and Maj and Lex are incomparable as shown by the example in Table 5.3.

Rank is incomparable with Pareto, and thus with all the others semantics. As an example consider five agents which have as ordering that shown in Figure 5.2. All voters will say that $cde \succ cdl$, thus $cde \succ_p cdl$. However the rank associated to cde is 3 and that associated to cdl is 2. Thus $cdl \succ_r cde$.

A1	A2	A3	A4	A5	Maj	Max	Lex
$o_1 \prec o_2$	$o_1 \succ o_2$	$o_1 \succ o_2$	$o_1 \succ o_2$	$o_1 \succ o_2$	$o_1 \succ o_2$	$o_1 \succ o_2$	$o_1 \prec o_2$

Table 5.3: Some examples of possible votes among five agents (A1, A2, A3, A4, A5) showing that $\succ_{max}$ and $\succ_{maj}$ are incomparable with $\succ_{lex}$.

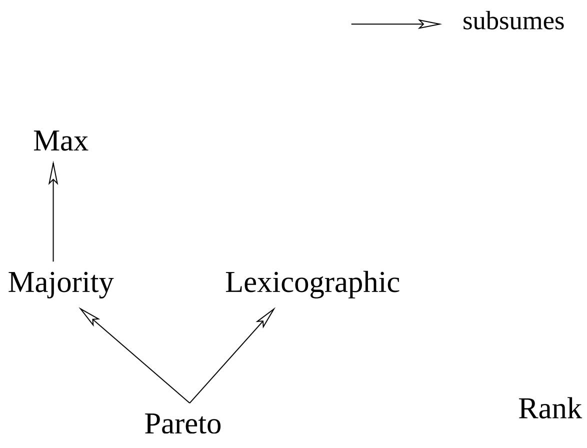


Figure 5.4: Subsumptions among the six relations corresponding to the voting semantics.

Figure 5.4 shows the subsumption schema over the five relations.

Given two outcomes, we can therefore first try to compare them with $\succ_p$. If they are incomparable, we have three options: either we try to compare them with $\succ_{maj}$ and if they remain incomparable with $\succ_{max}$, or we try to compare them with $\succ_{lex}$. All options enlarge the set of pairs of outcomes which are comparable. In alternative we can use the quantitative approach in $\succ_r$. Only this last option (comparing them with $\succ_r$) is guaranteed to order any two outcomes.

Notice that in all the five relations, except Rank, it is not possible for two outcomes to be indifferent, since we assume that each feature is ranked by at least one of the partial CP-nets, while indifference in the qualitative relations (Pareto, Max, Majority, and Lex) means indifference for everybody.

Optimality From these subsumption results, the following relationships hold between the various optimal outcomes.

Theorem 5.7 *A majority or lexicographical optimal outcome is necessarily Pareto optimal. However, the reverse is not true. A max optimal outcome is necessarily majority optimal, but the reverse is not true.*

Proof: Consider two outcomes o and o'. From Theorem 5.6, we have that if $o \succ_{maj} o'$ or $o \succ_{lex} o'$ then either $o \succ_p o'$ or $o \bowtie_p o'$, but not $o \prec_p o'$. Thus if o is optimal for $\succ_{maj}$ or $\succ_{lex}$ it is optimal for $\succ_p$ as well.

To see that the converse does not hold consider the example shown in the third row of Table 5.1. Assume there are only two outcomes o and o'. For Pareto they are both optimal, however o' is not optimal for Majority nor for Lex.

As above, from Theorem 5.6 we know that if $o \succ_{maj} o'$ then $o \succ_{max} o'$. This implies that if $o \succ_{max} o'$, then it can be that $o \succ_{maj} o'$ or $o \bowtie_{maj} o'$ but not $o \prec_{maj} o'$. Thus whatever is optimal for $\succ_{max}$ remains optimal for $\succ_{maj}$. As an example of an outcome which is optimal for Majority but not for Max consider the last row of Table 5.1. If there are only two outcomes, o and o', then for Majority they are both optimal, while o' is not optimal for Max.

Similarly, from the incomparability results, we can deduce incomparability also for the respective optimal outcomes.

A fundamental question is whether any optimal outcomes exist. In the following theorem we show that all the semantics we have considered induce an ordering which always has at least an optimal.

Theorem 5.8 *An optimal outcome exists for the Pareto, Majority, Max, Lex and Rank semantics.*

Proof: In Theorem 5.5 we have shown that all the induced orderings are strict partial orders (except rank which is a total order). This means that in the ordering corresponding to all five semantics there cannot be cycles, since that would contradict the antisymmetric property. Since we are assuming that there are a finite number of features with finite domains, we have strict partial orders or a total order over a finite set. This implies the existence of undominated outcomes.

Notice that the set of Pareto optimal outcomes does not necessarily coincide with the set of outcomes which are optimal for all the CP-nets. In fact, if an outcome is optimal for all, then it is Pareto optimal. However, the converse is not true in general. For example, consider the set of optimal outcomes of a single partial CP-net. If they are all incomparable (no matter if they are not optimal for the other CP-nets), then they are also Pareto optimal for the mCP-net.

We will now give several results on testing optimality and finding optimal outcomes. In what follows, we assume that there are n binary features in the mCP-net, k features in the largest partial CP-net, and at most l visible features in a partial CP-net.

Theorem 5.9 *Optimality can be tested with the following complexities:*

- *for Pareto, Majority and Max:* $O(m2^{n+k})$ *time;*
- *for Lex:* $O(m2^{l+k})$*;*
- *for Rank:* $O(m2^{l+k} + 2^n m2^k) = O(m2^{n+k})$.

Proof: We will now consider separately each of the statements of the theorem.

- To test if an outcome is Pareto optimal, we need to compare ($O(2^k)$) the given outcome with all other outcomes ($O(2^n)$) in all partial CP-nets (m).
- Similarly for Max and Majority.
- For lexicographical optimality, we check if the given outcome is optimal for the first agent ($O(k)$). If it is not, it is not lexicographically optimal. If it is, then, for any optimal outcome of the first agent to which it is indifferent (there may be 2^l of them), we must ask the second agent if the given outcome is better ($O(2^k)$). If also the second agent is always indifferent, we must turn to the next agent and so on (m agents).

- Finally, for rank optimality, we need to first rank all outcomes for each partial CP-net. To do this, we start with the optimal ones, which are $O(2^l)$ and have rank 0, and for each of them we build the flipping sequence to any other of the outcomes (which are 2^k). Then, for each outcome (2^n), we compute the global rank by summing the ranks in each partial CP-net (the rank tables have 2^k elements) and we check that the rank of the given outcome is minimal.

The above theorem can be summarized by saying that testing optimality can be easy for Lex if the size of the partial CP-nets (k) is bounded, while it is difficult for Pareto, Majority, Max and Rank.

We now consider the problem of finding an optimal outcome.

Theorem 5.10 *An optimal outcome can be found using algorithms with the following complexities:*

- *for Majority and Max:* $O(m2^{2n+k})$*;*
- *for Lex and Pareto:* $O(m2^{2l+k})$*; the same complexity holds also to find all optimal outcomes, not just one;*
- *for Rank:* $O(m2^{n+k})$ *(for the first optimal outcome, linear for the others).*

Proof:

- For Majority, we need to compare ($O(2^k)$) all outcomes (2^n) to all other outcomes ($O(2^n)$) in all partial CP-nets (m). To find a max optimal outcome, we must do the same steps as for Majority.
- To find a lexicographically optimal outcome, we compute all the optimal outcomes for the first agent ($O(k2^l)$), and then we do the same steps as for testing optimality ($O(m2^l2^k)$) for each one of them (they can be 2^l).

- For Pareto optimality, since lexicographically optimals are also Pareto optimals (Theorem 5.7), the complexity is the same as for Lex.
- To find a rank optimal outcome, we need to perform the same steps as for testing optimality, except that we don't have to compare with the given outcome. However, this does not change the overall complexity.

To summarize, an optimal outcome can be found easily for Pareto and Lex if the size of the partial CP-nets (k) is bounded, while the complexity of the procedures proposed for Majority, Max and Rank are exponential. Thus, with respect to optimality testing, Pareto now falls into the easy category.

Dominance We will now consider the complexity for dominance testing using the different semantics. We again assume that there are n binary features in the mCP-net, k in the largest partial CP-net, and l visible features.

Theorem 5.11 *Given two outcomes, testing if one dominates the other can be achieved with the following complexity:*

- *for Pareto, Majority, Max, and Lex:* $O(m2^k)$*;*
- *for Rank:* $O(m2^{n+k})$*.*

Proof:

- To determine if $o_1 \succ_p o_2$, we must test whether $o_1 \succ o_2$ or $o_1 \approx o_2$ for all m partial CP-nets. Each of these tests takes at most $O(2^k)$ time.
- To determine if $o_1 \succ_{maj} o_2$, we must test whether $o_1 \succ o_2$ for all m partial CP-nets. Only the last test may give us the required majority.
- Similarly, to determine if $o_1 \succ_{max} o_2$, we must test whether $o_1 \succ o_2$ for all m partial CP-nets.

- Even for Lex, in the worst case, we must check dominance or indifference in all CP-nets, since it may be that they are indifferent for all partial CP-nets except the last one.

- For Rank, a brute force algorithm would rank all outcomes and then compare the ranks of the given two. This takes $O(m2^{n+k})$ as explained above for testing Rank optimality.

Thus, although Rank gives us a quantitative ordering, it is easier to test dominance in the qualitative semantics (Pareto, Majority, Max, and Lex). In fact, if the number and size of the partial CP-nets are bounded, it takes constant time. The fact that Majority and Max may be not transitive implies that for such semantics we cannot optimize any dominance test exploiting the transitive property. Thus, while in the worst case the complexity is the same as for Pareto and Lex, in practice dominance testing can be much easier for such two semantics.

5.3.5 Comparison among the semantics

Table 5.4 summarizes the main properties of the semantics considered in this section for mCP-nets. In particular, we show if the semantics yields a strict order or not and we compare the complexity of testing optimality and dominance, and of finding an optimal outcome. This table lets us compare the five semantics easily.

For example, Lex is the semantics to choose if it is reasonable to order the agents. On the other hand, Pareto is the best in terms of complexity (except for optimum testing). However, as it looks for a consensus among all agents, it orders the fewest pairs of outcomes. Max and Majority ordering are less desirable in terms of complexity, since only dominance testing is easy (under certain assumptions). Since Majority subsumes Max, Majority has more optimal outcomes and a weaker ordering. Finally, the Rank semantics has bad complexity. This bad complexity seems a result of merging the qualitative nature of CP-nets with the quantitative spirit of a rank. Note that this complexity is only seen the first time

	Pareto	Maj	Max	Lex	Rank
S.O.	yes	no	no	yes	yes
Test opt.	diff	diff	diff	e-k	diff
Find opt.	e-k	diff	diff	e-k	diff
Dominance	e-k c-m	e-k c-m	e-k c-m	e-k c-m	diff

Table 5.4: Comparison among the five semantics. Legenda: S.O. = strict order, diff = difficult, e-k = easy if k bounded, c-m = constant if m bounded).

two outcomes are ordered, and subsequent queries can be answered in linear time.

5.4 Fairness of Preference Aggregation

As shown in Section 5.3 for situations involving multiple agents, we need to combine the preferences of several individuals. In this section, we consider each agent as voting on whether they prefer one outcome to another. We prove that, under certain conditions on the kind of partial order allowed to express the preferences in the result, if there are at least two agents and three outcomes to order, no preference aggregation system can be fair. That is, no preference aggregation system can be free (give any possible result), monotonic (improving a vote for an outcome only ever helps), independent to irrelevant assumptions (the result between two outcomes only depends on how the agents vote on these two outcomes), and non-dictatorial (there is not one agent who is never contradicted). This result generalizes Arrow's impossibility theorem for combining total orders [Arr51].

The formalisms for representing preferences we have considered in Chapter 4 and in this chapter, namely Soft Constraints and CP-nets, provide an ordering on outcomes. In general, this ordering is partial as outcomes may be incomparable. For example, when comparing wines, we might prefer a white wine grape like chardonnay to the sauvignon blanc grape, but we might not want to order chardonnay compared to a red wine grape like merlot.

The result of aggregating the preferences of multiple agents is itself naturally a partial order. If two outcomes are incomparable for each agent, it is reasonable for them to remain incomparable in the final order. Incomparability can also help us deal with disagreement between the agents. If some agents prefer A to B and others prefer B to A, then it may be best to say that A and B are incomparable (as in the Pareto semantics). Here, we consider this kind of scenario. We assume each agent has a preference ordering on outcomes represented via soft constraints, CP-nets or any other mechanism. A preference aggregation procedure then combines these partial orders to produce an overall preference ordering, and this again can be a partial order, as shown in Section 5.3. The question we address here is: can we combine such preferences fairly?

Suppose we have 100 agents, and 50 of them prefer A to B, whilst the other 50 prefer B to A. It might seem reasonable to decide that A is incomparable to B. But what if 99 agents prefer A to B, and only one prefers B to A? Rather than always declare A to be incomparable to B whenever someone votes against A being better to B, is there not a more sophisticated voting system that will decide A is better than B and compensate the agent who preferred B to A on some other outcomes? We will show that, if each agent can order the outcomes via a partial order, and if the result has to be a partial order with a unique top or a unique bottom, then any preference aggregation procedure is ultimately unfair.

This result is both disappointing and a little surprising. By moving from total order to a class of partial orders, we enrich greatly our ability to combine outcomes fairly. If agents disagree on two outcomes, we can always declare them to be incomparable. In addition, a partial ordering can have multiple outcomes which are optimal. Unlike an election, we need not declare a single winner. Nevertheless, we still do not escape the reach of Arrow's theorem. Any voting system will have one or more agents who dictate the overall preferences.

However, preferences may need more relaxed orders than the one considered in our results. For example, CP-nets, even acyclic ones, may produce partial orders with more than one bottom. Soft constraints produce arbitrary partial orders. So we still hope there is a voting semantics for preference aggregation which is fair.

5.4.1 Social welfare functions for partial orders

As we said in section 5.2.2, one property of preference aggregation which is highly desirable is fairness. We will now consider fairness also in the context of partially ordered preferences. We, thus, assume that each agent's preference specify a partial order over the possible outcomes. We aggregate the preferences of a number of agents using a social welfare function. A *social welfare function* f is a function from profiles p to orderings over outcomes. A *profile* p is a sequence of n orderings $p_1, \ldots, p_n$ over outcomes, one for each agent $i \in \{1, \ldots, n\}$.

There are a number of properties that a social welfare function might be expected to possess. Except in the case of dictator, they are straightforward generalizations of the corresponding properties for social welfare functions for total orders [ASS02]:

- *Freeness*: f is surjective, that is, it can produce any ordering.

- *Unanimity*: if all agents agree that a is preferable to b, then the resulting order must agree as well. That is, if $a >_{p_i} b$ for all agents i, then $a >_{f(p)} b$. Notice that a stronger notion could be defined, where unanimity over incomparability is also required. However, this is not needed for the results of our paper.

- *Independence to irrelevant alternatives*: the ordering between a and b in the result depends only on the relation between a and b given by the agents; that is, for all profiles p and p', for all a, b, for all agents i, if $p_i(a, b) = p'_i(a, b)$, then $f(p)(a, b) = f(p')(a, b)$, where, given an ordering o, $o(a, b)$ is the restriction of o on a and b.

- *Monotonicity*: We say that b improves with respect to a if the relationship between a and b does not move to the left along the following sequence: $>, \geq, (\bowtie \text{ or } =), \leq, <$. Given two profiles p and p', if passing from p to p' b improves with respect to a in one agent i and $p_j = p'_j$ for all $j \neq i$, then in passing from $f(p)$ to $f(p')$ b improves with respect to a.

Another desirable property of social welfare functions is the absence of a dictator. With partial orders, there are several possible notions of dictator:

Strong dictator: an agent i such that, in every profile p, $f(p) = p_i$, that is, her ordering is the result;

Dictator: an agent i such that, in every profile p, if $a \geq_{p_i} b$ then $a \geq_{f(p)} b$.

Weak dictator: an agent i such that, in every profile p, if $a \geq_{p_i} b$, then $a \not< _{f(p)} b$.

Nothing is said about the result if a is incomparable or indifferent to b for the dictator or weak dictator. Clearly a strong dictator is a dictator, and a dictator is a weak dictator. Note also that whilst there can only be one strong dictator or dictator, there can be any number of weak dictators.

A social welfare function with a dictator (resp., strong dictator, weak dictator) will be said to be dictatorial (resp., strongly dictatorial, weakly dictatorial). By using these three notions of dictator, we correspondingly define three notions of fairness:

- *strong fairness*: when the social welfare function is unanimous, independent to irrelevant alternatives, and not strongly dictatorial;

- *fairness*: when the social welfare function is unanimous, independent to irrelevant alternatives, and not dictatorial;

- *weak fairness*: when the social welfare function is unanimous, independent to irrelevant alternatives, and not weakly dictatorial.

Given the relationship among the three notions of dictator, it is easy to see that weak fairness implies fairness, and fairness implies strong fairness.

We first show that the absence of a strong dictator is a very weak property to demand of a social welfare function over partial orders. Even an "unfair" voting system like the Lex rule, which explicitly favours a particular agent, is not strongly dictatorial.

Proposition 5.1 *Given a set of agents and a set of outcomes, and assuming each agent gives a partial order with ties of the outcomes, and the result is a partial order with ties, a social welfare function can be strongly fair (that is, free, monotonic, independent of irrelevant assumptions, and without any strong dictator).*

The Lex function defined above is free, monotonic, and independent to irrelevant assumptions. The absence of a strong dictator might seem to be in contradiction to the nature of Lex as there is a most important agent. However, such an agent does not dictate incomparability (which is required to be a strong dictator), since whatever is left incomparable by this agent can then be ordered by some less important agent. Thus Lex is strongly fair.

Notice however that the first agent in the Lex function is always a dictator since if she states that a is better than b then a is better than b in the result. So the Lex function is not fair.

However, there are fair social welfare functions on partial orders, as the following theorem states.

Proposition 5.2 *Given a set of agents and a set of outcomes, and assuming each agent gives a partial order with ties of the outcomes, and the result is a partial order with ties, a social welfare function can be fair (that is, free, monotonic, independent of irrelevant assumptions, and not dictatorial).*

For example, the Pareto function defined above is free, monotonic, independent to irrelevant assumptions, and not dictatorial. A particular agent can only force the result by stating the incomparability of all possible outcomes. However, this is not considered to be dictatorial according to our definition of dictatorship.

5.4.2 Weak fairness: impossibility results

We will now show that, under certain conditions, it is impossible for a social welfare function over partial orders to be weakly fair. The conditions involve the shape of the partial orders. In fact, we assume the partial orders of the agents to be general (PO), but the resulting partial order must have all top or all bottom elements indifferent (rPO).

Theorem 5.12 *Given a social welfare function f over partial orders, assume the result is a rPO, there are at least 2 agents and 3 outcomes, and f is unanimous and independent to irrelevant alternatives. Then there is at least one weak dictator.*

Proof: The proof is similar in outline to one of the proof of Arrow's theorem [Gea01]. However, we must adapt each step to this more general context of partial orders. We assume the resulting ordering is a rPO with all bottom elements indifferent. The proof can then be repeated very similarly for the other case in which the resulting ordering is a rPO with all top elements indifferent.

1. First we prove that, if a element b is at the very top or at the very bottom in all POs of the agents, then it must be a top or bottom element in the resulting rPO. If b is not a top or bottom element in the result, there must be other elements a and c such that $a > b > c$. We will now adjust the profile so that c is above a for all agents. Since we assumed just one top and one bottom for all agents, we can always do that while keeping b at the extreme position and not changing the ordering relation between a and b and between c and b.

   By unanimity, we must have c above a in the result. By independence to irrelevant alternatives, we still have $a > b$ and $b > c$. By transitivity, we have $a > c$ which is a contradiction.

2. We now prove that there is a pivotal agent n^* such that, when he moves b from bottom to top, the same change happens in the result.

Assume all agents have b as the bottom. Then, b must be at the bottom in the result by unanimity. Let n^* be the first agent such that, when b moves from bottom to top, this happens in the result. Note that n^* must exist, because when all agents move b from bottom to top, by unanimity in the result we must have b at the top.

3. We continue by proving that n^* is a weak dictator for pairs of elements not involving b.

   Let us consider the following scenarios in the context of moving b from the bottom to the top of each agent's ranking.

   Π_1: b is still the bottom of n^*. In the result, b is the bottom element so we must have, $a > b$ for all a.

   Π_2: b has been moved to the top of n^*. In the result, b is a top element so we must have, $b \geq c$ or b incomparable to c for all c.

   Π_3: As in Π_2 but a has now been moved above b in n^* (and thus also above c), and all other agents move freely a and c leaving b in the top or bottom position.

   By independence to irrelevant alternatives, $a > b$ must be the result of Π_3, since all the ab preferences are the same as in Π_1. Also, $b \geq c$ or b incomparable to c must be the result of Π_3, since all $b-c$ preferences are the same as in Π_2. By transitivity, the result of Π_3 cannot have $c > a$ since it would imply $c > b$ which is contradictory with the assumption that b and c are either incomparable or $b \geq c$. Thus n^* is a weak dictator for pairs not involving b.

4. We now prove that there exists an agent n' which is a weak dictator for pairs with no c. To do this, it is enough to use the same construction as above but with c in place of b.

5. We show now that $n^* = n'$. On total orders, there can be only one dictator, so it follows immediately that $n^* = n'$. With partial orders, there can be

more than one weak dictator. The argument that $n^* = n'$ is therefore much more elaborate. Without loss of generality, assume $n^* \leq n'$. Suppose that $n^* < n'$. Let us consider the following profiles: we start with all agents having b at the bottom and c at the top. Then we swap b and c in each orderings, going through the agents in the same order as in the previous constructions. When we move b up for n^*, b goes to the top in the result (by the previous part of the proof). c goes down for n^*, and in the result it can go down as well, in which case we can repeat the same construction as in points 1,2,3 starting with c at the top instead of b at the bottom, and we can prove that n^* is also a weak dictator for pairs not involving c. Thus we would have $n^* = n'$, which is a contradiction. Otherwise, c could stay at the top. However, since n^* is a weak dictator for pairs not involving b, and since c is the bottom for n^*, then all the elements must be at the top with c (as incomparable or indifferent elements). This is true also in any other profile obtained from the current one by leaving all the other agents move freely a and b. Thus n^* is not contradicted on any pair.

As with total orders, we can also prove a weaker result in which we replace unanimity by monotonicity and freeness.

Theorem 5.13 *Given a social welfare function f over partial orders, assume the result is a rPO, there are at least 2 agents and 3 outcomes, and f is free, monotonic, and independent to irrelevant alternatives. Then there is at least one weak dictator.*

Proof: Suppose the social welfare function is free and monotonic, and that $a \geq b$ for all agents. If a is moved to the top of the ordering for all agents then, by independence to irrelevant alternatives, this leaves the result between a and b unchanged. Suppose in the result $a < b$ or a is incomparable to b. By monotonicity, any change to the votes of a over b will not help ensure $a \geq b$. Hence, the election cannot be free. Thus it must be $a \geq b$ in the result. The voting system is therefore unanimous. By Theorem 5.12, there must be at least one weak dictator.

Consider, for example, the Pareto rule. With this rule, every agent is a weak dictator since no agent can be contradicted. Note that we could consider a social welfare function which modifies Pareto by applying the rule only to a strict subset of the agents, and ignores the rest. The agents in the subset will then all be weak dictators.

Implied impossibility results

A number of results follow immediately from these theorems. If we denote the class of all social welfare functions from profiles made with orders of type A to orders of type B by $A^n \mapsto B$, then we have proved the impossibility of being weakly fair for functions in $PO^n \mapsto rPO$.

If all functions in $A^n \mapsto B$ are not weakly fair, then also functions in $A^n \mapsto B'$, where B' is a subtype of B, are not weakly fair, since by restricting the codomain of the functions we are just considering a subset of them. Therefore this impossibility theorem implies also that the functions in $PO^n \mapsto O$, where O is anything more ordered than a rPO, cannot be weakly fair. For example, we can deduce that functions in $PO^n \mapsto TO$ cannot be weakly fair.

The same reasoning applies when we restrict the domain of the functions, that is, we pass from $A^n \mapsto B$ to $A'^n \mapsto B$ where A' is a subtype of A. In fact, by doing this, we restrict the kind of profiles we consider, so whatever is true in all the profiles of the larger set is also true in a subset of the profiles. In particular, if a function from A^n to B has a weak dictator, then the same function, restricted over the profiles in A'^n, also has the same weak dictator. Thus if the functions in $A^n \mapsto B$ cannot be weakly fair, also the functions in $A'^n \mapsto B$ cannot be weakly fair. In particular, from our result we can deduce that all functions in $TO^n \mapsto rPO$ cannot be weakly fair. Finally, we can deduce that all functions in $TO^n \mapsto TO$ cannot be weakly fair, which is exactly Arrow's theorem. In fact, we have a lattice of impossibility results for classes of social welfare functions, as described by Figure 5.5.

$$\begin{tikzcd}
 & PO^n \mapsto rPO \arrow[dl] \arrow[dr] & \\
PO^n \mapsto TO \arrow[dr] & & TO^n \mapsto rPO \arrow[dl] \\
 & TO^n \mapsto TO &
\end{tikzcd}$$

Figure 5.5: Lattice of impossibility results. rPO stands for partial order where top elements or bottom elements are all indifferent, PO stands for partial order, TO stands for total order. Arrow's theorem applies to $TO^n \mapsto TO$. $\swarrow$ and $\searrow$ stand for the lattice ordering.

5.4.3 Two possibility results

We now consider ways of assuring that a preference aggregation function is weakly fair. To do this, we will, focus here on the condition that Sen has proved sufficient for fairness in the case of total orders, that is, triplewise value-restriction [Sen70]. We recall that this property requires that for every triple of outcomes x_1, x_2, x_3, there exists $x_i \in \{x_1, x_2, x_3\}$ and $r \in \{1, 2, 3\}$ such that no agent ranks x_i as his r-th preference among x_1, x_2, x_3.

In our scenario, where agents may use any PO to express their preferences, and the result of a social welfare function can be any PO, Sen's theorem does not directly apply. However, we will give a generalization which directly applies the statement of Sen's theorem to profiles derived from our profiles which are sequences of POs, without the need to modify its proof.

Now that we have partial orders, to assure transitivity in the resulting order, we must avoid both cycles (as in the total order case) and incomparability in the wrong places. More precisely, if the result has $a > b > c$, we cannot have $c > a$, which would create a cycle, and not even a incomparable to c, since in both cases transitivity would not hold.

A profile is a sequence of strict POs. We will say that a profile is *good* if all the profiles obtained by linearizing the strict POs to obtain strict TOs have the triplewise value-restriction property.

Theorem 5.14 *Consider the social welfare function* f *defined as Majority in Section 5.3.3. Assume* f *takes a profile* p *and returns an ordering* o*. If* p *is good, then* o *is transitive. That is,*

1. *it is not possible that* o *has a cycle;*

2. *for any* a, b, c*, it is not possible to have* $a > b > c$ *and* a *incomparable to* c *in* o*.*

Proof: We recall here the definition of the Majority function: $\alpha \succ_{maj} \beta$ iff the number of agents which say that $\alpha \succ \beta$ is greater than the number of agents which say that $\beta \succ \alpha$ plus the number of those that say that α and β are incomparable.

Take any profile p' which is a linearization of p. Since p is good, then p' has the triplewise value-restriction property, thus $Majority(p')$ produces an ordering without cycles, by Sen's theorem. Since p' is a linearization of p, p has a smaller or equal set of ordered pairs w.r.t. p'. Therefore, if Majority has not produced any cycle starting from p', it cannot produce any cycle if it starts from p. In fact, Majority counts the number of people who order a pair, so if the profile has less ordered pairs, a smaller or equal number of pairs are ordered also in the result.

Assume now we have $a > b > c$ and a incomparable to c in the ordering o. We will now show that if this is the case then there is a linearization which doesn't satisfy the triplewise value-restriction property.

In fact, since $a > b$ in o we know that $|S_{a>b}| > |S_{a<b}| + |S_{a \bowtie b}|$. Similarly $|S_{b>c}| > |S_{b<c}| + |S_{b \bowtie c}|$. Since we are assuming that $a \bowtie c$ then $|S_{a>c}| \leq |S_{c>a}| + |S_{c \bowtie a}|$. We know that a majority of agents says that $a > b$. Let us now assume that no agent says that a is incomparable to b and that no agent says that b is incomparable to c. Let us consider the agents that say $a > b$, then for each such agent she must have one of the following orderings: (1) $a > b > c$; (2) $a > b \wedge c > b$; (3) $a > c > b$; (4) $c > a > b$. We want to prove that that there is at least one agent that says (1) and that there is at least one agent that says either (2) or (4). In fact it is not possible that all the agents which have said $a > b$ have all ordering (1), since that would mean that there is also a majority that says $a > c$, while $a \bowtie c$ by hypothesis. Moreover, it is not possible that all the agents that say $a > b$ vote as (2) or (3) or (4) but none using (1) since this would imply that $c > b$. Thus we can conclude that at least one agent must vote as in (1) and at least one agent votes either (2) or (4).

Now let us consider the agents in the majority that vote $b > c$. Each of them can have one of the following orderings: (i) $a > b > c$; (ii) $b > c \wedge b > a$; (iii)$b > c > a$; (iv) $b > a > c$. As before they cannot all vote (i), otherwise using Majority we would have $a > c$, and not $a \bowtie c$ as assumed. However at least one must vote (i), since (ii), (iii) and (iv) order b above a, while we know there is a majority saying $a > b$. Moreover, it is also not possible that all voters that say $b > c$ say (i) or say (iv) since again that would imply $a > c$. Thus there is at least an agent such that either she says (ii) or she says (iii).

Summarizing, we have an agent that says (1) (or (i)) $a > b > c$, then we have an agent that says (3) or (4). Notice that if she says (3) we can linearize (3) into (4) by adding $c > a$. Thus for the second agent we have $c > a > b$. Finally we have an agent that says either (ii) or (iii). Again (ii) can be linearized into (iii) by adding $c > a$. Thus for the third agent $b > c > a$. This is a linearization which violates the triplewise value-restriction property.

Notice that if we allow the agents to express incomparability between a and b and/or b and c this means that agents that voted (3) or (4) now could vote (2) and that agents that voted (iii) or (iv) now could vote (ii) or give even more incomparability. However this does not prevent the possibility to linearize the orderings as above.

We have therefore generalized Sen's theorem to allow for partial orders.

Informally, we may say that the above result can be used when the profiles are very ordered, and in the right way. In fact, if we take the profile where all agents say that all outcomes are incomparable, then the set of linearizations of such a profile contains all possible profiles, so some will not be good ones.

To the other extreme, we will now give another possibility result which can be applied to profiles with very little ordering among the outcomes. This result assures transitivity of the resulting ordering by a more rough approach: it just forbids the presence of chains in the ordering. That is, for any triple x_1, x_2, x_3 of outcomes, it is never possible that the resulting ordering contains $x_i > x_j > x_k$ where i, j, k is any permutation of $\{1, 2, 3\}$. This is done by restricting the classes of orderings allowed for the agents.

Definition 5.6 (Majority-good) *A profile is Majority-good if, for any triple of outcomes, only one of the following situations can happen (situation α):*

- *the outcomes are all incomparable;*
- *only two of them are ordered, or*
- *there is one of them, which is more preferred than the other two.*

or (situation β),

- *the outcomes are all incomparable;*
- *only two of them are ordered, or*
- *there is one of them, which is more less preferred than the other two.*

Theorem 5.15 *Consider a profile which is Majority-good, and apply the Majority function to such a profile. Then the resulting ordering is transitive.*

Proof:

Let us first consider situation α. We will prove that for every triple a,b, and c it cannot be $a > b > c$, that there are no transitive chains in the ordering. Let us assume that $a > b > c$. Since $a > b$ then:

- (1) $|S_{a>b}| > |S_{a>b}| + |S_{a \bowtie b}|$

Similarly from $b > c$ we have that

- (2) $|S_{b>c}| > |S_{c>b}| + |S_{b\bowtie c}|$

From the fact that we are situation α we have that:

- (3) $|S_{a>b}| \leq |S_{b\bowtie c}|$
- (4) $|S_{b>c}| \leq |S_{b>a}| + |S_{b\bowtie a}|$

The reason for inequality (3) is that all the voters that say $a > b$ cannot order b and c. The reason for inequality (4) is that all the voters that put b above c must either put a below b or incomparable to b.

From the above inequalities we get the following inconsistency:

$$|S_{a>b}| \overset{(3)}{\leq} |S_{b\bowtie c}| \overset{(2)}{<} |S_{b>c}| \overset{(4)}{\leq} |S_{b>a}| + |S_{b\bowtie a}| \overset{(1)}{<} |S_{a>b}|$$

In situation β, (1) and (2) still hold while we have, from the fact that we are situation β:

- (3) $|S_{b>c}| \leq |S_{a\bowtie b}|$
- (4) $|S_{a>b}| \leq |S_{c>b}| + |S_{c\bowtie b}|$

The reason for inequality (3) is that all the voters that say $b > c$ cannot order a and b. The reason for inequality (4) is that all the voters that put a above b must either put b below c or incomparable to c.

From the above inequalities we get the following inconsistency:

$$|S_{b>c}| \overset{(3)}{\leq} |S_{a\bowtie b}| \overset{(1)}{<} |S_{a>b}| \overset{(4)}{\leq} |S_{c>b}| + |S_{c\bowtie b}| \overset{(2)}{\leq} |S_{c>b}|$$

Therefore we have identified three sufficient conditions for Majority to be weakly fair when both profiles and the result are POs.

5.5 Related Work

We would like to start this section considering the approach to partial CP-nets proposed in [BBD$^+$04a]. In such a paper, incompletely specified preferences and indifference are briefly addressed. As in our case the notion of flipping sequence is redefined. In particular, given a variable X with parents $\mathbf{U}$ and two values $x_1, x_2 \in D(X)$, if x_1 and x_2 are equally preferred given assignment to $\mathbf{U}$, namely $\mathbf{u}$, then flipping the value from x_1 to x_2 is said to be indifferent. Notice that this is only part of the definition of indifferent flip we give (see Definition 5.2). As in our case, they introduce incomparable flips. An incomparable flip is one from x_1 to x_2, such that it is not improving nor worsening, given the assignment to the parents of X, $\mathbf{U}$. Such definition is different from ours since we have different notion of improving flip (which considers also changing the value of a non-ranked feature, see Definition 5.2). With their definitions of flips they show that flipping sequence search in non-binary valued CP-nets with partially specified preferences is not in NP. If introducing indifference does not make things better with respect to dominance testing, it doesn't make things worse in outcome optimization. They propose to adapt Forward-sweep [BBHP99] by adding branching on each variable X for which the (already generated) assignment to its parents induces more than one undominated outcome. The complexity is as in what we have proposed in Section 5.3.1, $O(n\alpha)$ where n is the number of features and α is the total number of undominated outcomes.

Preference aggregation has attracted interest from the AI community fairly recently. This has motivated work on both preference representation and aggregation. In particular, voting theory has been considered in light of AI specifications and in terms of combinatorial sets of candidates.

In [Fis74] the author considers relaxing the completeness of the ordering produced by a social welfare function and studies the impact of this change on Arrow's theorem. As noted by the author, Arrow's theorem assumes that every subset of two elements of the set of candidates X, is a potential feasible subset, also called and *environment*. In other words Arrow's assumptions imply that for every pair of candidates, a and b, in the final ordering, either a and b are strictly ordered or they are in a tie. In [Fis74] the possibility that some two-element subset of X is not admitted as a potential environment is taken into consideration. In short, the author allows the existence of pairs of candidates on which the social order is not required. Two elements belonging to such a pair are said to be *incomparable*. This scenario is modeled by having a social welfare functions where the codomain is a partial order, and profiles are allowed to be strict weak orders, which are negatively transitive and assymetric. Notice that this structure is more general than total orders but less general than partial orders, since, for example, it does not allow situations where $a > b$ and c is incomparable to both a and b. Thus, if we compare this setting to ours, the orders allowed in the profiles by [Fis74] are less general than the ones we have, while the final ordering can be any partial order with no specific restrictions. As we do, the author redefines the classical notions of freeness, unanimity and independence by irrelevant alternatives in terms of the new orderings used. Moreover, the notions of dictator and vetoer which are introduced correspond resp. to our notions of dictator and weak dictator.

In [Bar82], an extension of Arrow's theorem which goes in the direction of changing the ordering allowed in the profiles and weakening the ordering in the codomain. In particular social orders can be partial, and agents are allowed to vote using a partial order. However, the set of profiles must also be *regular*, meaning that for any three alternatives, every configuration of their orders must be present in a profile. We can thus summarize saying we are more general in the profiles and less general in the final ordering. It should be noticed that, in contrast with the impossibility result we propose, the restriction imposed on the profiles in [Bar82] is not compatible with total orders and thus the classical Arrow's theorem cannot be obtained as a consequence.

In [Wey84] Arrow's theorem is reconsidered in terms of social welfare functions where profiles are total orders and which have as codomain quasi orderings, that is, binary relations which are reflexive and transitive (preorders, in our terminology). Thus, the setting is different from the one we consider since we allow generic partial orders in the profiles and we require rPOs in the codomain. The paper shows how the relaxation on codomains considered leads to the existence of groups of voters which are characterized by collective dictatorship properties. Such groups are called α-oligarchies and β-oligarchies depending on which property they satisfy. In particular β-oligarchy (which is the weakest notion) is a set of voters, L, such that, for every pair of candidates, a and b, whenever $a >_i b$, for every agent i in L then $a > b$ in the result and if there is at least an agent j in L for which $a >_j b$ then it cannot be that $a < b$ or $a = b$ in the result. According to this definition, it is easy to see that a β-oligarch is weak dictator according to our definition but not vice-versa.

A similar setting is considered in [DFP02]. The paper is written in terms of multi-criteria decision making, rather than aggregating preferences of different agents. This means that what is an agent in our context is a criteria, i.e., a different aspect of the decision problem in [DFP02]. The authors consider two decision paradigms. The one which is closest to what we consider is that called *decision making under uncertainty* since it requires all the attributes to have the same domain. Translated in our context this is equivalent to saying that the set of candidates is the same for all agents. The orderings on the single criteria are assumed to be total. This is in contrast with what we propose wince we allow agents to vote with partial orders. The resulting ordering is instead a quasi-ordering and is thus more general the $rPOs$. However, the result in the paper, which states the existence of an oligarchy, defined in a similar way as in [Wey84], requires several additional assumptions. For example an hypothesis is the discrimination axiom which requires that each agent orders strictly at least one triple of candidates.

The work by Jerome Lang in [Lan02, Lan03] is also related to ours since it considers a scenario where there are multiple agents expressing preferences which must be combined in a unifying result. This work is a very interesting survey in the field since it presents the various logical preference representation approaches and considers them in terms of complexity for answering the usual queries of interest. Secondly, in [Lan02, Lan03], various voting rules are considered and the focus is set on those which are more efficient when the set of candidate is combinatorial. The set of candidates is indicated in [Lan02, Lan03] with X and it is assumed to be finite. As the author points out, there has not been a unique answer to which should be the mathematical model underlying the preferences that an agent has concerning a set of candidates. In [Lan02, Lan03] the author identifies two main frameworks which have been proposed $cardinal\ preference\ structures$ and $relational\ preference\ structures$. The first consists of an evaluation function (i.e. the utility function) which maps every candidate into a numerical value.The second family of preference representations are characterized by a binary relation R on X. R is generally a preorder (thus reflexive and transitive) and the preference structure coincides with an order on the set of candidates, thus it is said to be ordinal. Notice that, as in our case, the order does not need to be complete, which means that some candidates can remain incomparable. As noted on [Lan02, Lan03], an explicit representation of a preference structure consists of the data of all candidates with their utilities (for cardinal preferences) or the whole relation R (for ordinal preferences). These representations have a spatial complexity in $O(|X|)$ for cardinal structures and $O(|X|^2)$ for ordinal structures, respectively, where $|X| = 2^n$, where n is the number of variables. In most of AI problems the set of candidates X has a combinatorial nature, namely, X is described as the set of possible assignments to a a set of features and thus coincides with the Cartesian product of the domains of such features, $X = D_1 \times \cdots \times D_n$, where D_i is the set of possible values for feature v_i. The size of X is exponentially large in the number of features, hence specifying a preference structure explicitly in such a case is unreasonable. In order to overcome this problem several compact preference representation languages have been proposed by the AI community. In this thesis we have considered Soft constraints and conditional CP-nets, in [Lan02, Lan03] the author focuses on representation languages which are logical (and propositional), i.e., each feature v_i is a binary variable: $D_1 = ... = D_n = 0, 1$. Notice that this has never been a requirement in the work we have proposed here. It should be noticed that such languages have the advantage of being very expressive and close to intuition. Moreover they benefit from the well-worked machinery developed (as SAT-solvers). Given a set of preferences expressed using one of those languages the author, in [Lan02, Lan03] considers three main queries that can be asked on the preferences:

- dominance testing (called "comparison", in [Lan02, Lan03]), i.e. given two candidates, x and y decide if $x \geq y$ in the induced ordering;
- optimality testing (called non-dominance in [Lan02, Lan03]) that is determining whether a given outcome is undominated in the ordering;
- CAND_OPT_SAT that is, given a propositional formula ψ, determining if there is an undominated outcome which satisfies ψ.

Notice that we have considered dominance testing and optimality testing but also the problem of finding a undominated outcome or all undominated outcomes. In [Lan02, Lan03], such problems are defined but are not considered since they are computationally hard. The complexity of answering such queries is considered in the context of several different **logical representations**. In other words the preferences of an agent depend on a set of "goals", that is a set of propositional formulas that represent what the agent wants to achieve. Given the set of goals, a preference ordering is induced on the outcomes according in relation to how is the outcome in relation to the goals. For example in *basic propositional representation* there is a single goal G (i.e. a conjunction of propositional formulas). and the utility function u_G generated by G is defined as follows: for each possible assignment (i.e. candidate) x, $u_G(x) = 1$ if x satisfies G, otherwise $u_G(x) = 0$. This representation is very rough, however in [Lan02, Lan03] it is shown that it has attractive complexity properties, e.g. dominance testing is in P. However testing if a given candidate is undominated is Co-NP-complete. In [Lan02, Lan03], several other logical representation as *weighted goals*, where each goal is the utility of a candidate and is computed by first gathering the valuations of the goals it satisfies, the valuations of the goals it violates, and then by aggregating these valuations in a suitable way and *conditional logics* where each goal is associated to a context in which the goal holds. After considering the different preference representations, several voting rules, among the most popular in voting theory, are examined. Such rules are evaluated according to their relevance for combinatorial votes and to their computational complexity. Notice that all the rules considered in [Lan02, Lan03] are social welfare rules, i.e. they provide the winner or a set of winners of an election and not an ordering as we have considered in Section 5.2.2. Aggregation rules of cardinal preferences, in other words numerical aggregators of utilities are briefly considered. In fact, as noted in [Lan02, Lan03] the problem with these methods is, again, the difficulty to elicit a numerical utility function. The first class of rules considered is that of *scoring rules*. The general principle of scoring rules consist in translating the preference relation of voters (expressed by an ordering) into score functions that are consistent with the given ordering. Notice that these rules apply directly when the ordering given is a total order. Among them T*Borda count*, which assigns to a candidate a number equal to the cardinality of the set of candidates minus its rank (or position) in the ordering. and more arbitrary ones as *plurality* and *veto*. Also the famous *Condorcet criterion* is considered, which states that a candidate X is a winner if for any other candidate y there are strictly more agents preferring x to y than agents preferring y to x. A major drawback, pointed out in [Lan02, Lan03] is that a Condorcet winner almost never exists in practical cases, because the number of candidates being much larger than the number of voters, there are generally pairs of candidates up on which all voters are indifferent.

Next, we would like to consider a work taken from the Multi Agent literature which proposes to use, as a preference aggregation method, a criterion taken from economic decision theory. In [ER96] the authors propose to derive consensus in a multiagent scenario using the Clarke Tax mechanism [Cla71]. This mechanism is similar to those we have considered since consensus is reached through the process of voting. The problem tackled in [ER96] is that of a designer of an heterogeneous multiagent environment which must determine the rules by which the agents will interact. The heterogeneity is given by the assumption that each agent acts in a selfish way, trying to maximize her own interest. The scenario described is that of N agents operating in a world which is in an initial state s_0 and are asked to decide what to do next. Although we have never taken this point of view, where outcomes are states of the world, nothing in what we have presented precludes this interpretation. The representation of the preferences of each agent considered in [ER96] is quantitative. More precisely, each agent has a utility function, called $worth$, with which she expresses how much a given state is good with respect to her personal goal. Moreover, a decision procedure maps the preferences declared by the agents into a group of decisions concerning how the world will be transformed. In [ER96] such decision procedure is implemented with the Clarke Tax mechanism. Such technique is a one-shot voting-by-bid method, that differs from the classic sealed-bid mechanism, since instead of collecting the bid of the winning bidder, each agent is fined with a tax which equals the portion of her bid that made a difference to the outcome. For example consider a scenario where there are 3 states, s_1, s_2, and s_3, and 3 agents, a_1, a_2 and a_3. Each agent ranks the states using her worth function, denoted as w_i for agent i. Such utilities can be positive or negative and are combined by summing them. For example, if agent a_1 says $w_1(s_1) = 27$ and $w_1(s_2) = -33$ and $w_1(s_3) = 6$ and agent a_2 says $w_2(s_1) = -36$ and $w_2(s_2) = 12$ and $w_2(s_3) = 24$ and agent a_3 says $w_3(s_1) = -9$ and $w_3(s_2) = -24$ and $w_3(s_3) = -15$, then if we sum the utilities we get -18 for s_1, -45 for s_2 and 15 for s_3, thus state s_3 wins. If agent a_2 had not voted then the result would have been 18 for s_1, -57 for s_2 and -9 for s_3, thus state s_1 would have won. According to the Clarke Tax rule agent a_2 must be fined 27, that is the total utility of s_1 minus the total utility of s_3 in the case he had not voted. From the example we see that this rule is different from the ones we have considered since, not only it gives an ordering on the states, but it also assumes the existence of some monetary-like entity that can be traded among the agents. One of the advantages in terms of social welfare of the Clarke Tax mechanism is non manipulabilty, that is, no agent can get a better result by not declaring the truth. Additionally, the winner according to this method is also a Condorcet winner, that is it beats each other candidate in all pair-wise votes. In terms of the properties we have defined in Section 5.2.2, the Clarke Tax mechanism is monotonic, independent of irrelevant alternatives and also satisfies neutrality. Moreover it respects individual rationality, that is, an agent may only gain utility by taking part to the process, anonymity, i.e., the identity of the voters has no influence on the outcomes and expressiveness, which means that it uses cardinal utilities. This last property is the main difference with our approach since we need not to quantify preferences.

Finally, we consider the work presented in [DW91], which analyzes preference-based nonmonotonic logics in terms of properties taken from social choice theory. Default theories as nonmonotonic theories can be seen as characterized by rational inference, in which the agent selects maximally preferred states of belief [JS89, Sho87]. A nonmonotonic logic is, in its more general form, defined as partial preorder on a set, M, of interpretations which, depending on the base logical language L, may represent truth assignments, models, Kripke structures, or other similar objects [JS89]. In [DW91] each agent models her preferences using a non monotonic logic. Thus, it gives a preorder on the outcomes. The concept of *aggregation policy* is defined as a function that specifies the global preorder corresponding to any given set of individual preorders. A result similar to the Arrow's impossibility theorem [Arr51] is given for this scenario where the agents use preorders and also the result is a preorder. However an additional hypothesis is added, namely *conflict resolution*. This property requires that if a pair is ordered in some way by at least an agent, then it must be ordered also in the global preorder. In other words, all pairs that are comparable for some agent cannot be incomparable in the resulting preorder. In [DW91] it is shown that no aggregation policy can be free, unanimous, independent of irrelevant alternatives, non dictatorial and respect also conflict resolution. It should be noticed that the definition of non dictatorship used in [DW91] corresponds to that of dictator which we have given in Section 5.4.1. This result is thus different from the one we propose here. On one side, they are more general, since they allow preorders while we allow only partial orders. However conflict resolution is a very strong property to require. For example the Pareto semantics defined in 5.3.3 does not respect it. Moreover we use a weaker version of dictatorship while they use a stronger version not adapted to deal with incomparability.

5.6 Future Work

In many situations we may only be interested in the best outcome for all the agents. Such a situation can be described by means of a social choice function. A **social choice** function is a mapping from a profile to one outcome, the optimal outcome. The Muller-Satterthwaite theorem is a generalization of Arrow's theorem on total orders which shows that a dictator is inevitable if we have two agents, three or more outcomes and the social choice function collecting votes is unanimous and monotonic [MS77]. With a partial order, there can be several outcomes which are incomparable and optimal. We can therefore consider a generalization of social choice functions. A **social choice** function f is a mapping from a profile to a non-empty set of outcomes, the optimal outcomes.

We say that a social choice function f is **unanimous** iff when the outcome a is optimal for each agent in Π then $a \in f(\Pi)$, is **monotonic** iff given two profiles Π and Π' in which $a \in f(\Pi)$, and for any b O'_i in Π' ranks a better than b whenever O_i in Π does then $a \in f(\Pi')$, and is **weak dictatorial** iff there is an order O_i in Π and a is optimal in O_i implies $a \in f(\Pi)$. For example, consider the social choices function $f(\Pi)$ which returns the set of optimals for each order O_i in Π. This is unanimous and monotonic. In addition, every agent in the profile is a weak dictator. It is an interesting open question if the Muller-Satterthwaite theorem can be generalized to social choices functions. That is, are weak dictators inevitable if we have two or more agents, three or more outcomes and the social choices function is unanimous and monotonic? A further extension would be to the generalization of the Gibbard-Statterthwaite theorem [Gib73]. That is, are weak dictators inevitable if we have at least two agents and three outcomes, and the social choices function is strategy proof and onto. A social choice function is strategy proof if it is best for each agent to order outcomes as they prefer and not to try to vote tactically.

Recently a line of research [DM03], seems to believe that Majority voting, although not transitive, has some properties which imply that is more "fair" than other approaches. We would like to investigate whether incomparability affects such properties and if the same holds in the context of partial orders.

Chapter 6

Summary and Future Directions

In this chapter we discuss what has been achieved in this thesis and we describe a number of possible future directions we would like to pursue.

6.1 Summary

In this thesis we have considered reasoning with preferences in a number of different AI problems. We have started with a specific area as, temporal reasoning, for which there is specialized reasoning machinery. In particular, we have considered quantitative temporal constraints and we have added preferences in such a context. We have discussed the complexity for solving temporal constraint satisfaction problems with preferences in general, and we have also identified a tractable subclass: simple temporal problems with semi-convex preference functions. For this subclass, we have considered finding optimal solutions with respect to the (fuzzy) maximin criterion and to the Pareto criterion. In the first case, we have proposed two solvers based on two different approaches. One enforces the notion of path consistency adapted to deal also with preferences. We have shown that enforcing such a property is sufficient to solve an STPP and it is a polynomial time algorithm. The second solver, reduces the problem to that of solving several hard constraint problems. We have shown that also this solver is polynomial. We have also designed a solver that finds Pareto optimal solutions. It applies one of the solvers for fuzzy-optimal solutions several times, to problems obtained from the original one by changing some constraints. Also this solver is polynomial. We have implemented and tested the solvers for fuzzy optimals. The tests, performed on randomly generated problems, have show that the second solver is much faster, although it allows to represent less general preferences. We have also considered the problem of actually obtaining the preferences on all the temporal constraints. Practically, it may not be feasible to retrieve all such preferences by a user. We have, thus, applied a machine learning technique, inductive learning, to overcome this problem. In particular, we have considered inducing local preferences (i.e., preferences on the constraints) from global preferences on complete solutions. We have implemented and tested the learning module. The experimental results have been encouraging since they show that the learned constraint problem approximates well the original preference functions defined on the solutions.

Next we have studied the more general class of temporal constraint satisfaction problems under uncertainty. We have tackled the problem of finding optimal schedules that have desirable properties w.r.t. the occurrence of uncertain events. In order to do so, we have taken the notions of controllabilty (strong, dynamic and weak), from the context of Simple Temporal Problems with Uncertainty and we have extended them in order to handle preferences as well. This has lead to the definition of Optimal Strong, Dynamic and Weak Controllability. Such definitions require different levels of robustness with respect to uncontrollable events and optimality with respect to preference in all possible situations. We have given algorithms to test such properties and we also have considered the execution of optimal schedules in the context of dynamic control of plans. The key result in this respect has been to have shown that adding preference does not make the problem harder. This is important, since in many application the tradeoff between quality and time is a key issue.

We have then considered more general scenarios characterized by the coexistence of preference represented in different ways, qualitative (as in CP-nets) and quantitative (as in soft constraints). We have also considered problems in which there are hard constraints as well. We have shown that the two approaches, CP-nets and soft constraints, are complementary and none can substitute the other. However, we have given a mapping algorithm that approximates the ordering induced on the outcomes by a set of conditional preference statements via a soft constraint satisfaction problem. We have shown that it is possible to do so by linearizing the partial ordering of acyclic CP-nets in a consistent way and in a way that respects the fundamental properties of the semantics. The motivation for such an approximation has been the need to overcome the bad complexity for dominance testing in the qualitative framework of CP-nets. Next we have focused on outcome optimization, that is, finding undominated outcomes, in a constrained environment. In particular, we have defined a new semantics for constrained CP-nets and we have reduced the problem of finding the optimals with respect to this semantics to that of solving a hard constraint problem. We have also extended this approach to problems where there are soft constraints as well, taking the position of giving a higher priority to the soft constraints rather than to optimality in the CP-net. Finally, we have tried to use the hard constraint reduction in the context of another semantics defined for CP-nets. In this respect we have given an algorithm able to detect the presence of undominated outcomes even in cyclic CP-nets which is based on solving a hard constraint problem. We have used such a technique in order to find solutions that are both undominated in the CP-net and feasible with respect to a set of hard constraints. This solver, whose complexity is exponential, has the advantage of dealing with cyclic CP-nets as well and of allowing the use of powerful state of the art constraint solvers.

Finally, we have enlarged our target even more, encompassing also multi-agent scenarios. We have considered preference aggregation when the users reason about their own preferences either with CP-nets or soft constraints. In particular, we have defined the multiple agent framework of mCP-nets and we have considered different aggregation criteria. Such criteria have been mutated for the field of voting theory. It hasn't been possible to apply them directly due to the incompleteness of the orderings specified by the agents in our context. In particular the issue of fairness of the aggregation schemes has risen. We reconsidered the main results on this topic from social welfare theory, namely Arrow's impossibility theorem and Sen's possibility theorems, with respect to partial orders. We have shown that the results continue to hold if the properties required are conveniently adapted to partial orders.

6.2 Lessons Learnt

Writing this Ph.D thesis I have learned some lessons.

Given a class of problems with preferences, a good starting point for designing a solver is to decompose the preference problem into several crisp problems and then to merge the results. This can be seen as the interpretation of the famous ancient roman approach "divide et impera" in this specific context. I have learned this from the study of temporal problems especially. In fact, both in the case without and with uncertainty, the key to find the optimal solutions has been to consider the solutions of the hard constraint problems obtained at different preference levels. A similar approach has worked for the dominance testing and outcome optimization in scenarios with quantitative and qualitative preferences, and constraints. In fact, by approximating, we have reduced a hard dominance testing problem into a linear one. Moreover, we have shown how a hard constraint problem is also sufficient when the focus is on finding undominated outcomes.

Semiring-based soft constraint problems have shown to be a good framework both with respect to representation power and also with respect to compatibility with other frameworks. In particular, such a framework has proven its flexibility and adaptability in the case of temporal constraints. In this context it has allowed to extend the existing hard constraint systems by modeling preferences in a natural way. Also in the case of conditional statements soft constraints have allowed both to incorporate the two approaches into a single one, by approximation, and to combine the different reasoning engines in a cooperative way maintaining them separated and exploiting the best of both. I have thus acquired confidence that soft constraints represent a very well balanced tradeoff between generality and expressiveness.

Finally I have learned that preferences are not an option, they are necessary. One thing in common among the different results presented in this thesis is the impressive gain in terms of expressive power obtained by adding preferences. A part of the AI will (and should) continue to be concerned with the size of the problems that are tackled. However, I have learned that the theoretical work and the solving tools that have been developed recently allow researchers to consider sophisticated problems as representing non crisp entities as preferences. I have learned that preferences are a key aspect in the process of making machines closer and more useful to humans.

6.3 Future Directions

There are many future directions which will be interesting to pursue.

The results of this thesis are a strong motivation to continue exploring fields of knowledge representation searching for contexts to which preferences can be added in a convenient way. In this respect, we plan to consider other temporal frameworks, as Conditional Temporal Problems [TVP03], and also to add preferences at the planning level rather than just the scheduling one.

Moreover, we believe that the “soft” connotation of preferences represented using soft constraints is particularly suitable to situations where there is uncertainty. We would like to, similarly to what has been done in the case of fuzzy constraints [DDP93], use soft constraints in order to define objects with a dual nature of preference and uncertainty representation. Notice that also in this case temporal reasoning systems are an ideal starting point, since uncertainty has an intrinsic temporal component. Here we have considered a worst case scenario, with respect to temporal uncertainty where there is no knowledge on which events are more likely than others. However in many situations such a knowledge is available either in terms of probability distributions or in a weaker version as possibility distributions [Zad75]. We plan to reason about problems where there are preferences expressed using soft constraints and temporal uncertainty described using probabilities and possibilities. As in the STPPU framework presented in Chapter 3, we believe that different approaches will come from the different calibrations of the tradeoff between optimization in terms of preferences and robustness to uncertain events. Another interesting direction is stochastic programming in which a problem is divided into several stages and at each stages some decisions must be taken and a part of the uncertain variables are revealed. To the best of our knowledge the role of preferences has not been considered in depth in these scenarios. The soft constraint framework seems suitable also in this case since it allows to represent in a unifying framework purely preferential constraints, temporal constraints with preferences, and constraints over uncertain variables. In a first scenario, the uncertainty may be allowed only on the values of some variables, while in a more complex one uncertainty could be also on the constraints, as on which must be satisfied or on how the preference functions are defined on them.

Another line of research that we foresee as interesting is that of investigating the relation between the soft constraint approach for handling preference and other existing frameworks. Here we have started considering CP-nets, however there are other frameworks as the logic-based ones [DT99, Lan03] which have not yet been considered. Merging soft constraints with such frameworks is an attractive perspective since it would allow qualitative and quantitative representations to coexist and interact in a unique automated reasoning process.

Finally, we plan to investigate further preference aggregation in multiagent scenarios. The study of societies of artificial agents is a topic which is attracting an increasing amount of attention. One of the main goals of AI is to build tools that allow agents to reason in increasingly sophisticated ways. Moreover, when embedded into a distributed system, such agents must interact with others and negotiate common decisions while pursuing their personal goals. We believe that a powerful preference reasoning engine can make a difference, since it allows the representation of the agent's goal in a way that is amenable to negotiation and coordination with other agents, avoiding deadlocks. A considerable gain can be obtained by reconsidering many important results of social welfare theory and social choice theory in light of this new perspective. In fact, many properties which are desirable in human societies, are desirable also in all automated agents scenarios. As examples consider unanimity, monotonicity, neutrality and non-manipulability (also known as strategy-proofness). Moreover, such properties allow one to structure and characterize different aggregation schemes from a different point of view than the usual one of complexity.

To conclude, we point out that, in addition to formalizing new frameworks and approaches we also plan to apply our techniques to some real world problems. The space-application domain, as we have pointed out in Chapter 2 and 3, is very attractive for the work proposed in the temporal context. The combination of a soft constraint solvers and machine learning techniques, as well as reasoning tools able to handle both quantitative and qualitative preferences, are also of interest in an on-line scenario. For example, we are currently developing a tool for searching large web-based catalogs using constraints and preferences to find optimal solutions and machine learning to provide feedback to the system in terms of preferences on proposed products.

References

[ALAP98] L.B. Almeida, T. Langlois, J.D. Amaral, and A. Plankhov. Parameter adaptation in stochastic optimization. In D. Saad, editor, *Online Learning in Neural Networks*. Cambridge University Press, 1998.

[All83] J. Allen. Mantaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.

[AP91] S. K. Andersen and J. Pearl. Probabilistic reasoning in intelligent systems: Networks of plausible inference. *Artif. Intell.*, 48(1):117–124, 1991.

[Arr51] K. Arrow. *Social Choice and Individual Values*. John Wiley and Sons, 1951.

[ASS02] K. J. Arrow, A. K. Sen, and K. Suzumara. *Handbook of Social Choice and Welfare.* North-Holland, Elsevier, 2002.

[Bar82] J.P. Barthelemy. Arrow's theorem: unusual domains and extended codomains. *Matematical Social Sciences*, 3:79–89, 1982.

[BBB01] C. Boutilier, F. Bacchus, and R. I. Brafman. UCP-networks: A directed graphical representation of conditional utilities. In J. S. Breese and D. Koller, editors, *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5*, pages 56–64. Morgan Kaufmann, 2001.

[BBD$^+$04a] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res. (JAIR)*, 21:135–191, 2004.

[BBD$^+$04b] C. Boutilier, R. I. Brafman, Carmel Domshlak, H. H. Hoos, and D. Poole. Preference-based constraint optimization with CP-nets. *Computational Intelligence*, 20(2):137–157, 2004.

[BBHP99] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In K. B. Laskey and H. Prade, editors, *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30-August 1*, pages 71–80. Morgan Kaufmann, 1999.

[BCR02] S. Bastarelli, P. Codognet, and F. Rossi. Abstracting soft constraints: Framework, properties, examples. *Artif. Intell.*, 139(2):175–211, 2002.

[BD02] R. I. Brafman and C. Domshlak. Introducing variable importance Tradeoffs into CP-nets. In A. Darwiche and N. Friedman, editors, *UAI '02, Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta, Canada, August 1-4*, pages 69–76. Morgan Kaufmann, 2002.

[BD04] R. Brafman and Y. Dimopoulos. Extended semantics and optimization algorithms for CP-networks. *Computational Intelligence*, 20(2):218– 245, 2004.

[BDSar] R. I. Brafman, C. Domshlak, and Eyal Shimony. Qualitative decision making in adaptive presentation of structured information. *ACM Transaction on Information Systems*, To appear.

[Beh77] F. Behringer. On optimal decisions under complete ignorance: A new criterion stronger than both Pareto and Maximin. *European Journal of Operation Research*, 1:295–306, 1977.

[BFM^{+}96] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and Valued CSPs: Basic Properties and Comparison. In M. Jampel, E. Freuder, and M. Maher, editors, *Over-Constrained Systems (Selected papers from the Workshop on Over-Constrained Systems at CP'95, reprints and background papers)*, volume 1106, pages 111–150. 1996.

[BG95] F. Bacchus and A. Grove. Graphical models for preference and utility. In *Uncertainty in Artificial Intelligence. Proceedings of the Eleventh Conference (1995)*, pages 3–10, San Francisco, 1995. Morgan Kaufmann Publishers.

[BG96] F. Bacchus and A. Grove. Utility independence in qualitative decision theory. In Luigia Carlucci Aiello, Jon Doyle, and Stuart Shapiro, editors, *KR'96: Principles of Knowledge Representation and Reasoning*, pages 542–552. Morgan Kaufmann, San Francisco, California, 1996.

[BG00] S. Badaloni and M. Giacomin. Flexible temporal constraints. In *8th Information Processing and Management of Uncertainty in knowledge-Based System Conference (IPMU 2000)*, pages 1262–1269, 2000.

[BMR95] S. Bistarelli, U. Montanari, and F. Rossi. Constraint solving over semirings. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95*, pages 624–630, Montreal, Quebec, Canada, August 20-25, 1995. Morgan Kaufmann.

[BMR97] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint solving and optimization. *Journal of the ACM*, 44(2):201–236, mar 1997.

[BRS00] A. Biso, F. Rossi, and A. Sperduti. Experimental results on learning soft constraints. In *KR 2000, Principles of Knowledge Representation and Reasoning Proceedings of the Seventh International Conference*, pages 435–444, Breckenridge, Colorado, USA, April 11-15, 2000. Morgan Kaufmann.

[Cas58] H. N. Castaneda. Review of hallden ”on the logic of better”. *Philosophy and Phenomenological Research*, 19:266, 1958.

[Cla71] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT press, Cambridge, MA, 1990.

[Coo94] M. Cooper. On the complexity of fuzzy constraint satisfaction problems. In H. Fragier, editor, *Problemes de Satsfaction de constraints flexibles application a l'ordonnancement de production*. These n. 1760, Universitè P. Sabatier, Toulouse, 1994.

[Cou77] P. Cousot. Asynchronous iterative methods for solving a fixed point system of monotone equations in a complete lattice. Technical Report R. R. 88, Institut National Polytechnique de Grenoble, 1977.

[DB02] C. Domshlak and R. I. Brafman. CP-nets: Reasoning and consistency testing. In D. Fensel, F. Giunchiglia, D. L. McGuinness, and M. Williams, editors, *Proceedings of the Eight International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25*, pages 121–132. Morgan Kaufmann, 2002.

[DDP93] H. Fargier D. Dubois and H. Prade. The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. In *IEEE International Conference on Fuzzy Systems*, 1993.

[Dec03] R. Dechter. *Constraint processing*. Morgan Kaufmann, 2003.

[DF99] D. Dubois and P. Fortemps. Computing improved optimal solutions to max-min flesible constraint satisfaction problems. *European Journal of Operational Research*, 118:95–126, 1999.

[DFP93] D. Dubois, H. Fargier, and H. Prade. Flexible constraint satisfaction problems with application to scheduling problems. Technical Report Report IRIT/'93-30-R, 1993.

[DFP95] D. Dubois, H. Fargier, and H. Prade. Fuzzy constraints in job shop-scheduling. *Journal of Intelligent Manufacturing*, 6:215–234, 1995.

[DFP96] D. Dubois, H. Fargier, and H. Prade. Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. *Appl. Intell.*, 6(4):287–309, 1996.

[DFP02] D. Dubois, H. Fargier, and P. Perny. On the limitations of ordinal approaches to decision making. In *KR 2002*, pages 133–144, 2002.

[DFP03] D. Dubois, H. Fargier, and H. Prade. Fuzzy scheduling: Modelling flexible constraints vs. coping ith incomplete knowledege. *European Journal of Operational Research*, 147:231–252, 2003.

[DHP03] D. Dubois, A. HadjAli, and H. Prade. Fuzziness and uncertainty in temporal reasoning. *J. UCS*, 9(9):1168–, 2003.

[DM03] P. Dasgupta and E. Maskin. Is Majority rule the best voting method? Technical report, Internal Report Dept. of Economics University of Cambridge, UK, 2003.

[DMP91] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artif. Intell.*, 49(1-3):61–95, 1991.

[DMR$^+$02] R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington. Contingency planning for planetary rovers. In *3rd Intl. Workshop on Planning and Scheduling for Space*, 2002.

[Dom02] C. Domshlak. *Modeling and Reasoning about preferences with CP-nets*. PhD thesis, 2002.

[DOP00] D. Dubois, W. Ostasiewicz, and H. Prade. *Fundamentals of Fuzzy Sets*, chapter Fuzzy sets: History and basic notions, pages 21–124. The Handbooks of Fuzzy Sets Series. Kluwer Academic, Boston, 2000.

[Doy80] J. Doyle. A model for deliberation, action and introspection. Technical Report 419, MIT, 1980.

[Doy96] J. Doyle. Toward rational planning and replanning: rational reason maintance, reasoning economies and qualitative preferences. *In Advanced Planning Technology*, pages 130–135, 1996.

[DP85] D. Dubois and H. Prade. A review of fuzzy set aggregation connectives. *Inf. Sci.*, 36(1-2):85–121, 1985.

[DT99] J. Doyle and R. H. Thomason. Background to qualitative decision theory. *AI Magazine*, 20(2):55–68, 1999.

[DW91] J. Doyle and M. P. Wellman. Impediments to universal preference-based default theories. *Artif. Intell.*, 49(1-3):97–128, 1991.

[DW94] J. Doyle and M. Wellman. Representing preferences as *ceteris paribus* comparatives. In *AAAI Spring Symposium on Decision-Theoretic Planning*, pages 69–75, 1994.

[ER96] E. Ephrati and J. S. Rosenschein. Deriving consensus in multiagent systems. *Artif. Intell.*, 87(1-2):21–74, 1996.

[Fis69] P.C. Fishburn. *Utility Theory for Decision Making*. John Wiley & Sons, 1969.

[Fis74] P. C. Fishburn. Impossibility theorems without the social completeness axiom. *Econometrica*, 42:695–704, 1974.

[FJMS01] J. Frank, A. Jonsson, R. Morris, and D. Smith. Planning and scheduling for fleets of earth observing satellites. In *6th Intl. Symposium on AI, Robotics, and Automation in Space (i-SAIRAS'01)*, 2001.

[FL93] H. Fargier and J. Lang. Uncertainty in constraint satisfaction problems: a probalistic approach. In M. Clarke, R. Kruse, and S. Moral, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty, European Conference, ECSQARU'93, Granada, Spain, November 8-10, Proceedings*, volume 747 of *Lecture Notes in Computer Science*, pages 97–104. Springer, 1993.

[FLP93] H. Fargier, J. Lang, and H. Prade. Selecting preferred solutions in fuzzy constraint satisfaction problems. In *1st European Congress on Fuzzy and Intelligent Technologies (EUFIT)*, 1993.

[FW92] E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *Artif. Intell.*, 58(1-3):21–70, 1992.

[GA89] M. Ghallab and A. M. Alaoui. Managing efficiently temporal relations through indexed spanning trees. In N. S. Sridharan, editor, *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI'89)*, pages 1297–1303, Detroit, MI, USA, August, 1989. Morgan Kaufmann.

[Gea01] J. Geanakoplos. Three brief proofs of Arrow's impossibility theorem. *Economic Theory*, 2001.

[Gib73] A. Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41:587–601, 1973.

[GNT04] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning*. Morgan Kaufmann Publishers, 2004.

[Gol97] C. Goller. A connectionist approach for learning search-control heuristics for automated deduction. Technical report, Technical University Munich,Computer Science, 1997.

[GS95] A. Gerevini and L. K. Schubert. Efficient algorithms for qualitative reasoning about time. *Artif. Intell.*, 74(2):207–248, 1995.

[Hal57] S. Hallden. *On the Logic of 'Better'*. Lund, 1957.

[Han01] S. O. Hansson. Preference logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 4, pages 319–394. Kluwer, 2001.

[Hay94] S. Haykin. *Neural Networks: a comprehensive Foundation*. IEEE Press, 1994.

[JS89] A. L. Brown Jr. and Y. Shoham. New results on semantical nonmonotonic reasoning. In M. Reinfrank, J. de Kleer, M. L. Ginsberg, and E. Sandewall, editors, *Non-Monotonic Reasoning, 2nd International Workshop, Grassau, FRG, June 13-15, Proceedings*, volume 346 of *Lecture Notes in Computer Science*, pages 19–26. Springer, 1989.

[Jun02] U. Junker. Preferences in AI and CP: Symbolic approaches. Technical Report WS-02-13, AAAI Technical Report, 2002.

[Kel78] J. S. Kelly. *Arrow Impossibility Theorems*. Academic Press, New York, 1978.

[Kel87] J. S. Kelly. *Social Choice Theory: An Introduction*. Springer Verlag, Berlin, 1987.

[KM75] A. Kron and V. Milanovic. Preference and choice. *Theory of Deccision*, 6:185–196, 1975.

[KMMR01] L. Khatib, P. H. Morris, R. A. Morris, and F. Rossi. Temporal constraint reasoning with preferences. In B. Nebel, editor, *Proceedings of* *the Seventeenth International Joint Conference on Artificial Intelligence, IJ-CAI 2001, Seattle, Washington, USA, August 4-10*, pages 322–327. Morgan Kaufmann, 2001.

[KMMV03] L. Khatib, P. H. Morris, R. A. Morris, and K. B. Venable. Tractable Pareto optimization of temporal preferences. In G. Gottlob and T. Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15*, pages 1289–1294. Morgan Kaufmann, 2003.

[KR76] R. L. Keeney and H. Raiffa. *Decision with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley & Sons, 1976.

[Lan02] J. Lang. From preference representation to combinatorial vote. In D. Fensel, F. Giunchiglia, D. L. McGuinness, and M. Williams, editors, *Proceedings of the Eight International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25*, pages 277–290. Morgan Kaufmann, 2002.

[Lan03] J. Lang. Logical preference representation and combinatorial vote. *Anals of Mathematics and Artificial Intelligence (Special Issue on the Computational Properties of Multi-Agent Systems)*, To appear, 2003.

[LS88] C. E. Leiserson and J. B. Saxe. A mixed-integer linear programming problem which is efficiently solvable. *J. Algorithms*, 9(1):114–128, 1988.

[LSIR02] N. Layaïda, L. Sabry-Ismaïl, and C. Roisin. Dealing with uncertain durations in synchronized multimedia presentations. *Multimedia Tools Appl.*, 18(3):213–231, 2002.

[Mac77] A. K. Mackworth. Consistency in networks of relations. *Artif. Intell.*, 8(1):99–118, 1977.

[Mac92] A.K. Mackworth. Constraint satisfaction. In S. C. Shapiro, editor, *Encyclopedia of AI (second edition)*, volume 1, pages 285–293. John Wiley & Sons, 1992.

[MD02] M. McGeachie and J. Doyle. Efficient utility functions for ceteris paribus preferences. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pages 279–284. AAAI Press, 2002.

[Mit97] T.M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997.

[MK90] S. Morris and I. Kroo. Aircraft design optimization with dynamic performance constraints. *Journal of Aircraft Design*, Dec 1990.

[MM99] Paul H. Morris and Nicola Muscettola. Managing temporal uncertainty through waypoint controllability. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99*, pages 1253–1258. Morgan Kaufmann, 1999.

[MMK$^+$04] P. Morris, R. Morris, L. Khatib, S. Ramakrishnan, and A. Bachmann. Strategies for global optimization of temporal preferences. In M. Wallace, editor, *Proceeding of the 10th International Conference on Principles and Practice of Constraint Programming (CP-04)*, volume 3258 of *Lecture Notes in Computer Science*, pages 588–603. Springer, 2004.

[MMPS98] N. Muscettola, P. H. Morris, B. Pell, and B. D. Smith. Issues in temporal reasoning for autonomous control systems. In *Agents*, pages 362–368, 1998.

[MMV01] P. H. Morris, N. Muscettola, and T. Vidal. Dynamic control of plans with temporal uncertainty. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001,* *Seattle, Washington, USA, August 4-10*, pages 494–502. Morgan Kaufmann, 2001.

[Mon74] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inf. Sci.*, 7:95–132, 1974.

[MRSV04] P. Meseguer, F. Rossi, T. Schiex, and K.B. Venable, April 2004. Private communication.

[MS77] E. Muller and M.A. Satterthwaite. The equivalence of strong positive association and strategy-proofness. *Economic Theory*, 14:412–418, 1977.

[MS99] P. La Mura and Y. Shoham. Expected utility networks. In K. B. Laskey and H. Prade, editors, *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30-August 1*, pages 366–373. Morgan Kaufmann, 1999.

[NB95] B. Nebel and H. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *J. ACM*, 42(1):43–66, 1995.

[Nis98] T. Nishikawa. Water-resources optimization model for Santa Barbara, California. *Journal of Water Resources Planning and Management*, 124(5):252–263, September 1998.

[OC00] A. Oddi and A. Cesta. Incremental forward checking for the disjunctive temporal problem. In W. Horn, editor, *ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, Germany, August 20-25*, pages 108–112. IOS Press, 2000.

[PP04] B. Peintner and M. E. Pollack. Low-cost addition of preferences to DTPs and TCSPs. In D. L. McGuinness and G. Ferguson, editors, *Proceedings of the Nineteenth National Conference on Artificial Intelligence,* *Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, San Jose, California, USA*, pages 723–728. AAAI Press / The MIT Press, 2004.

[PR94] M. Pelillo and M. Refice. Learning compatibility coefficients for relaxation labeling processes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(9):933–945, 1994.

[RBD$^+$00] K Rajan, D. E. Bernard, G. Dorais, E. B. Gamble, B. Kanefsky, J. Kurien, W. Millar, N. Muscettola, P. Pandurang Nayak, N. F. Rouquette, B. D. Smith, W. Taylor, and Y. W. Tung. Remote Agent: An autonomous control system for the new millennium. In W. Horn, editor, *ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, Germany, August 20-25*, pages 726–730. IOS Press, 2000.

[RHWI99] J. Roslof, I. Harjunkoski, T. Westerlund, and J. Isaksson. A short-term scheduling problem in the paper-converting industry. *Computers chem. Eng.*, 23:971–974, 1999.

[RHZ76] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems,Man, and Cybernetics*, 6(6):173–184, 1976.

[RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[Ros00] F. Rossi. Constraint (logic) Programming: a survey on reserach and applications. In K. R. Apt, A. C. Kakas, E. Monfroy, and F. Rossi, editors, *New Trends in Contraints, Joint ERCIM/Compulog Net Workshop, Paphos, Cyprus, October 25-27, Selected Papers*, volume 1865 of *Lecture Notes in Computer Science*. Springer, 2000.

[RSV$^+$02] F. Rossi, A. Sperduti, K. B. Venable, L. Khatib, P. H. Morris, and R. A. Morris. Learning and solving soft temporal constraints: An experimental study. In P. Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, Proceedings*, volume 2470 of *Lecture Notes in Computer Science*, pages 249–263. Springer, 2002.

[Rut94] Z. Ruttkay. Fuzzy constraint satisfaction. In *Proceedings 1st IEEE Conference on Evolutionary Computing*, pages 542–547, Orlando, 1994.

[Saa98] D. Saad. *Online Learning in Neural Networks*. Cambridge University Press, 1998.

[San99] T. W. Sandholm. Distributed rational decision making. In Gerhard Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 201–258. The MIT Press, Cambridge, MA, USA, 1999.

[Sch92] T. Schiex. Possibilistic Constraint Satisfaction problems or "How to handle soft constraints?". In D. Dubois and M. P. Wellman, editors, *UAI '92: Proceedings of the Eighth Annual Conference on Uncertainty in Artificial Intelligence, July 17-19, Stanford University, Stanford, CA, USA*, pages 268–275. Morgan Kaufmann, 1992.

[SD93] E. Schwalb and R. Dechter. Coping with disjunctions in Temporal Constraint Satisfaction Problems. In *11th National Conference on Artificial Intelligence (AAAI 93)*, pages 127–132. The AAAI Press/The MIT Press, 1993.

[Sen70] A. Sen. *Collective Choice and Social Wellfare*. Holden-Day, 1970.

[SF89] N. Sadeh and M. S. Fox. Preference propagation in temporal/capacity constraint graphs. Technical Report CMU-CS-88-193, 1989.

[SFV95] T. Schiex, H. Fargier, and G. Verfaillie. Valued Constraint Satisfaction Problems: Hard and easy problems. In *Fourteenth International Joint Conference on Artificial Intelligence, (IJCAI 95)*, pages 631–639. Morgan Kaufmann, 1995.

[Sho81] R. E. Shostak. Deciding linear inequalities by computing loop residues. *J. ACM*, 28(4):769–779, 1981.

[Sho87] Y. Shoham. Nonmonotonic logics: Meaning and utility. In John P. McDermott, editor, *10th International Joint Conference on Artificial Intelligence (IJCAI'87)*, pages 388–393. Morgan Kaufmann, 1987.

[Sho97] Y. Shoham. Conditional utility, utility independence, and utility networks. In *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, August 1-3, Brown University, Providence, Rhode Island, USA*, pages 429–436. Morgan Kaufmann, 1997.

[Sim01] H. Simonis. Building industrial applications with Constraint Programming. In H. Comon, C. Marché, and R. Treinen, editors, *Constraints in Computational Logics: Theory and Applications, International Summer School, CCL'99 Gif-sur-Yvette, France, September 5-8, Revised Lectures*, Lecture Notes in Computer Science, pages 271–309. Springer, 2001.

[SK00] K. Stergiou and M. Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artif. Intell.*, 120(1):81–117, 2000.

[Str80] P. Straffin. *Topics in the Theory of Voting*. Birkhauser, 1980.

[Sut92] R. S. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*, pages 171–176, 1992.

[SV98] E. Schwalb and L. Vila. Temporal Constraints: A survey. *Constraints*, 3(2/3):129–149, 1998.

[SW93] R. S. Sutton and S. D. Whitehead. Online learning with random representations. In *ICML*, pages 314–321, 1993.

[SW98] D. Sabin and R. Weigel. Product configuration frameworks - a survey. *IEEE Intelligent Systems*, 13(4):42–49, 1998.

[Tar72] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[TP03] I. Tsamardinos and M. E. Pollack. Efficient solution techniques for disjunctive temporal reasoning problems. *Artif. Intell.*, 151(1-2):43–89, 2003.

[TPR03] I. Tsamardinos, M. E. Pollack., and S. Ramakrishnan. Assessing the probability of legal execution of plans with temporal uncertainty. In *Workshop on Planning Under Uncertainty and Incomplete Information of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003)*, Trento Italy, 2003.

[Tsa02] I. Tsamardinos. A probabilistic approach to robust execution of temporal plans with uncertainty. In I. P. Vlahavas and C. D. Spyropoulos, editors, *Methods and Applications of Artificial Intelligence, Second Hellenic Conference on AI, SETN 2002. Thessaloniki, Greece, April 11-12, Proceedings*, volume 2308 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 2002.

[TVP03] I. Tsamardinos, T. Vidal, and M. E. Pollack. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints*, 8(4):365–388, 2003.

[vB89] P. van Beek. Approximation algorithms for temporal reasoning. In *In Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI 89)*, pages 1291–1296, Detroit MI USA, August 1989. Morgan Kaufmann.

[VF99] T. Vidal and H. Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *J. Exp. Theor. Artif. Intell.*, 11(1):23–45, 1999.

[VG94] L. Vila and L. Godo. On fuzzy temporal constraint networks, 1994.

[VG96] T. Vidal and M. Ghallab. Dealing with uncertain durations in temporal constraint networks dedicated to planning. In W. Wahlster, editor, *12th European Conference on Artificial Intelligence, Budapest, Hungary, August 11-16, Proceedings (ECAI'96)*, pages 48–54. John Wiley and Sons, Chichester, 1996.

[VKB89] M. Vilain, H. Kautz, , and P. Van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In D. S. Weld and J. de Kleer, editors, *Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kauffman, 1989.

[vW63] G. H. von Wright. *The Logic of Preference: An Essay*. Edinburg University Press, 1963.

[vW72] G. H. von Wright. The logic of preference reconsidered. *Theory and Decisions*, 3:140:167, 1972.

[Wal96] M. Wallace. Practical applications of Constraint Programming. *Constraints*, 1(1/2):139–168, 1996.

[Wey84] J. A. Weymark. Arrow's theorem with social quasi-orderings. *Publich Choice*, 42:235–246, 1984.

[Wil04] N. Wilson. Extending CP-nets with stronger conditional preference statements. In D. L. McGuinness and G. Ferguson, editors, *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, San Jose, California, USA*, pages 735–741. AAAI Press / The MIT Press, 2004.

[Zad75] L.A. Zadeh. Calculus of fuzzy restrictions. *In Fuzzy Sets and their Applications Cognitive and Decision Processes*, pages 1–39, 1975.

[ZR96] G. Zlotkin and J. S. Rosenschein. Mechanism design for automated negotiation, and its application to task oriented domains. *Artif. Intell.*, 86(2):195–244, 1996.