Fast (Incremental) Algorithms for Useful Classes of Simple Temporal Problems with Preferences

T. K. Satish Kumar
Computer Science Division
University of California, Berkeley
tksk@eecs.berkeley.edu

Abstract

In this paper, we will provide a fast polynomial-time algorithm for solving *simple temporal problems* (STPs) with *piecewise linear convex preference functions* and a *utilitarian* objective function. Our algorithm is motivated by the study of the *linear programming* (LP)-dual of a given *mincost circulation problem* (MCCP). We will also show how this *duality* relationship between *simple temporal problems with preferences* (STPPs) and MCCPs leads to fast *incremental* algorithms for solving the former. Our algorithms bear important implications in planning, scheduling and execution monitoring scenarios where (partial) plans are subject to repeated changes, and the most preferred solutions to the underlying STPPs have to be computed and updated fast (incrementally).

1 Introduction

Many applications in AI depend crucially on our ability to efficiently deal with combinatorial problems that involve both a *satisfaction* component and an *optimization* component—hence requiring us to reason about both *hard* and *soft* constraints. (The hard and soft constraints capture the satisfaction and optimization components of a problem respectively.) We are often required to express natural factors like fuzziness, probabilities, preferences and/or costs, and are subsequently interested in finding an optimal solution with respect to one or more criteria. Important problems in planning and scheduling, for example, involve metric temporal constraints and various kinds of associated preferences.

Simple Temporal Problems (STPs) [Dechter *et al*, 1991] constitute a well studied formalism used for reasoning with metric time. An STP is characterized by a graph $G = \langle \mathcal{X}, \mathcal{E} \rangle$, where $\mathcal{X} = \{X_0, X_1 \ldots X_N\}$ is a set of events (X_0 is the "beginning of the world" node and is set to 0 by convention), and $e = \langle X_i, X_j \rangle$ in $\mathcal{E}$, annotated with the bounds $[LB(e), UB(e)]$, is a *simple temporal constraint* between X_i and X_j indicating that X_j must be scheduled between $LB(e)$ and $UB(e)$ seconds after X_i is scheduled ($LB(e) \leq UB(e)$).[1] The formalism of *Simple Temporal Problems with Preferences* (STPPs) [Khatib *et al*, 2001] was introduced to deal with situations where the factors guiding the relative execution times of certain pairs of events are better modeled (and abstracted) using *preference functions* rather than *hard constraints*. Roughly, an STPP is characterized by a graph $G = \langle \mathcal{X}, \mathcal{E} \rangle$, where $\mathcal{X} = \{X_0, X_1 \ldots X_N\}$ is a set of events (X_0 is the "beginning of the world" node and is set to 0 by convention), and $e = \langle X_i, X_j \rangle$ in $\mathcal{E}$, is either a *hard* simple temporal constraint annotated with $[LB(e), UB(e)]$, or a *soft* constraint annotated with a *preference function* $F_e(t) : R \rightarrow R$ indicating the *preference* associated with scheduling X_j exactly t seconds after X_i. Given such *local* temporal preference functions, the *objective function* can be defined in a number of ways. One simple way is to maximize the *minimum* of all the local temporal preferences; this is referred to as *Weakest Link Optimization* (WLO). WLO in STPPs has been shown to be tractable when all the preference functions are *semi-convex*[2] [Khatib *et al*, 2001]. The WLO model, however, corresponds to *egalitarianism* in *utility theory*, and it is easy to illustrate its myopic focus. A much better optimization criterion is to maximize the *sum* of the preferences (corresponding to *utilitarianism* in *utility theory*).[3]

A polynomial-time algorithm for solving STPPs with piecewise linear convex preference functions and a utilitarian objective function is provided in [Morris *et al*, 2004]. This algorithm, however, employs a general LP solver, and therefore does not tightly characterize the complexity of solving such problems. Moreover, the algorithm is not incremental. We note that there is great value in designing simple and fast polynomial-time algorithms for solving metric temporal reasoning problems. This is evident, for example, even in the simple case of STPs. Although STPs are characterized by a set of simple linear (difference) constraints amenable to a polynomial-time algorithm using a general LP solver, it is well understood that it is much better to solve a given STP using shortest path computations (say, by employing the Bellman-Ford algorithm). This is because the latter approach would lead to a simple and fast polynomial-time algorithm—tightly characterizing the complexity of solving STPs. In the

[1]An STP can be solved in polynomial time [Dechter *et al*, 1991]; but in general, STPs are not as expressive as DTPs [Stergiou and Koubarakis, 1998] or RDTPs [Kumar, 2005].

[2]$g(y)$ is *semi-convex* iff for any horizontal line L, the set of all y, such that $g(y)$ is not below L, constitutes a single interval.

[3]In [Kumar, 2004], a polynomial-time algorithm for certain other useful classes of STPPs is provided.

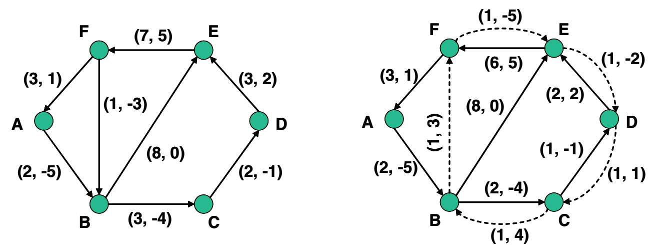

Figure 1: Illustrates how flow is augmented along negative cost cycles in the residual graph (for solving a given MCCP). Every edge is annotated with two numbers: the first one being the capacity, and the second one being the cost. (a) (left-hand side) shows a scenario where a negative cost cycle BCDEF is chosen to augment flow along. If a unit amount of flow is pushed along this cycle, then it leads to a situation as shown in (b) (right-hand side) with the appropriate residual edges. We note that there exists a negative cost cycle even in (b)—namely FAB—therefore indicating that further flow has to be augmented to achieve optimality.

same spirit, it is imperative for us to develop simple and fast polynomial-time algorithms for solving STPPs with piecewise linear convex preference functions and a utilitarian objective function—especially because these classes of STPPs are fairly general and very useful in practice (see [Morris *et al*, 2004] for motivating arguments). Further, it is also beneficial to develop incremental algorithms that can readily update the solutions to (these kinds of) STPPs when (small) changes are made to them (as in, say, a *refinement planner*).

In this paper, we will first provide a fast polynomial-time algorithm for solving STPPs of the above mentioned kinds. Our algorithm is motivated by the study of the LP-dual of a given *mincost circulation problem* (MCCP). We will then show how this *duality* relationship between STPPs and MCCPs leads to fast incremental algorithms for solving the former. Our algorithms bear important implications in planning, scheduling and execution monitoring scenarios where (partial) plans are subject to repeated changes, and the most preferred solutions to the underlying STPPs have to be computed and updated fast (incrementally).

2 Brief Overview of MCCPs

A class of linear programs that can be solved better (compared to employing a generic LP solver) using a specialized set of techniques is the class of *flow* problems; these include variants like the well studied *maxflow problems*, *mincost circulation problems*, *generalized flow problems*, etc. Many insights about a given LP problem can be drawn from studying its *dual*. Given a *primal* LP problem, one can construct its *dual* LP problem using a prescribed set of rules [Sultan, 1993]. A number of interesting theorems guide the relationship between a given primal LP problem and its dual; these include the *weak duality theorem*, the *strong duality theorem* and *complementary slackness* [Sultan, 1993].[4]

[4]A well-known technique for designing approximation algorithms, for example, is the so-called *primal-dual* method; LP-duality is also extensively used in the study of *flows* and *cuts*.

In this section, we will briefly review MCCPs and some of the well-known results/algorithms associated with solving them. An MCCP is characterized by a directed graph $G = \langle V, E \rangle$, where $V = \{v_1, v_2 \dots v_n\}$ is the set of nodes, and E is the set of edges.[5] An edge $e = \langle v_i, v_j \rangle \in E$ is annotated with two numbers: (a) a positive real number $U(e)$ representing the *capacity* of the edge (maximum amount of flow that can be pushed through the edge), and (b) a real number (not necessarily positive) $C(e)$ representing the *cost* of pushing a unit amount of flow through the edge. The goal is to come up with a *circulation* (an amount of flow on each edge) such that: (a) the *conservation constraints* hold at every node (i.e. the total incoming flow is equal to the total outgoing flow at every node), (b) the *capacity constraints* hold true for every edge (i.e. the amount of flow pushed through any edge is less than or equal to the capacity of that edge), and (c) the overall cost of the flow is *minimized*.

Algorithms for solving MCCPs work by *augmenting* flow along *negative cost cycles* in the residual graph. Given a candidate circulation f, its *residual graph* G_f is defined by replacing each edge $\langle u, v \rangle \in E$ by two edges: (a) $\langle u, v \rangle$ of capacity $U(\langle u, v \rangle) - f(\langle u, v \rangle)$ and cost $C(\langle u, v \rangle)$, and (b) $\langle v, u \rangle$ of capacity $f(\langle u, v \rangle)$ and cost $-C(\langle u, v \rangle)$. A circulation f^* can be shown to be optimal if and only if there are no negative cost cycles in the residual graph G_{f^*}. Figure 1 shows an example of flow augmentations in an MCCP.

Although repeatedly pushing flow along *any* negative cost cycle (in the residual graph) leads us eventually to the optimal circulation, better running times are achieved by pushing flow along the *min-mean cost cycles* [Goldberg and Tarjan, 1989]. The best known algorithms for solving MCCPs work in time $O(m(\log n)(m + n(\log n)))$ [Orlin, 1988]. (Here, n is the number of nodes, and m is the number of edges.) We note, however, that when all the capacities are integers, there are two things that characterize the number of (flow-augmenting) iterations required to solve an MCCP: (a) the strongly polynomial running time of $O(m(\log n)(m + n(\log n)))$, and (b) the amount of the flow itself; so, if the amount of the flow (not the cost of the flow) is itself small, we need only a few iterations. As we will allude to later, this can be exploited in designing incremental algorithms for solving STPPs.

The LP formulation of an MCCP is as follows: ($f(e)$ for all $e \in E$ are the variables)
(1) $\forall e \in E$: $0 \leq f(e) \leq U(e)$
(2) $\forall v \in V$: $\sum_{\langle v,w \rangle \in E} f(\langle v, w \rangle) - \sum_{\langle u,v \rangle \in E} f(\langle u, v \rangle) = 0$
(3) Maximize $-\sum_{e \in E} C(e) f(e)$
We note that the *capacity constraints* and the *conservation constraints* are reflected in (1) and (2) respectively. The dual of the above LP is as follows: ($l(e)$ for all $e \in E$, and $p(v)$ for all $v \in V$ are the variables)
(1) $\forall e \in E$: $l(e) \geq 0$
(2) $\forall \langle u, v \rangle \in E$: $l(\langle u, v \rangle) + p(u) - p(v) \geq -C(\langle u, v \rangle)$
(3) Minimize $\sum_{e \in E} l(e) U(e)$

[5]Unlike in a *maxflow* problem, there are no explicitly nominated *source* and/or *terminal* nodes. However, MCCPs are more general than *maxflow* problems; a given *maxflow* problem can be modeled as an instance of the MCCP by adding a back-edge from the terminal to the source of sufficiently large capacity and negative cost.

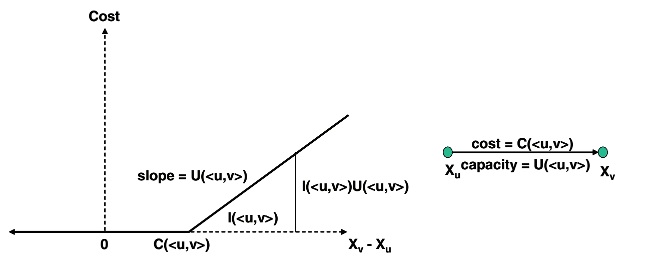

Figure 2: (a) (left-hand side) illustrates a primitive kind of *cost* function for the time difference between two events X_v and X_u. (b) (right-hand side) shows the corresponding cost and capacity annotations on the edge $\langle X_u, X_v \rangle$ in the dual formulation of the problem.

We note that each capacity constraint in the primal corresponds to a variable $l(e)$ in the dual; further, the $l(e)$s are non-negative because the capacity constraints are inequalities. Similarly, each conservation constraint in the primal corresponds to a variable $p(v)$ in the dual; further, the $p(v)$s are unrestricted because the conservation constraints are equalities. Each variable in the primal corresponds to an inequality in the dual. The left-hand sides of these inequalities follow from tracing which variables in the primal participate in which constraints (and what the respective coefficients are). The right-hand sides follow from the primal's objective function. Finally, the coefficients of the variables in the dual's objective function come from the right-hand sides of the corresponding constraints in the primal.

$p(u)$ is referred to as the *potential* of the node u, and $p(u) + C(\langle u, v \rangle) - p(v)$ is referred to as the *reduced cost* $C_p(\langle u, v \rangle)$ of the edge $\langle u, v \rangle$. It is easy to see that the sum of the reduced costs along a cycle is equal to the sum of the costs (along the same cycle). Further, it can be shown that a circulation f^* is optimal if and only if there exists a potential function p such that for all $e \in G_{f^*}$: $C_p(e) \geq 0$ (see [Ahuja *et al*, 1993]).

3 LP-Duality Perspectives on STPPs

Examining the dual of an MCCP more closely, we notice that the only constraints involving the potentials of the nodes are difference inequalities of the form: $l(\langle u, v \rangle) + p(u) - p(v) \geq -C(\langle u, v \rangle)$ i.e. $p(v) - p(u) \leq C(\langle u, v \rangle) + l(\langle u, v \rangle)$. These resemble the inequalities arising in STPs (with a few beneficial differences). First, we notice that if $l(\langle u, v \rangle)$ is equal to 0 for all $\langle u, v \rangle$, then these inequalities directly correspond to simple temporal constraints.[6] In the more general case that $l(\langle u, v \rangle) \geq 0$, we can reinterpret $l(\langle u, v \rangle)$ as indicating the "*amount by which the simple temporal constraint* $p(v) - p(u) \leq C(\langle u, v \rangle)$ *is violated*." Further, the dual of an MCCP allows us to minimize a positive linear combination of the $l(\langle u, v \rangle)$s (with the weight corresponding to $l(\langle u, v \rangle)$ being the positive capacity $U(\langle u, v \rangle)$). This can be reinterpreted as minimizing "*a positive linear combination of how much we violate each of the specified simple temporal constraints by*." In principle, therefore, STPPs with linear *cost* functions (as shown in Figure 2) fit the LP-dual model of MCCPs. Here (see Figure 2), every edge $\langle X_u, X_v \rangle$ is annotated with a *cost* function (instead of a *preference* function) that remains 0 as far as the difference $X_v - X_u$ is $\leq C(\langle u, v \rangle)$, but begins to increase linearly with positive slope $U(\langle u, v \rangle)$ beyond this bound; and the objective is to *minimize* the overall cost (summed over all edges). Figure 2 also shows that such a cost function translates to the edge $\langle X_u, X_v \rangle$ with capacity $U(\langle u, v \rangle)$ and cost $C(\langle u, v \rangle)$ in the dual formulation of the problem as an MCCP.

[6]Note that $C(\langle u, v \rangle)$ is the cost associated with the edge $\langle u, v \rangle$, and is allowed to be negative—maintaining the complete generality of a difference inequality arising in STPs.

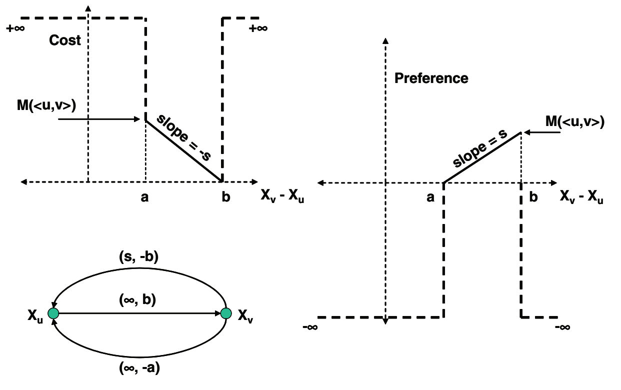

Figure 3: (b) (right-hand side) shows a gadget where $a \leq X_v - X_u \leq b$, and the preference function in the feasible region has a positive slope of s between $X_v - X_u = a$ and $X_v - X_u = b$. (a) (upper left-hand side) shows the equivalent cost function (ignoring additive constants like $M(\langle u, v \rangle)$). (c) (lower left-hand side) shows the required MCCP representation (in the dual formulation).

We note/recollect two things before proceeding further. First, a *hard* simple temporal constraint of the form $p(v) - p(u) \leq C(\langle u, v \rangle)$ can be handled by modeling it as $p(v) - p(u) \leq C(\langle u, v \rangle) + l(\langle u, v \rangle)$ with $U(\langle u, v \rangle)$ being practically set to ∞. Second, although the flow-based techniques allow us to efficiently solve only the dual LP problem (MCCP), the *strong duality theorem* yields the *value* of the optimal solution for the (original) primal LP problem (STPP). In the next section (where we will deal with STPPs having piecewise linear convex preference functions and a utilitarian objective function), we will show that the *value* of the optimal solution alone can be used to iteratively construct the required optimal schedule in its entirety.

3.1 Designing Gadgets

We will now introduce combinatorial structures called *gadgets* that will be used later in the paper to handle STPPs with piecewise linear convex preference functions. These gadgets are in turn built using the basic building block of Figure 2.[7]

Consider designing the gadget where $a \leq X_v - X_u \leq b$, and the preference function in the feasible region is as indicated in Figure 3(b) (with a positive slope of s between

[7]The introduction of such gadgets is partly inspired by the techniques used in [Morris *et al*, 2004].

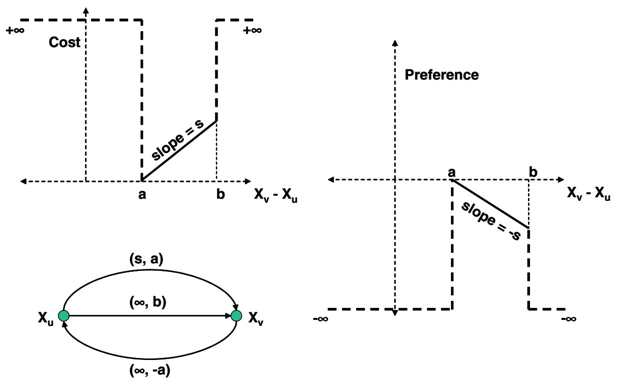

Figure 4: Shows the combinatorial structures relevant to preference functions with negative slopes. The three diagrams shown above are in direct analogy with those of Figure 3.

$X_v - X_u = a$ and $X_v - X_u = b$). Ignoring additive constants, such a *preference function* is equivalent to a *cost function* as shown in Figure 3(a). This cost function is positive everywhere, and can be modeled using two *hard* constraints (namely, $a \leq X_v - X_u$ and $X_v - X_u \leq b$), and one *soft* constraint (namely, $X_v - X_u \geq b$ with a cost of violation by 1 unit being equal to s). From Figure 2, therefore, the required MCCP representation (in the dual formulation of the problem), is as shown in Figure 3(c). Using similar arguments (see Figure 4), we can design a gadget where $a \leq X_v - X_u \leq b$, and the preference function in the feasible region is as indicated in Figure 4(b) (with a negative slope of $-s$ between $X_v - X_u = a$ and $X_v - X_u = b$).

4 STPPs with Piecewise Linear Convex Preference Functions

In this section, we will build the final algorithm for solving STPPs with piecewise linear convex preference functions[8] and a utilitarian objective function. A key idea in this algorithm involves the transformation of such a given STPP to one that has preference functions limited to the types in Figures 3 and 4. The transformation works as follows. For every piecewise linear convex preference function $F_e(t)$, and corresponding to every (intermediate) landmark of this function, we add a new variable. The difference between such consecutive variables is constrained to be between 0 and the length of the corresponding interval,[9] and the associated preference function is designed to be in direct correspondence with the slope of $F_e(t)$ within this interval. Figure 5 illustrates this process for a single preference function. The MCCP representation of the new problem can be built by simulating preference functions with positive and negative slopes using the gadgets in Figures 3 and 4 respectively. Figure 6 presents an

[8]For simplicity, we will assume that every such preference function is defined to be within an interval; for example, Figure 5 is taken to not only specify a preference function for $X_j - X_i$, but also implicitly constrain $X_j - X_i$ to be within $[r_1, r_6]$.

[9]Note that we use hard constraints for this purpose.

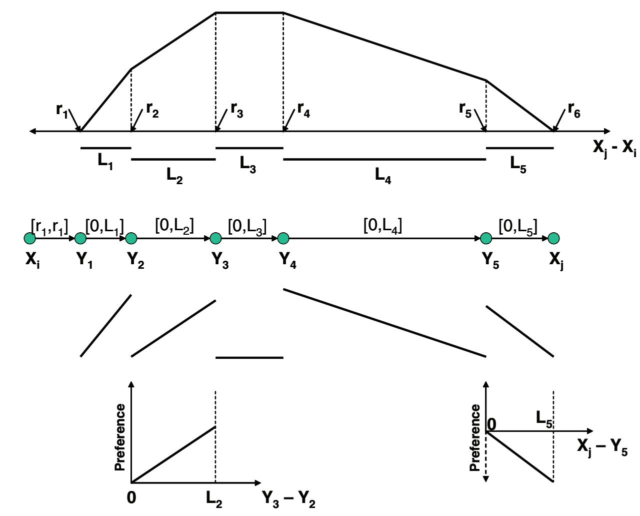

Figure 5: The top diagram shows an example piecewise linear convex preference function with the appropriate landmarks and lengths defined by it. The second diagram shows the intermediate variables we would introduce between X_j and X_i. It also shows the hard simple temporal constraints we would add between consecutive variables created for this edge. The third diagram shows the slopes of the line segments in the corresponding intervals (simply as defined in the original preference function). The bottom diagram shows two examples of the kinds of preference functions we would like to construct corresponding to each interval. In each case, the preference value is 0 at the origin, and either increases or decreases linearly up to a certain value; everywhere else, the preference value is assumed to be $-\infty$ (directly corresponding to the gadgets in Figures 3 and 4).

elaborate example of the dual formulation of an STPP (with piecewise linear convex preference functions) as an MCCP.

We will now argue that an optimal solution for the new problem (with the additional variables) yields an optimal solution for the original problem. The key insight stems from the fact that in a piecewise linear convex preference function, the slopes of the line segments (defined by the function) are monotonically decreasing (from left to right). Consider any edge $e = \langle X_i, X_j \rangle$ with the associated preference function $F_e(t)$. Let the landmarks defined by $F_e(t)$ be $r_1^e, r_2^e \ldots r_{k_e+1}^e$, and let the corresponding intermediate variables be $Y_1^e, Y_2^e \ldots Y_{k_e}^e$ (see Figure 5 for an example). The slopes of the lines segments (defined by $F_e(t)$) in the intervals $[r_1^e, r_2^e], [r_2^e, r_3^e] \ldots [r_{k_e}^e, r_{k_e+1}^e]$ are monotonically decreasing. Because of this property, and from the standard arguments associated with the optimality of the greedy algorithm for solving the "*fractional knapsack problem*," it follows that the optimal setting of the intermediate variables $Y_1^e, Y_2^e \ldots Y_{k_e}^e$ for *any* values of X_i and X_j is:[10] $Y_2^{*e} \leftarrow Y_1^e + \min(r_2^e - r_1^e, X_j - Y_1^e); Y_3^{*e} \leftarrow Y_2^{*e} + \min(r_3^e - r_2^e, X_j - Y_2^{*e}); \ldots Y_{k_e}^{*e} \leftarrow Y_{k_e-1}^{*e} + \min(r_{k_e}^e - r_{k_e-1}^e, X_j - Y_{k_e-1}^{*e})$ (see Figure 7 for an example). For *any* values of X_i and X_j, therefore, the best way to set the intermediate variables is to exhaust the slack in the difference between one consecutive pair of (intermediate) variables before going to the next pair (left to right). Since

[10]Note that Y_1^e is fixed to $X_i + LB(\langle X_i, X_j \rangle)$; see Figure 5.

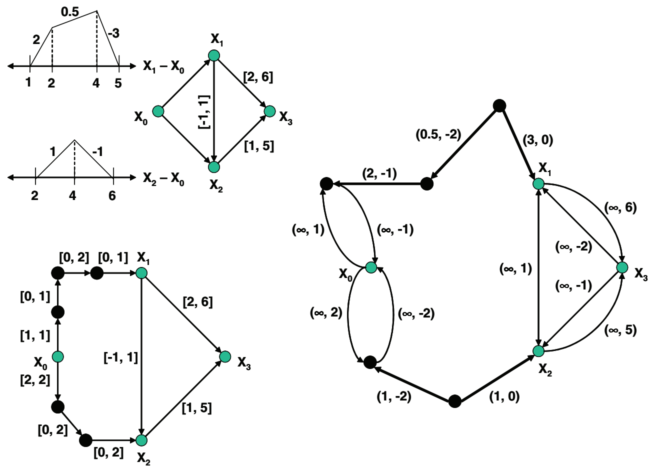

Figure 6: The top-left diagram shows an example STPP with hard simple temporal constraints and two piecewise linear convex preference functions. The bottom-left diagram shows the intermediate variables introduced for each preference function and the hard simple temporal constraints between them (dark nodes indicate newly added variables). The right-hand side diagram shows the MCCP representation of the problem. The dark lines indicate edges that encode different pieces of the corresponding preference functions. For clarity, some hard simple temporal constraints that flank the dark edges are not shown encoded explicitly.

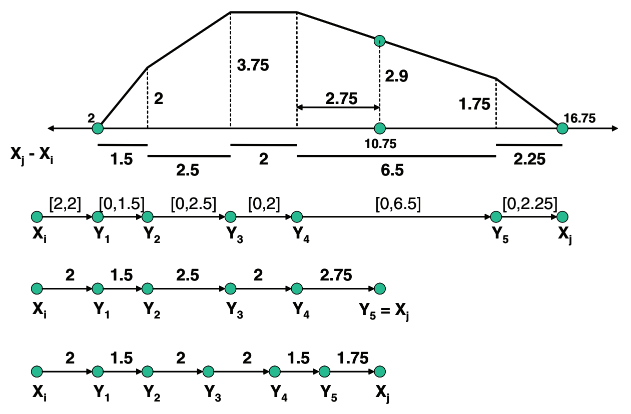

Figure 7: Illustrates that for any (values of) X_i and X_j, the optimal way to set the intermediate variables is to exhaust the slack in one consecutive pair of (intermediate) variables before going to the next (left to right). When $X_j - X_i = 10.75$, for example, the first way of setting the variables (third diagram from top) exhausts all the leftmost slacks, creating a total value of $(1.5)((2-0)/1.5) + (2.5)((3.75-2)/2.5) + (2)((3.75-3.75)/2) + (2.75)((1.75-3.75)/6.5) = 2.9$. This is indeed equal to the true value of the preference function at $X_j - X_i = 10.75$. We note that any other way of setting the intermediate variables leads to a non-optimal value. For example, in the second way (bottom diagram), the total preference is equal to $(1.5)((2-0)/1.5) + (2)((3.75-2)/2.5) + (2)((3.75-3.75)/2) + (1.5)((1.75-3.75)/6.5) + (1.75)((0-1.75)/2.25) = 1.58$ which is less than the actual preference value at this point.

the optimal setting of the intermediate variables is completely determined by the values assigned to the end-point variables X_i and X_j, and since the intermediate variables do not participate in any constraint of the original problem, an optimal solution to the original STPP can be obtained simply by taking the *projection* of an optimal solution to the new STPP.

Figure 8 presents the algorithm for reporting the *value* of an optimal solution to a given STPP (with piecewise linear convex preference functions and a utilitarian objective function).[11] We will now argue that in fact the value of the optimal solution alone is enough to efficiently construct the required optimal schedule. Suppose we have K preference functions in the STPP; let us call them $F_{e_1}(t), F_{e_2}(t) \dots F_{e_K}(t)$. Let s^* be an optimal schedule, and let the value of s^* be v^*. The idea is to replace the preference functions in the STPP (one by one) by additional simple temporal constraints that would serve to rule out all the non-optimal solutions. In each iteration, we will replace an arbitrarily chosen preference function $F_{e_h}(t)$ by a simple temporal constraint such that any optimal solution to the new problem (with one less preference function) has a value $\geq v^*$. In the remaining smaller problem (with respect to the number of preference functions), we will follow the same procedure to replace another arbitrarily chosen preference function by a simple temporal constraint, and so forth. After K iterations, therefore, we will be left with an STP—any solution to which has a value $\geq v^*$. That is, all the non-optimal solutions would have been ruled out, and an optimal solution to the original problem can be obtained simply by solving the final STP using shortest path computations.

The crucial step in the above process, however, is to replace an arbitrarily chosen preference function $F_{e_h}(t)$ by an appropriate simple temporal constraint such that an optimal solution to the new problem has a value $\geq v^*$. This can essentially be done by a standard backtrack-free search algorithm that utilizes the procedure of Figure 8 in every iteration. Suppose $F_{e_h}(t)$ is associated with the edge $\langle X_i, X_j \rangle$. It is easy to see that the requirement $F_{e_h}(X_j - X_i) \geq v$ is in fact equivalent to a simple temporal constraint between X_i and X_j (because of *semi-convexity*). Now, suppose $F_{e_h}(t)$ is always a non-negative integer in the range $[0, M]$.[12] We can consider the simple temporal constraint equivalent to $F_{e_h}(X_j - X_i) \geq d$, and examine whether d plus the value of the optimal solution to the smaller problem (that incorporates the new simple temporal constraint instead of $F_{e_h}(t)$) is equal to the value of the optimal solution to the original problem. If it is, then the newly added simple temporal constraint is the one that can replace $F_{e_h}(t)$. It is easy to see that ranging over all possible values of d (viz. from 0 to M), we will always be able to find an appropriate cutoff value—the corresponding simple temporal constraint of which can replace the function $F_{e_h}(t)$.

The running time complexity of Figure 8 is dominated by that of solving the intermediate MCCP (step 3 in Figure 8).

[11]We assume that the additive constants that arise in the transformations involved (like in Figure 3) are appropriately factored in.

[12]If $F_{e_h}(t)$ is not always integral, then we can scale all the numbers appropriately so that M will be indicative of the precision of the numbers appearing in the range of the function.

```
ALGORITHM: Solve-Piecewise-Linear-Convex-STPP-Value
INPUT: An STPP with piecewise linear convex preference functions and a utilitarian objective function.
OUTPUT: Value of an optimal schedule that satisfies all hard constraints (if possible) and maximizes sum of the preferences.
(1) For every preference function $F_e(t)$ (let $e = \langle X_i, X_j \rangle$):
    (a) Identify the landmarks $r^e_1, r^e_2 \dots r^e_{k_e+1}$, and introduce the corresponding new variables $Y^e_1, Y^e_2 \dots Y^e_{k_e}$ (see Figure 5).
    (b) Create simple temporal constraints between consecutive points in $Y^e_1, Y^e_2 \dots Y^e_{k_e}$, between $X_i$ and $Y^e_1$, and between $Y^e_{k_e}$ and $X_j$ as shown in Figures 5,6.
    (c) For each of the linear preference functions thus defined (see Figures 5,6), build gadgets as shown in Figures 3,4.
(2) Encode hard constraints as explained in the text.
(3) Solve the resulting MCCP using efficient "flow-based" techniques (see [Orlin, 1988] and [Goldberg and Tarjan, 1989]).
(4) RETURN: the value of the solution.
END ALGORITHM
```

Figure 8: A fast polynomial-time algorithm for reporting the *value* of an optimal solution to a given STPP with *piecewise linear convex preference functions* and a *utilitarian* objective function. We assume that a value of $-\infty$ is returned when the problem is infeasible.

This complexity is $O(m(\log\ n)(m + n(\log\ n)))$ where n (number of nodes) and m (number of edges) take into account the new variables and constraints/edges created for the landmarks of the preference functions. The total running time of the algorithm (for constructing the optimal schedule) is $K.R.T$ where K is the total number of preference functions, R is the largest value of any preference function, and T is the complexity of Figure 8 (as analyzed above).

5 Incremental Computation

Since our algorithms for solving STPPs (of the foregoing kinds) are based on solving MCCPs, they can also readily be made incremental. Incremental algorithms for flow problems can be found in [Hartline and Sharp, 2005], and incremental computation of *resource-envelopes* based on flow techniques can be found in [Kumar, 2003] and [Muscettola, 2004]. MCCPs can also be solved incrementally using techniques very similar to that in [Kumar, 2003] (as illustrated in Figures 5 and 6 of that paper). The main idea is the same—reverse the flow on edges that are not present in the new instance of the MCCP by first computing the amount by which such a reverse flow would violate the conservation constraints at each of the nodes; after the excess at each node is measured, a *maxflow* is staged between two auxiliary nodes S' and T' to regain conservation consistency; a subsequent optimization phase is carried out that respects this consistency—hence solving the new instance of the problem. The running time of this incremental procedure depends on "*how much flow remains to be augmented*"—and this characterizes the "difference" between the original problem instance and the new problem instance (see [Kumar, 2003] for more details).

It is worth noting that an incremental procedure for solving MCCPs finds immense use in iteratively building the optimal schedule for an STPP (as explained before). Further, incremental computation of solutions to STPPs is extremely important for an active management of planning, scheduling, and execution monitoring. In a *refinement planner*, for example, an STPP might characterize the various preferences and temporal constraints in a partial plan—the optimal solution to which might provide a useful heuristic value in the search for a good plan. Because refinement operators used to extend a partial plan include various kinds of incremental changes to it, an incremental algorithm for solving STPPs is likely to be much more useful than recomputation (from scratch) at each node in the search tree.

6 Conclusions and Future Work

We presented a fast polynomial-time algorithm for solving STPPs with *piecewise linear convex preference functions* and a *utilitarian* objective function. Our algorithm was motivated by the LP-duality relationships between STPPs and MCCPs, and could be made *incremental*—thereby bearing important implications in many real-life applications. In future work, we are interested in identifying richer classes of STPPs that can be analyzed and solved efficiently using LP-duality.

References

[Ahuja *et al*, 1993] Ahuja R. K., Magnanti T. L. and Orlin J. B. 1993. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, New Jersey.

[Dechter *et al*, 1991] Dechter R., Meiri I. and Pearl J. 1991. Temporal Constraint Networks. *Artificial Intelligence, Vol. 49.*

[Goldberg and Tarjan, 1989] Goldberg A. V. and Tarjan R. E. 1989. Finding Minimum-Cost Circulations by Cancelling Negative Cycles. *Journal of ACM 36(1989), 873-886.*

[Hartline and Sharp, 2005] Hartline J. and Sharp A. 2005. Incremental Flow. *www.cs.cornell.edu/w8/~jhartlin/incflow.pdf*

[Khatib *et al*, 2001] Khatib L., Morris P., Morris R. and Rossi F. 2001. Temporal Constraint Reasoning with Preferences. *Proceedings of IJCAI'2001.*

[Kumar, 2003] Kumar T. K. S. 2003. Incremental Computation of Resource-Envelopes in Producer-Consumer Models. *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP'2003).*

[Kumar, 2004] Kumar T. K. S. 2004. A Polynomial-Time Algorithm for Simple Temporal Problems with Piecewise Constant Domain Preference Functions. *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI'2004).*

[Kumar, 2005] Kumar T. K. S. 2005. On the Tractability of Restricted Disjunctive Temporal Problems. *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS'2005).*

[Morris *et al*, 2004] Morris P., Morris R., Khatib L., Ramakrishnan S. and Bachmann A. 2004. Strategies for Global Optimization of Temporal Preferences. *Proceedings of CP'2004.*

[Muscettola, 2004] Muscettola N. 2004. Incremental Maximum Flows for Fast Envelope Computation. *ICAPS'2004.*

[Orlin, 1988] Orlin J. B. 1988. A Faster Strongly Polynomial Minimum Cost Flow Algorithm. *STOC'1988.*

[Stergiou and Koubarakis, 1998] Stergiou K. and Koubarakis M. 1998. Backtracking Algorithms for Disjunctions of Temporal Constraints. *Proceedings of AAAI'1998.*

[Sultan, 1993] Sultan A. 1993. *Linear Programming: An Introduction with Applications.* San Diego: Academic Press, 1993.