Optimising Flexibility for Simple Temporal Networks

Cees Witteveen
Department of Software and Computer Technology, Delft University of Technology, Mekelweg 4, 2628 CD, Delft, The Netherlands

Keywords: Simple Temporal Planning, Scheduling, Flexibility, Linear Programming.

Abstract: We generalise a recently proposed concurrent flexibility metric to overcome some of its shortcomings. We show that these shortcomings can be removed if one selects an optimal subset of variables for which the concurrent flexibility is determined. The flexibility of the remaining variables does not play a role in the determination of the flexibility of the system. We present a preliminary experimental evaluation of the improvement in concurrent flexibility that can be obtained by comparing some (approximation) algorithms. Their performance on several benchmark sets is evaluated. As a result, in some cases the concurrent flexibility of an STN can be enhanced by 20 - 50%.

1 INTRODUCTION

Simple Temporal Networks (STNs) (Dechter, 2003) offer an elegant framework for analysing temporal aspects of scheduling problems. An STN is a tuple $S = (T, C)$ where $T = \{t_0, t_1, \ldots, t_n\}$ is a set of temporal variables and C is a finite set of binary constraints of the form $t_j - t_i \leq c_{ij}$, for some $c_{ij} \in \mathbb{R}$. Solutions to an STN $S = (T, C)$ are schedules σ, assigning values $\sigma(t)$ to variables $t \in T$ such that all constraints in C are satisfied. It is well-known that deciding whether an STN has a schedule, finding such a schedule, as well as integrating new constraints are problems that all can be solved in low polynomial time (Dechter, 2003).

The focus in this paper, however, is not on finding (single) schedules for an STN: Instead of determining exact times to schedule events, we concentrate on the inherent *flexibility* of an STN. Intuitively, flexibility of a scheduling system refers to the amount of freedom we have in choosing the exact times to schedule an event. While a conventional schedule σ determines a fixed time-to-execute value for each temporal variable $t \in T$ in advance, a flexible schedule offers an *interval of values* for each t to choose from. Such a flexibility could be of great value if, for example, some delay has affected the execution of a temporal variable and a predetermined time-to-execute value for a to be dispatched variable would violate a constraint in C. In such a case flexibility could ensure that a different value can be chosen for t, without affecting the execution of the remaining part of the schedule. In this way we could avoid a costly recomputation of the current schedule.

In order to quantify the available flexibility in an STN, we use *flexibility metrics*. A flexibility metric should reflect some intuitive properties of flexibility we expect to hold and also should allow us to compare STNs w.r.t. their inherent flexibility. Previously, a so-called *naive flexibility (NF)* metric for STNs has been proposed, based on the difference between latest and earliest starting times of events. However, quite often this metric does not reflect our intuition about flexibility of an STN (Wilson et al., 2013). The same authors proposed an alternative, the *concurrent flexibility (CF)* metric, promising such a reflection of intuitive characteristics of flexibility. However, as we will show, there are some cases where this CF metric, too, does not correspond to intuitive properties we would expect such a metric to have. We identify these cases and we propose a modification of CF (ICF: an improved concurrent flexibility metric) to remove the observed deficiencies.

While the two existing metrics (NF and CF) can be computed efficiently, we show that computing our improved flexibility metric is NP-hard. Therefore, we offer some heuristics to approximate our new improved flexibility metric. Using some existing set of benchmark problems we show that, using these heuristics, in some cases the flexibility of an STN can be significantly improved compared with the aforementioned concurrent flexibility.

Witteveen, C.
Optimising Flexibility for Simple Temporal Networks.
DOI: 10.5220/0005824205240531
In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016) - Volume 2*, pages 524-531
ISBN: 978-989-758-172-4

2 PRELIMINARIES

Given an STN $S = (T,C)$, let $var(C) \subseteq T$ denote the set of variables mentioned in C. If $T' \subseteq T$, then $C_{T'} = \{c \in C \mid var(\{c\}) \subseteq T'\}$ and $S_{T'} = (T', C_{T'})$. 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*.[1] In order to be able to express absolute time constraints, the time-point $t_0 \in T$, also denoted by z, is used. It represents a fixed reference point and is assigned the value 0.

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). These tightest constraints are represented as the elements of the $n \times n$ *distance matrix* D_S, containing for every pair (t_i, t_j) of variables the length $D_S(i, j)$ of the *shortest path*[2] from t_i to t_j. If D_S contains the entries $D_S(i, j)$ and $D_S(j,i)$, then $t_i - t_j \leq D_S(j,i)$ and $t_j - t_i \leq D_S(i, j)$ are the strongest constraints implied by C with respect to the temporal difference between t_j and t_i. Slightly abusing terminology, we call $S_{min} = (T, D_S)$ the minimal STN equivalent[3] to S,

The latest (lst) and earliest (est) starting times of $t_i \in T$ can be found by using the first row and first column of D, respectively: $lst(t_i) = D(0,i)$ and $est(t_i) = -D_S(i,0)$. Given an STN S, its distance matrix D_S can be computed in $O(n^3)$-time (Dechter, 2003). Hence, using the STN-machinery we can find earliest and latest starting times for variables $t \in T$ efficiently.

Remark 1. Here, we always assume that, for each $t \in T$, there is a finite interval for scheduling it. More exactly, we assume for each STN $S = (T,C)$ there exist finite constants (horizons) $h_S^- < h_S^+$, such that for all $t \in T$, $h_S^- \leq t \leq h_S^+$. This avoids the occurrence of unbounded time intervals such as $-\infty \leq t_i - t_j \leq \infty$. ■

3 NAIVE FLEXIBILITY

Intuitively, flexibility of a schedule reflects the freedom to choose a time-to-execute for a temporal variable t. Hence, it seems quite self-evident to choose the interval $[est(t), lst(t)]$ to characterise such a freedom of choice: First of all, for every value v outside this interval, there exists no schedule σ such that $\sigma(t) = v$. Moreover, this interval does not contain useless values: it is well-known that for each $t \in T$ and for every $v \in [est(t), lst(t)]$, there exists a schedule σ for S such that $\sigma(t) = v$. No wonder then that one of the first ideas for a flexibility metric has been based on these $[est(t), lst(t)]$ intervals. More exactly, the so-called *naive flexibility* $flex_N(S)$ of an STN $S = (T,C)$ has been defined as $flex_N(S) = \sum_{t \in T}(lst(t) - est(t))$.

However, as has already been pointed out (Wilson et al., 2013; Wilson et al., 2014), this metric will often overestimate the available flexibility in an STN due to the fact that the flexibility intervals assigned to temporal variables are not independent. We will illustrate this fact with a very simple example.

Example 1. Consider the following STNs $S_1 = (T_1, C_1)$ and $S_2 = (T_2, C_2)$, where

$$\begin{aligned} T_1 = T_2 = &\ \{z, t_1, t_2\}, \\ C_1 = &\ \{0 \leq t_i - z \leq 100 \mid i = 1,2\} \\ C_2 = &\ \{0 \leq t_i - z \leq 100 \mid i = 1,2\} \cup \\ &\ \{0 \leq t_1 - t_2 \leq \infty\}. \end{aligned}$$

For both STNs we have $est(t_i) = 0$ and $lst(t_i) = 100$, for $i = 1,2$. Hence, $flex_N(S_1) = flex_N(S_2) = 200$. Intuitively, however, in S_2, due to the ordering constraint between t_1 and t_2, the flexibility should be much lower, since not all combinations of values chosen in these intervals are allowed simultaneously. For example, while $t_1 = 75$ and $t_2 = 10$ is allowed, $t_1 = 10$ and $t_2 = 75$ is not. More in general, a flexibility of f_1, $0 \leq f_2 \leq 100$ given to t_1, results in a flexibility of $f_2 = 100 - f_1$ for t_2. Hence, intuitively, for S_2 we would expect a total flexibility of $f_1 + (100 - f_1) = 100$ instead of 200. ■

The reason for this overestimation is that, in general, the intervals $[est(t), lst(t)]$ are not independent: choosing a value v in one interval might affect the feasible choices in an other interval. In fact, although for each $t \in T$ and every $v \in [est(t), lst(t)]$ there exists a schedule σ for S such that $\sigma(t) = v$, this property does not hold if we consider two or more temporal variables *simultaneously*.

4 CONCURRENT FLEXIBILITY

In (Witteveen et al., 2014) it has been shown that computing the naive flexibility (NF) metric $flex_N(S)$ for an STN $S = (T,C)$ is identical to determining the maxi-

[1]Without loss of generality, in the remainder of the paper, we simply assume that an STN is consistent.

[2]Interpreting the constraints as specifications of the distance between variables.

[3]That is, S_{min} and S have the same set of solutions.

mum of the following LP:

$$\begin{aligned}
\text{maximize} \quad & \sum_{t \in T} (t^+ - t^-) && \text{(LP1)} \\
\text{subject to} \quad & t^- \leq t^+ \quad , && \forall\, t \in T \\
& t^+ - t'^+ \leq c \ , && \forall\, (t - t' \leq c) \in C \\
& t^- - t'^- \leq c \ , && \forall\, (t - t' \leq c) \in C
\end{aligned}$$

Geometrically speaking the solution space of an STN S is a *polytope*. In case of a consistent STN it is a bounded polytope. The (value of the) naive flexibility metric can be shown to be equal to the sum of the lengths of the n (spanning) edges of the smallest n-dimensional *outer bounding box* containing the polytope generated by the underlying STN.

Example 2. Consider the STNs discussed in Example 1. In Figure 1 we have depicted the solution spaces of both STNs. The polytope constituting the solution space of S_1 is the large rectangle with edges of length 100. This rectangle is also the smallest outer bounding box for this polytope. The sum of the lengths of the two spanning edges for this bounding box equals $100 + 100 = 200$. This is the value of the naive flexibility $flex_N(S_1)$ of S_1.

The polytope belonging to S_2 is the shaded upper left triangle with corners $(0,0),(0,100)$ and $(100,100)$. The smallest bounding box containing this triangle is the large rectangle, i.e., the same outer bounding box as the one computed for S_1. Hence, $flex_N(S_2) = flex_N(S_2) = 200$. As we can see, however, half of the values assigned to t_1 and t_2 in this larger bounding box will lie outside the solution polygon. ■

Clearly, if the polytope generated by an STN S is not a hypercube, there are points in this outer bounding box which do not belong to the solution space of the STN. In these cases $flex_N(S)$ will *overestimate* the amount of flexibility of S.

To improve upon the shortcomings of the naive flexibility metric, it was decided in (Wilson et al., 2013) to base an alternative metric, the so-called *concurrent flexibility (CF) metric*, on the *largest inner bounding box* of the solution polytope generated by S. Note that the inner bounding box is contained in the solution space. Hence, it guarantees that for every constraint $t_j - t_i \leq c_{i,j}$ the corresponding flexibility intervals $[t_i^-, t_i^+]$ and $[t_j^-, t_j^+]$ associated with these variables do not violate any constraint. This implies that for every value $v_i \in [t_i^-, t_i^+]$ and $v_j \in [t_j^-, t_j^+]$ it should hold that $v_j - v_i \leq c_{i,j}$. But this is exactly the case whenever t_j assumes a *maximal* value and t_i assumes a *minimal* value, i.e., when $t_j^+ - t_i^- \leq c_{i,j}$.

Hence, taking into account that we want to maximise the length of the intervals $[t^-, t^+]$, the correct value for this new concurrent flexibility metric $flex(S)$ for $S = (T, C)$ can be found by solving the following LP:

$$\begin{aligned}
\text{maximize} \quad & \sum_{t \in T} (t^+ - t^-) && \text{(LP2)} \\
\text{subject to} \quad & t^- \leq t^+ \quad , && \forall\, t \in T \\
& t^+ - t'^- \leq c \ , && \forall\, (t - t' \leq c) \in C
\end{aligned}$$

Example 3. Consider again the STNS $S_1 = (T_1, C_1)$ and $S_2 = (T_2, C_2)$ specified in Example 1. Since the polytope generated by S_1 is a rectangle (see Figure 1), its largest inner bounding box is identical to its smallest outer bounding box. Hence, the naive flexibility $flex_N(S_1)$ equals the concurrent flexibility $flex(S_1)$.

The polytope generated by S_2 is the shaded upper triangle depicted in Figure 1. Its largest inner bounding box (the smaller rectangle in the shaded area in Figure 1) determines the concurrent flexibility $flex(S_2)$ of S_2: in this case $flex(S_2) = 50 + 50 = 100 < flex_N(S_2) = 200$. This is what would be expected. ■

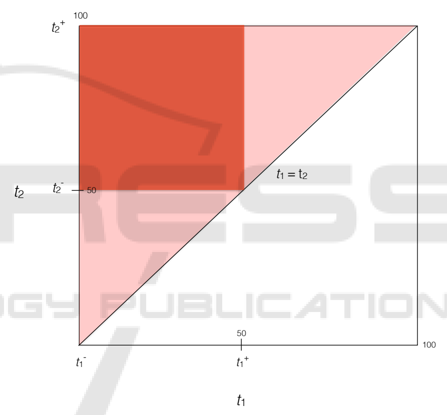

Figure 1: The polytope generated by S_1 is the large rectangle with edges of length 100; the polytope generated by S_2 is the red upper triangle. The inner bounding box of this triangle is the smaller rectangle with edges of length 50. The inner bounding box of S_1 (also the outer bounding box of both S_1 and S_2) equals the polytope of S_1.

5 SHORTCOMINGS OF CF

Two advantages of the CF metric are that *(i)* it never *overestimates* the intuitive flexibility of an STN and *(ii)* it allows one to choose an arbitrary value in each temporal variable interval without risking infeasibility. There are, however, some clear cases where the concurrent flexibility seriously *underestimates* the intuitive flexibility of an STN. We discuss these cases and then propose an improved concurrent flexibility (ICF) metric.

5.1 Rigid Components

While concurrent flexibility offers the possibility to choose from all flexibility intervals $[t^-, t^+]$ simultaneously, the mere fact that all time-point variables are included, paradoxically enough, might severely restrict the concurrent flexibility. Consider the following example:

Example 4. Let $S = (T, C)$ where $T = \{z, t_1, t_2\}$ and $C = \{0 \leq t_1 - z \leq 100,\ 0 \leq t_2 - t_1 \leq 0\}$. Note C implies $t_1 = t_2$. Intuitively, the flexibility of this STN should be equal to 100: we can choose a value for t_1 (or t_2) and then the value for the other variable is simply determined. Computing the concurrent flexibility, however, gives $\mathit{flex}(S) = \max\{(t_1^+ - t_1^-) + (t_2^+ - t_2^-)\} = 0$, while the naive flexibility results in $\mathit{flex}_N(S) = 200$. Again, the naive flexibility is a serious overestimation of the available flexibility, since t_2 is determined as soon as t_1 is determined. On the other hand, the concurrent flexibility seems to be a serious underestimation of the available flexibility. So, intuitively, the concurrent flexibility metric fails for these kind of instances. ■

The reason for this underestimation is not difficult to see: if there is a rigid functional relationship between two or more variables, this relationship will reduce the concurrent flexibility of *all* the variables involved to zero, since a choice v for one of the variables involved will immediately reduce the possible values for all the other variables to a fixed value determined by v. Hence, v itself cannot be allowed to vary, therefore all intervals associated with these variables have length 0. Intuitively, this is unwanted, since, in general, there is always one variable in such a collection of rigidly coupled variables we can choose several values for. Therefore, in such a situation, we should choose a *representative* for the set of coupled variables and let the values of the other variables in the set be determined as soon as we have chosen a value for the representative. This means that in such cases, we have to rescue flexibility by giving up the idea of assigning a flexibility interval to every variable in T.

Let us make these ideas more precise. First of all, we define what it means to for a set of variables to be *rigidly coupled*:

Definition 1. *Given an STN $S = (T, C)$, a subset of variables $X \subseteq T$ is rigidly coupled, if for every pair $t_i, t_j \in X$ it holds that $t_i - t_j = c_{i,j}$ is a constraint implied by C. A* rigid component[4] *is a maximal subset of rigidly coupled variables.*

[4]The term *rigid components* has been borrowed from Hunsberger (Hunsberger, 2002).

So X is a rigid component if taking a value for any variable $t_k \in X$, the values of the remaining variables in X are completely determined. Since the distance matrix D_S contains the tightest upper bounds for the constraints between any pair of variables $t_i, t_j \in T$, it is easy to see that the following observations do hold:

Observation 1. *A subset of variables $X \subseteq T$ in an STN $S = (T, C)$ is rigidly coupled, if for every pair $t_i, t_j \in X$ it holds that $D_S(t_i, t_j) = -D_S(t_j, t_i)$. Hence, for such a pair of variables we have $D_S(t_i, t_j) + D_S(t_j, t_i) = 0$.*

Due to the strict dependencies between them, the flexibility of every variable in a rigid component X is 0 and therefore the total sum of all individual flexibilities in X is 0. Furthermore, it is easy to see that rigid components, due to their maximality must be disjoint. Therefore, we have:

Observation 2. *Let $S = (T, C)$ be an STN. Let X and Y be rigid components. Then:* (i) $flex(X) = flex(Y) = 0$[5] *and* (ii) $X \cap Y = \emptyset$.

Note that rigid components can be easily found by inspecting the distance matrix D_S.

The following proposition states that whenever an STN has rigid components, we might safely remove all elements but one from each rigid component and then concentrate the flexibility computation on the remaining variables:

Proposition 1. *Let $S = (T, C)$ and $X \subseteq T$ a rigid component. Let $t \in X$ be arbitrary. Then*

1. *for every schedule σ' for $S_{(T-X)\cup\{t\}}$, there exists a schedule σ for S extending σ'.*
2. *$\mathit{flex}(S_{(T-X)\cup\{t\}}) \geq \mathit{flex}(S)$.*

Based on Proposition 1 we propose to improve the concurrent flexibility of a system by contracting every rigid component to a single variable. The concurrent flexibility of this reduced system then should represent the flexibility of the original system. First of all, it avoids the disadvantage of assigning a flexibility of 0 to every variable in a rigid component and assigns a flexibility at least as large as the concurrent flexibility of the original system. Secondly, it ensures that whatever schedule for the reduced system is taken, such a schedule can be extended to a schedule of the original system. We therefore call the resulting flexibility metric the *improved flexibility metric*, denoted by $\mathit{flex}^*(S)$. In this context, the metric is based on removal of rigid components while keeping only representatives of them and then computing the concurrent flexibility on the remaining system S_{RC}. So our

[5]Slightly abusing terminology, $\mathit{flex}(X)$ stands for the total flexibility of all vars in X.

first proposal for improvement comes down to defining $flex^*(S) = flex(S_{RC})$.

Example 5. Consider Example 4 with the rigidly coupled variables t_1 and t_2. Clearly, $X = \{t_1, t_2\}$ is a rigid component. Let t_1 be the representative of X. The resulting system $S' = S_{(T-X)\cup\{t_1\}} = (T', C')$ where $T' = \{z, t_1\}$ and $C' = \{t_1 - z \leq 100, z - t_1 \leq 0\}$. The concurrent flexibility of S' equals $flex(S') = 100$. Hence, $flex^*(S) = flex(S') = 100$. ■

We conclude that the disadvantage of concurrent flexibility when dealing with rigid components can be easily eliminated by computing the concurrent flexibility of a reduced STN. This computation takes only polynomial overhead.

5.2 Flexibility Restricting Constraints

Unfortunately, there are other, more general, cases where the flexibility of the system as measured by CF also underestimates the intuitively available flexibility. Let us again consider a simple example.

Example 6. Consider the system $S = (T, C)$ where $T = \{z, t_1, t_2\}$ and $C = \{0 \leq t_1 - z \leq 100, 0 \leq t_2 - t_1 \leq 2\}$. Note that $flex_N(S) = 100 + 102 = 202$, while $flex(S) = 2$. This system does not contain rigid components, but almost all the original flexibility for the variables t_1 and t_2, taken separately, is ”absorbed” by a strong constraint between them. Clearly, $flex_N(S) = 202$ is overestimating the 'real' flexibility, because choosing a value for t_1, or for t_2, leaves at most 2 units of flexibility for the other variable. On the other hand, insisting that t_2 and t_1 should be chosen concurrently, reduces the total concurrent flexibility of the system to 2. Intuitively, however, we should reason as follows: Taken separately, t_2 has a higher flexibility (102) than t_1 (100). Taken together they are strongly dependent: it is almost impossible to choose concurrently a value for t_1 and t_2 in their flexibility intervals without violating the constraints. Therefore, we should determine the flexibility based on t_2 only, thereby possibly reducing the flexibility of t_1 to 0 (note that choosing $t_2 = 102$ does determine the value of t_1 completely). ■

This example suggests a more general strategy to improve the concurrent flexibility of an STN: *(i)* Select a suitable subset of variables for which the concurrent flexibility is maximised, while *(ii)* making sure that a suitable assignment of values to the remaining variables is always guaranteed.

We start with investigating the extendability issue *(ii)*, given a subset T' of variables selected. In order to guarantee an extension to a complete solution, the method we need to apply is called the *Fourier-Motzkin Elimination* (FME) (Khachiyan, 2009). This elimination method is applied to a set A of linear equations to eliminate variables t in such a way that the solutions to the resulting system A' (without t) are exactly those solutions of A where the assignments to the variable t have been removed. In other words: solutions to the reduced set $A' = FME(A, t)$ always can be extended to solutions of A. We discuss the details of the FME method relevant to our problem.

The Fourier-Motzkin Elimination Method. It suffices to explain the FME procedure specialised to STNs.

Proposition 2. *An FME of a variable $t_i \in T$ from an STN $S = (T, C)$ results in replacing all combinations of constraints $t_j - t_i \leq c_{i,j} \in C$ and $t_i - t_k \leq c_{i,k} \in C$ by the constraint*

$$t_j - t_k \leq \min\{c_{j,k}, c_{i,j} + c_{i,k}\}$$

The resulting system $S' = (T - \{t_i\}, C')$ does not contain any occurrence of t_i and every solution σ' to S' can be extended to a solution σ to S.

Definition 2. *Let $S = (T, C)$ be an STN. The reduced system S' resulting from S after the FME of the variables $t_1, t_2, \ldots, t_k$ is denoted by $S' = \exists t_1, t_2, \ldots t_k S$.*

Example 7. Take the STN S presented in Example 6. The Fourier-Motzkin Elimination of t_1 w.r.t. S results in the system $S' = (T', C') = \exists t_1 S$, where $T' = \{z, t_2\}$ and $C' = \{0 \leq t_2 - z \leq 102\}$. ■

While, in general, the FME enjoys double exponential complexity ($O(n^{2^n})$), for STNs the FME can be performed in $O(n^3)$-time as the removal of a single variable t_i results in a set of binary difference constraints that can be computed in $O(n^2)$-time and there are at most $O(n)$ variables to be eliminated.

When using the minimal distance matrix D_S to represent all the minimal constraints, the application of FME to a set of variables T' comes down to the simple removal of corresponding rows and columns in the matrix D_S:

Observation 3. *Let $S = (T, C)$ be an STN. Let $S' = \exists t_i S$ be the STN derived by FME of the i-th variable $t_i \in T$. Then $S'_{min} = (T - \{t_i\}, D_S^{-i})$ is the minimal system equivalent to S', where $D_{S'} = D_S^{-i}$ is the distance matrix derived from D_S by deleting the i-th row and i-th column from D_S.*

Proof. Let $S = (T, C)$ be an STN with distance matrix D_S. Since D_S is the minimal distance matrix of S we have:

$$\forall\ 1 \leq i, j, k \leq n,\ D_S(k, i) + D_S(i, j) \leq D_S(k, j) \quad (1)$$

Applying the FME w.r.t. t_i implies that all combinations of inequalities $t_j - t_i \leq D_S(i,j)$ and $t_i - t_k \leq D_S(k,i)$ are replaced by the inequality

$$t_j - t_k \leq \min\{D_S(k,j), D_S(k,i) + D_S(i,j)\}.$$

Then, using equation 1, we have

$$\min\{D_S(k,j), D_S(k,i) + D_S(i,j)\} = D_S(k,j)$$

implying that the entries in the remaining rows and columns of D_S are not changed due to the removal of the i-th variable. So, a minimal system after applying a Fourier Motzkin elimination step w.r.t. t_i will have a distance matrix $D_{S'}$ where the row and column corresponding to t_i have been removed and all remaining entries remain the same. □

Now we are ready to discuss the second problem we mentioned: selecting a subset T' of variables such that the concurrent flexibility based on this subset of variables is maximal.

Given the above discussed correspondence between applying the FME and removing rows and columns from D_S, we would like to compute the concurrent flexibility for a subset T' of variables using the distance matrix $D_{T'}$ directly. Here, a recently discovered equivalence between *flex*(S) and the value of a minimum cost assignment based on the distance matrix D_S is very useful (Mountakis et al., 2015):

Proposition 3. *Let $S = (T,C)$ be a consistent STN having a minimum distance matrix D_S. Then the concurrent flexibility flex(S) equals the value $c(M)$ of a minimum cost assignment given a cost matrix D^*_S, where*

$$D^*_S(i,j) = \begin{cases} lst(t_i) - est(t_i) & \text{if } i = j, \\ D_S(i,j) & \text{else} \end{cases}$$

Here, the problem to find a minimum cost assignment given an $n \times n$ cost matrix D comes down to find a bijective function $M : \{1,\ldots,n\} \to \{1,\ldots,n\}$ such that $c(M) = \sum_{i=1}^{n} D(i, f(i))$ is minimised. The value $c(M)$ is known as the min-cost assignment given D. The proposition states the equality $c(M) = \mathit{flex}(S)$, where D^*_S is the underlying cost matrix.

Example 8. Consider the STN presented in Example 6. Its distance matrix D_S and corresponding cost matrix D^*_S are

$$D_S = \begin{bmatrix} 0 & 100 & 102 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix}, \quad D^*_S = \begin{bmatrix} 0 & 100 & 102 \\ 0 & 100 & 2 \\ 0 & 0 & 102 \end{bmatrix}$$

The value of a minimum cost assignment $c(M)$ based on D^*_S equals $c(M) = 0 + 2 + 0 = 2 = \mathit{flex}(S)$. ■

Given the correspondence between *flex*(S) and computing a min-cost assignment for a cost matrix D^*_S, it is now easy to establish a relationship with an improved concurrent flexibility metric *flex*$^*(S)$:

Definition 3. *Given a cost matrix D^*_S, the cost matrix based on $T' \subseteq T$, is denoted by $D^*_{S,T'}$, a submatrix of D^*_S obtained by removing the rows and columns corresponding to variables in $T - T'$.*

Definition 4. *A maximal min-cost assignment based on D_S is the maximal cost of all min-cost assignments $M_{T'}$ based on $D^*_{S,T'}$ for some $T' \subseteq T$. We then define*

$$\mathit{flex}^*(S) = \mathit{max}\{c(M_{T'}) : T' \subseteq T \text{ and } M_{T'} \text{ is a min-cost assignment for } D^*_{S,T'}\}$$

Using this correspondence, we can show that computing *flex*$^*(S)$ is NP-hard:

Proposition 4. *Given an STN $S = (T,C)$ it is NP-hard to compute flex$^*(S)$.*

Proof. We use a reduction from SET PACKING. An instance of the (unweighted variant of the) set packing problem is a tuple (U,C), where U is a non-empty set set and $C \subseteq 2^U$ a nonempty collection of subsets of U. We have to find a subset C' (a set-packing) of C such that *(i)* no subsets occurring in C' do overlap and *(ii)* the number of sets in this set packing, i.e. $|C'|$, is maximal.

The reduction to finding the maximal concurrent flexibility problem is quite straightforward: Given an instance (U,C) of the set-packing problem use a fixed enumeration $< c_1, c_2, \ldots c_m >$ of the subsets in C. Assign to each subset $c_i \in C$ a fixed cost $D(i,i) = 1$. Furthermore, for every pair of subsets $c_i, c_j \in C$, define $D(i,j) = 1$ if $c_i \cap c_j = \emptyset$ and $D(i,j) = -|U| - 1$, else. Consider the following claim:

Claim: There exists a subset C' of C with coverage $m > 0$ iff there exists a maximal min-cost assignment based on D of total value $m > 0$.

Clearly, if this claim holds, computing the improved flexibility metric *flex*$^*(S)$ for the obtained STN S would solve the set packing problem. This implies that computing *flex*$^*(S)$ is NP-hard as it solves the NP-hard set-packing problem.

Proof of the Claim

(only if). Suppose such a subset C' exists. Without loss of generality, we may assume that the first m rows in the matrix D correspond to the subsets occurring in C'. But then the first m rows and columns form a submatrix filled with the value 1 only. Hence, selecting this submatrix would result in a minimum cost assignment equal to m. Therefore, *flex*$^*(S) \geq m$.

(if) Suppose a subset maximal min-cost solution with value m has been found. We show that there exists a

subset C' of C with coverage m. Given the construction of D, at least $m' \geq m$ rows must have been chosen for the minimum cost solution of value m.
1. If $m' = m$, then, clearly, these m rows must form a sub matrix filled with ones only, hence there exist m subsets c_i that do not overlap with each other. Therefore, there exists a subset C' of C such that $|C'| = m$.
2. If $m' > m$, then there are $m' - m$ rows where at least one entry $-|U| - 1$ appears. This means that at least once the value $-U - 1$ will contribute to the minimum cost assignment, resulting in a negative value. Since $m > 0$ was assumed, this is not possible. Hence, only the first case applies and we are done. □

5.3 ICF: Experimental Results

In this section we present an experimental evaluation of our improved flexibility metric. First of all, these experiments are used to give an impression of *(i)* the practical value of our metric, i.e., whether the improved flexibility metric really improves the existing concurrent flexibility metric and *(ii)* if so, what is a good (approximation) algorithm (among the ones we investigated) that can be applied.

We investigated a couple of algorithms and applied them to a dataset of STN benchmark instances (see (Planken et al., 2013)). This dataset contains a series of STN-instances grouped as *bdh-agent-problems* (300 instances), *diamonds* (130 instances), *chordalgraphs* (400 instances), *NeilYorkeExamples* (180 instances) The size of the STN instances in these datasets varies from instances with 41 variables with 614 constraints to instances with 4882 nodes and 14110 constraints. In total, these sets contain 1010 STN-instances. We used MATLAB (R2014b) to perform the experiments. In particular, we used a fast implementation of the Jonker-Volgenant algorithm (Jonker and Volgenant, 1987) to compute minimal cost assignments to obtain the concurrent flexibility of an STN.

Detecting Rigid Components. The first algorithm we applied is a simple rigid component detection algorithm as suggested in Section 5.1. Whenever it detects a rigid component X in an STN S it removes all but one variable from X. When no more rigid component can be found, the number of rigid components detected, their size and the improvement w.r.t. the concurrent flexibility of the original system is determined.

The results are simply disappointing: In none of the 1010 instances rigid components have be found. We conclude that detecting rigid components in these benchmark sets does not help at all in improving the concurrent flexibility.

Removing Tightly Coupled Variables. Next, generalising from rigid components, we investigate whether removal of variables that are tightly coupled improves the concurrent flexibility.

To each instance S in the benchmark sets, we applied the following algorithm:

1. Determine the distribution F_S of sums $D_S(i,j) + D_S(j,i)$ for all entries D_S. Remember that for each (i,j) this sum indicates the tightness of the constraints between variable t_i and t_j.
2. To remove the δ-percent tightest constraints, collect all pairs of variables t_i and t_j such that $D_S(i,j) + D_S(j,i)$ occurs in the δ-percent lowest part of the distribution F_S.
3. For each such a pair (i,j), remove the variable (t_i or t_j) from T whose removal results in the largest increase of the concurrent flexibility for the remaining set of variables.
4. Compute the concurrent flexibility $\mathit{flex}(S')$ of the subsystem S' obtained in this way.

For each instance, this procedure was repeated for 10 threshold values δ ranging from $\delta = 0.05$ to $\delta = 10$. We computed for each instance the maximum ratio $\mathit{flex}(S')/\mathit{flex}(S)$ (**max ratio**) obtained and the corresponding threshold value δ (**threshold**) for which it was obtained. Note that values $\mathit{flex}(S')/\mathit{flex}(S) > 1$ indicate an improvement that can be obtained by looking at a subset of variables instead of all variables.

The results are summarised in Table 1. Observe that, in general, as all improvements are 5% or less, no significant increase in flexibility can be observed in any set of benchmark problems. Furthermore, looking at the threshold values, it hardly seems to make sense to go beyond threshold values of $\delta = 1$ to detect an improvement in concurrent flexibility. This seems to show that a simple greedy approach based on tightness of constraints does not help that much in improving the flexibility.

Table 1: Removing tightly coupled variables for improving concurrent flexibility.

benchmark set	max ratio	threshold
bdh	1	0
diamonds	1.05	0.62
chordal	1.01	0.05
NeilYorke	1.04	0.37

Iteratively Removing Least Contributing Variables. In the first two experiments we applied algorithms attempting to improve the concurrent flexibility by removing variables that were rigidly or tightly coupled. Since we observed that the improvements in concurrent flexibility were quite modest, we decided to use another approach not based on the detection and removal of tightly coupled variables. Instead, we used a fairly simple greedy approach using the minimum cost assignment algorithm as a subroutine. The algorithm iteratively removes that variable $t \in T$ whose removal results in the largest increase in concurrent flexibility until no further improvement is possible. Given an STN $S = (T, C)$ the algorithm is as follows:

```
compute $flex(S_T)$;
while $\exists t \in T$ s.t. $flex(S_{T-\{t\}}) \geq flex(S_T)$ do
  $t := \text{argmax}_t flex(S_{T-\{t\}})$;
  $flex(S_T) = flex(S_{T-\{t\}})$;
  $T := T - \{t\}$;
end while
return $flex(S') = flex(S_T)$;
```

In Table 2, we have presented for each benchmark set the average ratio $flex(S')/flex(S)$ (**ratio**) of the concurrent flexibility based on a subset of variables (S') vss the concurrent flexibility of the original system (S) and the average reduction (**reduction**) in the number of variables obtained when computing $flex(S')$. It turns out that this procedure greatly improves the concurrent flexibility by removal of variables, although showing remarkable differences between the benchmark sets. While the bdh and chordal instances were not sensitive to this approach, the flexibility in the NeilYorkeExamples and the diamonds benchmark sets could be improved significantly (up to 50%). This improvement was established using a relatively small part of the original set of variables due to a significant reduction (0.48 and 0.70) of the number of variables involved. Clearly, we need to further investigate which properties are responsible for the (in)sensitivity to these greedy approaches.

Table 2: Iteratively removing variables to maximising concurrent flexibility.

benchmark set	**ratio**	**reduction**
bdh	1	0.01
diamonds	1.50	0.70
chordal	1.02	0.30
NeilYorke	1.32	0.48

6 CONCLUSIONS

We introduced an improved concurrent flexibility metric to determine the inherent flexibility of an STN. The main reason for introducing this new metric was the observation that in some cases requiring flexibility for all events concurrently is unrealistic, especially if there are strict or nearly strict dependencies between some temporal variables. Our new metric avoids these difficulties by looking at a subset of the variables for which the concurrent flexibility is maximised.

A question that still has to be answered is what happens to the flexibility of the variables not included in this subset. Clearly, their flexibility depends on the values chosen for the variables occurring in the subset. This idea we would like to explore and generalise in the near future: instead of distinguishing between two types of variables, we envision a partial flexibility order over the variables: a variable t occurs before t' if the flexibility of the first one determines the flexibility of the latter one. Following this idea one could envision *flexibility policies* that given such a partial flexibility order guarantee a minimal amount flexibility for subsets of variables.

REFERENCES

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

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

Hunsberger, L. (2002). Algorithms for a temporal decoupling problem in multi-agent planning. In *Proceedings AAAI*.

Jonker, R. and Volgenant, A. (1987). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.

Khachiyan, L. (2009). Fourier-motzkin elimination method. In Floudas, C. A. and Pardalos, P. M., editors, *Encyclopedia of Optimization*, pages 1074–1077. Springer US.

Mountakis, S., Klos, T., and Witteveen, C. (2015). Temporal flexibility revisited: Maximizing flexibility by computing bipartite matchings. In *Proceedings Twenty-Fifth International Conference on Automated Planning and Scheduling*.

Planken, L. R., Boerkoel, J. C., and Wilcox, R. J. (2013). Multiagent STN benchmark instances.

Wilson, M., Klos, T., Witteveen, C., and Huisman, B. (2013). Flexibility and decoupling in the simple temporal problem. In Rossi, F., editor, *Proceedings International Joint Conference on Artificial Intelligence (IJCAI)*, pages 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.

Witteveen, C., Wilson, M., and Klos, T. (2014). Optimal decoupling in linear constraint systems. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, pages 2381–2387. AAAI Press, Menlo Park, CA.