Experimental Evaluation of Search and Reasoning Algorithms, 1994.

[11] Debasis Mitra, Padmini Srnivasan, Mark L. Gerard, and Adrian E. Hands. A probabilistic interval constraint problem: fuzzy temporal reasoning. In *World Congress on Computational Intelligence (FUZZ-IEEE'94)*, 1994.

[12] Raul E. Valdes-Perez. The satisfiability of temporal constraint networks. In *Proceedings of AAAI*, 1987.

[13] Peter van Beek. *Ph.D. dissertation: Exact and Approximate Reasoning about Qualitative Temporal Relations*. University of Alberta, Edmonton, Canada, 1990.

5 Conclusion

In this article we have described some advantages of having the capability to generate all possible consistent singleton models (CSM), from a given set of temporal intervals and some incomplete constraints between them. The problem of finding all consistent singleton models is plagued with two sources of expected exponential growth with respect to the number of temporal entities in the data base. First, the problem of finding one consistent model is itself NP-complete. Second, intuition suggests that the number of models should increase exponentially with respect to the number of nodes. The latter apprehension is possibly not appropriate, at least from our preliminary experimental results with randomly generated temporal networks. Number of models actually reduces as constraining level increases with the number of nodes[6]. Ladkin's experiment[4] has also allayed the fear about hardness of the problem of finding one consistent model.

An algorithm is described here which systematically searches and prunes search space to achieve not only one CSM, but all CSM's. The systematic nature of its searching (and pruning) seem to be the key to its efficiency. Significance of this algorithm in temporal reasoning and its capability to find all CSM's in reasonable amount of time is the focus of this article. Theoretical implications involve the capability of studying global consistency problem empirically at much greater depth than has been done so far. Our mechanism also makes us capable to handle non-monotonicity (of a particular type) in temporal reasoning. Other theoretical implications are also discussed in this article.

There are quite a few implications of this algorithm from the point of view of applied research. Some of the most important implications involve temporal data base, critical-domain application areas, prediction/explanation, generating all feasible (interval-based) plans, plan recognition, and image recognition There are many other implications which have not been discussed in this article (for example, parallelizability of the algorithm and associated speed up). Generating all CSM's were considered to be a very hard problem. This is the main reason why no one (as far as we know) has looked into the significance of having all CSM's available. Higher efficiency of our algorithm makes such modeling feasible, and hence studying the significance of such modeling necessary.

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):510–521, 1983.

[2] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 26(2), 1984.

[3] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In *Proccedings of International Joint Conference on Artificial Intelligence*, pages 331–337, 1991.

[4] Peter B. Ladkin and Alexander Reinefield. Effective solution of qualitative interval constraint problems. *Artficial Intelligence*, 57:105–124, 1992.

[5] David Mitchell, Bart Seleman, and Hector Levesque. Hard and easy distributions of sat problems. In *Proceedings of the Tenth National Conference of Am. Asso. of Artificial Intelligence*, 1992.

[6] Debasis Mitra. *Ph.D. dissertation: Efficiency issues in temporal reasoning.* University of Southwestern Louisiana, Lafayette, 1994.

[7] Debasis Mitra and Rasiah Loganantharaj. All feasible plans using temporal reasoning. In *Proceedings of AIAA/NASA conference on intelligent robotics in field, factory, services and space, '94*, 1994.

[8] Debasis Mitra and Rasiah Loganantharaj. Efficent exact algorithm for finding all consistent singleton labelled models. In *Proceedings of IEEE Robotics and Automation conference, Nice, France*, 1991.

[9] Debasis Mitra and Rasiah Loganantharaj. Weak-consistency algorithms for interval-based temporal constraint propagation. In *AAAI'94 workshop note on Spatial and Temporal reasoning*, 1994.

[10] Debasis Mitra, Nabendu Pal, and Rasiah Loganathatraj. Complexity studies of a temporal constraint propagation algorithm: a statistical analysis. *AAAI'94 workshop note on*

example, we may study how such models grow in planning domain with respect to the structural parameters mentioned above.

(c) *Studying real-life problems*: Instead of studying average number of models (or average-case complexity) for randomly generated networks, as suggested in previous paragraphs, one can study networks from classes of real life of temporal consistency-problems, *e.g.*, planning, scheduling and diagnosis.

(d) *Topological distance between local consistency problems and global-consistency problem*: Local consistency problems and global-consistency problem form a hieararchy[9]. An efficient global-consistent algorithm makes it feasible to empirically study the topological distance[6] between global-consistency problem and any other local-consistency problem (*e.g.* 3-consistency), for which efficient polynomial algorithm already exists. Such study will expose a quantitative measure of the usefulness of these local consistency algorithms.

(e) *Predictability of complexity*: In a recent set of experimental studies we are trying to have the capability of predicting run-time of an algorithm based on the *structure* of a problem instance[10]. In addition to predicting time-complexity, now we could predict the number of CSM's a problem instance can have, which is more related to the problem-structure. In order to gain such statistical prediction power, we need to experiment with our algorithm, in line with such experiments for empirically studying time-complexity.

(f) *Non-monotonic temporal reasoning:* The algorithm requires maintenance of all CSM's at each stage of updating the network. This enables one to go back and redo the reasoning from any stage, in case some information on any *old* arc gets changed in future. This non-monotonicity is not feasible unless all CSM's are preserved at each stage.

4 Practical implications of the algorithm

Practical implication of being able to incrementally developing models, and having all such models available in a data base are immense. Some of them are discussed in this section.

(a) *Temporal data base*: A temporal data base will grow incrementally as and when a new temporal assertion comes to the knowledge of the user. So far temporal reasoning community has not paid much attention to this fact. The present algorithm is particularly tuned to such application for efficient incremental growth, making it a very strong candidate for utilization in temporal data bases.

(b) *Critical applications*: In any critical application temporal queries on multiple arcs have to be answered in real-time. Most of the temporal reasoning scheme of present day are not suitable for handling query involving multiple temporal relations in real time. Having all CSM's available in data base makes this feasible.

(c) *Prediction of possible scenarios*: Having all CSM's available again makes one capable of finding out a suitable model for the purpose of prediction, such that the chosen model is consistent with the currently supplied partial model.

(d) *Explanation*: Similar to prediction, developing scenarios involving past temporal assertions can act as explanations for some phenomena which are occurring at present or which have occurred in the past.

(e) *All feasible plans*: A special case of predictive capability of temporal reasoning system is that of generating all feasible plans[7].

(f) *Plan recognition*: A special advantage of having a data base of all feasible plans available, is to recognize a plan based on executed operations up to a '*present*' time.

(g) *Image recognition*: If images are represented in terms of spatial relations (a two dimensional extension of temporal relations) between objects in the image (as number of possible models derived from incomplete information), then image recognition problem becomes a problem of search within models.

(h) *Incorporating uncertainty in temporal reasoning*: Some work has been done recently in studying propagation of fuzzy plausibility values of temporal constraints while running 3-consistency algorithm[11]. In the temporal reasoning system for finding all CSM's, presented in this article, such uncertainty propagation will result in an ordering on all possible models, and possibly eliminating some weak models, thus gaining further pruning power for the sake of efficiency, paying for the overhead incurred in uncertainty propagation.

2 Consistent Singleton Modeling Algorithm

The algorithm described here for finding all consistent temporal models is based on *forward pruning* technique which we have developed sometime back[8]. The formalism used is based on constraint propagation technique, which was first developed for interval based-temporal reasoning by Allen[1]. In this formalism propositional temporal entities (on time-intervals) are represented as nodes in a graph, and relations between them are represented as directed arcs between them. Labels on each arc express possible temporal relations between two end nodes. There are 13 primitive temporal relations feasible between time intervals[1]. Disjunction of them forms an incomplete information between two temporal entities. Three types of such labels are of special interest. (1) Disjunction of all 13 primitive relations signifies lack of any information (or constraint) between two nodes. (2) A null relation as any label in the network signifies contradiction of constraints, or an inconsistent network. (3) Single primitive relation on each of the labels signify a singleton model of the temporal data base.

Our algorithm adds each node one by one to the data base of temporal intervals. Thus the n-th node is added to the database of (n -1) nodes. The data base of (n -1) nodes consists of a set of consistent singleton models (CSM). New constraints from n-th node to all other previously added nodes are used to find n-node consistent singleton models. This is done for all CSM's available in the old data base. New data base consists of n-node CSM's.

At each stage an approximate algorithm is run first as preprocessor to prune primitive relations from labels which do not have any possibility of forming a CSM. After running the preprocessor, the *forward pruning* (FP) algorithm is run. FP-algorithm systematically picks up singletons (primitive relations) from the *next* new arc from an ordered set of new arcs, and updates labels on all remaining new arcs by standard temporal interval–constraint updating technique[2]. If in any updating stage a null relation is generated it is considered to be a contradiction, and the algorithm backtracks over, first the primitive relations on current arc (we call it *sidetrack*), and if it is exhausted, then over the previous arc in the ordered set (of new arcs). If backtracking exhausts all primitive relations from label on the first picked up arc, then the present old-CSM can not find a progeny. A new n-node CSM is found when a primitive relation is picked up final updated label of on the last arc of the set. A new CSM is added for each primitive relation on the last arc to the new data base. This process is repeated for all old CSM's. If no model is found for all old CSM's then the new given constraints on n-th node is not consistent with respect to old data base.

Using this dynamic programming technique, the algorithm improves upon the unsystematic approach of backtracking which have been used for complete interval-based reasoning algorithms developed before. The cost of this improved time efficiency is in the additional space requirement of storing list of singletons at each stage. The data structure design for the implementation becomes quite important in achieving space and time efficiency. The algorithm is already implemented in its preliminary form.

3 Theoretical implications of the algorithm

In this section some theoretical significance of having this algorithm for completely solving the problem of interval-based temporal constraint propagation is discussed.

(a) *Empirical studies of temporal constraint satisfaction problem (TCSP)*: Recently a new approach of empirically studying NP-complete problems[3, 5] is taking shape. Apart from gaining insight into the structure of the problem, such empirical studies also help in developing more efficient algorithms. Our algorithm should provide an efficient means to empirically study the temporal constraint propagation problem, which also happened to be NP-complete problem. This algorithm, being more efficient and powerful (in providing all models), should yield better insight into the problem, with respect to some identified problem structures like average number of constrained arcs, average number of primitive relations on those arcs, or higher moments of their distributions[10].

(b) *Average number of models*: Since our algorithm is particularly suitable for generating all CSM's, it gives an opportunity to study average number of models with respect to different problem structures. Such studies could be very valuable in acquiring domain-specific information. For

Theoretical and practical implications of an algorithm for finding all consistent temporal models

Debasis Mitra
dmitra@ccaix.jsums.edu
Department of Computer Science
1400 Lynch St, P.O. Box 18839
Jackson State University, Jackson MS 39217

Abstract

In this article we have discussed an algorithm for finding all consistent temporal models (with interval-based representation) in an incremental fashion, and significance of having this algorithm. Reasoning with time in interval-based representation is NP-complete problem. This has deterred researchers from studying the significance of having a data base of all consistent models. Advantage of having such models is multi-fold, as we have tried to show here. Our algorithm is efficient enough so that obtaining all consistent models become practicable. This makes such study necessary. Significance elaborated here spreads over both theoretical as well as practical aspects.

1 Introduction

Reasoning with time is an important part of most cognitive activities. Detecting consistency for a given set of temporal constraints is an NP-complete problem, when temporal entities are considered as intervals in time. So far there exist only four algorithms for completely detecting temporal consistency. First one, based on dependency-directed backtracking was proposed by Valdes-Perez[12]. It was not complete, and there is no reported implementation of it.

Second one, forwarded by Peter van Beek[13] is based on first isolating a polynomially solvable (pointizable) subset of the problem, next solving it using polynomial algorithm, then converting the result back to original domain, and on failure, backtracking to find different subset of the original problem. Third algorithm, based on an unsystematic approach of backtrack, was developed and implemented by Ladkin et al[4]. Experimentation with their algorithm has shown that, on an average, the problem is not a very hard one. Almost at the same time, we have developed another algorithm which not only detects consistency, but also finds all possible consistent (temporal) models. Ladkin et al's algorithm finds one consistent model as a side effect. Although, if sufficient time is given, their depth-first search algorithm also can be extended to find all consistent models, our dynamic programming-based algorithm is particularly geared towards efficiently finding all consistent models. Its forward-pruning heuristic prunes the search space in each stage, thus gaining efficiency on average case. Another aspect of our algorithm is that it adds each temporal entity individually, thus developing the data base of the temporal entities incrementally. Other algorithms are not particularly geared towards such incremental development of data base.