Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling

Temporal Flexibility Revisited: Maximizing Flexibility by Computing Bipartite Matchings

Simon Mountakis and **Tomas Klos** and **Cees Witteveen**
Algorithmics group, Department of Computer and Software Technology,
Delft University of Technology, NL-2628 CD, Delft, The Netherlands

Abstract

We discuss two flexibility metrics for Simple Temporal Networks (STNs): the so-called naive flexibility metric based on the difference between earliest and latest starting times of temporal variables, and a recently proposed concurrent flexibility metric. We establish an interesting connection between the computation of these flexibility metrics and properties of the minimal distance matrix D_S of an STN S: the concurrent flexibility metric can be computed by finding a *minimum* weight matching of a weighted bipartite graph completely specified by D_S, while the naive flexibility metric corresponds to computing a *maximum* weight matching in the same graph. From a practical point of view this correspondence offers an advantage: instead of using an $O(n^5)$ LP-based approach, reducing the problem to a matching problem we derive an $O(n^3)$ algorithm for computing the concurrent flexibility metric.

Introduction

Simple Temporal Networks (STNs) (Dechter, Meiri, and Pearl 1991; Dechter 2003), offer a convenient framework for modelling temporal aspects of scheduling problems, distinguishing between a set T of temporal variables and a set C of linear difference constraints between them. STNs have been used quite extensively in a number of different domains where automated planning and scheduling are key issues, such as manufacturing, maintenance, unmanned spacecraft and robotics. One of the attractive properties of STNs is that various important search and decision problems can be solved quite efficiently. For example, deciding whether an STN admits a schedule, finding such a schedule, as well as finding a complete schedule extending a partial one are all problems that can be solved in low-polynomial time.

Constructing one single schedule, however, for an STN is often not sufficient. In quite some applications one has to react immediately to possible disturbances without having the time to recompute a new schedule. In such a case, instead of one single schedule, we would like to have a *set of schedules* available from which we can select in constant time a suitable alternative. The availability of such a set of alternative schedules implies that the scheduler could offer some amount of *flexibility* in scheduling: instead of determining a fixed value for a temporal variable, a flexible schedule offers for each temporal variable a set of values to choose from. Clearly, such a flexibility is determined by properties of the set C of constraints of an STN. Therefore, one would like to specify a *flexibility metric* for STNs based on the properties of this set C. One of the first proposals (Policella et al. 2004; Pollack and Tsamardinos 2005) for such a flexibility metric has been based on considering a (non-empty) interval $[est(t), lst(t)]$ for each temporal variable $t \in T$ and defining the flexibility of an STN S as $flex(S) = \Sigma_{t \in T}(lst(t) - est(t))$. Here, $est(t)$ refers to the earliest time t can be executed in any feasible schedule for S and $lst(t)$ to its latest time. We will discuss this idea in more detail below.

As has been pointed out recently, this so-called *naive flexibility metric flex* has some serious shortcomings. Its main disadvantage is that, in general, it is far too optimistic with respect to the available flexibility in an STN. The reason is that the flexibility intervals $[est(t), lst(t)]$ assigned to temporal variables t, in general, are not independent: The flexibility of two temporal variables together might be substantially smaller than the sum of their individual flexibilities (Wilson et al. 2013; 2014). As an alternative, these authors proposed to assign flexibility intervals to temporal variables such that these intervals are *independent*: for every $t \in T$, its flexibility interval $[t^-, t^+]$ enables one to make a choice for a time point in the interval without any consequence for the validity of the choices for other temporal variables. In their paper they use a Linear Programming (LP) based approach to compute this so-called *concurrent flexibility metric* $flex^*$ to show that it can be computed in polynomial time. They do not, however, mention exact upper bounds for the time complexity of computing this metric.

In this paper, we start by presenting a geometric interpretation of the concurrent and the naive flexibility metric. Then, by using duality theory, we show that there exists a nice correspondence between (a slightly modified) minimal distance matrix D_S associated with an STN S and both flexibility metrics. If this matrix D_S is conceived as the specification of weights on the edges of a (complete) weighted bipartite graph G_S, then the value of the concurrent flexibility metric equals the costs of a minimum matching in G_S, while the naive flexibility metric corresponds to the costs of a maximum matching in G_S. Using this correspondence, we are able to deduce an efficient $O(n^3)$ algorithm for determining both the concurrent and naive flexibility of an STN.

Preliminaries

An STN is a pair $S = (T, C)$, where $T = \{t_0, t_1, \ldots, t_{n-1}\}$ is a set of temporal variables and C is a finite set of binary constraints on T, each constraint having the form[1] $t_j - t_i \leq c_{ij}$, for some real number c_{ij}. A solution to S is a *schedule* for S, that is, a function $\sigma : T \to \mathbb{R}$, assigning a real value (time-point) to each temporal variable in T such that all constraints in C are satisfied. If such a schedule exists, we say that the STN is *consistent*.[2] The time-point $t_0 \in T$, also denoted by z, is used to be able to express absolute time constraints. It represents a fixed reference point on the timeline, and is assigned the value 0.

Example 1. *We consider two STNs $S_1 = (T_1, C_1)$ and $S_2 = (T_2, C_2)$ where $T_1 = T_2 = \{z, t_1, t_2, t_3\}$ and*

$$\begin{aligned} C_1 = &\quad \{0 \leq t_i - z \leq 50 \mid i = 1, 2, 3\}, \\ C_2 = &\quad C_1 \cup \{0 \leq t_i - t_j \leq \infty \mid 1 \leq j < i \leq 3\}. \end{aligned}$$

Every assignment $\sigma_1(t_i) = v_i \in [0, 50]$ is a schedule for S_1. On the other hand, a schedule σ_2 for S_2 has to satisfy $0 \leq \sigma_2(t_1) \leq \sigma_2(t_2) \leq \sigma_2(t_3) \leq 50$.

Using a shortest path interpretation of an STN $S = (T, C)$ (Dechter 2003), there is an efficient method to find all tightest constraints implied by C, by searching for all shortest paths between the time points in T using e.g. Floyd and Warshall's all-pairs shortest paths algorithm (Floyd 1962). The $n \times n$ *minimum distance matrix* D_S contains for every pair of time-point variables t_i and t_j the length $D_S[i, j]$ of the *shortest path* from t_i to t_j in the distance graph. In particular, the latest starting time $lst(t_i)$ and earliest starting time $est(t_i)$ can be obtained directly from the first row and first column of D_S: $lst(t_i) = D[0, i]$ and $est(t_i) = -D[i, 0]$. Given an STN S, this matrix D_S can be computed in low-order polynomial ($O(n^3)$) time (Dechter 2003). Here, n denotes the number of temporal variables.

The following proposition is a restatement of Theorem 3.3 in (Dechter, Meiri, and Pearl 1991) and essentially states that every STN is decomposable via its distance graph D_S:

Proposition 1. *Let $S = (T, C)$ be an STN and D_S its distance matrix. For $i = 1, \ldots, n - 1$, let $est(t_i) = -D_S[i, 0]$ and $lst(t_i) = D_S[0, i]$. Then, for every schedule σ for S, and every $t \in T$, it holds that $\sigma(t) \in [est(t), lst(t)]$. Moreover, given any $t \in T$ and $v \in [est(t), lst(t)]$, there exists a schedule σ for S such that $\sigma(t) = v$.*[3]

Finally, we assume that for each $t \in T$ there is a finite interval for scheduling it. In particular this means that for each STN $S = (T, C)$ we assume that for all $t \in T$, $t \geq z$ holds and there exists a finite constant (horizon) h_S such that for all $t \in T$ it holds that $t \leq h_S$. This avoids the use of unbounded time intervals such as $-\infty \leq t_i - t_j \leq \infty$.

[1]If both $t_j - t_i \leq c_{ij}$ and $t_i - t_j \leq c_{ji}$ are specified, we will also use the more compact notation $-c_{ji} \leq t_j - t_i \leq c_{ij}$.

[2]Without loss of generality, in the remainder of the paper, we assume that an STN to be consistent. Note that consistency of an STN can be determined in low-order polynomial time (Dechter, Meiri, and Pearl 1991).

[3]In fact it can be shown that there exists a simple backtrack-free polynomial-time algorithm to construct such a schedule σ for S.

The naive and the concurrent flexibility metric

Intuitively, a flexibility metric should indicate our freedom of choice in choosing values for temporal variables such that the constraints are satisfied. In the naive flexibility metric one has chosen, for each variable $t \in T$, the interval $[est(t), lst(t)]$ to indicate this freedom of choice.[4] The naive flexibility of an STN S then is defined as $flex(S) = \sum_{t \in T} (lst(t) - est(t))$. Obviously, since D_S is computable in $O(n^3)$ time, $flex(S)$ can be computed in $O(n^3)$ time.

As we already mentioned, using $flex(S)$ has a serious disadvantage, due to *dependencies* that might exist between temporal variables. We give a simple example to illustrate these dependencies and their consequences.

Example 2. *Consider the STNs presented in Example 1. For both STNs it holds that $est(t_i) = 0$ and $lst(t_i) = 50$, for $i = 1, 2, 3$. Hence, $flex(S_1) = flex(S_2) = 150$. Intuitively, however, due to the ordering constraints between t_1, t_2 and t_3, the flexibility of S_2 should be much lower: giving a flexibility of f_1 to t_1 results in a flexibility of $f_2 = v_2 - f_1$ for t_2, where $50 \geq v_2 \geq f_1$, and a flexibility of $50 - v_2$ for f_3. The sum of these flexibilities equals $f_1 + (v_2 - f_1) + (50 - v_2) = 50$, one-third of the value of the naive flexibility for S_2.*

One way to see how these dependencies are neglected in computing the naive flexibility $flex(S)$ is to give a geometric interpretation of this metric.

The solution space of a set of linear constraints is a polytope. In (our) case of a consistent STN it is a bounded polytope. The set of intervals $\{[est(t_i), lst(t_i)]\}_{i=1}^{n}$ can be thought of as determining the smallest (hyper)cube containing this polytope. Clearly, if the polytope itself is not a hypercube, there are points in the hypercube that do not belong to the solution space of the STN. Exactly in these cases $flex(S)$ will overestimate the amount of flexibility of S.

Example 3. *Consider again the STNs S_1 and S_2 as presented in Example 1. The polytope generated by C_1 is the large (hyper)cube depicted in Figure 1. This hypercube equals the smallest hypercube containing the polytope, hence $flex(S_1) = 50 + 50 + 50$ is the real flexibility of S_1. The polytope generated by C_2 is the shaded subspace depicted in Figure 1. The smallest hypercube containing it, however, equals the hypercube specified for S_1. Hence, $flex(S_2) = flex(S_1) = 150$: an overestimation of the flexibility.*

A new flexibility metric for STNs (Wilson et al. 2013) has been introduced to overcome the disadvantages of $flex(S)$. Instead of determining the smallest *outer bounding box* (hypercube) containing the polyhedron generated by a set of constraints C, they determine the largest *inner bounding box* contained in this polyhedron (see below for a computation).

[4]One of the reasons for this choice is that taking any value $v \in [est(t), lst(t)]$ for t allows us, by decomposability (Proposition 1), to extend the partial schedule $\sigma(t) = v$ to a total schedule σ for an STN S.

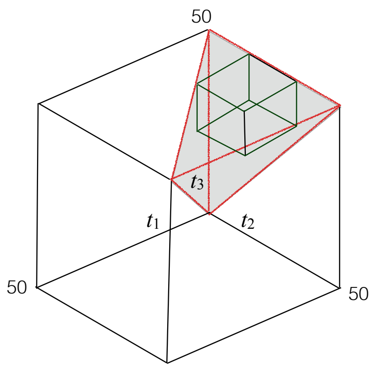

Figure 1: The polytope P_1 generated by S_1 is a cube with edges of size 50; the polytope P_2 generated by S_2 is the shaded subspace of P_1. The inner bounding box of P_2 is the small green cube in the shaded area. The inner bounding box of S_1 and also the outer bounding box of S_1 and S_2 equal P_1.

An obvious advantage of such an inner bounding box is that every point in the box belongs to the polyhedron (i.e. solution space) containing it.

Example 4. *Consider again the STN S_2. A largest inner bounding box contained in the polytope determined by C_2 is the small box contained in the shaded area in Figure 1. All its points are contained in the solution space of S_2.*

To construct a flexibility metric based on the largest inner bounding box we have to make sure that for every constraint $t_j - t_i \leq c_{i,j}$ the flexibility intervals $[t_i^-, t_i^+]$ and $[t_j^-, t_j^+]$ associated with t_i and t_j are independent, i.e., every value v_i and v_j for t_i and t_j, respectively, chosen in these intervals has to satisfy the constraint $v_j - v_i \leq c_{i,j}$. This is exactly the case if t_j assumes the maximum value and t_i assumes the minimum value in its interval, i.e., we have to ensure that $t_j^+ - t_i^- \leq c_{i,j}$. Therefore, the value $\mathit{flex}^*(S)$ for the concurrent flexibility for S can be found by solving the following LP (see (Wilson et al. 2014)):

$$\begin{aligned} \text{maximize} \quad & \sum_{t_i \in T} (t_i^+ - t_i^-) && \text{(LP1)} \\ \text{subject to} \quad & t_i^- \leq t_i^+ \quad , && \forall\, t_i \in T \\ & t_j^+ - t_i^- \leq c_{i,j} \, , && \forall\, (t_j - t_i \leq c_{i,j}) \in C \end{aligned}$$

Here, $\mathit{flex}^*(S)$ indicates the maximum flexibility that can be obtained when all flexibility intervals of temporal variables are independent from each other and values in these intervals can be chosen concurrently.

Example 5. LP1 *will return* $\mathit{flex}^*(S_1) = 150$*, but* $\mathit{flex}^*(S_2) = 50$*, showing that here the concurrent flexibility metric corresponds to our intuition.*

Concurrent flexibility by minimum matching

In this section we show that computing the maximum concurrent flexibility is identical to computing a perfect minimum weight matching of a weighted bipartite graph G_S completely specified by D_S. On the other hand, computing the naive flexibility corresponds to computing a maximum matching in this graph G_S.

To establish these results, first, we specify an LP equivalent to LP1 using the minimum distance matrix D_S instead of the original set C of constraints. Then we show that the entries of D_S can be used to compute a least upper bound on $\mathit{flex}^*(S)$. Finally, by using duality theory in Linear Programming and some adaptations, we show that this least upper bound is a maximum matching in a bipartite graph specified by D_S and in fact equals $\mathit{flex}^*(S)$.

Remember that the entries $d(i,j)$ of the minimum distance matrix D_S of an STN S specify the upper bound of the strongest difference constraint between the temporal variables t_j and t_i: $t_j - t_i \leq d(i,j)$. Hence, we can replace the specification LP1 by the following equivalent LP, since both LP's specify the same solution space:

$$\begin{aligned} \text{maximize} \quad & \sum_{t_i \in T} (t_i^+ - t_i^-) && \text{(LP2)} \\ \text{subject to} \quad & t_i^- \leq t_i^+ \quad , && \forall\, t_i \in T \\ & t_j^+ - t_i^- \leq d(i,j) \quad , && \forall\, t_i \neq t_j \in T \end{aligned}$$

To remove the condition $\forall\, t_i \neq t_j \in T$, it is sufficient to realize that $t_i^+ - t_i^- = (t_i^+ - t_0^-) + (t_0^+ - t_i^-) \leq d(0,i) + d(i,0) = \mathit{lst}(t_i) - \mathit{est}(t_i)$, since $t_0^- = t_0^+ = 0$. Hence, setting $d(i,i) = d(0,i) + d(i,0)$ for all $t_i \in T$, we change LP2 to the following equivalent LP:

$$\begin{aligned} \text{maximize} \quad & \sum_{t_i \in T} (t_i^+ - t_i^-) && \text{(LP3)} \\ \text{subject to} \quad & t_i^- \leq t_i^+ \quad , && \forall\, t_i \in T \\ & t_j^+ - t_i^- \leq d(i,j) \quad , && \forall\, t_i, t_j \in T \end{aligned}$$

We will denote this modified distance matrix D_S, where $d(i,i) = \mathit{lst}(t_i) - \mathit{est}(t_i)$, by D_S^*.

Now consider the sum $\sum_{i=1}^n (t_i^+ - t_i^-)$ we want to maximize. This sum can be rewritten as $\sum_{i=1}^n (t_i^+ - t_i^-) = \sum_{i=1}^n (t_i^+ - t_{\pi(i)}^-)$ for an arbitrary permutation π of $(1, 2, \ldots, n)$. Since $t_i^+ - t_{\pi(i)}^- \leq d(\pi(i), i)$, we can derive an upper bound on this sum for every permutation π: $\sum_{i=1}^n (t_i^+ - t_{\pi(i)}^-) \leq \sum_{i=1}^n d(\pi(i), i)$. In particular, there are permutations π^* such that for all permutations π it holds that $\sum_{i=1}^n d(\pi^*(i), i) \leq \sum_{i=1}^n d(\pi(i), i)$. Such a (smallest) permutation π^* specifies a least upper bound on $\mathit{flex}^*(S) = \max \sum_{i=1}^n (t_i^+ - t_i^-)$.

There is an efficient way to compute such a smallest permutation π^* by obtaining a *minimum weighted matching* between $T^+ = \{t_1^+, t_2^+, \ldots, t_n^+\}$ and $T^- = \{t_1^-, t_2^-, \ldots, t_n^-\}$: Consider the weighted complete graph $G_S = (V, E, w)$ where $V = T^+ \cup T^-$, the edges $\{t_i^+, t_j^-\} \in E$ for $i, j = 1, \ldots n$, and for every edge $\{t_i^+, t_j^-\}$ its weight $w_{i,j} = d_{j,i}$. Note that G_S is completely specified by D_S^*. By Hall's theorem, this graph has a minimum weighted perfect matching, that is a set $M \subseteq E$ of n edges covering T^+ and T^- such that the sum of these edges is minimum. Clearly, such a minimum weighted matching determines a permutation π^*

such that $\sum_{i=1}^{n} d(\pi^*(i), i)$ is minimum. Since it holds that $\sum_{t_i \in T}(t_i^+ - t_i^-) \leq \sum_{i=1}^{n} d(\pi^*(i), i)$, the cost $c(M)$ of such a minimum matching provides an upper bound for *flex*$^*(S)$.

Actually, by applying simple LP-duality theory, we can do better and show that there exists an *exact* correspondence between a minimum matching in D_S^* and the maximum concurrent flexibility *flex*$^*(S)$:

Proposition 2. *Let $S = (T, C)$ be a consistent STN having a minimum distance matrix D_S. Then the concurrent flexibility flex$^*(S)$ of S equals the cost $c(M)$ of a minimum matching M of the complete weighted graph G_S specified by D_S^*.*

Proof. The most elegant way to prove this result is making use of duality in linear programming. The dual of LP3 is the following LP:

$$\begin{aligned}
\text{minimize} \quad & \sum_{1 \leq i,j \leq n} d(i,j)\, y_{j,i} && \text{(LP4)} \\
\text{subject to} \quad & \sum_{j=1}^{n} y_{i,j} = 1 + y_{0,i} && , i \in \{1, 2, \ldots n\} \\
& \sum_{i=1}^{n} y_{i,j} = 1 + y_{0,j} && , j \in \{1, 2, \ldots n\} \\
& y_{i,j} \geq 0 && , i, j \in \{0, 1, \ldots, n\}
\end{aligned}$$

Since the original LP (LP3) has an optimal solution, by strong duality, the objective values of optimal solutions of both LP's coincide. The constraint matrix associated with both LP's is total unimodular, hence, LP4 has integral optimal solutions. This dual LP4 can be interpreted as follows: Let G_S be a complete bipartite graph having two sets of nodes $A = \{1, \ldots, n\}$ and $B = \{1, \ldots n\}$, representing the rows and columns of D_S^*. Let D_S^* specify the weights $d(j, i)$ of the edges $e = (i, j)$ of G_S. Then, in case $y_{0,i} = 0$ for all $i = 1, 2, \ldots n$), it is not difficult to see that a minimizer for LP4 specifies a minimum weight matching M in G_S. By strong duality, the cost $c(M)$ of M corresponds to *flex*$^*(S)$.

Therefore, the only part left to prove is to show that the assumption $y_{0,i} = 0$ for all $i = 1, 2, \ldots n$ does not affect the *value* of the solutions to LP4:

Claim Whenever LP4 has an optimal integral solution, there also exists an integral solution of LP4 with at most the same cost such that $y_{0,i} = 0$, for all $i = 1, 2, \ldots, n$.

Proof of the Claim Suppose there exists an integral solution $\underline{y}^* = (y_{i,j}^*)$ for LP4 where $y_{0,i}^* > 0$ for some $1 \leq i \leq n$. Then there exist $j, k \in \{1, \ldots, n\}$, $j \neq i$ and $k \neq i$, such that $y_{i,j}^* = 1$ and $y_{k,i}^* = 1$.[5] This condition violates the matching conditions. But then, since $d(j, k) \leq d(j, i) + d(i, k)$, there exists a solution $\mathbf{y} = (y_{i,j})$ such that $c(\mathbf{y}) \leq c(\mathbf{y}^*)$ where $y_{i,j} = y_{k,i} = 0$ and $y_{k,j} = 1$, and $y_{0,i} = y_{0,i}^* - 2 + y_{k,j}^*$. Hence, there exists a solution $\mathbf{y}$ with at most the same cost as $\mathbf{y}^*$ such that $0 \leq y_{0,i} < y_{0,i}^*$. Iterating this procedure, we conclude that there exists a solution such that $y_{0,i} = 0$ as well, with cost not exceeding the cost of the original solution. This procedure can be repeated until we have obtained an optimal solution where it holds that $y_{0,i} = 0$ for all i, i.e., a perfect minimum weight matching, thereby proving the claim. $\square$

[5]Wlog. we may assume $y_{i,j}^* \in \{0, 1\}$, $1 \leq i, j \leq n$: If $y_{i,j}^* > 1$ then there exists a solution $\mathbf{y}'$ with $c(\mathbf{y}') \leq c(\mathbf{y}^*)$ s.t. $y'_{i,j} = 1$ and $y'_{0,i} = y_{0,i}^* - (y_{i,j}^* - 1)$.

A maximum matching M for D_S^* can be determined immediately: Note that the value of any matching M' realised by a permutation π' satisfies $\sum_{i=1}^{n} d(i, \pi'(i)) \leq \sum_{i=1}^{n} d(i, 0) + d(0, \pi'(i)) = \sum_{i=1}^{n} d(i, 0) + d(0, i) = \sum_{i=1}^{n} lst(t_i) - est(t_i) = flex(S)$. Since $d(i, i) = lst(t_i) - est(t_i)$, the matching $M = \{(i, i) : i = 1, 2, \ldots, n\}$ realized by $\pi(i) = i$, is a maximum matching for D_S^*:

Proposition 3. *Let $S = (T, C)$ be a consistent STN having a minimum distance matrix D_S. Then the* naive flexibility *flex(S) of S equals the cost $c(M)$ of a* maximum matching *M of the weighted graph specified by D_S^*.*

From $O(n^5)$ to $O(n^3)$ to compute flexibility

The complexity of computing the concurrent flexibility of an STN by an LP method depends on the exact method used for the LP solver. Currently, the best (interior-point based) LP-solvers have a complexity of $O(n^3 L)$ (Potra and Wright 2000) where n is the number of unknowns to solve for (the dimension of the vector x) and L is the input complexity, i.e., the bit length of the input description. This means that given an STN $S = (T, C)$, the currently best LP-solvers could need $O(n^3 m)$-time to find the concurrent flexibility *flex*$^*(S)$, where $n = |T|$ and $m = |C|$. Since $m = O(n^2)$, we could expect a worst-case running time $O(n^5)$ to compute *flex*$^*(S)$.

The correspondence (flexibility $\equiv$ matching) obtained in the previous section allows us to present a better upper bound of the running time needed to compute the maximum concurrent flexibility. For, we know that given an STN S its minimal distance matrix D_S can be computed in $O(n^3)$ time. A minimum matching algorithm based on the specification of D_S also requires $O(n^2 \log n + n \times n^2) = O(n^3)$-time (Fredman and Tarjan 1987). Hence, the total computation time for determining a minimum matching is $O(n^3)$-time. Note that computing the naive flexibility also requires an $O(n^3)$ computation of earliest and latest times for the temporal variables, both of which can be obtained via D_S.

Hence, in conclusion, we obtain the following result:

Proposition 4. *The concurrent flexibility as well as the naive flexibility of an STN S can be computed in $O(n^3)$-time.*

Conclusion

We established a connection between the computation of two flexibility metrics and properties of the minimal distance matrix D_S of an STN S: computing the concurrent flexibility metric is identical to computing a minimum weight matching of a weighted bipartite graph completely specified by the minimum distance matrix D_S. On the other hand, computing the naive flexibility metric corresponds to computing a maximum matching in this graph.

References

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Floyd, R. 1962. Algorithm 97: Shortest path. *Communications of the ACM* 5(6).

Fredman, M. L., and Tarjan, R. E. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM* 34(3):596–615.

Policella, N.; Smith, S. F.; Cesta, A.; and Oddi, A. 2004. Generating robust schedules through temporal flexibility. In *Proceedings ICAPS*.

Pollack, M. E., and Tsamardinos, I. 2005. Efficiently dispatching plans encoded as simple temporal problems. In Vlahavas, I., and Vrakas, D., eds., *Intelligent Techniques for Planning*. Idea Group Publishing. 296–319.

Potra, F. A., and Wright, S. J. 2000. Interior-point methods. *Journal of Computational and Applied Mathematics* 124(1–2):281 – 302. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

Wilson, M.; Klos, T.; Witteveen, C.; and Huisman, B. 2013. Flexibility and decoupling in the simple temporal problem. In Rossi, F., ed., *Proceedings International Joint Conference on Artificial Intelligence (IJCAI)*, 2422 – 2428. AAAI Press, Menlo Park, CA.

Wilson, M.; Klos, T.; Witteveen, C.; and Huisman, B. 2014. Flexibility and decoupling in Simple Temporal Networks. *Artificial Intelligence* 214:26–44.