Optimal Design of Consistent Simple Temporal Networks

Romeo Rizzi and Roberto Posenato

Computer Science Department
University of Verona
Verona, Italy
{romeo.rizzi,roberto.posenato}@univr.it

***Abstract*—Simple Temporal Networks (STNs) are used in many applications, as they provide a powerful and general tool for representing conjunctions of minimum and maximum distance constraints between pairs of temporal variables.**

During construction of an STN, it is possible that the network presents some constraint violations that need to be resolved. One way to solve such violations is to remove a minimal number of constraints, already shown to be an APX-hard problem. Another way is relaxing some constraints in different ways till violations are solved and choosing the best configuration according to one or more criteria.

In this paper, assuming that it is possible to increase any constraint bound of an STN paying a constraint-specific cost, we exhibit a polynomial-time algorithm that repairs an STN eliminating all constraint violations at minimum global cost.

***Index Terms*—Simple Temporal Networks; STNs design; optimization problem; consistency; minimum cost flow; linear programming.**

I. Introduction

In many areas of Artificial Intelligence (AI), including knowledge representation and planning & scheduling, the representation and management of quantitative temporal aspects is of crucial importance. Examples of possible quantitative temporal aspects are: constraints on the earliest start time and latest end one of activities, constraints over the minimum and maximum temporal distance between activities, etc. In many cases, these constraints can be represented as an instance of a *Simple Temporal Network (STN)* [1], a directed weighted graph where a node represents a time-point variable (timepoint), usually corresponding to the beginning/end of activities, and an arc represents a temporal distance constraint between a pair of timepoints, e.g., $x \overset{v}{\to} y$ stands for $y - x \le v$, where $v \in \mathbf{R}$. In an STN it is possible to represent constraints like $u \le y - x \le v$ (minimal and maximal temporal distance between x and y) as a pair of arcs, $x \overset{v}{\to} y$ and $y \overset{-u}{\to} x$. An STN is said to be *consistent* if it is possible to assign a real value to each timepoint so that all temporal constraints are satisfied. The consistency property can be verified by searching for negative cycles in the graph and it has been shown that the consistency check and the determination of the earliest/ latest value for each timepoint can be calculated in polynomial time [1].

During the development of an STN, new timepoints or constraints may be added or existing constraints may be restricted requiring to check the consistency frequently. In the case that an STN becomes (is) inconsistent, it is desirable to easily adjust constraints to make the network consistent again.

One method to solve the inconsistency is finding a minimum set of constraints that has to be removed in order to make the graph free of negative cycles. It is known that this problem is NP-hard [2], and in fact it is APX-hard because the Feedback Arc Set (FAS) problem can be reduced to it. Indeed, the feedback arc sets of a graph $G = (V, E)$ are precisely the sets of constraints whose removal makes consistent the STN made of G with weight -1 associated to each one of its arcs. FAS is APX-hard [3], and can be approximated to within $O(\log(n)\log(\log(n)))$ [4], [5]. In this paper, we denote by n and m the number of nodes and arcs of the graph of reference, respectively.

Therefore, in the literature, many alternatives to tackle the problem have been proposed. To the best of our knowledge, the most interesting ones are the following.

Planken et al. [6] presented a polynomial-time algorithm that maintains partial path consistency when new constraints are added or existing constraints are tightened. In such a way it is possible to verify if the set of new/modified constraints can be accepted or not. Even if the proposed system exhibits a practically good performance, it does not determine the minimal network and, therefore, the set of all possible values for each timepoint. Indeed, the algorithm has to be applied iteratively in order to successfully add/restrict constraints to a consistent STN.

Dasdan [2] presented the problem to solve constraint violations in an STN as a more general problem, called *Temporal and Spatial Difference Constraint Violations Problem*. After showing that the problem is NP-hard, Dasdan considered specific domain driven conditions on how to modify constraints and, then, gave a polynomial-time algorithm to assist a designer to plan a consistent network by an iterative debugging procedure. The proposed algorithm determines which constraints are violated supporting the ordering of violations using their inherent criticality or user-defined priority.

This paper introduces a different approach that allows one to solve the constraint violations problem in only one step producing, in polynomial time, a globally optimal network. Our viewpoint is that each violated constraint should be repaired (i.e., its bound can be augmented, not reduced) paying a linear cost that can be specified for each constraint. Within such setting, we exhibit a $\tilde{O}(nm)$ time algorithm that, given a not consistent network, returns a consistent one at minimum global reparation cost. Since each constraint may have a specific cost function, it is possible to customize in a detailed way the fixing phase and obtain the consistent network avoiding the iterative fixing procedures proposed by other approaches. Our framework also allows for more general convex cost functions to be dealt with, and we offer algorithms that are efficient both in theory and in practice.

A related line of research was fostered by Khatib et al. [7]. In [7], the authors introduced a generalization of STNs: *Simple Temporal Problems with Preferences (STPPs)*. In an STPP each constraint is represented by an arc, an interval, and a *preference* function. An assignment of reals to the timepoints is a feasible solution for an STPP if, for every arc, the difference between its two timepoint values falls within the arc interval. The preference function specifies a quality measure for each point in the interval and, therefore, the goal is to find a solution optimizing the *global preference value*. They provide polynomial-time algorithms for the case where the preference function of each constraint is convex. Though the formalism they proposed is rich, their global preference value essentially amounts however to the maximum of the preference values on the single constraints. This can be regarded as a bottleneck problem and can hence be solved by means of a binary search on the optimum solution value. Indeed, it is a property of this framework that, once the optimal solution value has been guessed, then checking whether a solution of that value exists reduces to a classical STN consistency problem. These results were improved in [7], where the solution to the bottleneck STPP was extended to semi-convex preference functions and with the observation that the sum of costs STTP (i.e., *utilitarian global temporal value*) can be formulated as a linear program when each preference function is a linear one.

Morris et al. [8] generalized the positive result on the utilitarian version of STPP to convex piece-wise linear functions. Moreover, proposed a characterization of different notions of global temporal value and the identification of tractable subclasses of STPPs.

Finally, Peintner et al. [9] proposed a greedy algorithm, and offered an experimental evaluation, for solving STPPs where the preference functions have not to be convex.

The remainder of the paper is organized as follows. in Sect. II we introduce some definitions and well-know results in order to state a new property of STNs. In Section III we introduce the problem to repair an inconsistent STN as a linear programming one, while in Section IV we present and discuss the combinatorial algorithm that solves the repair problem most effectively. In Section V we suggest a suitable approach to get practical solutions when each one of the arc cost functions is convex. Finally, Section VI concludes the paper.

II. Background and Notation

In this section, we introduce our basic terminology, some definitions and well-know results about circulations and conservative graphs; we also restate a characterization of the consistency property of STNs.

Our graphs are directed. For every node v, δ_v^+ denotes the set of arcs exiting v and δ_v^- denotes those entering. A graph is *eulerian* if $|\delta_v^+| = |\delta_v^-|$ at every node v. It is well known that eulerian graphs can be decomposed into directed circuits [10].

Given a function $c : E \mapsto \mathbf{R}$, we write c_e or $c(e)$ interchangeably, and when F is a set, $c(F)$ stands for $\sum_{e \in F} c_e$. Given a directed graph $G = (V, E)$, together with an upper bound capacity $c_e \in \mathbf{R}^+$, and a value $\ell_e \in \mathbf{R}$ specified for every arc e, a *circulation* in (G, c) is the specification of an amount of flow f_e to be sent along every arc e, subject to the conservation of flow constraints:

$$f(\delta_v^+) = f(\delta_v^-) \text{ for every node } v.$$

A circulation f is called *admissible* if $0 \leq f_e \leq c_e$ for every arc e. The value of a circulation f is $\sum_{e \in E} \ell_e f_e$.

The problem of finding a maximum value circulation can be solved by linear programming, since we optimize a linear function, and all constraints are linear. As for more efficient solutions, combinatorial algorithms are known for this problem (for a comprehensive survey see [11]). The main algorithmic approaches pursued in the literature and in applications are:

Minimum Mean Cycle Canceling: it is the first special case of cycle canceling (a general primal method) proven to involve a polynomial number of iterations. In [12], the authors proposed a simple polynomial-time algorithm which runs in $O(nm(\log n) \min\{\log(nC), m \log n\})$ time on a network of n vertices, m arcs, and arc costs of maximum absolute value C.

Successive Shortest Path and Capacity Scaling: in [13] a dual method is presented which can be viewed as the generalizations of the Ford-Fulkerson algorithm.

Cost Scaling: in [14] a primal-dual approach is presented which can be viewed as the generalization of the push-relabel algorithm [15].

Network Simplex: a specialized version of the linear programming simplex method.

In practice, the circulation problem is the classical problem that we have to solve when, once a maximum flow has been found, we seek to find a minimum cost one by iteratively redirecting the flow along cycles of negative costs in an auxiliary network. Indeed, even if the traditional approaches improve the flow considering one negative circuit at each iteration, it can be argued that these approaches are ultimately interested in finding a circulation of maximum value, that, in turn, can be regarded as a maximum-value eulerian subgraph in the auxiliary network, which decomposes into a packing of cycles.

A weighted graph (G, ℓ) is called *conservative* when $\ell(C) := \sum_{e \in C} \ell_e \geq 0$ for every directed circuit C of G.

The Bellman-Ford algorithm [16] can be used to produce in $O(nm)$ time:

- either a proof that (G, ℓ) is not conservative in the form of a negative circuit C in G;
- or a proof that (G, ℓ) is conservative in the form of a potential function $\pi : V \mapsto \mathbf{R}$ such that for every arc $e = (u, v)$ of G, its *reduced length* $\ell_e^\pi := \ell_e - \pi_v + \pi_u$ is non-negative.

An STN can be viewed as a weighted graph (G, ℓ) whose nodes are timepoints that must be placed on the real line and whose arcs express mutual constraints on the allocations of their endpoints. An STN is called *consistent* if there exists a scheduling $t : V \mapsto \mathbf{R}$ such that

$$t_v \leq t_u + \ell_{(u,v)} \quad \forall \text{ arc } (u, v) \text{ of } G. \tag{1}$$

Theorem 1 ([1], [16], [17]). *An STN (G, ℓ) is consistent if and only if it is conservative.*

Proof. If $t : V \mapsto \mathbf{R}$ is a scheduling for (G, ℓ), then each reduced length $\ell^t_{(u,v)} := \ell_{(u,v)} - t_v + t_u$ is non-negative by (1); thus (G, ℓ^t) is conservative and, since $\ell(C) = \ell^t(C)$ for every circuit C of G, then (G, ℓ) is conservative as well.

On the other side, if (G, ℓ) is conservative, then, using the Bellman-Ford algorithm, it is possible to determine a potential function $\pi : V \mapsto \mathbf{R}$ such that for every arc $e = (u, v)$ of G, $\pi_v \leq \pi_u + \ell_e$. Such π satisfies (1) and, therefore, (G, ℓ) is consistent. □

III. Problem Formulation

In this section, we formalize the problem to solve the inconsistency of an STN as a linear programming problem.

Let us consider a graph $G = (V, E)$ augmented by the following two functions:

- *length* $\ell : E \mapsto \mathbf{R}$, and,
- *cost* $c : E \mapsto \mathbf{R}^+$, where $\mathbf{R}^+$.

c_e represents the unitary cost for increasing ℓ_e. Here, the weighted graph (G, ℓ) may contain negative cycles.

We are interested at a minimum cost plan to transform an inconsistent STN (G, ℓ) into a consistent one by suitably relaxing some of its constraints (increasing the lengths of some of the arcs) and paying a cost specified by the cost function c. Our minimum cost network design problem can be expressed by means of the following linear programming problem:

$$\min \textstyle\sum_{(u,v)\in E} c_{(u,v)} \ell^+_{(u,v)}$$
$$\begin{cases} t_v \leq t_u + \ell_{(u,v)} + \ell^+_{(u,v)} & \forall \text{ arc } (u, v) \text{ of } G, \\ \ell^+ \geq \mathbf{0}, \end{cases} \tag{P1}$$

where $\ell^+ : E \mapsto \mathbf{R}^+$ is the decision variables vector specifying the length increment adopted for each arc.

Example 1. *Fig. 1 depicts a STN where cycles ACB, ACD, and $ADCB$ are negative. The associated minimum cost network design problem is given by the following linear programming problem:*

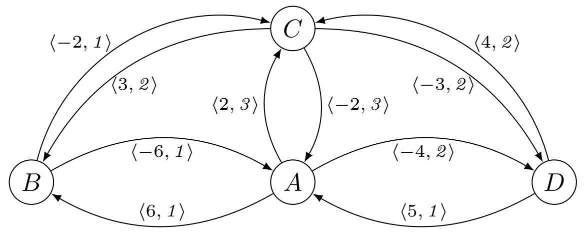

Fig. 1. A graph with two negative cycles. Each label has the form $\langle l, c\rangle$, where l is the length and c is the cost of the associated edge.

$$\min \ell^+_{(A,B)} + \ell^+_{(B,A)} + 3\,\ell^+_{(A,C)} + 3, \ell^+_{(C,A)} + 2\,\ell^+_{(A,D)}$$
$$+ \ell^+_{(D,A)} + \ell^+_{(B,C)} + 2\,\ell^+_{(C,B)} + 2\,\ell^+_{(C,D)} + 2\,\ell^+_{(D,C)}$$
$$\begin{cases} t_B \leq t_A + 6 + \ell^+_{(A,B)} \\ t_A \leq t_B - 6 + \ell^+_{(B,A)} \\ t_C \leq t_A + 2 + \ell^+_{(A,C)} \\ t_A \leq t_C - 2 + \ell^+_{(C,A)} \\ t_D \leq t_A - 4 + \ell^+_{(A,D)} \\ t_A \leq t_D + 5 + \ell^+_{(D,A)} \\ t_C \leq t_B - 2 + \ell^+_{(B,C)} \\ t_B \leq t_C + 3 + \ell^+_{(C,B)} \\ t_D \leq t_C - 3 + \ell^+_{(C,D)} \\ t_C \leq t_D + 4 + \ell^+_{(D,C)} \\ \ell^+ \geq \mathbf{0}. \end{cases} \tag{Ex1}$$

By well-known results about linear programming, and convex programming, this model will be amenable of polynomial-time solution algorithms even when the linear addenda $c_e \ell^+_e$ in the objective function get replaced by general convex functions and even when the whole objective function is a generic (smooth) convex function in the variables ℓ^+_e, $e \in E$. However, in the next section, we will provide extremely efficient combinatorial algorithms in order to more conveniently solve the problem.

IV. The Combinatorial Algorithm

In this section, we propose an efficient combinatorial algorithm that repairs, at a minimum global cost, an inconsistent STN where constraint bounds are allowed to be augmented paying a cost.

By Theorem 1, the problem to repair an inconsistent STN can be regarded as the request for a minimum cost plan to transform (G, ℓ) into a weighted graph with no negative cycles (conservative weighted graph) by suitably increasing the lengths of some arcs. This view leads to an alternative mathematical formulation of the Problem P1:

$$\min \textstyle\sum_{e\in E} c_e \ell^+_e$$
$$\begin{cases} \ell^+(C) \geq -\ell(C) & \forall \text{ directed cycle } C \text{ of } G \\ \ell^+ \geq \mathbf{0}. \end{cases} \tag{P2}$$

Example 2. *Considering the Problem (Ex1) presented in Example 1, the only meaningful constraints are those associated* *to the negative cycles in the graph. Therefore, Problem (Ex1) can be rewritten as the following linear programming problem:*

$$\begin{aligned}&\min \ell^+_{(A,B)} + \ell^+_{(B,A)} + 3\,\ell^+_{(A,C)} + 3, \ell^+_{(C,A)} + 2\,\ell^+_{(A,D)}\\ &\quad + \ell^+_{(D,A)} + \ell^+_{(B,C)} + 2\,\ell^+_{(C,B)} + 2\,\ell^+_{(C,D)} + 2\,\ell^+_{(D,C)}\\ &\begin{cases}\ell^+_{(A,C)} + \ell^+_{(C,B)} + \ell^+_{(B,A)} \geq 1 & \textit{(cycle ACB)}\\ \ell^+_{(C,A)} + \ell^+_{(A,D)} + \ell^+_{(D,C)} \geq 2 & \textit{(cycle CAD)}\\ \ell^+_{(A,D)} + \ell^+_{(D,C)} + \ell^+_{(C,B)} + \ell^+_{(B,A)} \geq 3 & \textit{(cycle ADCB)}\\ \ell^+ \geq \mathbf{0}.\end{cases}\end{aligned} \tag{Ex2}$$

Even though the above linear programming problem has an exponential number of constraints, the separation problem associated to the polytope of its feasible solutions essentially amounts to the detection of a negative cycle C in the weighted graph $(G, \ell + \ell^+)$, whence can be solved in polynomial time by means of the Bellman-Ford algorithm. It follows then by the milestone equivalence result [15] between separation and optimization that an optimal solution to Problem (P2) can be found in polynomial time by means of the ellipsoid algorithm. Indeed, it was this observation that suggested us to propose and pursue our STN design framework, safe in the knowledge that we were proposing a model amenable of efficient algorithmic solutions.

Now, we want to show how it is possible to solve the problem in a more effective way by means of a combinatorial algorithm.

As a starting point, we consider the dual of Problem (P2):

$$\begin{aligned}&\max \textstyle\sum_C -\ell(C)\lambda_C\\ &\begin{cases}\sum_{C\ni e}\lambda_C \leq c_e & \text{for every edge } e \in E,\\ \lambda \geq \mathbf{0},\end{cases}\end{aligned} \tag{P3}$$

where there is a variable λ_C for every directed circuit C of G. Problem (P3) essentially asks for packing negative cycles into (G, c) (clearly, if $\ell(C) > 0$, then $\lambda_C = 0$ in every optimal solution to Problem (P3)).

Example 3. *Considering the Problem (Ex2) presented in Example 2, its dual is the following linear programming problem:*

$$\begin{aligned}&\max \lambda_{ACB} + 2\,\lambda_{CAD} + 3\,\lambda_{ADCB}\\ &\begin{cases}\lambda_{ACB} + \lambda_{ADCB} \leq 1 & \textit{for the arc } (B,A)\\ \lambda_{ACB} \leq 3 & \textit{for the arc } (A,C)\\ \lambda_{CAD} \leq 3 & \textit{for the arc } (C,A)\\ \lambda_{CAD} + \lambda_{ADCB} \leq 2 & \textit{for the arc } (A,D)\\ \lambda_{ACB} + \lambda_{ADCB} \leq 2 & \textit{for the arc } (C,B)\\ \lambda_{CAD} + \lambda_{ADCB} \leq 2 & \textit{for the arc } (D,C)\\ \lambda \geq \mathbf{0}.\end{cases}\end{aligned} \tag{Ex3}$$

On this small example, it can be easily seen that the optimal solution is given by $\lambda_{ACB} = 0$, $\lambda_{CAD} = 1$, and $\lambda_{ADCB} = 1$. The value of the optimal solution is 5.

Unfortunately, due to the exponential number of constraints of the primal Problem (P2), the dual Problem (P3) is bound to have an exponential number of variables. However, we can faithfully express this problem by means of a more compact (only a polynomial number of variables) linear programming formulation by making the link with the classical concept of circulation. Indeed, as recalled in Section II, a circulation can be regarded as an eulerian subgraph, which then can be decomposed into a packing of cycles. Therefore, Problem (P3) can be regarded as an instance of the more classical and compact max circulation problem. The following two lemmas are meant to prove this formally.

Lemma 2. *From any feasible solution λ to Problem (P3), we can determine a circulation f of (G, c) with value $\sum_{e\in E} -\ell_e f_e = \sum_C -\ell(C)\lambda_C$.*

Proof. For every arc e of G, let $f_e := \sum_{C\ni e}\lambda_C$. Note that $f_e \geq 0$ since $\lambda \geq \mathbf{0}$ and does not exceed the capacity c_e by the feasibility of λ. Also, this f obeys the conservation of flow constraints since it is a linear combination of characteristic vectors of directed circuits. Finally,

$$\begin{aligned}\sum_{e\in E} -\ell_e f_e &= \sum_{e\in E} -\ell_e \sum_{C\ni e}\lambda_C\\ &= \sum_C \lambda_C \sum_{e\in C} -\ell_e = \sum_C -\ell(C)\lambda_C\,.\end{aligned}$$

□

Lemma 3. *From any circulation f of (G, c) we can determine a feasible solution λ to Problem (P3) with $\sum_C -\ell(C)\lambda_C = \sum_{e\in E} -\ell_e f_e$.*

Proof. Let d be a natural number such that $d \cdot f_e$ is integer for every e. Let $G(d \cdot f)$ be the multigraph obtained form G by taking $d \cdot f_e$ parallel copies of arc e, for every e. Since f obeys the conservation of flow constraints, then $G(d\cdot f)$ is an eulerian digraph and can be decomposed into directed circuits. Clearly, each one of these circuits maps down naturally to a unique circuit of G (when all copies of an edge are read just as that original edge of G). For every directed circuit C of G, let n_C the number of circuits in the decomposition of $G(d\cdot f)$ that map down to C. Then we define $\lambda_C := \frac{n_c}{d}$. □

This simple reduction already delivers an efficient purely combinatorial algorithm to solve Problem (P3), whence its consequences deserve to be explicitly pinned down in the following fact.

Fact 4. *An optimal solution to Problem (P3) can be obtained in $O(nm(\log n)\min\{\log(nC), m\log n\})$ time. Furthermore, when c is an integer valued function, then the optimal solution $\overline{\lambda}$ obtained is also integer.*

Proof. The proof follows from Lemma 2 and 3. The time bound refers to the algorithm in [12]. The integrality considerations follow from the well known total unimodularity properties of network flows. □

Now we concentrate on how to solve Problem (P2), the problem of true pertinence to us and core of the framework we intend to propose in this paper, assuming the optimal solution $\overline{\lambda}$ to Problem (P3) is given.

The following algorithm will do the job in $O(nm)$ time even in case $\overline{\lambda}$ is not integral.

Let $E^c := \{e \in E \mid \sum_{C \ni e} \overline{\lambda}_C = c_e\}$, $E^0 := \{e \in E \mid \sum_{C \ni e} \overline{\lambda}_C = 0\}$, and $E^{\neq} := E \setminus (E^c \cup E^0)$.
Starting from (G, ℓ), we determine a new weighted graph (G', ℓ), with $G' = (V, E')$, as follows.
First, we copy all $e \in E$ in E'; then, for every arc $e = (u, v) \in E^c \cup E^{\neq}$, we introduce in E' a new arc $e^R = (v, u)$ and extend ℓ by setting $\ell(e^R) = -\ell(e)$; next, we remove from E' all arcs in E^c.
In practice, for every arc $e \in E^c$ we replace e with one reversed both in direction and in length while for every arc $e \in E^{\neq}$ we add a parallel arc e^R with opposite direction and length, this time without removing e.

The *mate operator* $[\cdot]^R$ is defined as $[e]^R := e^R$ for every $e \in E \setminus E^0$ and $[e^R]^R := e$ for every $e^R \in E' \setminus E^0$, and can be extended to every edge set $F \subseteq (E \cup E') \setminus E^0$ by setting $[F]^R = \{[f]^R \ : \ f \in F\}$. Note that $\ell(F \cup [F]^R) = 0$ for every F. With this notation, $E' = E^0 \cup E^{\neq} \cup [E^{\neq}]^R \cup [E^c]^R$.

Lemma 5. *If $\overline{\lambda}$ is an optimal solution to Problem (P3), then (G', ℓ) is conservative.*

Proof. Assume that (G', ℓ) is not conservative and let C be a negative circuit in (G', ℓ). Also, represent $\overline{\lambda}$ as a circulation f of (G, c) with value $\sum_{e \in E} -\ell_e f_e = \sum_C -\ell(C)\lambda_C$. This is done by setting $f_e := \sum_{C \ni e} \lambda_C$ for every arc e of G, as more exaustively explained in the proof of Lemma 2.

Since $\overline{\lambda}$ is an optimal solution to Problem (P3), it follows that f is a maximum value circulation for (G, c) by Lemma 3. We will contradict this fact by showing how the negative circuit C can be used to produce a better circulation.

Since C is a directed circuit of G' then $C \subseteq E' = E^0 \cup E^{\neq} \cup [E^{\neq}]^R \cup [E^c]^R$. We suggest to modify f as follows:

$$\tilde{f}_e = \begin{cases} f_e + \varepsilon & \text{if } e \in C \cap (E^0 \cup E^{\neq}), \\ f_e - \varepsilon & \text{if } e^R \in C \cap [E^{\neq} \cup E^c]^R, \\ f_e & \text{otherwise.} \end{cases}$$

Clearly,

$$\begin{aligned} \sum_{e \in E} \ell_e \tilde{f}_e &= \sum_{e \in E} \ell_e f_e + \varepsilon \left(\sum_{e \in C \cap (E^0 \cup E^{\neq})} \ell_e \right) - \varepsilon \left(\sum_{e \in C \cap ([E^{\neq}]^R \cup [E^c]^R} \ell_{e^R} \right) \\ &= \sum_{e \in E} \ell_e f_e + \varepsilon \left(\sum_{e \in C \cap (E^0 \cup E^{\neq})} \ell_e \right) + \varepsilon \left(\sum_{e \in C \cap ([E^{\neq}]^R \cup [E^c]^R} \ell_e \right) \\ &= \sum_{e \in E} \ell_e f_e + \varepsilon \ell(C) < \sum_{e \in E} \ell_e f_e \, , \end{aligned}$$

as long as $\varepsilon > 0$, since $\ell(C) < 0$.

Also, for every $\varepsilon > 0$, $\tilde{f}$ is a circulation for G since we can think of it as obtained from the circulation f through the following three steps:

1) first, increase f_e by a same quantity ε on all the arcs e of the directed circuit C; this step could extend f to arcs not present in G;
2) then, decrease it back by ε on each arc $e \in C \setminus C \cap (E^0 \cup E^{\neq}) = C \cap ([E^{\neq}]^R \cup [E^c]^R)$; this step nullify the f on arcs nor present in G;
3) finally, decrease it by ε on each arc e such that $e^R \in C \cap [E^{\neq} \cup E^c]^R$, i.e., $e \in [C]^R \cap (E^{\neq} \cup E^c)$ as required by the definition of $\tilde{f}$.

The first step does clearly not spoil the property of f being a circulation since C is eularian. But the very same argument works for the next two steps combined since $(C \cap ([E^{\neq}]^R \cup [E^c]^R)) \cup ([C]^R \cap (E^{\neq} \cup E^c))$ is also eulerian as $[C \cap ([E^{\neq}]^R \cup [E^c]^R)]^R = [C]^R \cap (E^{\neq} \cup E^c)$ for the operator $[\cdot]^R$ is idempotent and $[A \cup B]^R = A^R \cup B^R$ and $[A \cap B]^R = A^R \cap B^R$.

Finally, by taking $\varepsilon := \overline{\varepsilon} := \min\{\min_{e \in C \cap (E^0 \cup E^{\neq})}(c_e - f_e), \min_{e^R \in [C]^R \cap (E^{\neq} \cup E^c)} f_e\}$, then the circulation $\tilde{f}$ is admissible. Note that $\overline{\varepsilon} > 0$. ☐

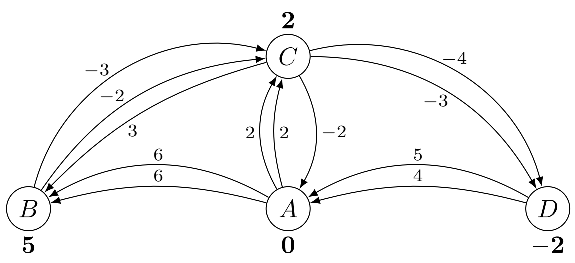

Fig. 2. The auxiliary graph (G', ℓ) based on the optimal solution $\lambda_{CAD} = \lambda_{ADCB} = 1$ and the instance graph depicted in Fig. 1. Bold labels associated to nodes represent values of the potential function π returned by Bellman-Ford algorithm with A as starting node.

Example 4. *Considering the dual problem presented in Example 3, we now build up the auxiliary weighted graph (G', ℓ) based on the optimal solution $\lambda_{CAD} = \lambda_{ADCB} = 1$ with value $\overline{\lambda} = 5$. Here, $E^c = \{(B, A), (A, D), (D, C)\}$, $E^{\neq} = \{(C, B), (C, A)\}$, and $E^0 := E \setminus (E^c \cup E^{\neq})$. The auxiliary weighted graph (G', ℓ) is represented in Fig. 2.*

Now, since (G', ℓ) is conservative, then we can can compute in $O(nm)$ time a potential function $\pi : V \mapsto \mathbf{R}$ such that for every edge of G', that is, for every edge $e = (u, v) \in E' = E^0 \cup E^{\neq} \cup [E^{\neq}]^R \cup [E^c]^R$, the reduced length $\ell_\pi(e) := \ell(e) - \pi_v + \pi_u$ is non-negative. In this way, for every edge $e \in E^0 \cup E^{\neq}$ we have that

$$\ell_\pi((u, v)) \geq 0 \, ,$$

whereas, for every edge $e = (u, v) \in E^c$ we have that

$$\begin{aligned} \ell_\pi((u, v)) &= \ell((u, v)) - \pi_v + \pi_u \\ &= -\ell((v, u)) - (-\pi_u) + (-\pi_v) \\ &= -\ell_\pi((v, u)) \\ &\leq 0 \, . \end{aligned}$$

Based on this, we propose considering the following solution to Problem (P2):

$$\ell^+(e) = \begin{cases} 0 & \text{for every } e \in E^0 \cup E^{\neq}, \\ -\ell_\pi(e) & \text{for every } e \in E^c. \end{cases}$$

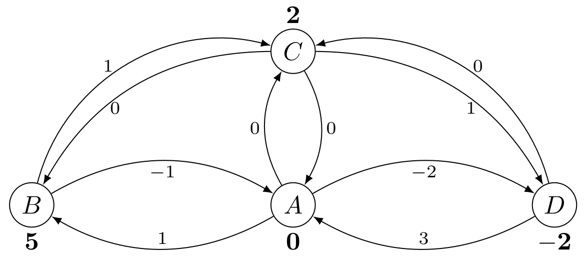

Fig. 3. The values of ℓ_π for the STN of Fig. 1 determined using the potential function depicted in Fig. 2.

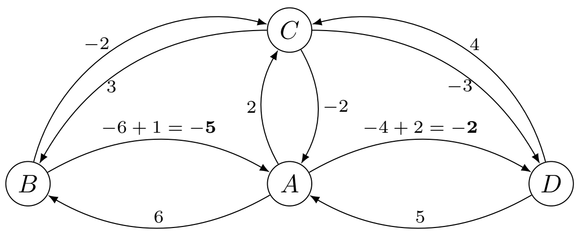

Fig. 4. A minimal-cost consistent STN determined from the original STN of Fig. 1. The bold labels represent the new length values of the associated edges.

This is a feasible solution to our original problem simply because $(\ell + \ell^+)_\pi = \ell_\pi + \ell^+ \geq \mathbf{0}$, and the feasibility of the resulting STN network is certified by the same potential computed above. In order to prove its optimality, we analyze its cost as follows:

$$\begin{aligned}\sum_{e\in E} c_e \ell^+(e) &= \sum_{e\in E^c} -c_e \ell_\pi(e)\\ &= -\sum_{e\in E^c}\left(\sum_{C\ni e}\overline{\lambda}_C\right)\ell_\pi(e)\\ &= -\sum_{C}\overline{\lambda}_C \sum_{e\in C\cap E^c}\ell_\pi(e)\\ &\leq -\sum_{C}\overline{\lambda}_C\sum_{e\in C}\ell_\pi(e)\\ &= -\sum_{e\in E}\overline{\lambda}_C \ell_\pi(C)\\ &= -\sum_{e\in E}\ell(C)\overline{\lambda}_C\,.\end{aligned}$$

Since we could rewrite the cost of the proposed solution to Problem (P2) as the optimal solution value of its dual Problem (P3), then the optimality follows by the weak duality theorem.

Example 5. *Fig. 3 depicts ℓ_π obtained applying the potential function π defined in the auxiliary graph (G,ℓ) of Fig. 2 to the original length ℓ function of the STN in Fig. 1, as previously described.*

The optimal solution to the primal problem (i.e., the minimal-cost consistent STN) is determined considering only the negative edges values in Fig. 3 and adding their absolute values to the corresponding original edge lengths in Fig. 1, as depicted in Fig. 4. Thus constraint (B,A) is relaxed by 1 and constraint (A,D) is relaxed by 2. The cost of these relaxations is $1\cdot 1 + 2\cdot 2 = 5$ and is indeed the same as the cost of the solution to the dual problem, whence optimality is confirmed and we have produced the optimal solution to our original problem.

The most expensive step in computing this optimal solution to Problem (P2) along the above lines, once the $\overline{\lambda}$ are given, is the computation of the potential function π, which can be done in $O(nm)$ by a standard Bellman-Ford algorithm.

V. Convex Costs

Simple reductions from FAS, like those mentioned in the introduction, already show that the problem becomes APX-hard as soon as the costs are non convex. We have already observed in Section III that our model must be in class P when the cost function is convex. Here we want to indicate more effective approaches for the case when the global cost function is of the form $\sum_{e\in E} c_e(\ell^+_e)$, with each $c_e(\cdot)$ convex. Assume first $c_e(\cdot)$ is a convex piecewise linear function: in this case $c_e = \sum_{i=1}^k f_{\overline{x}_i,m_i}$, where

$$f_{\overline{x}_i,m_i}(x) = \begin{cases} 0 & \forall\ 0\leq x\leq \overline{x}_i,\\ m_i(x-\overline{x}_i) & \forall\ x\geq \overline{x}_i.\end{cases}$$

Note that the arc $e=(u,v)$ can then be represented by k arcs (where k was the number of pieces) $e_j = (u,v)$ with $\ell(e_j) = \ell(e) + \sum_{i\leq j}\overline{x}_i$ and $c(e_j) = m_j$, for $j = 1,\dots,k$. Notice that the cost function associated to each one of these k arcs is now linear so that we are back to the model investigated into the previous section.

The same trick essentially works in order to obtain an effective FPTAS for the more general case when all the costs $c_e(\cdot)$ are semiderivable convex functions, since, for any $\varepsilon > 0$, and for each $B > 0$, each such function can be approximated by a convex piecewise fuction $p_e(\cdot)$ with $c_e(x) \leq p_e(x) \leq (1+\varepsilon)c_e(x)$ for each $x\geq 0$ with $x\leq \overline{X}$, and made of $O\left(\frac{1}{\varepsilon}\log B\right)$ pieces, where $B\leq c_e(\overline{X})$.

VI. Conclusions

In the literature, there are different frameworks and approaches aimed to solve the constraint violations in an inconsistent STN. Some authors proposed semi-automated protocols in which a designer interacts with a solving algorithm proposing a set of constraints to be removed/relaxed.

In this paper, we propose a novel framework where the problem of eliminating constraint violations is reformulated as an optimization problem where violated constraints can be relaxed paying a cost. In this way, we show that the problem is tractable and can be solved by linear programming. Therefore, the framework allows a designer to solve the inconsistency of an STN in only one step.

Moreover, we offer an analysis of the mathematical structure of the model that leads to effective (time complexity $\tilde{O}(nm)$)

combinatorial algorithms that allow the resolution of bigger instances.

As for future work, we aim to evaluate the practical performances of the proposed algorithms and the effectiveness and applicability of the framework in some application domains.

References

[1] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, vol. 49, no. 1-3, pp. 61–95, 1991.

[2] A. Dasdan, "Provably efficient algorithms for resolving temporal and spatial difference constraint violations," *ACM Tran. on Design Aut.of Elect. Sys. (TODAES)*, vol. 14, no. 1, p. 8, 2009.

[3] V. Kann, "On the approximability of NP-complete optimization problems," Ph.D. dissertation, Royal Inst. of Tech. Stockholm, 1992.

[4] G. Even, J. S. Naor, B. Schieber, and M. Sudan, "Approximating minimum feedback sets and multicuts in directed graphs," *Algorithmica*, vol. 20, no. 2, pp. 151–174, 1998.

[5] E. Speckenmeyer, "On feedback problems in digraphs," in *Graph-Theoretic Concepts in Computer Science: WG'89*, vol. 15. Springer Verlag, 1990, p. 218.

[6] L. Planken, M. de Weerdt, and N. Yorke-Smith, "Incrementally solving STNs by enforcing partial path consistency," in *Proc. of ICAPS*, 2010, pp. 129–136.

[7] L. Khatib, P. Morris, R. Morris, and F. Rossi, "Temporal constraint reasoning with preferences," in *Proc. of the 17th int. joint conf. on Artif. Intelligence*, vol. 1. Morgan Kaufmann, 2001, pp. 322–327.

[8] P. Morris, R. Morris, L. Khatib, S. Ramakrishnan, and A. Bachmann, "Strategies for global optimization of temporal preferences," in *Principles and Practice of Constraint Programming – CP 2004*, ser. LNCS, M. Wallace, Ed. Springer, 2004, vol. 3258, pp. 408–422. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30201-8_31

[9] B. Peintner and M. E. Pollack, "Any time, complete algorithm for finding utilitarian optimal solutions to STPPs," in *Proc. of the 20th nat. conf. on Artificial Intelligence*, ser. AAAI'05, vol. 1. AAAI Press, 2005, pp. 443–448. [Online]. Available: http://dl.acm.org/citation.cfm?id=1619332.1619404

[10] D. B. West, *Introduction to graph theory*, 2nd ed. Prentice, 2001.

[11] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications*. Prentice H., 1993.

[12] A. V. Goldberg and R. E. Tarjan, "Finding minimum-cost circulations by canceling negative cycles," *J. of the ACM (JACM)*, vol. 36, no. 4, pp. 873–886, 1989.

[13] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. of the ACM (JACM)*, vol. 19, no. 2, pp. 248–264, 1972.

[14] A. V. Goldberg and R. E. Tarjan, "Finding minimum-cost circulations by successive approximation," *Math. of Oper. Res.*, vol. 15, no. 3, pp. 430–466, 1990.

[15] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, 2nd ed., ser. Algorithms and Combinatorics. Springer, 1993, vol. 2.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 2001.

[17] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, 1958.