2014

Risk minimization of disjunctive temporal problem with uncertainty

Hoong Chuin LAU
Singapore Management University, hclau@smu.edu.sg

Tuan Anh HOANG
Singapore Management University, tahoang.2011@smu.edu.sg

Risk Minimization of Disjunctive Temporal Problem with Uncertainty

Hoong Chuin Lau and Tuan Anh Hoang

School of Information Systems, Singapore Management University

Abstract. The Disjunctive Temporal Problem with Uncertainty (DTPU) is a fundamental problem that expresses temporal reasoning with both disjunctive constraints and contingency. A recent work (Peintner *et al*, 2007) develops a complete algorithm for determining Strong Controllability of a DTPU. Such a notion that guarantees 100% confidence of execution may be too conservative in practice. In this paper, following the idea of (Tsamardinos 2002), we are interested to find a schedule that minimizes the risk (i.e. probability of failure) of executing a DTPU. We present a problem decomposition scheme that enables us to compute the probability of failure efficiently, followed by a hill-climbing local search to search among feasible solutions. We show experimentally that our approach effectively produces solutions which are near-optimal.

1 Introduction

Expressive and efficient temporal reasoning is a significant task in planning and scheduling. A typical assumption is that all time points or temporal events such as starting and ending actions are under the complete control of the execution agent, and this can be modeled by the simplest and widely used Simple Temporal Network. The Simple Temporal Problem (STP) and the Disjunctive Temporal Problem (DTP) (that allows for disjunctive constraints) are concerned with checking temporal consistency of a given temporal network. While STP is polynomial-time solvable [1], solving DTP is known to be NP-hard [7] in general.

In real world practice, there are often needs to perform inference on events in the presence of exogenous factors (often referred as "Nature") which cannot be directly controlled by the agent, whose realization can only be observed. Due to these so-called "observable" time points, uncertainty is introduced into the above problems. Correspondingly, we have the Simple Temporal Problem with Uncertainty (STPU) and Disjunctive Temporal Problem with Uncertainty (DTPU), and the concept of temporal consistency is extended by varying notions of *controllability*, such as Strong, Weak and Dynamic Controllability, depending on how "observable" events affect controllable events.

Most works on temporal constraint problems with uncertainty focus on execution with 100% confidence. From the practical standpoint, this may lead to wasteful resource allocation or even infeasible networks. A interesting approach to deal with this is to take a probabilistic perspective. In [9], the Probabilistic

Simple Temporal Problem (PSTP) was introduced by representing each uncontrollable event as a probability density function (PDF), where the goal was to find a schedule that maximizes the probability of execution under strong controllability (i.e. a one-size-fits-all schedule that maximizes the probability of successful execution against all possible realizations of uncertain events. [11] then presented heuristic techniques for determining the probability of executing a dynamically controllable strategy for PSTP. The concept of Robust Controllability was proposed in [2] to ensure the dynamic controllability of STPU within a specified degree of risk.

The Disjunctive Temporal Problem with Uncertainty (DTPU) was first defined in [12] for the purpose of modeling and solving planning and scheduling problems that feature both disjunctive constraints and contingency. [5] investigated the semantics of DTPU constraints and proposed a way to check if Strong Controllability holds. That work focused on checking if there exists a solution that ensures all constraints will be satisfied regardless of Nature's realization of uncontrollable events. Since dealing with execution under 100% confidence might be too conservative, a logical extension is in finding solutions that minimizes the risk of failure instead.

This paper is concerned with executing a DTPU from a risk minimization perspective as presented in [9]. Given a DTPU instance where the contingent constraints are modeled as random variables of known distributions, we are interested to find a solution that minimizes execution risk - or put in more positively, maximizes the probability of successful execution - that all temporal constraints will be satisfied. We propose a method for decomposing a DTPU into components so that the probability of success can be efficiently computed. Following that, we propose a computationally efficient hill-climbing local search that enables near-optimal solutions to be obtained.

2 Background and Literature Review

In this section, we briefly review the preliminaries and recent literature on the temporal constraint problem.

2.1 Temporal Constraint Problems

An STP [1] is defined as a pair $< V, S >$, where V is a set of temporal variables representing temporal events or time points, and S is a set of constraints between points, each taking the form $v_j - v_i \in [a, b]$, where $v_i, v_j \in V$ and a and b are some constants. Since STP contains only binary constraints, it can be represented by a weighted graph whose variables are represented as nodes and each directed edge (v_i, v_j) is labeled by an interval $[a, b]$. An STP is consistent iff there exists at least one solution (an assignment to all temporal variables) such that all constraints in S are satisfied, which can be determined in polynomial time [1].

A DTP [7] extends an STP by admitting disjunctive constraints. A DTP constraint consists of a disjunction of STP constraints of the form: $v_{j1} - v_{i1} \in$

$[a_{i1j1}, b_{i1j1}] \bigvee v_{j2} - v_{i2} \in [a_{i2j2}, b_{i2j2}] ... \bigvee v_{jk} - v_{ik} \in [a_{ikjk}, b_{ikjk}]$. A DTP instance is consistent iff it contains at least a consistent component STP obtained by selecting one disjunct from each constraint. Efficient solvers for this NP-hard problem have been developed (e.g. Epilitis in [10]) to search through the meta space of component STPs. A number of pruning techniques are embedded in DTP solvers to reduce the search space, including conflict-directed backjumping, removal of subsumed variables, semantic branching, and no-good recording. [3] applied local search to DTP to generate solutions with minimal constraint violation and the operation is within the total assignment space of the underlying CSP rather than the partial assignment space of the related meta-CSP. [8] showed how their end-point ordering model can be used to express the qualitative interval algebra in a quantitative constraint solver with finite domains, and how to convert an interval algebra network into an equivalent non-binary CSP with finite integer domains. They observed that the relative positions of interval endpoints in an interval algebra network can be used to determine consistency.

2.2 Temporal Constraint Problems with Uncertainty

To model uncertainty, two classes of temporal variables are defined: executable V_e controlled by the execution agent, and uncontrollable V_u controlled by Nature. In addition to the deterministic constraints S defined above, S_e and S_c respectively denote the sets of executable and contingent constraints. S_e model execution requirements in response to uncertain events (e.g.,"Activity can only be started (controllable) at least 30 minutes after the rain stops (uncontrollable)"). Hence, each executable constraint takes the form $y - x \in [a, b]$, where $y \in V_e$ and $x \in V_u$. S_c model the temporal behavior of an uncontrollable event (e.g.,"The dinner will be ready (uncontrollable) between 20 and 30 minutes after cooking starts (controllable)"). They are used usually to model durational uncertainty whose values are controlled by Nature and can only be observed by the agent. In our paper, each contingent constraint takes the form $x - y = \tilde{d}$, where $y \in V_e, x \in V_u$ and $\tilde{d}$ is a random variable with a certain probability distribution.

Based on different conditions of guaranteeing all constraints will be satisfied, three standard levels of controllability have been defined in the literature: Strong Controllability (i.e. existence of a universal solution), Weak Controllability (i.e. existence of a solution for each scenario), and Dynamic Controllability (i.e. existence of a solution that can always be built incrementally based on outcomes of contingent edges in the past). Tractable algorithms for checking them were provided in [13]. [4] proposed a pseudo-polynomial algorithm to handle dynamic controllability of STPUs based on constraint satisfaction. In [14], techniques were proposed to optimize the bounds on durations of contingent edges such that the resulting STPU is dynamic controllable.

Another line of work to deal with temporal uncertainty takes on a probabilistic context, i.e. allowing for probabilistic violation of constraints. [2] modeled contingent edges as random variables. Under some assumptions on the relationship between controllable and uncontrollable points, they provided an efficient polynomial-time approach based on second-order cone programming to check

if an STPU is Robust Controllable, i.e., can be executed dynamically within a given level of risk. [9] dealt with the problem of maximizing the probability of successful execution of PSTP (Probabilistic STPU) that models STPUs with continuous conditional PDF for each uncontrollable event. He formulated and solved the problem as a non-linear constrained optimization problem, and in general, the approach does not guarantee finding a globally optimal solution. [11] proposed heuristic techniques for approximating upper and lower bounds on the probability of executing a dynamically controllable strategy for PSTP.

3 Problem Definition

Definition 1. A Disjunctive Temporal Problem with Uncertainty is a tuple $< V_e, V_u, C, C_u, P >$ where

- $V_e = \{y_1, ..., y_n\}, V_u = \{x_1, ..., x_m\}$ respectively denote the sets of executable (or controllable) and observable (or uncontrollable) variables (or temporal events, time points) taking real values;
- C is the set of controllable temporal constraints on V_e and V_u. In the following, we describe five different types of constraints $\{S, S_e, D, D_e, D_{mix}\}$. (For brevity, the reader may skip this and come back for more details later.)
  - S: set of standard STP constraints between controllable points;
  - S_e: set of executable STPU constraints, each specifying the temporal requirement between an uncontrollable and a controllable point;
  - D: set of DTP constraints, each being a disjunction of two or more STP constraints S;
  - D_e : set of disjunctions of two or more executable STPU constraints;
  - D_{mix}: set of disjunctions of a mix of STP and executable STPU constraints.
- C_u is the set of contingent constraints, one for each uncontrollable time point in V_u, representing duration uncertainty determined by Nature. Each contingent constraint in a DTPU links an uncontrollable time point x_j to a unique parent executable time point, denoted $pa(x_j)$, and is expressed as a disjunction of K_j random variables taking nonnegative values. Hence, a contingent constraint can be expressed algebraically as $x_j - pa(x_j) = d_j^1 \vee ... \vee d_j^{K_j}$. In plain terms, this means that the point x_j takes a value that Nature would determine, which is a realization of one of the random variables in $\{d_j^1, ..., d_j^{K_j}\}$; and this value gives the duration of the event whose start time is given by the executable time point $pa(x_j)$. For consistency of notations with controllable constraints, we write $C_u = \{S_c, D_c\}$, where S_c and D_c denote the set of contingent STPU and DTPU constraints respectively.
- P is a set of continuous PDFs, one for each random variable $\tilde{d}$ providing the probability distribution over time duration of the uncontrollable event x_j occurring after the executable point $pa(x_j)$ is started. The probability distributions are assumed to be independent of each other.

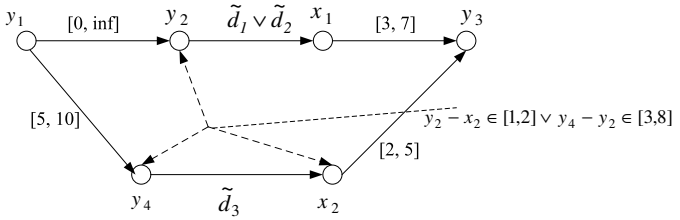

Fig. 1. DTPU Example.

Figure 1 gives an example of DTPU with four executable variables y_1 to y_4 (representing the start times of Tasks 1 to 4, where Task 1 is the dummy Time Reference point with no duration uncertainty), plus two observable variables x_1 and x_2 (representing the uncertain durations, i.e. uncontrollable events of Tasks 2 and 4's end times respectively). The constraint between executable events, such as that between y_1 and y_4, means that Task 4 is to start between 5 to 10 time units after y_1. The contingent constraint between y_4 and x_2 means the duration of Task 4 is controlled by Nature whose value follows a certain probability distribution. According to Definition 1, these constraints are classified as follows.

$$S : y_2 - y_1 \in [0, \infty] \;\; y_4 - y_1 \in [5, 10]$$
$$S_e : y_3 - x_1 \in [3, 7] \;\; y_3 - x_2 \in [2, 5]$$
$$D_{mix} : y_2 - x_2 \in [1, 2] \vee y_4 - y_2 \in [3, 8]$$
$$S_c : x_2 - y_4 = \tilde{d}_3$$
$$D_c : x_1 - y_2 = \tilde{d}_1 \vee x_1 - y_2 = \tilde{d}_2$$

As a side note, we like to comment that in practice, the number of disjuncts for disjunctive constraints in C and C_u is typically very small. For C, disjunctions are used to model scheduling constraints such as either x occurs before y or y before x and hence the number of disjuncts is 2. For C_u, disjunctions can be used to model rare events where a duration can either assume a usual (albeit uncertain) length or occasionally take an unusually long time, and hence the number of disjunct is again equal to 2.

In this paper, we use probability distributions to model contingent durations. This frees us from the requirement for bounded intervals, which we believe is less realistic, since it is hard to provide bounds for the nature-controlled temporal events and there is always a non-zero probability that exogenous factors may cause an event to occur outside the bounds.

Given a DTPU, a *schedule* (or solution) is an assignment to all executable time points in V_e.

Definition 2. The Risk-Minimal DTP problem (RDTP) is defined as: given a DTPU instance, find a solution such that the probability that all constraints are satisfied is maximized.

This problem belongs to the class of "inequality constrained discrete MIN-MAX problem" [6] in optimization theory, which is known to be notoriously hard to solve computationally.

4 Solution Approach

First, we show how to decompose the original DTPU into different sub-problems. We then discuss how the success probability of each sub-problem can be computed. This leads us to the next section where we discuss our proposed local search scheme that makes use this computation.

4.1 Problem Decomposition

Note that for each disjunctive contingency constraint in D_c, since Nature will dictate which disjunct will ultimately be chosen and which value for that single disjunct will be realized during execution, the planner need to take into consideration ALL combinations of disjuncts that Nature may choose. On the other hand, since the planner has the right to decide on the values of the executable variables, an assignment that satisfies ANY of the disjunct of a disjunctive constraint in$\{D, D_e, D_{mix}\}$ is deemed to have satisfied that constraint. For this reason, one can imagine decomposing the DTPU into component DTPUs which can be solved independently and in parallel. Hence, we propose to split the constraint set into two categories $\{S, S_e, S_c, D, D_e, D_{mix}\}$ (termed the Canonical set) and $\{D_c\}$.

Intuitively, our idea is to decompose the problem as an MAX/MIN tree where we seek the maxima over the branches of the Canonical constraints, each of which is in turn computed as the minima over the branches of the disjunctive contingent constraint set D_c (since Nature can pick any of these branches). Details are given as follows.

Definition 3. Component DTPU (C-DTPU). Given a DTPU, a C-DTPU is a DTPU with the set of temporal variables $V_e \cup V_u$, the set of disjunctive contingent constraints D_c, and a set of STPU constraints comprising the non-disjunctive constraints $\{S, S_e, S_c\}$ plus constraints obtained by selecting one disjunct from each constraint of the set $\{D, D_e, D_{mix}\}$. (In other words, a C-DTPU is a DTPU defined by $< V_e, V_u, S_{can}, D_c, P >$, where S_{can} is an STPU comprising a unique disjunct combination derived from the canonical set of the original DTPU. To avoid ambiguity, we also term this STPU as a *canonical* STPU.)

Given a C-DTPU, in order to handle the disjunctive contingent constraints D_c, we further decompose into Component STPUs (as the MIN branches):

Definition 4. Component STPU (C-STPU). Given a C-DTPU $< V_e, V_u, S_{Can}, D_c, P >$, a C-STPU is an STPU with the set of temporal variables $V_e \cup V_u$, the entire set of S_{Can} (defined above), a combination of contingent STPU constraints obtained by selecting one disjunct from each constraint of the set D_c, i.e. it is an STPU defined by $< V_e, V_u, S_{can}, S_{D_c}, P >$, where S_{D_c} is

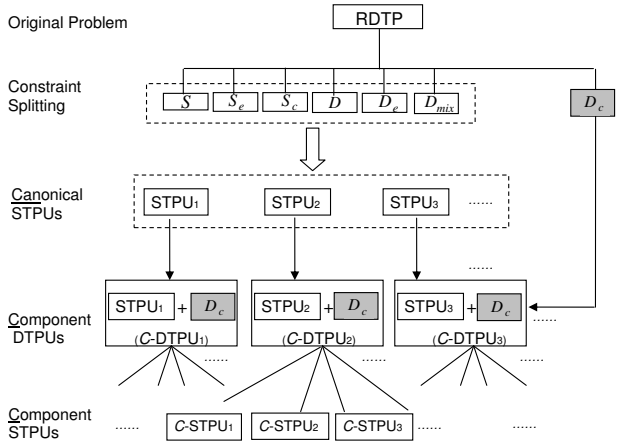

Fig. 2. DTPU Decomposition.

an STPU comprising a unique contingent disjunct combination from D_c of the original DTPU.

A pictorial representation of the decomposition is given in Figure 2.

4.2 Computing Probability of Success

Consider a DTPU T composed of $\varphi(T)$ number of C-DTPUs T_l ($l = 1, ..., \varphi(T)$). For each T_l, we are interested to find an optimal schedule (denoted as s_l^*) that maximizes its success probability (i.e. the probability that all temporal constraints will be satisfied during execution if controllable points are executed following the schedule). Let $P(T_l|s_l)$ represent the success probability of T_l when committing to schedule s_l (i.e. an assignment to all executable points of T_l), an optimal schedule can then be defined as, $s^* = arg\ max_s max_l(P(T_l|s))$.

Let $\theta(T_l)$ denote the total number of decomposed C-STPUs of T_l; T_{lk} denote the k^{th} C-STPU, and $P(T_{lk}|s_{lk})$ denote the success probability of T_{lk} with schedule s_{lk} ($l = 1, ..., \varphi(T)$ and $k = 1, ..., \theta(T_l)$). For simplicity, we will henceforth rewrite s_l and s_{lk} as s where there is no ambiguity.

As noted above, since Nature will select the disjunct combination of constraints D_c to realize, we adopt a worst-case approach in determining an optimal schedule s^* for a given T_l. Given a schedule s, the worst-case probability for T_l to be successfully executed when committing to s is equivalently the minimum success probability of all its C-STPUs, i.e. $P(T_l|s) = min_k P(T_{lk}|s)$, where $k = 1, ..., \theta(T_l)$. Therefore, the optimization problem of determining an optimal schedule s^* that maximizes the success probability of C-DTPU T_l is:

$$P(T_l|s^*) = max_s P(T_l|s) = max_s min_k P(T_{lk}|s) \tag{1}$$

subject to the standard STP constraints.

We now show how we compute the success probability for a single C-DTPU. For simplicity, we will drop the subscript l and write the C-DTPU as T and the C-STPU T_{lk} as T_k. The core computational part of our algorithm is to compute the success probability $P(T|s)$ of C-DTPU T when committing to schedule s. We first consider the success probability of C-STPU T_k and adopt the idea presented in (Tsamardinos, 2002) that deals with executing a PSTP so that the success probability is maximized. The difference is that in our work, the probability distribution is assumed with respect to a contingent constraint rather than an observable time point.

Recall that for a C-STPU T_k, three types of temporal constraints are involved: STP constraints, executable STPU constraints of the form $y_i - x_j \leq b_{x_j y_i}$ or $x_j - y_i \leq b_{y_i x_j}$ [1], and contingent STPU constraints of the form $x_j - pa(x_j) = d_j$. STP constraints are standard constraints that must be satisfied as hard constraints. Contingent constraints are instantiations of random variables determined by Nature. Thus, given a schedule s, the success probability is equal to the probability that the executable STPU constraints are satisfied given the probability distributions associated with the random variables.

Consider first an arbitrary observable variable x and its parent $pa(x)$. The success probability of C-STPU T_k associated with this x when committing to schedule s can be computed as the probability that all the executable STPU constraints linked to x are satisfied:

$$P(T_k|s) = P((\bigwedge_i y_i - x \leq b_{xy_i}) \wedge (\bigwedge_i x - y_i \leq b_{y_i x}) \tag{2}$$

Since $x - pa(x) = \tilde{d}$, we can rewrite the above as:

$$P(T_k|s) = P(L_s(d) \leq d \leq U_s(d)|s) \tag{3}$$

where $L_s(d) = Max_i(y_i - b_{xy_i}) - pa(x)$ is the lower bound for the contingent constraint to be satisfied for a given solution s, while $U_s(d) = Min_i(y_i + b_{y_i x}) - pa(x)$ is the upper bound.

Extending to the general case when multiple observable variables are involved and letting $\tilde{d_j}$ denote the random variable associated with the j^{th} contingent constraint, the above probability equation becomes:

$$P(T_k|s) = P(\bigwedge_j L_s(\tilde{d_j}) \leq \tilde{d_j} \leq U_s(\tilde{d_j})|s) \tag{4}$$

Since all executable variables have been fixed with a given schedule s, the values of $L_s(\tilde{d_j})$ and $U_s(\tilde{d_j})|s)$ are fixed for each contingent constraint j and not dependent on any other random variables. Also, the probability distribution for different contingent durations represented with random variables $\tilde{d_j}$ are assumed to be independent of each other. Hence, we have,

$$P(T_k|s) = \prod_j P(L_s(\tilde{d_j}) \leq \tilde{d_j} \leq U_s(\tilde{d_j})|s) \tag{5}$$

[1] Note that the original form of the constraint $y_i - x_j \in [a_{x_j y_i}, b_{x_j y_i}]$ has been rewritten by the two inequalities above for notational simplicity.

Thus, the success probability $P(T|s)$ of C-DTPU T when committing to schedule s can be represented as:

$$P(T|s) = min_k \prod_j P(L_s(\tilde{d_{kj}}) \leq \tilde{d_{kj}} \leq U_s(\tilde{d_{kj}})|s) \quad (6)$$

where $k = 1, ..., \theta(T)$ and $\tilde{d_{kj}}$ is the j^{th} contingent constraint of C-STPU T_k.

5 Local Search

In this section, we discuss our proposed local search algorithm to solve RDTP. Our search process is conducted on multiple C-DTPUs simultaneously and independently, and the one the gives the best solution will be returned. Note that such computations can occur in parallel without affecting the correctness of the solution. In the following, we therefore discuss our proposed local search algorithm to solve the problem associated with a single C-DTPU.

5.1 Search Space

A C-DTPU is made up of an STPU plus the D_c constraints. We perform search in the space of $feasible$ solutions which comprise schedules that do not violate any of the STP constraints in S. Note that for a C-DTPU, every executable constraint is a simple (i.e. non-disjunctive) constraint. This means, for each executable variable, we can calculate its minimal domain using the algorithm in [1] in polynomial time, which is also can be used to find a feasible solution the minimal domain.

5.2 Neighborhood Generation

The initial solution denoted as $s^0 = \{s_1^0, ...s_n^0\}$ is randomly selected from the feasible space, which is an assignment of all n executable variables. From equation (6) given in the previous section, we see that the overall probability of success of a solution s is a function of the probabilities that the individual random variables (representing the contingent constraints) will be realized within their respective interval bounds (lower and upper bounds) when committing to that schedule. Hence, the intuition for our proposed local search neighborhood is to adjust the value of one executable variable of s at a time in such a way that these probabilities might be increased while maintaining schedule feasibility. The following provides the neighborhood construction in detail.

We first make some observations any random variable d, before discussing the neighbor structure.

- For any interval $I \subseteq \mathcal{R}$, $P(d \in I)$ is not necessarily proportional to the length of I, and its value may vary, i.e., increase or decrease, when I is slided to the left or right.
- For any two intervals I_1 and I_2, we have $P(d \in I_1) \geq P(d \in I_2)$ if $I_2 \subseteq I_1$.

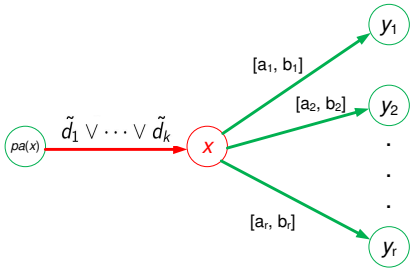

Fig. 3. An uncontrollable variable and its relatives in a C-DTPU

Therefore, at each local search iteration, the intuition is to explore candidate neighbors that will either: (a) slide the interval $[L_s(\tilde{d}_j), U_s(\tilde{d}_j)]$ to one where the area under the probability density function of $\tilde{d}_j$ over that interval has an equal or higher value; or (b) expand the interval $[L_s(\tilde{d}_j), U_s(\tilde{d}_j)]$ as much as possible. In both cases, care must be exercised to ensure the resulting schedule is still feasible. This hill-climbing process will lead us to iteratively improve the overall value of the success probability.

Assume that $s = \{s_1, ...s_n\}$ is a feasible schedule a certain C-DTPU in question. For each $y_i \in V_e$, denote:

$$\delta^-(y_i) = max\{\delta > 0 : \ s' = (s_1, ..., y_i - \delta, ..., s_n) \text{ is a feasible solution}\}$$
$$\delta^+(y_i) = max\{\delta > 0 : \ s' = (s_1, ..., y_i + \delta, ..., s_n) \text{ is a feasible solution}\}$$

That is, $\delta^-(y_i)$ and $\delta^+(y_i)$ provide the limits, when all other executable variables are fixed, on how much the value of y_i can be decreased or increased such that the resulting schedule is still feasible.

Since we must deal disjunctive contingent constraints (each of them comprising one or more random variables), rather than considering individual random variables, we instead focus on each uncontrollable variable. We say that two variables are said to be *related* if there is at least one constraint between them. Figure 3 gives an example of the relatives of an uncontrollable variable x. For each uncontrollable variable x_j, the bound interval $[L_s(x_j), U_s(x_j)]$ of x_j may be slided to the left or right by shifting the value of its parent variable $pa(x_j)$ along the interval $[pa(x_j) - \delta^-(pa(x_j)), pa(x_j) + \delta^+(pa(x_j))]$. Similarly, the bound interval may be expanded by shifting the values of the executable variables y_i emanating from x_j. In order to understand which of these variables to shift, we first provide the following definitions.

Definition 7. Optimal Relatives for an Uncontrollable Variable. Given a feasible solution s, the Optimal Relatives for an uncontrollable variable x_j $(j = 1, ...m)$ are the controllable variables y_j^l and y_j^u *related* with x_j that maximize the interval bounds of x_j when committing to s,i.e.,

$$y_j^l = argmax_{y_i}\{(y_i - b_{x_j y_i})|s\}$$
$$y_j^u = argmin_{y_i}\{(y_i + b_{y_i x_j})|s\}$$

where y_i's are the controllable variables *related* with x_j.

In other words, when $pa(x_j)$ is fixed, the expanded bounds of x_j ($[L_s(x_j), U_s(x_j)]$) for the executable constraints emanating from x_j (see Figure 3) to remain satisfied under the solution s is given by $[y_j^l - b_{x_j y_j^l} - pa(x_j), y_j^u + b_{y_j^u x_j} - pa(x_j)]$.

Definition 8. Optimal Relatives for an Executable Variable. Given a feasible solution s, the Optimal Relatives for an executable (controllable) variable y_i ($i = 1, ...n$) are the controllable variables Y_i^l and Y_i^u *related* with y_i that satisfy:

$$Y_i^l = argmax_{y_j}\{y_j - a_{y_i y_j}|s\}$$
$$Y_i^u = argmin_{y_j}\{y_j + a_{y_j y_i}|s\}$$

where y_j's are the controllable points *related* with y_i.

Then, $\delta^-(y_i)$ and $\delta^+(y_i)$ for y_i can be derived from its Optimal Relatives, i.e. $\delta^-(y_i) = y_i - Y_i^l + a_{y_i Y_i^l}$ and $\delta^+(y_i) = Y_i^u + a_{Y_i^u y_i} - y_i$.

With the above definitions, we are ready to define the neighboring solutions of s. For each $j = 1, ..., m$ we can obtain new neighboring solutions by the two following operators:

- Operator $\mathcal{O}_1$: Decrease or increase the value of the corresponding parent to get two neighbors:
$$s^l_{\mathcal{O}_1} = (s_1, \cdots, pa(x_j) - \delta^-(pa(x_j)), \cdots, s_n)$$
$$s^u_{\mathcal{O}_1} = (s_1, \cdots, pa(x_j) + \delta^+(pa(x_j)), \cdots, s_n)$$
which are also feasible and may improve the probability of success as they slide the interval of $\tilde{d}_j$.
- Operator $\mathcal{O}_2$: Decrease or increase the value of the corresponding optimal relatives to get two neighbors:
$$s^l_{\mathcal{O}_2} = (s_1, \cdots, y_j^l - \delta^-(y_j^l), \cdots, s_n)$$
$$s^u_{\mathcal{O}_2} = (s_1, \cdots, y_j^u + \delta^+(y_j^u), \cdots, s_n)$$
which are also feasible and may improve the probability of success since they expand the interval of $\tilde{d}_j$.

At each iteration, the neighbors of the current solution is constructed by changing either the parent or an optimal relative of a uncontrollable variable to the value that guarantees the largest interval while keeping the other elements fixed.

5.3 Algorithm

Now we formally describe one local search iteration that moves from a current solution $s = (s_1, \cdots, s_n)$ to the next by calculating the Optimal Relatives Y_i^l and Y_i^u for each y_i, and y_j^l and y_j^u for each x_i. Our local search neighborhood generation algorithm can be summarized as in Algorithm 1.

We have three variants of the Algorithm 1 by applying either one or both of the operators $\mathcal{O}_1$ and $\mathcal{O}_2$. We will show in Section 6 the comparison among those variants.

The analysis of worst-case computational complexity is given as follows. Since we compute Y_i^l and Y_i^u for each y_i, and y_j^l and y_j^u for each x_j for the initiate solution, the cost for initial phase is $O((n+m)r)$, where n and m are the number of controllable and uncontrollable variables respectively, r is maximum number

Algorithm 1 Searching for the Best Neighbor in a Component DTPU

Input: Component DTPU T and feasible schedule s.
Output: BN, the best neighbor of s.

```
Initialize BN ← s
for each uncontrollable variable x do
    Apply O_1 to get two neighbors s^l_{O_1} and s^u_{O_1}
    if P(T|s^l_{O_1}) > P(T|s) then BN ← s^l_{O_1};
    end if
    if P(T|s^u_{O_1}) > P(T|s) then BN ← s^u_{O_1};
    end if
    Apply O_2 to get two neighbors s^l_{O_2} and s^u_{O_2}
    if P(T|s^l_{O_2}) > P(T|s) then BN ← s^l_{O_2};
    end if
    if P(T|s^u_{O_2}) > P(T|s) then BN ← s^u_{O_2};
    end if
end for
return BN
```

of related (adjacent) variables (which is the max degree of the graph). The cost for computing the probability of success is $O(mK)$ where K is the maximum number of disjunctions in a disjunctive contingent constraint (which is typically a small value). Since each solution keeps information about Y_i^l and Y_i^u for each y_i, and y_j^l and y_j^u for each x_j, the cost for updating one of them when we change the value of a y_j is constant. Therefore, the cost for the optimal relative update step is $O(r)$. Consequently, the cost for a local search move is $O(mr)$. Finally, the complexity of our local search algorithm is $O((n+m)r) + O(Lm(r+K))$ where L is the number of local search iterations.

6 Experimental Analysis

In this section, we present experimental results that verify the performance of our proposed local search algorithm. We implemented our algorithm in C# on a Core(TM) 2 Duo CPU 2.33GHz processor under Windows7 operating system with a main memory of 2GB.

6.1 DTPU Instance Generation

In general, to evaluate the performance of a heuristic optimization algorithm, it would be ideal to have baselines such as the exact optimal solutions for a set of problem instances. But since there are no DTPU benchmark problems in the literature and it is impractical to optimally solve reasonably large instances, we resort to using reasonably tight lower bounds. More precisely, we generate problem instances which satisfy the condition that there is at least one feasible solution with the probability of success of at least $1-\epsilon$ for a given threshold value ϵ between 0 and 1. On small values of ϵ (e.g., 0.2, 0.1, or 0.05), the value

$1 - \epsilon$ provides a tight lower bound for the optimal solution, and our aim is to show that our algorithm produces a solution very near to $1 - \epsilon$.

For our experiments, we restrict to two disjunctions per disjunctive constraint. As described in the Problem Definition section, this is a reasonable setting in modeling real-world uncertainty.

With the above considerations in mind, we apply the following steps to generate the problem instances with n_c controllable variables, n_u uncontrollable variables, m_c controllable (i.e. STP) constraints, $2 \cdot n_c$ executable constraints, and roughly $q \cdot m_c$ disjunctive constraints. q is a small value between 0 and 1 to ensure that the number of disjunctive constraints is small.

Generating a DTP To ensure that a generated instance has at least one feasible solution, we reverse-engineer the problem generation; that is, we first generate a schedule s' and then generate the set of constraints that are satisfied given s'. More precisely, we employ the following steps:

1. Generate a connected graph G with n_c nodes variables and m_c edges.
2. Schedule s': assign 0 to the *Time Reference* point and assign uniformly distributed random values to other nodes.
3. If (y_i, y_j) is an edge in G, and values assigned to y_i and y_j are $val(y_i)$and $val(y_j)$ respectively, and if $val(j_i) < val(y_j)$ then:
   - construct a constraint $y_j - y_i \in [U(0, val(y_j) - val(y_i)), U(val(y_j) - val(y_i), 2(val(y_j) - val(y_i)))]$. (This is to ensure that the generated constraint is satisfied.)
   - with probability q, construct the disjunction constraint between y_i and y_j in a similar fashion.

Generating uncontrollable variables, executable and contingent constraints Executable constraints and contingent constraints are generated in a similar fashion as above. That is, we first generate a realization of all uncontrollable variables. Based on this realization, we generate the executable constraints that are satisfied given s'(generated in the DTP generation phrase) and the realization. The set of contingent constraints are generated such that for each uncontrollable variable x, the set of generated executable constraints related to x is satisfied with probability $\hat{\epsilon} = (1 - \epsilon)^{1/n_u}$. To ensure that the generated instance has the probability of success given s' is $(\hat{\epsilon})^{n_u} = 1 - \epsilon$, we employ the following steps:

1. Assign random number $U(1, max_i\{val(y_i)\})$ to each uncontrollable variable.
2. For each uncontrollable variable x with assigned number $val(x)$:
   - Randomly choose two controllable variables y_i and y_j such that $val(y_i) > val(x)$ and $val(y_j) > val(x)$, then construct executable constraints between x and y_i, as well as x and y_j.
   - Randomly choose a variable y such that $val(y) < val(x)$ as $pa(x)$
   - Compute $L_s(x)$ and $U_s(x)$ associated with s'.
   - Randomly generate $p \in (0, 1)$, and set the two distributions $\tilde{d}_1$, $\tilde{d}_2$ such that the probability that $p\tilde{d}_1 + (1 - p)\tilde{d}_2 \in [L_s(x), U_s(x)]$ is $\hat{\epsilon}$.

In the case that $\tilde{d}_1$ and $\tilde{d}_2$ are of normal distributions with deviations σ_1 and σ_2, if the probability that the realization falls within the interval $[L_s, U_s]$ is $\hat{\epsilon}$, we have the following:

$$\begin{aligned}\hat{\epsilon} = p \cdot [\Phi(\tfrac{U_s-\mu_1}{\sigma_1}) - \Phi(\tfrac{L_s-\mu_1}{\sigma_1})] + \\ +(1-p) \cdot [\Phi(\tfrac{U_s-\mu_2}{\sigma_2}) - \Phi(\tfrac{L_s-\mu_2}{\sigma_2})]\end{aligned} \tag{7}$$

where Φ is cumulative function for the standard normal distribution. A simple way to solve this equation is to set both $[\Phi(\frac{U_s-\mu_1}{\sigma_1}) - \Phi(\frac{L_s-\mu_1}{\sigma_1})]$ and $[\Phi(\frac{U_s-\mu_2}{\sigma_2}) - \Phi(\frac{L_s-\mu_2}{\sigma_2})]$ equal to $\hat{\epsilon}$. That is,

$$\begin{aligned}\Phi(\tfrac{U_s-\mu_1}{\sigma_1}) &= 1 - \nu \cdot (1-\hat{\epsilon}) \\ \Phi(\tfrac{L_s-\mu_1}{\sigma_1}) &= (1-\nu) \cdot (1-\hat{\epsilon}) \\ \Phi(\tfrac{U_s-\mu_2}{\sigma_2}) &= 1 - \tau \cdot (1-\hat{\epsilon}) \\ \Phi(\tfrac{L_s-\mu_2}{\sigma_2}) &= (1-\tau) \cdot (1-\hat{\epsilon})\end{aligned} \tag{8}$$

where ν and τ are random numbers in $(0, 1)$.

In the case that $\tilde{d}_1$ and $\tilde{d}_2$ are of uniform distributions, for the purpose of simplicity, we may let both $P(\tilde{d}_1 \in [L_s, U_s])$ and $P(\tilde{d}_2 \in [L_s, U_s])$ equal to $\hat{\epsilon}$, and L_s and U_s are the lower bound of $\tilde{d}_1$ and upper bound of $\tilde{d}_2$ respectively. This leads to the following:

$$\begin{aligned}\tilde{d}_1 &\sim U[L_s, U_1] \text{ where } U_1 = L_s + (U_s - L_s)/\hat{\epsilon} \\ \tilde{d}_2 &\sim U[L_2, U_s] \text{ where } L_2 = U_s - (U_s - L_s)/\hat{\epsilon}\end{aligned} \tag{9}$$

6.2 Results and Analysis

We ran experiments on 3 sets of random DTPU instances generated with parameter settings given in Table 1. For each setting, we generate 100 instances whose random variables follow the normal distribution, and 100 instances follow the uniform distribution.

	Set 1	Set 2	Set 3
#Controllable variables	15	20	30
#Uncontrollable variables	5	10	10
#DTP constraints	50	75	100
#Executable constraints	10	20	20
#Contingent constraints	4-6	6-12	6-12

Table 1. Parameters used in instance generation

For every instance, we generated random initial feasible solutions and applied three variants of our proposed local search: $\mathcal{O}_1$ denotes using only the operator $\mathcal{O}_1$ is applied to search for the best neighbor; $\mathcal{O}_2$ denotes using only operator $\mathcal{O}_2$; and $\mathcal{O}_1 + \mathcal{O}_1$ denotes using both operators. We then computed the mean and variance of the probabilities of success obtained by our three local search

variants on each set of problem instances. And as a benchmark comparison, we also show the mean probability of success of the best random initial solutions.[2] In addition, to show the actual quality of solutions, we show the number of times the obtained probability exceeds the tight lower bound $1-\epsilon$ (for $\epsilon = 0.2, 0.1$ and 0.05). The results are summarized in Table 2.

Set 1							
		Normal distribution			Uniform distribution		
$\boldsymbol{\epsilon}$		**0.2**	**0.1**	**0.05**	**0.2**	**0.1**	**0.05**
Mean of best initial solutions		0.60	0.69	0.78	0.65	0.69	0.67
$\mathcal{O}_1$	Mean	0.82	0.90	0.95	0.94	0.94	0.91
	Variance	0.07	0.11	0.06	0.08	0.10	0.13
	#Instances exceeding $\mathbf{1-\epsilon}$	85	74	87	93	84	55
$\mathcal{O}_2$	Mean	0.79	0.89	0.93	0.85	0.86	0.85
	Variance	0.12	0.10	0.12	0.16	0.15	0.18
	#Instances exceeding $\mathbf{1-\epsilon}$	73	72	70	74	51	33
$\mathcal{O}_1 + \mathcal{O}_2$	Mean	**0.86**	**0.94**	**0.97**	**0.97**	**0.97**	**0.97**
	Variance	**0.03**	**0.01**	**0.01**	**0.05**	**0.06**	**0.06**
	#Instances exceeding $\mathbf{1-\epsilon}$	**98**	**100**	**99**	**95**	**89**	**80**
Set 2							
Mean of best initial solutions		0.57	0.69	0.80	0.50	0.49	0.50
$\mathcal{O}_1$	Mean	0.80	0.88	0.93	0.81	0.80	0.81
	Variance	0.07	0.06	0.05	0.15	0.13	0.13
	#Instances exceeding $\mathbf{1-\epsilon}$	64	55	40	61	24	8
$\mathcal{O}_2$	Mean	0.77	0.88	0.93	0.72	0.73	0.72
	Variance	0.14	0.06	0.07	0.17	0.16	0.17
	#Instances exceeding $\mathbf{1-\epsilon}$	63	58	54	35	15	5
$\mathcal{O}_1 + \mathcal{O}_2$	Mean	**0.86**	**0.93**	**0.97**	**0.93**	**0.92**	**0.91**
	Variance	**0.02**	**0.01**	**0.01**	**0.10**	**0.09**	**0.08**
	#Instances exceeding $\mathbf{1-\epsilon}$	**98**	**98**	**97**	**91**	**69**	**39**
Set 3							
Mean of best initial solutions		0.52	0.65	0.70	0.45	0.46	0.45
$\mathcal{O}_1$	Mean	0.80	0.88	0.93	0.81	0.80	0.81
	Variance	0.07	0.06	0.05	0.15	0.13	0.13
	#Instances exceeding $\mathbf{1-\epsilon}$	64	55	40	61	24	8
$\mathcal{O}_2$	Mean	0.75	0.85	0.90	0.68	0.67	0.68
	Variance	0.14	0.11	0.16	0.19	0.17	0.16
	#Instances exceeding $\mathbf{1-\epsilon}$	45	41	38	25	8	2
$\mathcal{O}_1 + \mathcal{O}_2$	Mean	**0.86**	**0.93**	**0.96**	**0.91**	**0.90**	**0.89**
	Variance	**0.03**	**0.02**	**0.03**	**0.11**	**0.10**	**0.11**
	#Instances exceeding $\mathbf{1-\epsilon}$	**98**	**96**	**93**	**90**	**59**	**39**

Table 2. Performance of Local Search

[2] We generated one random initial feasible solution for each C-DTPU, and these figures are based on the *best* solutions generated. It is noteworthy that many of the initial solutions have very low probability of success, almost near to zero.

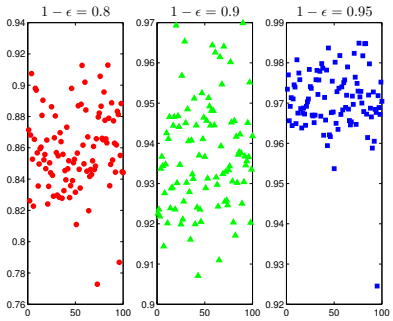

Fig. 4. Results for running $\mathcal{O}_1+\mathcal{O}_2$ on Set 1 instances under normal distribution.

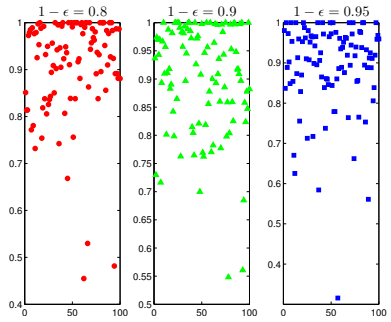

Fig. 5. Results for running $\mathcal{O}_1+\mathcal{O}_2$ on Set 3 instances under uniform distribution.

We observe that all local search variants performed well in general, since they generate solutions which are reasonably close to the value $1-\epsilon$ whereas the initial solutions are not so. This observation is consistent across all problem instance sets tested with different parameters and under both the normal and uniform distributions. Table 2 also clearly shows that, while the performance of $\mathcal{O}_1$ and $\mathcal{O}_2$ are not significantly different, the combined approach $\mathcal{O}_1+\mathcal{O}_2$ always gives the best solutions. Moreover, the solutions obtained by $\mathcal{O}_1+\mathcal{O}_2$ are more reliable, in the sense that they are closer to the optimal solutions (having the highest mean of probability of success among the three variants) and have the smallest variance.

Figure 4 shows the detailed results on the actual distribution of the 100 solution values obtained from running $\mathcal{O}_1+\mathcal{O}_1$ on Set 1 instances with contingent variables that follow the normal distribution. Similarly, Figure 5 shows results obtained from running $\mathcal{O}_1+\mathcal{O}_2$ on Set 3 instances with contingent variables that follow the uniform distribution. Both these figures illustrate that most solutions obtained by our algorithm $\mathcal{O}_1+\mathcal{O}_2$ are indeed very close to the optimal solutions (or more precisely, the respective tight lower bounds).

7 Conclusion

In this paper, we presented what we believe to be the first efficient scheme to tackle the optimization problem associated with DTPUs, based on a decomposition mechanism and local search. Experimental results demonstrate the computational efficiency of our approach on a range of problem instances. We believe it is possible to make the search process more effectively with more powerful local search algorithms. For future works, it is interesting to relax the assumption of independence among the random variables, which we inherited from [9], since such events are seldom independent. It is a challenging analytical and computational task to compute the probability of successful execution taking the correlation of random variables into consideration.

References

[1] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artif. Intell.*, 49, May 1991.
[2] H. C. Lau, J. Li, and R. H. Yap. Robust controllability of temporal constraint networks under uncertainty. In *Proc. ICTAI*, 2006.
[3] M. D. Moffitt and M. E. Pollack. Applying local search to disjunctive temporal problems. In *Proc. IJCAI*, 2005.
[4] P. Morris and N. Muscettola. Temporal dynamic controllability revisited. In *Proc. AAAI*, 2005.
[5] B. Peintner, K. B. Venable, and N. Yorke-Smith. Strong controllability of disjunctive temporal problems with uncertainty. In *Proc. CP*, 2007.
[6] B. Rustem and Q. Nguyen. An algorithm for the inequality-constrained discrete Min–Max problem. *SIAM J. Optimization*, 8(1), 1998.
[7] K. Stergiou and M. Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artif. Intell.*, 120, June 2000.
[8] J. Thornton, M. Beaumont, A. Sattar, and M. Maher. A local search approach to modelling and solving interval algebra problems. *J. Logic and Comput.*, 14, February 2004.
[9] I. Tsamardinos. A probabilistic approach to robust execution of temporal plans with uncertainty. In *Proc. SETN*, 2002.
[10] I. Tsamardinos and M. E. Pollack. Efficient solution techniques for disjunctive temporal reasoning problems. *Artif. Intell.*, 151, December 2003.
[11] I. Tsamardinos, M. E. Pollack, and S. Ramakrishnan. Assessing the probability of legal execution of plans with temporal uncertainty. In *Proc. ICAPS-04 Workshop*, 2004.
[12] K. B. Venable and N. Yorke-Smith. Disjunctive temporal planning with uncertainty. In *Proc. IJCAI*, 2005.
[13] T. Vidal and H. Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *J. Experimental and Theoretical Artificial Intelligence*, 11, 1999.
[14] B. W. Wah and D. Xin. Optimization of bounds in temporal flexible planning with dynamic controllability. In *Proc. ICTAI*, 2004.