Analysis of Approximation Algorithms for Maximal Temporal Constraint Satisfaction Problems

Malek Mouhoub
Department of Math & Computer Science, University of Lethbridge
4401 University Drive,
Lethbridge, AB, Canada, T1K 3M4

Abstract ***This paper presents an experimental study of the different local search techniques to solve Maximal Temporal Constraint Satisfaction Problems. Comparisons were carried out using Min-Conflicts-Random-Walk, Steepest-Descent-Random-Walk and Tabu Search methods. Empirical evidence shows that the Min-Conflicts-Random-Walk method finds almost always solutions of better quality, i.e solutions having smaller number of violated constraints and is faster than the other approximation methods to find solutions of the same quality.***

Keywords: Temporal Reasoning, Constraint Satisfaction Techniques, Heuristic Searching, Planning, Scheduling.

1 Introduction

Many applications such as planning, scheduling and natural language processing involve managing both symbolic and numeric aspect of time. We have developed a temporal model, TemPro, based on the interval algebra[1], to express such applications in terms of qualitative and quantitative temporal constraints[2]. TemPro Translates an application involving temporal constraints into a Temporal Constraint Satisfaction Problem. A Temporal Constraint Satisfaction Problem consists in finding one or more complete assignments of temporal variables (time events) that satisfy all the constraints. Solving temporal constraint satisfaction problems can be handled by complete algorithms based on backtracking and filtering techniques[3][2, 4][5][6].

A maximal temporal constraint satisfaction problem is a temporal constraint satisfaction problem where the goal consists in finding an assignment of temporal variables (time events) that satisfies the maximum number of constraints. This is the case of many real world applications where a complete solution (assignment of variables satisfying all the constraints of the problem) cannot be found. Another example concerns real time applications where a solution should be provided within a given deadline. The problem here consists in providing a solution with a quality (number of satisfied constraints) depending on the time allocated for computation.

Methods for solving maximal temporal constraint satisfaction problems include exact algorithms based on branch-and-bound techniques[7][4] and approximation ones based on local search methods[8]. In this paper we present a study of the different local search techniques to solve Maximal Temporal Constraint Satisfaction Problems. Comparisons were carried out using Min-Conflicts + Random-Walk, Steepest-Descent-Random-Walk and Tabu Search. Empirical evidence shows that the Min-Conflicts-Random-Walk method finds almost always solutions of better quality, i.e solutions having smaller number of violated constraints and is faster than the other approximation methods to find solutions of the same quality.

The rest of the paper is organized as follows : in the next section, we will present through an example, the different components of the model TemPro. The different approximation methods used for our analysis are then presented in section 3. Section 4 is dedicated to the experimental comparison of the different local search techniques. Concluding remarks and possible perspectives of our work are then presented in section 5.

2 Background and Basic Concepts

Consider the following typical temporal reasoning problem :

> *John, Mike and Lisa work for a company in Calgary. It takes John* **20 minutes***, Mike* **25 minutes** *and Lisa* **30 minutes** *to get to work. Every day, John left home* **between 7:20 and 7:26***. Mike arrives at work* **between 7:55 and 8** *and Lisa arrives at work* **between 7:50 and 8***. We also know that John and Mike* **meet** *on their way to work, that Mike arrives to work* **before** *Lisa and that Lisa and John go to work* **at the same time***.*

The above story includes numeric and qualitative information (words in boldface). There are three main events : John, Mike and Lisa are going to work respectively. Some numeric constraints specify the duration of the different events, e.g *15 minutes is the duration of Mike's event.* Other numeric constraints describe the temporal windows in which the different events occur. And finally, symbolic constraints state the relative positions between events e.g *John and Mike meet on their way to work.*

Given these kind of information, our main goal is to represent and reason about such knowledge and answer queries such as : "is the above problem consistent ?", "what are the possible times at which Lisa arrived at work ?", . . ., etc.

To reach this goal, we first translate the temporal problem into a constraint satisfaction problem. Our model TemPro is used for this purpose. Resolution techniques (exact or approximative) are then used to check for the consistency of the problem and to look for possible solutions. In the following, we will see the different objects of our model TemPro.

2.1 Events

In TemPro, temporal objects are called events. Events have a uniform reified representation made up of a proposition and its temporal qualification : $Evt = OCCUR(p, I)$ defined by Allen [1] and denoting the fact that the proposition p occurred over the interval I. For the sake of notation simplicity, an event is used in this paper to denote its temporal qualification. According to our model, in our example we have three main events : J, L and M representing the fact that John, Lisa and Mike are going to work respectively.

2.2 Qualitative Constraints

Qualitative constraints specify the relative temporal position of an event with respect to other events. The qualitative constraint between two events ev_1 and ev_2 can take the following forms : $I_1\ r_1\ r_2 \ldots r_n\ I_2$ where $ev_1 = OCCUR(p, I_1)$, $ev_2 = OCCUR(p, I_2)$ and each of the r_i 's is one of the thirteen Allen primitives : *Precedes*, *during*, *overlaps*, *meets*, *starts*, *finishes* noted respectively P, D, O, M, S and F; their converses $P^{\smile}$, $D^{\smile}$, $O^{\smile}$, $M^{\smile}$, $S^{\smile}$ and $F^{\smile}$; and the equality relation E.

2.3 Quantitative Constraints

Additional information about an event can be expressed, thus restricting its temporal qualification to belong to a given SOPO, i.e, the Sets Of Possible Occurrences where the given event can take place. In our case, a SOPO is restricted to be a finite set of intervals with con-

stant duration. We represent a SOPO by the fourfold $[begintime, endtime, duration, step]$ where *begintime* and *endtime* are respectively the earliest begin time and the latest end time of the corresponding event, *duration* is the duration of the event and *step* defines the distance between the starting time of two adjacent intervals within the SOPO. Thus, if e_i is an event numerically constrained by the SOPO $[inf_i, sup_i, d_i, s_i]$, then the set of possible occurrences of e_i is defined as :
$I = \{occ_j \mid begin(occ_j) = inf_i + k * p_i, end(occ_j) = begin(occ_j) + d_i, end(occ_j) <= sup_i, \; k \in [0, \frac{sup_i - inf_i - d_i}{s_i}] \cap \aleph\}$.

begin and *end* are functions on intervals that return the begin and the end points of a given interval, respectively.

2.4 TemPro based Temporal Constraint Satisfaction (TCSP)

Using the concepts described before, TemPro transforms a temporal problem involving numeric and symbolic information into a temporal constraint satisfaction problem involving:

- a set of variables $\{EV_1, \ldots, EV_n\}$, each defined on a discrete domain D_i standing for the set of possible occurrences in which the corresponding event can hold,
- and a set of binary constraints, each representing a qualitative disjunctive relation between a pair of events and thus restricting the values that the events can simultaneously take.

Using TemPro, our example can be represented by the following temporal constraint network (see figure 1).

A **solution to a TCSP** is an assignment of a value from its domain to every variable, in such a way that every constraint is satisfied. We may want to find:

- just one solution, with no preference as to which one,
- all solutions,

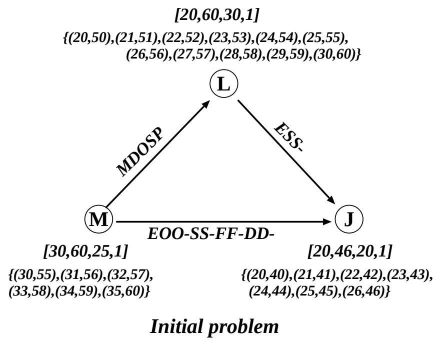

Figure 1: A Temporal Constraint Network.

- an optimal, or at least a good solution, given some objective function defined in terms of some or all of the variables.

Deciding the consistency of a TCSP is NP-hard. Indeed, looking for a complete solution of a temporal problem represented by our model TemPro requires a backtracking algorithm of complexity $O((Max(\frac{sup_i - inf_i - d_i}{s_i}))^N)$ where inf_i, sup_i, d_i and s_i are respectively the earliest start time, latest end time, duration and step of a certain event e_i and N is the number of events.

In the real world, a main challenge for temporal constraint problems is the ability to run on-line, that is to give a solution quickly enough so that the system can use it without missing the deadline of the first events. a solution with a quality (number of satisfied constraints) depending on the time allocated for computation. We are dealing here with a Maximal Temporal Constraint Satisfaction Problem (MTCSP) which is an optimization problem consisting in looking for an assignment that satisfy the maximal number of temporal constraints.

Methods for solving MTCSP include exact algorithms based on branch-and-bound techniques and approximation methods based on local search techniques.

In the following section we will present the different approximation methods.

3 Approximation methods for Maximal Temporal Constraint Satisfaction Problems

We will use the following terms:

State : one possible assignment of all events i.e set of couples $< ev_i, occ_j >$, where e_i is an event and occ_j is a possible interval belonging to the domain of e_i; the number of states is equal to the product of domains sizes.

State or solution quality : the number of constraint violations of the state or the solution.

Neighbor : the state which is obtained from the current state by changing one event value.

local-minimum : the state that is not a solution and the evaluation values of all of its neighbors are larger than or equal to the evaluation value of this state

Strict local-minimum : the state that is not a solution and the quality of all of its neighbors are larger than the evaluation value of this state

The different algorithms that we will consider in the following are based on a common idea known under the notion local search. In local search, an initial configuration (assignment of events) is generated randomly and the algorithm moves from the current configuration to a neighborhood configurations until a complete solution (TCSP problems) or a good solution (MTCSP problems) has been found or the resources available are exhausted.

3.1 Min-Conflict-Random-walk method

The Min-conflicts method chooses randomly any conflicting event, i.e., the event that is involved in any unsatisfied constraint, and then picks a value which minimizes the number of violated constraints (break ties randomly). If no such value exists, it picks randomly one value that does not increase the number of violated constraints (the current value of the event is picked only if all the other values increase the number of violated constraints). The problem of this method is that it is not able to leave local-minimum. In addition, if the algorithm achieves a strict local-minimum it does not perform any move at all and, consequently, it does not terminate. Thus noise strategies should be introduced. Among them, the random-walk strategy that works as follows : for a given conflicting event, the random-walk strategy picks randomly a value with probability p, and apply the MC heuristic with probability 1-p.
In the worst case, the time cost required in each move corresponds to the time needed to determine the value that minimizes the number of violated constraints. This time is of order $O(NMax(\frac{sup_i - inf_i - d_i}{s_i}))$

3.2 Steepest-Descent-Random-Walk

In the Steepest-Descent method, instead of selecting the event in conflict randomly, this algorithm explores the whole neighborhood of the current configuration and selects the best neighbor (neighbor with the best quality). This algorithm can be randomized by using the random-walk strategy in the same manner as for Min-Conflicts to avoid getting stuck at "local optima". The time cost required in each iteration corresponds to the time needed to find the best neighbor and is of order $O(N^2 Max(\frac{sup_i - inf_i - d_i}{s_i}))$ in the worst case.

3.3 Tabu Search

This method is based on the notion of tabu list used to maintains a selective history, composed of previously encountered configurations in order to prevent Tabu from being trapped in short term cycling and allows the search process to go beyond local optima. In each iteration of the algorithm, a couple $< event, v >$ that does not belong to the tabu list and corresponding to the best performance is selected and considered as an assignment of the current configuration. The previous assignment $< event, v' >$ will then replace the oldest move in the tabu list. The time cost required in each iteration is the same as for SDRW, i.e $O(N^2 Max(\frac{sup_i - inf_i - d_i}{s_i}))$ in the worst case.

4 Experimentation

In this section, we present comparative results concerning the different approximation methods we have seen in the previous section i.e MCRW, SDRW and Tabu search. Tests were performed on consistent and inconsistent problems randomly generated.

4.1 Comparison criteria

We use two criteria to compare the above methods. The first one is the quality of the solution, i.e the minimum number of violated constraints of the solution provided by the method. The second criterion is the computing effort needed by an algorithm to find its best solution. This last criterion is measured by the number of moves and the running time in seconds required by each algorithm. The experiments were performed on a SUN SPARC Ultra 5 station. All the procedures are coded in C/C++.

4.2 Parameter tuning for MCRW, SDRW and Tabu search

The performances of the approximation algorithms we use are greatly influenced by some parameters: the size of the tabu list, tl_size, in the case of tabu search; and the random walk probability, p, in the case of MCRW and SDRW. Preleminary tests determine the following ranges of parameter values:

- $10 \leq tl\_size \leq 20$
- $0.05 \leq p \leq 0.15$

Different discrete values between these ranges were further tested and the best values were identified for each class of problems.

4.3 TCSP instances

The tests presented in this section are performed on randomly generated consistent and inconsistent problems, each having 200 variables. The generated problems can be characterized by their tightness, which can be measured, as shown in [9], using the following definition :

> *The tightness of a CSP problem is the fraction of all possible pairs of values from the domain of two variables that are not allowed by the constraint.*

4.4 Results

Table 1 presents tests performed on consistent problems randomly generated. It gives a summary of the best results of MCRW, SDRW and Tabu search for the chosen instances in terms of quality of the solutions. The results correspond to the average time, number of moves and quality of solution provided by each method. To obtain these results, the algorithms were run 100 times on each instance, each run being given a maximum of 100,000 moves in the case of MCRW and 10,000 moves in the case of SDRW and tabu search. The parameter of each algorithm (the size of the tabu list tl_size and the random-walk probability p) is fixed according to the best value found during the parametric study. Note that, as mentioned before in section 3, the cost in time of a move in the case of Tabu Search and SDRW is equal to N times the cost of a move in the case of the MCRW method, where N is the number of variables (events).

From the data of Table 1, we can make the following observations. For under-constrained and middle-constrained problems, the MCRW method always provides the best results. It always found a complete solution within a reasonable amount of time which is not the case of the other two methods. It is also faster than the other two methods to find solutions of the same quality. However for over-constrained problems (see last row of

Tightness of the problem	MCRW				SDRW				Tabu Search			
	p	qual	time	# moves	p	qual	time	# moves	tl_size	qual	time	# moves
0.0002	5	0	0.12	5	15	0	2.67	80	10	0	0.17	4
0.0004	5	0	0.28	18	15	0	4.95	136	10	1	185	5000
0.001	5	0	0.46	28	5	0	8.24	193	10	0	0.6	16
0.002	5	0	0.95	68	5	0	11.22	212	10	2	294	10000
0.0037	5	0	1.74	145	10	0	126	712	10	1	270	10000
0.006	5	0	4	255	10	0	33	336	10	3	286	10000
0.03	15	0	86	3713	10	33	33802	10000	10	12	349	10000
0.044	5	0	73	1633	5	4	9595	10000	15	25	355	10000
0.045	5	0	72	1633	5	4	9614	10000	10	16	376	10000
0.058	5	0	15	433	5	74	12333	10000	15	12	364	10000
0.1	5	0	12	332	5	0	34	225	10	0	112	211
0.14	5	0	8.47	304	5	0	39	243	10	0	112	193
0.35	5	0	181	2009	15	0	66	210	20	68	714	10000
0.44	5	0	137	1291	5	220	8346	10000	10	63	646	10000
0.55	5	0	315	2505	5	0	66	210	10	0	262	190
0.67	5	372	13945	100000	5	0	130	297	10	0	422	224

Table 1: Comparative results of Tabu search, MCRW and SDRW for consistent problems

Tightness of the problem	MCRW				SDRW				Tabu Search			
	p	qual	time	# moves	p	qual	time	# moves	tl_size	qual	time	# moves
0.0002	5	**8**	0.44	32	15	8	4.5	107	10	8	0.28	6
0.001	5	**10**	0.7	53	5	10	10.26	199	10	11	242	5000
0.002	5	**2**	0.68	43	5	3	7.77	183	10	2	194	5000
0.0037	5	**14**	1237	100000	10	14	14.62	238	10	18	230	5000
0.006	5	**20**	5.83	425	10	20	33	336	10	22	377	10000
0.03	15	**21**	190	5406	10	32	3663	10000	10	85	341	10000
0.044	5	**43**	853	25	5	46	4827	10000	15	45	255	10000
0.1	5	**41**	10	318	5	106	41	233	10	91	25	230
0.14	5	**208**	10.14	279	5	208	37	215	10	230	22	197
0.35	5	**141**	259	3015	15	141	439	554	20	141	201	415
0.44	5	**531**	105	271	5	531	82	216	10	531	48	195
0.67	5	**858**	156	315	5	858	98	206	10	924	58	224

Table 2: Comparative results of Tabu search, MCRW and SDRW for non consistent problems

table 1) SDRW and Tabu Search have better performance. We can explain this by the fact that, for under constrained problems the initial configuration is in general of good quality. A complete solution can be obtained in this case by only changing the values of some conflicting variables (case of MCRW) instead of looking for the best neighbor which is much more expensive.

Table 2 presents the tests performed on inconsistent problems randomly generated. For each instance, an exact method based on branch and bound techniques[4] was first performed in order to get the optimal solution (solution with the minimum number of violated constraints). The three algorithms are then run 100 times on each instance, each run being given a maximum of 100,000 moves in the case of MCRW and 10,000 moves in the case of SDRW and tabu search. Quality values in boldface correspond to the best quality (optimal solution) found by the exact method.

From table 2 we can make the same observations as for table 1 i.e the MCRW method is the algorithm of choice if we have to deal with under-constrained or middle-constrained problems. The effort made by SDRW and tabu search methods to look for the best neighbor helps only in the case of over constrained problems.

5 Conclusion and Future Work

In this paper we have presented an experimental analysis of different approximations algorithms to solve Maximal Temporal Constraint Satisfaction Problems, namely : Min-Conflict-Random-Walk, Steepest-Descent-Random-Walk and Tabu Search. These algorithms are based on a common idea known under the notion of local search and where an initial configuration (assignment of events) is generated randomly and the algorithm moves from the current configuration to a neighborhood configurations until a complete solution (TCSP problems) or a good one (MTCSP problems) has been found. Empirical evidence shows that the Min-Conflict-Random-Walk method almost always provides the best results and founds a complete solution within a reasonable amount of time. It is also faster than the other two methods to find solutions of the same quality.

In the near future, we intend to integrate the approximation techniques into a planning system that handles temporal constraints. Indeed, the planning system will be able in this case to deal in real time, that is to provide a plan violating the minimum number of time constraints and without missing a given deadline. Approximation methods are also useful when there is no solution (complete plan) satisfying all temporal constraints.

References

[1] J.F. Allen. Maintaining knowledge about temporal intervals. *CACM*, 26(11):832–843, 1983.

[2] Malek Mouhoub, F. Charpillet, and J.P. Haton. Experimental analysis of numeric and symbolic constraint satisfaction techniques for temporal reasoning. *Constraints: An International Journal*, 2:151–164, 1998.

[3] I. Meiri. Combining qualitative and quantitative constraints in temporal reasoning. *Artificial Intelligence*, 87:343–385, 1996.

[4] M. Mouhoub. Reasoning about Numeric and Symbolic Time Information. The Twelfth IEEE International Conference on Tools with Artificial Intelligence(ICTAI'2000) . In *ICTAI'2000*, pages 164–172, Vancouver, 2000. IEEE Computer Society.

[5] R.M. Haralick and G.L. Elliott. Increasing tree search efficiency for Constraint Satisfaction Problems. *Artificial Intelligence*, 14:263–313, 1980.

[6] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[7] R. J. Wallace. Partial constraint satisfaction. *Lecture Notes in Computer Science*, 923:121–138, 1995.

[8] R. J. Wallace. Analysis of heuristic methods for partial constraint satisfaction problems. In *CP'96, LNCS 1118*, pages 308–322, Cambridge, MA, 1996.

[9] D. Sabin and E. C. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *Proc. 11th ECAI*, pages 125–129, Amsterdam, Holland, 1994.