A Metric for Quantifying Similarity between Timing Constraint Sets in Real-Time Systems

YUE YU and SHANGPING REN, Illinois Institute of Technology
XIAOBO SHARON HU, University of Notre Dame

Real-time systems are systems in which their timing behaviors must satisfy a specified set of timing constraints and they often operate in a real-world environment with scarce resources. As a result, the actual runtime performance of these systems may deviate from the design, either inevitably due to unpredictable factors or by intention in order to improve system's other Quality-of-Service (QoS) properties. In this article, we first introduce a new metric, *timing constraint set similarity*, to quantify the resemblance between two different timing constraint sets. Because directly calculating the exact value of the metric involves calculating the size of a polytope which is a #*P*-hard problem, we instead introduce an efficient method for estimating its bound. We further illustrate how this metric can be exploited for improving system predictability and for evaluating trade-offs between timing constraint compromises and the system's other QoS property gains.

Categories and Subject Descriptors: C.3 [**Computer Applications**]: Special-Purpose and Application-Based Systems—*Real-time and embedded systems*; J.6 [**Computer Applications**]: Computer-Aided Engineering—*Computer-aided design (CAD)*; B.6.3 [**Logic Design**]: Design Aids—*Automatic synthesis; optimization*

General Terms: Design, Measurement, Theory, Verification

Additional Key Words and Phrases: Timing constraints, timing constraint similarities, quality of service, real-time systems, scheduling, constraint satisfaction, timing constraint feasibility region

ACM Reference Format:
Yu, Y., Ren, S., and Hu, X. S. 2011. A metric for quantifying similarity between timing constraint sets in real-time systems. ACM Trans. Des. Autom. Electron. Syst. 16, 3, Article 34 (June 2011), 33 pages.
DOI = 10.1145/1970353.1970368. http://doi.acm.org/10.1145/1970353.1970368

1. INTRODUCTION

Software for real-world systems often operates in an unpredictable environment and interacts with physical machineries. Hence, for most of these software systems, it is difficult and unrealistic to implement them in such a way that they behave *precisely* as specified due to the following facts.

— *System Complexity.* The ever-increasing complexities of software systems have made guarantees of *exact* system behavior impractically expensive, if not impossible. For

The preliminary version of this article was presented at the *Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium,* 2009.
This work is supported in part by research grants from NSF CAREER CNS-0746643, CNS-0834180, CNS-0720457, CNS-1018731, CNS-1035894, and CPS-0931195.
This work was done when Y. Yu was a Ph.D. student at Illinois Institute of Technology.
Authors' addresses: Y. Yu, SAFE Investment Co., Ltd., 28/F Far East Finance Centre, Hong Kong; S. Ren (corresponding author), Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616; email: ren@iit.edu; X. S. Hu, Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556.

DOI 10.1145/1970353.1970368 http://doi.acm.org/10.1145/1970353.1970368

example, advances in computer architecture and software have made it difficult to predict the execution time of software, and networking techniques further introduce variability and stochastic behavior into the system [Lee 2005].

— *Unpredictable Operating Environment.* The intrinsically unpredictable nature of the environments in which software systems operate determines that even though software operates precisely as designed, its interactions with the outer world may not be totally expected. For example, Jackson et al. have shown that several aircraft accidents have been attributed to "mode confusion," where the software operated as designed but not as expected by the pilots [Jackson et al. 2007].

— *Computational Intractability.* From a theoretical point of view, achieving exactness in the verification of system properties is sometimes intractable. For example, Alur and Dill have shown that the satisfiability of a very simple class of real-time properties such as "every p-state is followed by a q-state *precisely* 5 time units later" turns out to be undecidable in a continuous-time model [Alur and Dill 1994]. Although several real-time logics are decidable under discrete approximations of real time [Alur and Henzinger 1989] or under interval timing constraints [Alur et al. 1991], these models unfortunately prohibit infinite precision.

On the other hand, real-time and embedded systems often face trade-offs between time and QoS under limited resources. Therefore, even if a system can be implemented precisely as specified, relaxing some of the specifications may reduce resource consumptions: for hard real-time systems, all timeliness requirements must be met and thus optimizing other properties such as minimizing energy consumption must not violate timing constraints; for soft real-time systems, on the other hand, the requirement for timing constraint satisfaction guarantees is not as stringent. Such timing flexibility allowed by soft real-time systems can often be utilized to improve the system's other QoS properties, such as reducing total energy consumption. A challenging task in investigating the trade-offs between timing constraint satisfaction and other QoS properties is how to quantify the degree of timing constraint satisfaction. That is, how do we measure the level of satisfaction for some given timing behavior with respect to a set of timing constraints? Another closely related challenge is to determine which timing constraints to be relaxed and by how much in order to achieve certain other QoS gains, for example, energy consumption reduction. Though some researchers have studied problems that are somewhat related to the preceding problems (to be discussed in the next section), we contend that there exists no systematic approach for tackling these challenges.

In this article, we propose a framework for measuring timing constraint satisfaction which can be used to address the previous challenges. Specifically, we introduce a novel metric, that is, *timing constraint set similarity*, to capture the resemblance between two timing constraint sets. It is defined in terms of the common feasible region of two systems constrained by the two given timing constraint sets. This value reflects the probability of timing constraint satisfaction when the original timing constraints are violated or intentionally modified for improving QoS properties.

However, directly calculating the exact value of similarity between two sets of timing constraints is computationally intractable. To overcome this difficulty, we leverage the concept of similarity bound and derive a closed-form formula for computing a tight similarity bound. This bound can be used to guide the design process and provide confidence guarantees on certain QoS properties.

To show how one may use the timing constraint similarity metric to predict real-world performances and guide design processes of real-time embedded systems, we:

(1) give a detailed design example in which a set of soft real-time tasks are executed on a multiprocessor system-on-chip (MPSoC) and the goal is to trade timing

constraint satisfaction for reducing energy consumption. This concrete example serves as a demonstration that the similarity metric provides an effective tool to measure and guide the trade-offs between different QoS properties;

(2) study the similarities between timing constraint sets in the specification of an object tracking system and its real-world deviation, and use the similarity to infer the relationships on properties, such as tracking precisions, fulfilled by different but similar systems.

The rest of this article is organized as follows. The next section provides a motivating example and reviews related work. Section 3 introduces the timing constraint set feasible region and its normal form. It is used to establish the constraint similarity metric. Section 4 presents the similarity metric that quantifies how much one set of timing constraints resembles another. Section 5 studies how the theory of timing constraint similarities can be utilized to reduce the total energy consumption of an MPSoC system with minimal changes to the satisfaction of original timing constraints. Section 6 applies the theory of timing constraint similarities to an object tracking system for predicting tracking error rates. Finally, we conclude and point out future work in Section 7.

2. MOTIVATION AND RELATED WORK

For a given specification, there may be different designs and implementations. Due to specification abstractions, often times, none of the implementations precisely satisfies the required timing constraints, especially in soft real-time systems. Hence, it is important to have a scheme that measures the level at which an implemented system satisfies the specified constraints. In addition, it allows studies of "what if" scenarios where certain timing constraints are relaxed to some extent in order to improve other QoS properties. Furthermore, it can be used to judiciously decide design specifications.

One intuitive way to quantify the level of timing constraint satisfaction is to measure the probability with which a system satisfying a set of modified timing constraints still satisfies the original timing constraints. With such a probability, design alternatives with different timing behavior can be easily compared. We use a simple example to illustrate this point.

Example 1. Consider scheduling a task j with a relative deadline of 22ms on an MPSoC with three cores m_1, m_2, and m_3. The worst-case execution times (WCETs) of j on m_1, m_2, and m_3 are 20ms, 25ms, and 30ms with peak powers $10W$, $7W$, and $6W$, respectively. For simplicity, we also assume that the actual execution times are uniformly distributed between 5ms and respective WCETs. Now, if we need to limit the peak power to be less than $8W$, but allow some deadline misses, we can schedule the task on either m_2 or m_3. If we schedule the task on m_2, for instance, what we can guarantee is the satisfaction of a constraint with a relative deadline of 25ms, rather than 22ms. Similarly, with the task on m_3, we can guarantee the satisfaction of a deadline of 30ms. In other words, in this example, to maintain the peak power below $8W$, we have two different approaches. Now, the question is from timing perspective, which one is a better option?

If task j is executed on m_2, the probability of the system satisfying the original timing constraint of 22ms is $\frac{22-5}{25-5} = 85\%$. The probability reduces to $\frac{22-5}{30-5} = 68\%$ if task j is executed on m_3. So for this simple example, the answer to the preceding question is obvious. That is, from the timing perspective, using m_2 is better than m_3. Note that this conclusion coincides with the intuition that 25ms is "closer" to 22ms than 30ms. However, this may not always be true; one could easily see this by considering the extreme case where the best-case execution time of j on m_2 is greater than 22ms.

From the previous simple example, one can see that the probability with which a system satisfying a set of modified timing constraints still satisfies the original timing constraints can be used effectively to compare design alternatives with different timing behaviors. Now the challenge is how to measure such a probability when there are more complex timing constraints involved. Furthermore, given the timing constraint satisfaction as one of the system comparison criteria, how can we find a subset of constraints from a given constraint set and modify them so that the required nontiming properties (e.g., power consumption) are satisfied, but the timing property change is minimal, in other words, the timing property is the most similar (closest) to the original one? The goal of this article is to address these questions by introducing a new metric.

As related work, many researchers have studied feasibility probabilities for tasks with varying execution times. Tia et al. [1995] propose a way to find the probability of a single task meeting its timing constraint, referred to as task feasibility probability. Kalavade and Moghé [1998] present an approach to compute the probability of any single task delay exceeding its deadline, which is equivalent to the task feasibility probability. However, Hu et al. [2001] point out that the probability of each individual task meeting its timing constraint is not sufficient in several situations since there often exists strong correlation among events of tasks meeting their deadlines. The authors give a new metric that considers the overall system probabilistic behavior where tasks have their individual deadlines and the correlations between tasks are captured by precedence constraints. With this metric in the system-level design exploration process, one can readily compare the probabilistic timing performance of alternative designs. Based on Hu et al. [2001], Wang et al. [2007] define a design metric called performance yield, which is the probability of the assigned schedule meeting the predefined performance constraints. However, none of these works considers the problem of measuring the level of timing constraint satisfaction when the original timing constraints cannot be satisfied or are intentionally modified.

In general cases, real-time systems have various types of timing constraints such as freshness, correlation, and output separation [Gerber et al. 1995]. Our study focuses on a more generalized constraint model where correlations between tasks are treated as linear timing constraints. As can be seen, the correlations between individual timing constraints are not as simple as precedence constraints; instead, they are captured by the *feasible region* of the entire set of timing constraints. We study the similarity relation between feasible regions of two different timing constraint sets and use the similarity value to infer constraint satisfaction probability of a system that satisfies one set of timing constraints satisfies the other.

Throughout our study, we assume that the feasible region of any timing constraint set is not empty, that is, the timing constraint set is feasible. It is well-known that the absence of negative cycles in a timing constraint graph implies the feasibility of the corresponding real-time constraint set [Raju et al. 1992]. When a set of real-time constraint is infeasible, the constraint set is debugged by freeing the corresponding constraint graph from all its negative cycles. Some algorithms for this task take time proportional to the number of negative cycles in the graph, which can grow exponentially [Dasdan 1999; Liao and Wong 1983]; Dasdan [2002a, 2002b, 2009] provides an algorithm whose time complexity is polynomial in the size of the input constraint graph.

Many notions on similarities have been defined in the literature for process models. Gupta et al. [2004] give a pseudometric analog of bisimulation for generalized semi-Markov processes and show that two metrically similar processes have similar observable quantitative properties. Thorsley and Klavins [2008] use Wasserstein pseudometrics to quantify the similarities between stochastic processes and introduce an algorithm to approximate the pseudometrics directly from sampled data rather than from process models themselves. The notion of similarity on other models are

also studied, for example, in de Alfaro et al. [2007], Song et al. [2007], and Julius et al. [2006]. However, the pseudometrics proposed in these works are used to compare processes. Though there are similarities between the idea of introducing quantitative metrics to measure two nonequivalent processes or constraints, the metrics introduced in this article not only measure the resemblance between two sets of timing constrains, but also provide quantitative design guidance in deciding the trade-offs between constraint satisfaction and other QoS properties.

Trading one QoS property for another has been studied in various contexts. For example, reducing energy consumption through compromising system performance has been considered in a wide spectrum of computing. To name a few, Moscibroda et al. discuss the trade-off between energy efficiency and rapidity of event dissemination in ad hoc and sensor networks [Moscibroda et al. 2006]; in high-performance computing, Feng et al. analyzed NAS and SPEC suites to determine the relationship between frequency and voltage settings and execution time, and show that a significant decrease in energy is possible with a small increase in time [Pan et al. 2005]. In fact, for real-time and embedded systems, dynamic voltage scaling techniques, which reduce system supply voltage for lower operation frequencies, has been extensively used in various power management schemes [Aydin et al. 2001; Pillai and Shin 2001; Saewong and Rajkumar 2003]. However, to our best knowledge, there is no quantitative study of trading timing constraint satisfaction in soft real-time systems for other QoS properties.

3. TIMING CONSTRAINT SET NORMAL FORM

In this section, we introduce the geometric foundations for characterizing timing constraint sets. The constraint normal form defined in this section will be used to establish constraint similarity metrics in Section 4.

In our system model, we take a commonly used approach in that system behaviors (or computations) are represented as *data streams*, that is, a sequence of event occurrences $(e_1, e_2, \ldots, e_n)$ [Woo et al. 2006], and a *timed data stream* is formed by pairing each event e_i with its corresponding occurrence time $t(e_i)$, as defined next [Arbab and Rutten 2002].

Definition 3.1 (*Timed Data Stream*). A Timed Data Stream (TDS) is a sequence $\big((e_1, t(e_1)), (e_2, t(e_2)), \ldots, (e_n, t(e_n))\big)$ where $(t(e_1), t(e_2), \ldots, t(e_n))$ is a monotonically increasing sequence with elements in $\Re^+ \cup \{+\infty\}$. Geometrically, a TDS is represented as a point in $|E|$-dimensional space where each axis represents an event and the projection of the point on the axis represents the occurrence time of the corresponding event.

Without timing constraints, events can occur at any time instances and thus the set of all TDS's occupies the entire nonnegative portion of the $|E|$-dimensional space. However, when a set of timing constraints of the form $t(e_i) - t(e_j) \leq d (d \in \Re^+ \cup \{+\infty\})$ exists, the set of TDS's satisfying the set of timing constraints is only a convex region in the $|E|$-dimensional space and we call it *feasible region* throughout the article. Feasible regions are the key in comparing timing constraint sets and we illustrate them in Examples 2 and 3.

Example 2 (*2-Dimensional Feasible Region*). Let s_j and f_j be the events that task j starts and finishes, the feasible region of the relative deadline constraint $0 < t(f_j) - t(s_j) \leq 22$ in Example 1 is shown in Figure 1 (shaded area).

In the figure, TDS $((s_j, 20), (f_j, 38))$ in the feasible region satisfies the relative deadline constraint, while TDS's $((f_j, 16), (s_j, 28))$ and $((s_j, 8), (f_j, 40))$ outside the feasible region violate causality $t(s_j) - t(f_j) < 0$ and deadline $t(f_j) - t(s_j) \leq 22$, respectively.

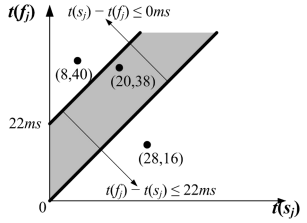


Fig. 1. The feasible region of constraint 0ms $< t(f_j) - t(s_j) \leq$ 22ms.

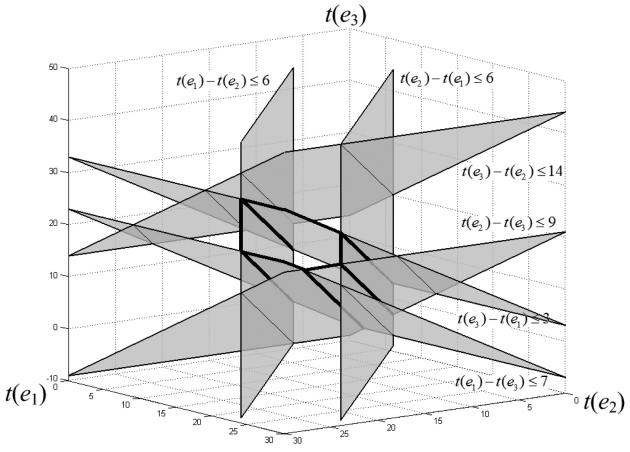


Fig. 2. The feasible region of a constraint set (1).

The dimension of feasible regions becomes higher when the number of constrained events increases. Consider the following example.

Example 3 (*3-Dimensional Feasible Region*). Let the set of timing constraints that specify the relative time spans among three events be

$$\left\{ \begin{array}{l} t(e_1) - t(e_2) \leq 6,\ t(e_2) - t(e_1) \leq 6,\ t(e_1) - t(e_3) \leq 7, \\ t(e_3) - t(e_1) \leq 3,\ t(e_2) - t(e_3) \leq 9,\ t(e_3) - t(e_2) \leq 14 \end{array} \right\} \tag{1}$$

Each timing constraint confines a half space in the 3-dimensional space and the intersection of such half spaces is the feasible region. The feasible region of (1) is shown in Figure 2 with its boundaries marked as bold lines.

In the figure, the pentagonal prism circumscribed by all but the plane representing the constraint $t(e_3) - t(e_2) \leq 14$ characterizes the feasible region, that is, each point $(t(e_1), t(e_2), t(e_3))$ in the region satisfies constraint set (1).

From Examples 2 and 3, we can see that a feasible region characterizes all valid execution time traces, that is, a system's valid timing behaviors under a set of timing constraints. However, when the dimension of a feasible region becomes higher, its shape becomes more complex and makes the graphical representation difficult. In order to compare feasible regions, alternative ways to represent high-dimensional feasible regions are needed.

We introduce an algebraic representation to describe feasible regions such that comparisons can be directly made between feasible regions. This representation builds on the concept of the most stringent constraints, which we explain next by

using Example 3 again. Examine the feasible region of Example 3 shown in Figure 2. Note that the shape of the feasible region of (1) does not change when the constraint $t(e_3) - t(e_2) \leq 14$ is changed to $t(e_3) - t(e_2) \leq 9$ (or any other constraint value larger than 9). In fact, $t(e_3) - t(e_2) \leq 9$ is the most stringent timing constraints between event e_3 and e_2 which can be implied by the given constraint set.

For a given set of timing constraints, we can find the most stringent constraint set by leveraging the approach of finding all-pairs shortest paths. Specifically, we construct a constraint graph G by defining the vertex set of G as the set of events in the timing constraint set; for every two vertices e_i, e_j in G, there is an edge from e_i to e_j with weight d if there is a constraint $t(e_i) - t(e_j) \leq d$. The most stringent timing constraint implied by the given constraint set between every pair of events, $t(e_i) - t(e_j) \leq d^*_{i,j}$, can hence be derived from applying the Floyd-Warshall all-pairs shortest paths algorithm on G [Jahanian and Mok 1987]. The most stringent constraint set has an important property which is summarized in Lemma 3.2. It states that the feasible region of a set of real-time constraints does not change when constraints between all event pairs are replaced by the corresponding most stringent constraints derived from the Floyd-Warshall algorithm.

LEMMA 3.2. *Given a set of m timing constraints of the form $t(e_i) - t(e_j) \leq d_k$ among n events, $\mathbf{At} \leq \mathbf{d}$, where $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{t} = \begin{bmatrix} t(e_1) & \dots & t(e_n) \end{bmatrix}^{\mathrm{T}}$, and $\mathbf{d} = \begin{bmatrix} d_1 & \dots & d_m \end{bmatrix}^{\mathrm{T}}$. We have $\{\mathbf{t} \,|\mathbf{At} \leq \mathbf{d}\} = \left\{\mathbf{t} \,\middle|\widetilde{\mathbf{A}}\mathbf{t} \leq \widetilde{\mathbf{d}}\right\}$, that is, the set of solutions of $\mathbf{At} \leq \mathbf{d}$ is the same as the set of solutions of $\widetilde{\mathbf{A}}\mathbf{t} \leq \widetilde{\mathbf{d}}$ where*

$$\widetilde{\mathbf{A}} = \left[\begin{array}{ccccc} 1 & -1 & & & \\ 1 & & -1 & & \\ \vdots & & & \ddots & \\ 1 & & & & -1 \\ \hline -1 & 1 & & & \\ & 1 & -1 & & \\ & \vdots & & \ddots & \\ & 1 & & & -1 \\ \hline \vdots & & \dots & & \vdots \\ \hline -1 & & & & 1 \\ & -1 & & & 1 \\ & & \ddots & & \vdots \\ & & & -1 & 1 \end{array}\right] \quad \textit{and} \quad \widetilde{\mathbf{d}} = \left[\begin{array}{c} d^*_{1,2} \\ d^*_{1,3} \\ \vdots \\ d^*_{1,n} \\ \hline d^*_{2,1} \\ d^*_{2,3} \\ \vdots \\ d^*_{2,n} \\ \hline \vdots \\ \hline d^*_{n,1} \\ d^*_{n,2} \\ \vdots \\ d^*_{n,n-1} \end{array}\right] \tag{2}$$

that is, $\widetilde{\mathbf{A}}$ is the incidence matrix of the corresponding complete graph of the timing constraint graph of $\mathbf{At} \leq \mathbf{d}$, and $d^*_{i,j}, i \neq j$ is the shortest path weight between $t(e_i)$ and $t(e_j)$.

PROOF.

(i) $\{\mathbf{t} \,|\mathbf{At} \leq \mathbf{d}\} \supseteq \left\{\mathbf{t} \,\middle|\widetilde{\mathbf{A}}\mathbf{t} \leq \widetilde{\mathbf{d}}\right\}$

This directly follows from the fact that $\mathbf{A}$ contains some rows of $\widetilde{\mathbf{A}}$ and the corresponding d's in $\mathbf{d}$ is no less than those in $\widetilde{\mathbf{d}}$ (the shortest path weights).

(ii) $\{\mathbf{t} \,|\mathbf{At} \leq \mathbf{d}\} \subseteq \left\{\mathbf{t} \,\middle|\widetilde{\mathbf{A}}\mathbf{t} \leq \widetilde{\mathbf{d}}\right\}$

Assume to the contrary that there is a vector $\mathbf{t}' = \left[\begin{array}{ccc} t_1 & \ldots & t_n \end{array}\right]^{\mathrm{T}}$ such that $\mathbf{t}' \in \{\mathbf{t} \,|\mathbf{At} \leq \mathbf{d}\} \wedge \mathbf{t}' \notin \left\{\mathbf{t} \,\middle|\widetilde{\mathbf{A}}\mathbf{t} \leq \widetilde{\mathbf{d}}\right\}$. This implies that the following set of linear inequalities has no solution.

$$\left[\begin{array}{c} \mathbf{I} \\ -\mathbf{I} \\ \widetilde{\mathbf{A}} \end{array}\right] \mathbf{t} \leq \left[\begin{array}{c} \mathbf{t}' \\ -\mathbf{t}' \\ \widetilde{\mathbf{d}} \end{array}\right] \tag{3}$$

Based on Farkas' lemma[1] [Fang and Puthebnpura 1993], together with the infeasibility of (3), we have that there exists an $(n^2 + n)$-vector $\left[\begin{array}{ccc} \mathbf{t}_1^{\mathrm{T}} & \mathbf{t}_2^{\mathrm{T}} & \mathbf{t}_3^{\mathrm{T}} \end{array}\right]^{\mathrm{T}}$ where $\mathbf{t}_1$ and $\mathbf{t}_2$ are two n-vector and $\mathbf{t}_3^{\mathrm{T}}$ is a $(n^2 - n)$-vector, such that (4), (5), and (6) hold.

$$\left[\begin{array}{ccc} \mathbf{I} & -\mathbf{I} & \widetilde{\mathbf{A}}^{\mathrm{T}} \end{array}\right] \left[\begin{array}{c} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \mathbf{t}_3 \end{array}\right] = \mathbf{0} \tag{4}$$

$$\left[\begin{array}{c} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \mathbf{t}_3 \end{array}\right] \geq \mathbf{0} \tag{5}$$

$$\left[\begin{array}{ccc} \mathbf{t}'^{\mathrm{T}} & -\mathbf{t}'^{\mathrm{T}} & \widetilde{\mathbf{d}}^{\mathrm{T}} \end{array}\right] \left[\begin{array}{c} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \mathbf{t}_3 \end{array}\right] < 0 \tag{6}$$

From (4) we have that

$$\mathbf{t}_1 - \mathbf{t}_2 = -\widetilde{\mathbf{A}}^{\mathrm{T}}\mathbf{t}_3. \tag{7}$$

Insert (7) into (6) we have that

$$-\mathbf{t}'^{\mathrm{T}}\widetilde{\mathbf{A}}^{\mathrm{T}}\mathbf{t}_3 + \widetilde{\mathbf{d}}^{\mathrm{T}}\mathbf{t}_3 = \left(\widetilde{\mathbf{d}}^{\mathrm{T}} - \mathbf{t}'^{\mathrm{T}}\widetilde{\mathbf{A}}^{\mathrm{T}}\right)\mathbf{t}_3 < 0. \tag{8}$$

Therefore, it must be that

$$\exists i, j : d^*_{i,j} < t_i - t_j, \tag{9}$$

since otherwise $\widetilde{\mathbf{d}}^{\mathrm{T}} - \mathbf{t}'^{\mathrm{T}}\widetilde{\mathbf{A}}^{\mathrm{T}} \geq \mathbf{0}$ together with (5) would imply $\left(\widetilde{\mathbf{d}}^{\mathrm{T}} - \mathbf{t}'^{\mathrm{T}}\widetilde{\mathbf{A}}^{\mathrm{T}}\right)\mathbf{t}_3 \geq 0$ which contradicts (8). However, (9) contradicts the fact that $d^*_{i,j}$ is the optimal solution to the linear program

$$\begin{array}{ll} \textbf{maximize} & t(e_i) - t(e_j) \\ \textbf{subject to} & \mathbf{At} \leq \mathbf{d} \end{array} \tag{10}$$

[1]Farkas' lemma: let $\mathbf{A}$ be a matrix and $\mathbf{x}$ and $\mathbf{b}$ vectors. Then the system $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq 0$ has no solution *iff* the system $\mathbf{A}^T\mathbf{y} \geq \mathbf{0}$, $\mathbf{b}^T\mathbf{y} < 0$ has a solution, where $\mathbf{y}$ is a vector.

that is, $d^*_{i,j}$ is the shortest path weight. Therefore, we have $\{\mathbf{t} \,|\mathbf{At} \leq \mathbf{d}\} \subseteq \left\{\mathbf{t} \,\middle|\widetilde{\mathbf{A}}\mathbf{t} \leq \widetilde{\mathbf{d}}\right\}$ and thus $\{\mathbf{t} \,|\mathbf{At} \leq \mathbf{d}\} = \left\{\mathbf{t} \,\middle|\widetilde{\mathbf{A}}\mathbf{t} \leq \widetilde{\mathbf{d}}\right\}$. □

An important implication of Lemma 3.2 is that the shape of the feasible region is determined solely by the most stringent timing constraints between all *pairs* of events. In fact, given two sets of real-time constraints C ($\mathbf{At} \leq \mathbf{d}$) and C' ($\mathbf{A}'\mathbf{t} \leq \mathbf{d}'$) on the same set of events, along with their all-pairs shortest paths matrices $\mathbf{D}^*$ and $\mathbf{D}'^*$, respectively, it is easy to see that $\widetilde{\mathbf{A}}$ is the same for C and C' since $\widetilde{\mathbf{A}}$ is the incident matrix of the complete timing constraint graphs. Moreover, we have the following.

$$\begin{aligned}\mathbf{D}^* = \mathbf{D}'^* &\Leftrightarrow \left\{\mathbf{t} \,\middle|\widetilde{\mathbf{A}}\mathbf{t} \leq \widetilde{\mathbf{d}}\right\} = \left\{\mathbf{t} \,\middle|\widetilde{\mathbf{A}}\mathbf{t} \leq \widetilde{\mathbf{d}}'\right\} \\ &\Leftrightarrow (\text{Lemma 3.2}) \{\mathbf{t} \,|\mathbf{At} \leq \mathbf{d}\} = \{\mathbf{t} \,|\mathbf{A}'\mathbf{t} \leq \mathbf{d}'\} \\ &\Leftrightarrow \text{feasible regions of } C \text{ and } C' \text{ are identical}\end{aligned}$$

In other words, there is a one-to-one correspondence between an all-pairs shortest paths matrix of a timing constraint set and its feasible region. Therefore, the constraint matrix that represents the most stringent constraints among all pairs of events uniquely characterizes the shape of the feasible region. We define this as the normal form of the constraint set.

Definition 3.3 (*Constraint Set Normal Form*). Given a timing constraint set C and the corresponding constraint graph G, its all-pairs shortest paths matrix, denoted as $\mathbf{D}^*$, where

$$\mathbf{D}^* = \begin{bmatrix} 0 & d^*_{1,2} & \cdots & d^*_{1,n} \\ d^*_{2,1} & 0 & \cdots & d^*_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d^*_{n,1} & d^*_{n,2} & \cdots & 0 \end{bmatrix} \tag{11}$$

and $d^*_{i,j}$ is the shortest path weight between $t(e_i)$ and $t(e_j)$ in the constraint graph G. $\mathbf{D}^*$ is called constraint set C's *normal form*.

The timing constraint set normal form bridges the geometric problem of a feasible region and their corresponding algebraic problem of linear inequalities. We can hence derive the relationship between feasible regions of two different constraint sets, such as inclusion, by studying the relationship between their normal forms.

THEOREM 3.4. *Given two sets of real-time constraints* $C = \mathbf{At} \leq \mathbf{d}$ *and* $C' = \mathbf{A}'\mathbf{t} \leq \mathbf{d}'$ *on the same set of events*[2]. *Let their corresponding normal forms be* $\mathbf{D}^*$ *and* $\mathbf{D}'^*$, *respectively. The feasible region of* C' *is included within that of* C *if and only if* $\mathbf{D}^* \geq \mathbf{D}'^*$, *that is,* $\forall i, j : d^*_{i,j} \geq d'^*_{i,j}$.

PROOF. Note that the feasible region of $\mathbf{A}'\mathbf{t} \leq \mathbf{d}'$ is included in that of $\mathbf{At} \leq \mathbf{d}$ iff the feasible region of $\begin{bmatrix}\mathbf{A} \\ \mathbf{A}'\end{bmatrix}\mathbf{t} \leq \begin{bmatrix}\mathbf{d} \\ \mathbf{d}'\end{bmatrix}$ is same as that of $\mathbf{A}'\mathbf{t} \leq \mathbf{d}'$. Hence, we can instead prove that the feasible regions of $\begin{bmatrix}\mathbf{A} \\ \mathbf{A}'\end{bmatrix}\mathbf{t} \leq \begin{bmatrix}\mathbf{d} \\ \mathbf{d}'\end{bmatrix}$ and $\mathbf{A}'\mathbf{t} \leq \mathbf{d}'$ are the same iff $\mathbf{D}^* \geq \mathbf{D}'^*$. In

[2]Note that the event sets of the two constraint sets need not be the same in order for the two feasible regions to be comparable. One can always extend both event sets to the same one by adding unconstrained events.

the following, for a constraint normal form $\mathbf{D}^*$, we use its equivalent system of linear inequalities representation $\widetilde{\mathbf{A}}\mathbf{t} \leq \widetilde{\mathbf{d}}$ where $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{d}}$ are defined in (2).

(i) $\mathbf{D}^* \geq \mathbf{D}'^* \Rightarrow C$ includes C': Suppose we have $\mathbf{D}^* \geq \mathbf{D}'^*$, that is, $\widetilde{\mathbf{d}} \geq \widetilde{\mathbf{d}}'$, it is not hard to see that $\begin{bmatrix} \widetilde{\mathbf{A}} \\ \widetilde{\mathbf{A}} \end{bmatrix} \mathbf{t} \leq \begin{bmatrix} \widetilde{\mathbf{d}} \\ \widetilde{\mathbf{d}}' \end{bmatrix}$ has the same feasible region as $\widetilde{\mathbf{A}}\mathbf{t} \leq \widetilde{\mathbf{d}}'$. Then, from Lemma 3.2, $\begin{bmatrix} \mathbf{A} \\ \mathbf{A}' \end{bmatrix} \mathbf{t} \leq \begin{bmatrix} \mathbf{d} \\ \mathbf{d}' \end{bmatrix}$ has the same feasible region as $\mathbf{A}'\mathbf{t} \leq \mathbf{d}'$.

(ii) C includes $C' \Rightarrow \mathbf{D}^* \geq \mathbf{D}'^*$: If $\begin{bmatrix} \mathbf{A} \\ \mathbf{A}' \end{bmatrix} \mathbf{t} \leq \begin{bmatrix} \mathbf{d} \\ \mathbf{d}' \end{bmatrix}$ has the same feasible region as $\mathbf{A}'\mathbf{t} \leq \mathbf{d}'$, then from Lemma 3.2, $\begin{bmatrix} \widetilde{\mathbf{A}} \\ \widetilde{\mathbf{A}} \end{bmatrix} \mathbf{t} \leq \begin{bmatrix} \widetilde{\mathbf{d}} \\ \widetilde{\mathbf{d}}' \end{bmatrix}$ has the same feasible region as $\widetilde{\mathbf{A}}'\mathbf{t} \leq \widetilde{\mathbf{d}}'$. To prove the necessary condition, we should have $\mathbf{D}^* \geq \mathbf{D}'^*$, that is, $\widetilde{\mathbf{d}} \geq \widetilde{\mathbf{d}}'$. Assume to the contrary that there is some $d^*_{i,j}$ in $\widetilde{\mathbf{d}}$ and $d'^*_{i,j}$ in $\widetilde{\mathbf{d}}'$ such that $d^*_{i,j} < d'^*_{i,j}$. Since $d'^*_{i,j}$ is the optimal solution to the linear program

$$\begin{aligned} &\textbf{maximize} \quad t(e_i) - t(e_j) \\ &\textbf{subject to} \quad \widetilde{\mathbf{A}}\mathbf{t} \leq \widetilde{\mathbf{d}}' \end{aligned} \tag{12}$$

and thus the optimal solution to the linear program (13)

$$\begin{aligned} &\textbf{maximize} \quad t(e_i) - t(e_j) \\ &\textbf{subject to} \quad \begin{bmatrix} \widetilde{\mathbf{A}} \\ \widetilde{\mathbf{A}} \end{bmatrix} \mathbf{t} \leq \begin{bmatrix} \widetilde{\mathbf{d}} \\ \widetilde{\mathbf{d}}' \end{bmatrix}. \end{aligned} \tag{13}$$

However, the optimal solution to (13) can be at most $d^*_{i,j}$ when the solution set of $\begin{bmatrix} \widetilde{\mathbf{A}} \\ \widetilde{\mathbf{A}} \end{bmatrix} \mathbf{t} \leq \begin{bmatrix} \widetilde{\mathbf{d}} \\ \widetilde{\mathbf{d}}' \end{bmatrix}$ is not empty. The contradiction implies that $\mathbf{D}^* \geq \mathbf{D}'^*$. □

From the preceding discussions, we can see that the constraint normal form bridges the geometric problem of a feasible region and their corresponding algebraic problem of linear inequalities and can serve as the algebraic representation that we stated earlier in this section. We can hence derive the similarity relationship between feasible regions of two different constraint sets by studying the constraint normal forms.

4. SIMILARITIES BETWEEN TIMING CONSTRAINT SETS

As discussed in Section 1, actual runtime performance of soft real-time systems may deviate from their initial designs, either inevitably due to unpredictable factors or by intention to improve system's other Quality-of-Service (QoS) properties. It is hence important to know quantitatively how much the timing behavior compromise is in these cases.

4.1 Similarities between Constraint Sets

In this section, we focus on quantifying timing behavior similarities and we base our model on the feasible regions of timing constraint sets discussed in Section 3. The following example of the similarities between feasible regions in 2 and 3 dimensions gives the intuition. Note that in the following discussions, for simplicity, we assume that event occurrence times allowed by a set of constraints are uniformly distributed in the feasible region of the constraint set.

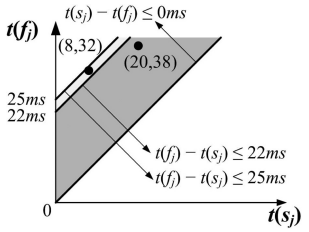


Fig. 3. The feasible regions satisfying constraint $0 < t(f_j) - t(s_j) \leq 22$ and $0 < t(f_j) - t(s_j) \leq 25$.

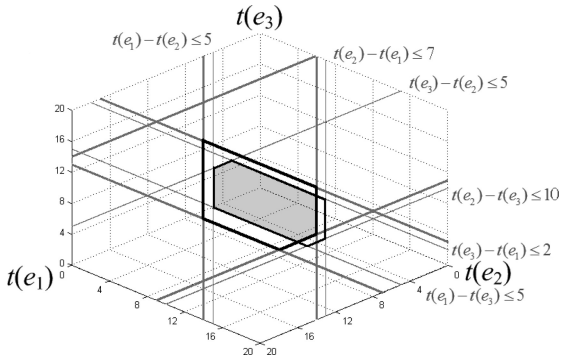


Fig. 4. The feasible regions satisfying constraint sets (1) (bold lines) and (14) (light lines), and their intersection (the shaded region).

Example 4 (*Feasible Region Similarity*). In Example 1, the original constraint was $0 < t(f_j) - t(s_j) \leq 22$ and the relaxed one is $0 < t(f_j) - t(s_j) \leq 25$. The relationship between the two corresponding feasible regions is depicted in Figure 3.

As can be seen from the figure, timed data stream $((s_j, 20), (f_j, 38))$ satisfies both constraint sets while $((s_j, 8), (f_j, 32))$ satisfies only the relaxed deadline. In fact, the common area of the two feasible regions occupies $\frac{22}{25} = 88\%$ of that of the relaxed deadline 25ms.

Advancing to 3-dimensional feasible regions, consider the feasible region of the following timing constraint set that has three events.

$$\left\{ \begin{array}{l} t(e_1) - t(e_2) \leq 5,\ t(e_2) - t(e_1) \leq 7, \quad t(e_1) - t(e_3) \leq 5, \\ t(e_3) - t(e_1) \leq 2,\ t(e_2) - t(e_3) \leq 10,\ t(e_3) - t(e_2) \leq 5 \end{array} \right\} \tag{14}$$

The relationship between feasible regions satisfying constraint sets (1) and (14) is illustrated in Figure 4, where bold lines, light lines, and the shaded region represent constraint sets (1), (14), and the intersection between their feasible regions, respectively.

From Figure 4, we can see that although feasible regions satisfying constraint sets (1) and (14) are not identical, they share some common region. Hence, we can expect that they have some timing behaviors in common.

Generalizing the previous discussions, we define the similarity between two timing constraint sets as the following.

Definition 4.1 (*Constraint Set Similarity*). Let $vol(C)$ denote the volume of the feasible region of a timing constraint set C. Given two timing constraint sets C and C',

the similarity relation is defined as $C \sim C' = \frac{vol(C^{\cap})}{vol(C')}$, where $C^{\cap}$ is the intersection of C and C'.

Intuitively, if $C \sim C' = P\%$, that is, the intersection of the feasible regions of constraint sets C and C' occupies $P\%$ of the feasible region of C', we know that $P\%$ of all the timed data streams satisfying C' satisfies C. Therefore, system satisfying C' will have a $P\%$ guarantee of satisfying C. Unfortunately, directly calculating the similarity between two sets of complete timing constraints is difficult. In fact, calculating the size of a polytope formed by a set of linear inequalities ($vol(C)$ in our context) has been shown to be $\#P$-hard [Dyer and Frieze 1988], and thus directly calculating the proportions of the intersection in any of the feasible regions, that is, the similarity metric, by comparing their sizes is costly. To overcome the computational impediment of evaluating directly the constraint set similarity between two constraint sets, we resort to finding a lower bound on the constraint set similarity that is easily computable. The following theorem defines such a bound.

THEOREM 4.2. *Given two timing constraint sets C and C', and corresponding normal forms be $\mathbf{D}^*$ and $\mathbf{D}'^*$, respectively. If the feasible region of C' is not included in that of C, that is, $\mathbf{D}^* \not\geq \mathbf{D}'^*$, then the similarity is bounded by*

$$\left(\inf_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{i,j} \leq d'^*_{i,j}}} \left\{ \frac{d^*_{i,j}}{d'^*_{i,j}} \right\} \right)^{|E|-1} \leq C \sim C' < 1, \tag{15}$$

where $|E|$ is the cardinality of the event set being constrained, $d^*_{i,j}$ and $d'^*_{i,j}$ are the corresponding entries in $\mathbf{D}^*$ and $\mathbf{D}'^*$, respectively. The similarity reaches upper bound 1 when feasible region of C' is included in that of C, that is, $\mathbf{D}^* \geq \mathbf{D}'^*$.

PROOF PRELIMINARY. Before giving the formal proof for the theorem, we briefly introduce some background in computing volumes of high-dimension polytopes. In Lawrence [1991], Lawrence gives an algorithm for computing polytope volume based on a combinatorial form of Gram's relation. A convex polytope P is given as

$$P = \left\{ \mathbf{x} \in \Re^n : \mathbf{x} \geq \mathbf{0}, \mathbf{A}\mathbf{x} \leq \mathbf{b} \right\}, \tag{16}$$

where $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{b}$ is a column vector in $\Re^m$ which has only nonnegative entries. If P is a simple (or nondegenerate)[3] polytope, then the volume of P can be derived by extracting parameters from basic feasible tableaux for the following linear programming problem

$$\begin{aligned} &\textbf{maximize} && f(\mathbf{x}) \\ &\textbf{subject to} && \mathbf{A}\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{17}$$

[3]An n-dimensional simple (or nondegenerate) polytope is a polytope where each vertex is the intersection of exactly n hyperplanes (defined by n of the m inequalities in (16) with $\leq$ replaced by $=$). This requirement is inherited from vertex enumeration algorithms used in the volume computation algorithm. Although feasible regions in this article are sometimes not nondegenerate, the requirement can be dropped by perturbing the auxiliary hyperplanes by a very small amount and the same result holds.

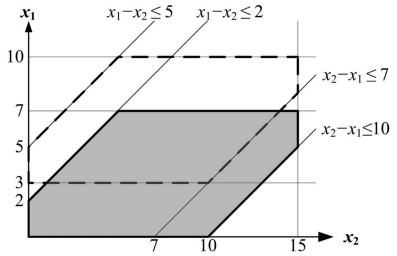


Fig. 5. Fixing $t(e_3)$ at 5 in (14) can be interpreted as using the plane $t(e_3) = 5$ to cut the feasible region of (14) in Figure 4 and view the slice in the $\mathbf{x}_1\mathbf{x}_2$ ($t(e_1)t(e_2)$) plane.

where $f(\mathbf{x}) = \mathbf{c}^t\mathbf{x} + \mathbf{d}$ can be any function that is not constant on any one of the hyperplanes defining P. The volume of P, denoted as $vol(P)$, is thus computed as [Lawrence 1991]

$$vol(P) = \sum_{\text{all vertices } v \text{ of } P} \frac{1}{n!}\frac{1}{\delta_v}\frac{\tilde{d}^n}{\gamma_{i_1}\cdots\gamma_{i_n}}, \tag{18}$$

where δ_v is the cumulative product of the pivot elements, $\gamma_{i_1}, \cdots, \gamma_{i_n}$ are the nonbasic feasible solutions, and $\tilde{d}$ is the value of the objective function. The pivot elements, nonbasic feasible solutions, and the values of the objective function can all be retrieved from the basic feasible tableaux for (17).

To apply the volume computation algorithm in our setting, the following issues need to be addressed.

(1) As can be seen in Figure 2, the feasible region formed by the intersections of half spaces corresponding to timing constraints is not bounded. We need to find a way to bound the feasible region.
(2) The volume computation algorithm crucially relies upon simplex-pivoting-based vertex enumeration algorithms. However, McMullen and Shepard [1971] have shown that the maximum number of vertices a polytope defined by m inequalities on n nonnegative variables can have is $\binom{m+\lfloor n/2\rfloor}{m} + \binom{m+\lceil n/2\rceil+1}{m}$. Therefore, in (18), summing for "all vertices v of P" generally has exponential cost. In fact, Dyer and Frieze [1988] have shown that computing $vol(P)$ is $\#P$-hard. Therefore, in our context, directly calculating the proportion of the intersection in any of the feasible regions by comparing the volumes is costly. We need to find a way to utilize the special properties of the feasible regions being compared.

The first issue can be addressed by "fixing" one of the n timers of events $e_1, \ldots, e_n$. As the selection of the "fixed" one does not change the ratio of the intersection in any of the feasible regions, without loss of generality, we choose to fix $t(e_n)$ at $d = \max_{i=1,\ldots,n-1} d^*_{n,i}$(in fact, as can be seen from (2), d could be any value larger than $\max_{i=1,\ldots,n-1} d^*_{n,i}$). The geometric interpretation for this is that the hyperplane $t(e_n) = d$ is used to "cut" the feasible region so that the resulting region is bounded in $\Re^{n-1}$. □

Example 5. For example, in (14), if we let $t(e_3) = 5$, the constraint set becomes

$$\left\{ \begin{array}{l} t(e_1) - t(e_2) \le 5, \quad t(e_2) - t(e_1) \le 7, \\ t(e_1) \le 10, \quad t(e_1) \ge 3, \quad t(e_2) \le 15 \end{array} \right\} \tag{19}$$

which is illustrated as dash line segments in Figure 5.

Since the volume computation algorithm requires that the origin in $\Re^n$ is a vertex of the polytope[4], we shift the feasible region to the origin (illustrated as the gray region in Figure 5), the constraint set will become

$$\left\{ \begin{array}{ll} t(e_1) - t(e_2) \leq 2, & t(e_2) - t(e_1) \leq 10, \\ t(e_1) \leq 7, & t(e_2) \leq 15 \end{array} \right\}. \tag{20}$$

As can be easily seen from the figure, the area of the feasible region is 80. To gain some insights into the volume computation algorithm which will facilitate our proof of Theorem 4.2, we illustrate the derivation of the area using the algorithm as follows.

Adopting the volume computation algorithm on the bounded feasible region, we choose the objective function $f((t(e_1), t(e_2))^T) = t(e_1) + t(e_2)$ and have the initial tableau

$$\left[\begin{array}{cccccc|l} \boxed{1} & -1 & 1 & 0 & 0 & 0 & 2 = d'^*_{3,1} + d'^*_{1,2} - d'^*_{3,2} \\ -1 & 1 & 0 & 1 & 0 & 0 & 10 = d'^*_{3,2} + d'^*_{2,1} - d'^*_{3,1} \\ 1 & 0 & 0 & 0 & 1 & 0 & 7 = d'^*_{3,1} + d'^*_{1,3} \\ 0 & 1 & 0 & 0 & 0 & 1 & 15 = d'^*_{3,2} + d'^*_{2,3} \\ \hline -1 & -1 & 0 & 0 & 0 & 0 & 0 \end{array} \right] \tag{21}$$

where we have $n = 3 - 1 = 2$, $\delta_v = 1$ (the initial pivot element enclosed in a box in (21)), the nonbasic feasible solutions are the two nonnegative numbers $(-1,-1)$ in the lower left partition of the tableau, and $\tilde{d} = 0$ (the lower right partition of the tableau, that is, the value of the objective function at the current vertex (0,0)). Then the first element of the summation in (18) is $\frac{1}{2!}\frac{1}{1}\frac{0^2}{(-1)(-1)} = 0$. Continuing pivoting at $(t(e_1), t(e_2)) = (2, 0)$ using the 1^{st} row, we have

$$\left[\begin{array}{cccccc|l} 1 & -1 & 1 & 0 & 0 & 0 & 2 = d'^*_{3,1} + d'^*_{1,2} - d'^*_{3,2} \\ 0 & 0 & 1 & 1 & 0 & 0 & 12 = d'^*_{1,2} + d'^*_{2,1} \\ 0 & \boxed{1} & -1 & 0 & 1 & 0 & 5 = d'^*_{1,3} + d'^*_{3,2} - d'^*_{1,2} \\ 0 & 1 & 0 & 0 & 0 & 1 & 15 = d'^*_{3,2} + d'^*_{2,3} \\ \hline 0 & -2 & 1 & 0 & 0 & 0 & 2 = d'^*_{3,1} + d'^*_{1,2} - d'^*_{3,2} \end{array} \right] \tag{22}$$

and $\frac{1}{2!}\frac{1}{1}\frac{2^2}{(-2)(1)} = -1$. Pivoting at $(t(e_1), t(e_2)) = (7, 5)$ using the 3rd row, we have

$$\left[\begin{array}{cccccc|l} 1 & 0 & 0 & 0 & 1 & 0 & 7 = d'^*_{1,3} + d'^*_{3,1} \\ 0 & 0 & 1 & 1 & 0 & 0 & 12 = d'^*_{1,2} + d'^*_{2,1} \\ 0 & 1 & -1 & 0 & 1 & 0 & 5 = d'^*_{1,3} + d'^*_{3,2} - d'^*_{1,2} \\ 0 & 0 & \boxed{1} & 0 & -1 & 1 & 10 = d'^*_{1,2} + d'^*_{2,3} - d'^*_{1,3} \\ \hline 0 & 0 & -1 & 0 & 2 & 0 & 12 = 2d'^*_{1,3} + d'^*_{3,1} + d'^*_{3,2} - d'^*_{1,2} \end{array} \right] \tag{23}$$

[4]This requirement can be discarded by lexicographic techniques for handling primal degeneracy in linear programming.

and $\frac{1}{2!}\frac{1}{1}\frac{12^2}{(-1)(2)} = -36$. Pivoting at $(t(e_1), t(e_2)) = (7, 15)$ using the 4th row, we have

$$\left[\begin{array}{cccccc|l} 1 & 0 & 0 & 0 & 1 & 0 & 7 = d'^*_{1,3} + d'^*_{3,1} \\ 0 & 0 & 0 & 1 & \boxed{1} & -1 & 2 = d'^*_{2,1} + d'^*_{1,3} - d'^*_{2,3} \\ 0 & 1 & 0 & 0 & 0 & 1 & 15 = d'^*_{2,3} + d'^*_{3,2} \\ 0 & 0 & 1 & 0 & -1 & 1 & 10 = d'^*_{1,2} + d'^*_{2,3} - d'^*_{1,3} \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 22 = d'^*_{1,3} + d'^*_{3,1} + d'^*_{2,3} + d'^*_{3,2} \end{array}\right] \tag{24}$$

and $\frac{1}{2!}\frac{1}{1}\frac{22^2}{(1)(1)} = 242$. Pivoting at $(t(e_1), t(e_2)) = (5, 15)$ using the 2nd row, we have

$$\left[\begin{array}{cccccc|l} 1 & 0 & 0 & -1 & 0 & \boxed{1} & 5 = d'^*_{2,3} + d'^*_{3,1} - d'^*_{2,1} \\ 0 & 0 & 0 & 1 & 1 & -1 & 2 = d'^*_{2,1} + d'^*_{1,3} - d'^*_{2,3} \\ 0 & 1 & 0 & 0 & 0 & 1 & 15 = d'^*_{2,3} + d'^*_{3,2} \\ 0 & 0 & 1 & 1 & 0 & 0 & 12 = d'^*_{2,1} d'^*_{1,2} \\ \hline 0 & 0 & 0 & -1 & 0 & 2 & 20 = 2d'^*_{2,3} + d'^*_{3,2} + d'^*_{3,1} - d'^*_{2,1} \end{array}\right] \tag{25}$$

and $\frac{1}{2!}\frac{1}{1}\frac{20^2}{(-1)(2)} = -100$. Pivoting at $(t(e_1), t(e_2)) = (0, 10)$ using the 1st row, we have

$$\left[\begin{array}{cccccc|l} 1 & 0 & 0 & -1 & 0 & 1 & 5 = d'^*_{2,3} + d'^*_{3,1} - d'^*_{2,1} \\ 1 & 0 & 0 & 0 & 1 & 0 & 7 = d'^*_{3,1} + d'^*_{1,3} \\ -1 & 1 & 0 & \boxed{1} & 0 & 0 & 10 = d'^*_{3,2} + d'^*_{2,1} - d'^*_{3,1} \\ 0 & 0 & 1 & 1 & 0 & 0 & 12 = d'^*_{2,1} + d'^*_{1,2} \\ \hline -2 & 0 & 0 & 1 & 0 & 0 & 10 = d'^*_{3,2} + d'^*_{2,1} - d'^*_{3,1} \end{array}\right] \tag{26}$$

and $\frac{1}{2!}\frac{1}{1}\frac{10^2}{(-2)(1)} = -25$. Pivoting at $(t(e_1), t(e_2)) = (0, 0)$ using the 3rd row, we have

$$\left[\begin{array}{cccccc|l} 0 & 1 & 0 & 0 & 0 & 1 & 15 = d'^*_{3,2} + d'^*_{2,3} \\ 1 & 0 & 0 & 0 & 1 & 0 & 7 = d'^*_{3,1} + d'^*_{1,3} \\ -1 & 1 & 0 & 1 & 0 & 0 & 10 = d'^*_{3,2} + d'^*_{2,1} - d'^*_{3,1} \\ 1 & -1 & 1 & 0 & 0 & 0 & 2 = d'^*_{3,1} + d'^*_{1,2} - d'^*_{3,2} \\ \hline -1 & -1 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \tag{27}$$

and $\frac{1}{2!}\frac{1}{1}\frac{0^2}{(-1)(-1)} = 0$. Therefore, from (18), the area of the slice is $0 - 1 - 36 + 242 - 100 - 25 + 0 = 80$, which conforms to our early observation. Similarly, the corresponding areas of the feasible region of (1) and its intersection with that of (14) are 112 and 73, respectively[5].

[5]As can be seen from (21) to (27), the linear combinations of $d^*_{i,j}$'s on the right side of each tableau correspond to cycles in the constraint graph of the timing constraint set. In fact, as shown in Provan's algorithm [Provan 1994] for enumerating vertices of a polytope related to a network linear program, the hyperplanes of the polytope P adjacent to a vertex v is in one-to-one correspondence with simple cycles of a directed graph modified (with respect to v) from the directed graph defined by the network linear program. Therefore, although a network linear program is significantly simpler than the general linear program as in (17), the number of terms in the summation (18) is still generally exponential. Therefore, deriving exact similarity ratio is of exponential cost.

From Example 5, we have the following observations that are crucial in our proof of Theorem 4.2.

Observation 1. Pivoting is essentially a Gaussian elimination, thus the value of the objective function at any pivoting step must be a linear combination of $d^*_{i,j}$'s.

Observation 2. As the corresponding hyperplanes of different timing constraint sets are parallel, the pivoting sequences for enumeration vertices of the two feasible regions are the same regardless of the value of $d^*_{i,j}$'s. This property overcomes the second issues identified earlier.

We now prove Theorem 4.2 based on these observations.

PROOF. Given constraint sets C and C' whose normal forms are $\mathbf{D}^*$ and $\mathbf{D}'^*$, respectively. Let $C^\cap$ denote the intersection of C and C' and $\mathbf{D}^{\cap *}$ denote its normal form. We define a new constraint set C'' whose normal form $\mathbf{D}''^*$ is

$$\mathbf{D}''^* = \sup_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{i,j}\leq d'^*_{i,j}}} \left\{\frac{d'^*_{i,j}}{d^*_{i,j}}\right\} \cdot \mathbf{D}^{\cap *}. \tag{28}$$

Since the constraint graph of $C^\cap$ comprises of edges from either C or C', for any entry $d^{\cap *}_{i,j}$ of $\mathbf{D}^{\cap *}$, we have

$$d^{\cap *}_{i,j} = d^{(')*}_{i,k_1} + d^{(')*}_{k_1,k_2} + \cdots + d^{(')*}_{k_{t-1},k_t} + d^{(')*}_{k_t,j}, \tag{29}$$

where $d^{(')*}_{k_s,k_{s+1}}$ denotes either $d^*_{k_s,k_{s+1}}$ or $d'^*_{k_s,k_{s+1}}$. Therefore, from (28), when $\sup_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{i,j}\leq d'^*_{i,j}}} \left\{\frac{d'^*_{i,j}}{d^*_{i,j}}\right\} \geq 1$, we have

$$d''^*_{i,j} = \left(d^{(')*}_{i,k_1} + \cdots + d^{(')*}_{k_t,j}\right) \cdot \sup_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{i,j}\leq d'^*_{i,j}}} \left\{\frac{d'^*_{i,j}}{d^*_{i,j}}\right\}. \tag{30}$$

If $d^{(')*}_{k_s,k_{s+1}}$ is $d^*_{k_s,k_{s+1}}$, since $\sup_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{i,j}\leq d'^*_{i,j}}} \left\{\frac{d'^*_{i,j}}{d^*_{i,j}}\right\} \geq \frac{d'^*_{k_s,k_{s+1}}}{d^*_{k_s,k_{s+1}}}$ we have

$$d^{(')*}_{k_s,k_{s+1}} \cdot \sup_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{i,j}\leq d'^*_{i,j}}} \left\{\frac{d'^*_{i,j}}{d^*_{i,j}}\right\} \geq d'^*_{k_s,k_{s+1}} \tag{31}$$

and if $d^{(')*}_{k_s,k_{s+1}}$ is $d'^*_{k_s,k_{s+1}}$, as $\sup_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{i,j}\leq d'^*_{i,j}}} \left\{\frac{d'^*_{i,j}}{d^*_{i,j}}\right\} \geq 1$, (31) also holds. Thus, from (30) and (31), we have

$$d''^*_{i,j} \geq d'^*_{i,k_1} + d'^*_{k_1,k_2} + \cdots + d'^*_{k_{t-1},k_t} + d'^*_{k_t,j} \geq d'^*_{i,j}, \tag{32}$$

that is, $\mathbf{D}''^* \geq \mathbf{D}'^*$. Therefore, from Theorem 3.4, the feasible region of C' is included within that of C'' and thus

$$vol(C'')/vol(C') \geq 1. \tag{33}$$

We now use the volume computation algorithm to calculate the ratio between $vol(C^\cap)$ and $vol(C'')$. From Observation 1, at each pivoting step, the value of the

objective function is a linear combination of $d^*_{i,j}$'s, that is, at the k'th pivoting step, the values of the objective functions for $vol(C^{\cap})$ and $vol(C'')$ are

$$\sum_{\substack{i,j=1,\ldots,n,\\ i\neq j}} a^{\cap(k)}_{i,j} \cdot d^{\cap *}_{i,j} \quad \text{and} \quad \sum_{\substack{i,j=1,\ldots,n,\\ i\neq j}} a''^{(k)}_{i,j} \cdot d''^{*}_{i,j} \tag{34}$$

respectively. From Observation 2, since the pivoting sequence for deriving $vol(C^{\cap})$ and $vol(C'')$ are the same, we have

$$\forall i, j, k : a^{\cap(k)}_{i,j} = a''^{(k)}_{i,j}. \tag{35}$$

Moreover, from (28), we have

$$\forall i, j = 1, \ldots, n, i \neq j : \frac{d^{\cap *}_{i,j}}{d''^{*}_{i,j}} = \inf_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{i,j}\leq d'^*_{i,j}}} \left\{ \frac{d^*_{i,j}}{d'^*_{i,j}} \right\}. \tag{36}$$

Therefore, from (35) and (36), we have

$$\frac{\sum_{\substack{i,j=1,\ldots,n,\\ i\neq j}} a^{\cap(k)}_{i,j} \cdot d^{\cap *}_{i,j}}{\sum_{\substack{i,j=1,\ldots,n,\\ i\neq j}} a''^{(k)}_{i,j} \cdot d''^{*}_{i,j}} = \frac{d^{\cap *}_{1,2}}{d''^{*}_{1,2}} = \cdots = \frac{d^{\cap *}_{n,n-1}}{d''^{*}_{n,n-1}} = \inf_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{i,j}\leq d'^*_{i,j}}} \left\{ \frac{d^*_{i,j}}{d'^*_{i,j}} \right\}. \tag{37}$$

From Observation 2, we also know that at each pivoting step, the cumulative products of pivoting elements and the nonbasic feasible solutions for deriving $vol(C^{\cap})$ and $vol(C'')$ are the same. Therefore, from (18) and (37), we have

$$\frac{vol(C^{\cap})}{vol(C'')} = \frac{\sum_{\forall v\in C^{\cap}} \frac{1}{n!}\frac{1}{\delta_v} \frac{\left(\sum a^{\cap(k)}_{i,j}\cdot d^{\cap *}_{i,j}\right)^{|E|-1}}{\gamma_{i_1}\cdots\gamma_{i_n}}}{\sum_{\forall v\in C''} \frac{1}{n!}\frac{1}{\delta_v} \frac{\left(\sum a''^{(k)}_{i,j}\cdot d''^{*}_{i,j}\right)^{|E|-1}}{\gamma_{i_1}\cdots\gamma_{i_n}}} = \left(\inf_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{i,j}\leq d'^*_{i,j}}} \left\{ \frac{d^*_{i,j}}{d'^*_{i,j}} \right\} \right)^{|E|-1}. \tag{38}$$

Finally, from (33) and (38), we have

$$\frac{vol(C^{\cap})}{vol(C')} = \frac{vol(C^{\cap})}{vol(C'')} \cdot \frac{vol(C'')}{vol(C')} \geq \left(\inf_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{i,j}\leq d'^*_{i,j}}} \left\{ \frac{d^*_{i,j}}{d'^*_{i,j}} \right\} \right)^{|E|-1}. \tag{39}$$

□

From Theorem 4.2, one can see that the similarity lower bound can be calculated easily once the normal forms of the constraint sets are available. Comparing similarities of different constraint set pairs then can be indirectly achieved through evaluating their similarity bounds. Before discussing various implications of using the similarity bound in Section 4.2, we demonstrate the use of Theorem 4.2 on the constraint sets given in Example 4. From Theorem 4.2, the ratio of the common region between (1) and (14) to the feasible region of (14) is bounded by $\left[\frac{36}{49}, 1\right)$ where $\frac{36}{49} = \left(\inf\left\{\frac{6}{7}, \frac{9}{10}\right\}\right)^{3-1}$. Therefore, we assume a uniform distribution of the event timing

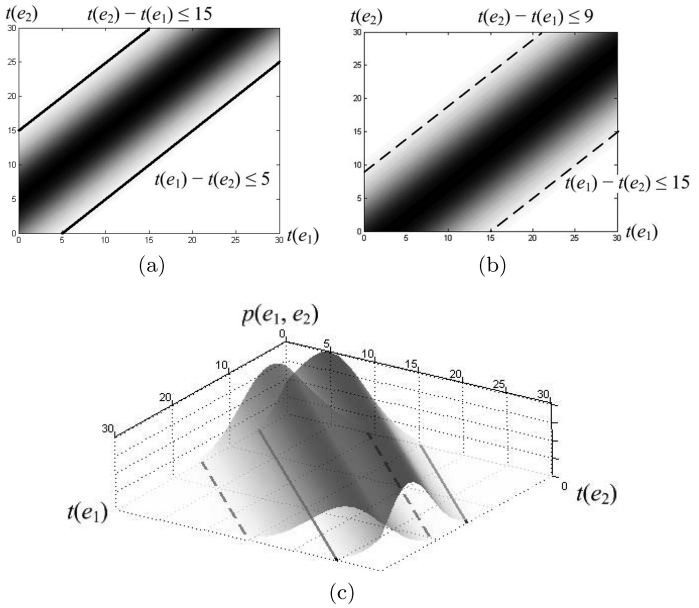


Fig. 6. Feasible region similarities of nonuniformly distributed timed data streams.

behavior in the feasible regions, Theorem 4.2 guarantees that at least $\frac{36}{49} = 73\%$ timed data streams that satisfy (14) also satisfy (1). This gives us a quantitative measure of the resemblance between systems constrained by (1) and (14), respectively. Actually, as shown in Example 5, the exact ratio of the common region between (1) and (14) to the feasible region of (14) is $\frac{73}{80} = 91.25\%$.

4.2 Discussions

4.2.1 Timed Data Stream Distribution in the Feasible Region. In the preceding discussions, we assume that timed data streams are uniformly distributed in the feasible region of the constraint set. The bound given in Theorem 4.2 is based on such an assumption. However, the definition of constraint feasible region similarities can be extended to nonuniform cases. For example, consider two 2-dimensional feasible regions of constraint sets $C = \{t(e_1) - t(e_2) \leq 5, t(e_2) - t(e_1) \leq 15\}$ and $C' = \{t(e_1) - t(e_2) \leq 15, t(e_2) - t(e_1) \leq 9\}$. Assuming timed data streams are not uniformly distributed in the regions, but are as shown in Figure 6(a) and 6(b), respectively. Obviously, in order to compare their similarities, not only their areas but also the densities within the areas must be considered. For instance, the intersection of the feasible regions of C and C' is denser than the complements of the regions as depicted in Figure 6(c). Therefore, when calculating the similarity of $\mathcal{S}(C)$ based on Definition 4.1, we must take into account distribution density of timed data stream within its feasible regions formed by a given constraint set.

In a soft real-time system, the distribution of timing values (such as the completion time of a task) can be evaluated by methods presented in existing work, for example, Tia et al. [1995], Kalavade and Moghé [1998], yi Huang and Liu [1995], and Li [1996]. The distribution can then be used in combination with our proposed similarity bound concept to compare timing behaviors of different designs. The detail of this is beyond the scope of this article.

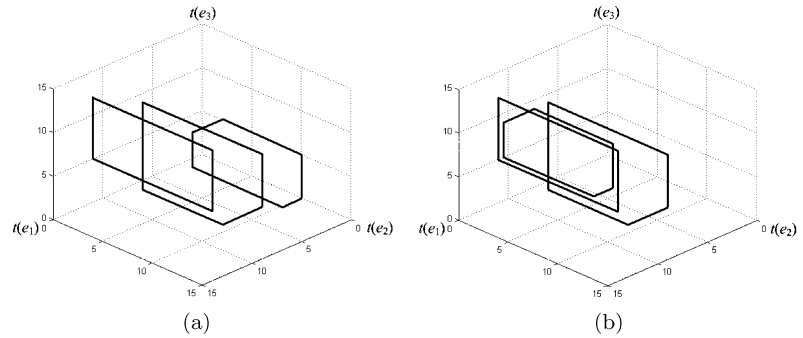


Fig. 7. Similarity relation is not transitive.

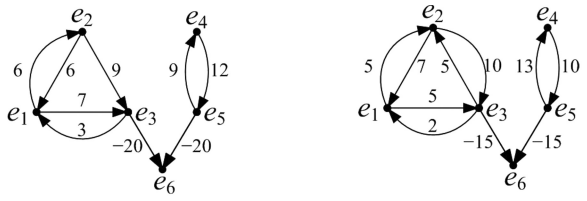
4.2.2 Symmetry and Transitivity of Constraint Set Similarity. It is worth pointing out that the constraint set similarity relation is neither symmetric nor transitive. From Definition 4.1, it is not hard to see that in general $C \sim C' \neq C' \sim C$. For instance, for constraint sets $C = \{0 < t(f_j) - t(s_j) \leq 22\}$ and $C' = \{0 < t(f_j) - t(s_j) \leq 25\}$ as given in Example 4, $C \sim C' = 88\%$, while $C' \sim C = 1$.

Similarly, neither can we infer $C \sim C''$ from $C \sim C'$ and $C' \sim C''$. Figure 7 shows an example. In the figure, the feasible regions of three constraint sets C, C', and C'' are represented as a tetragon, a pentagon, and a hexagon, respectively. The similarity between C and C' ($C \sim C'$) is the same for both figure Figure 7(a) and Figure 7(b). However, depending on the positions from which C'' similar to C', C and C'' can be either very similar (as shown in Figure 7(b)) or very dissimilar (as shown in Figure 7(a)).

4.2.3 The Tightness of the Similarity Bound. From Theorem 4.2, it is easy to see that as the dimension of feasible regions gets higher, the similarities between their corresponding constraint sets decrease significantly due to the exponent $|E| - 1$. This is quite intuitive since, on one hand, as more events and constraints get involved, the chance of timed data streams satisfying one constraint set but violating the other gets bigger; on the other hand, from a geometric point of view, the volume of a polytope is exponential to its dimension, and so does the similarity between two polytopes.

4.2.4 Dealing with Unconstrained Event Pairs in a Constraint Set. In Example 4, we illustrate the similarities between timing constraint sets where there is a constraint, either explicit or implicit, for every pair of events. However, there are cases where there are event pairs which are not constrained. For example, for constraint sets $C_1 = \{-5 \leq t(e_2) - t(e_1) \leq 22\}$ and $C_2 = \{t(e_2) - t(e_1) \leq 25\}$, the similarity $C_1 \sim C_2$ is close to 0 since in C_2 we implicitly have $t(e_1) - t(e_2) \leq +\infty$ and the feasible region is not bounded on the corresponding direction. In this case, the similarity relation stated in Theorem 4.2 still applies, but as it approaches to 0 ($C_1 \sim C_2 = \inf\{\frac{22}{25}, \frac{5}{+\infty}\} = 0$), such 0 similarities render the metric too coarse. Hence, a refinement that considers unconstrained events is needed.

For most real-time applications, we observe that events often form groups such that those within the same group are pairwisely constrained either explicitly or implicitly as shown in Section 4.1, and the timing relations between groups are either nonexistent or constrained by unidirectional constraints such as precedence constraints or delays. Therefore, given two timing constraint sets C and C' on the same set of events



(a) timing constraint graph of C (b) timing constraint graph of C'

Fig. 8. Similarity between general timing constraint sets.

E, in order to take the unconstrained event pairs into consideration, we take the following steps

(I) Partition E by strongly connected components of constraint graphs of C and C'. We only consider the case where both partitions are the same. It is not hard to see that each pair of events in a partition is explicitly or implicitly constrained.
(II) Let $E_1, \ldots, E_K$ denote the K partitions and $C_1, \ldots, C_K$ and $C'_1, \ldots, C'_K$ denote the constraints of C and C' within the partitions, respectively. Then $C \sim C'$ is bounded by

$$C \sim C' \geq \inf_{k=1,\ldots K} \{C_k \sim C'_k\} \tag{40}$$

$$\geq \inf_{k=1,\ldots K} \left\{ \inf_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{k_{i,j}} \leq d'^*_{k_{i,j}}}} \left\{ \frac{d^*_{k_{i,j}}}{d'^*_{k_{i,j}}} \right\}^{|E_k|-1} \right\}. \tag{41}$$

By partitioning events as well as the constraints among them, we reduce the dimensions of feasible regions of constraint sets, filter out constraints that are irrelevant to the measurement of similarities, and thus get a more fine-grained view of similarities between the constraint sets.

We demonstrate the approach through a simple example. Consider vote-and-decide applications where several groups of voters vote within groups and a decision unit collects decisions from all groups. A typical constraint set constrains events within each voting group by relative deadlines to guarantee voting consistency and defines certain delays for the decision unit to make decision after all votes are collected. Figure 8 shows the timing constraint graphs of two timing constraint sets. According to strongly connected components, we partition the events into $E_1 = \{e_1, e_2, e_3\}$, $E_2 = \{e_4, e_5\}$, and $E_3 = \{e_6\}$, where partitions E_1 and E_2 are events from the corresponding voting groups, and partition E_3 is the deciding event. The similarity between the two sets of constraints, $C \sim C'$, is then lower bounded by $\inf\{\frac{36}{49}, \frac{9}{13}, 1\} \approx 69\%$.

5. APPLICATION 1: IMPROVING SYSTEMS' QOS PROPERTIES WITH CONSTRAINT SIMILARITY GUARANTEES

The constraint similarity study is important as it has broad applications in areas where other types of QoS requirements, such as total energy consumption, are directly affected by a system's timing behaviors. As an example, we consider the energy-aware task assignment for soft real-time applications on a multiprocessor system-on-chip (MPSoC) which is similar to the one discussed in Chantem et al. [2008].

In particular, in this section, we will demonstrate: (a) given the similarity metric and its bound (Section 4), calculate the probability guarantee that the original timing constraints are still satisfied by a modified constraint set for the purpose of reducing total energy consumption; and (b) given a maximum allowed constraint comprise, determine the constraint relaxations that best reduce energy consumption.

It is worth pointing out that the example of reducing energy consumption is used only to illustrate our approach. The similarity metric and the methodologies of using the metric to guide the trade-offs between timing and other QoS properties can be applied in a broad spectrum of soft real-time applications where both timing and other resources are under constraints.

5.1 System and Task Model

The MPSoC under consideration consists of a set of heterogeneous cores M. Let J be the set of tasks to be executed on M. For each task $j \in J$, the following parameters are used in our discussions:

— $EX(j, m)$: j's *worst-case* execution time on core m,
— $ex(j, m)$: j's *actual* execution time on core m, $ex(j, m) \in (0, EX(j, m)]$,
— d_j: the *relative* deadline of j,
— s_j: the start event of task j,
— f_j: the finish event of task j, $t(f_j) = t(s_j) + ex(j, m)$,
— $P(j, m)$: the power consumption of core $m \in M$ when task j executes on m.

The goal is to determine a static assignment of tasks to cores to further reduce the energy consumption while ensuring the required probability of constraint satisfactions guarantees. The hard real-time version of the problem, where a 100% deadline satisfaction must be ensured, is discussed in Chantem et al. [2008]. From the constraint satisfaction perspective, a deadline miss indicates that an execution trace falls outside of the feasible region defined by the given timing constraint set. When we allow a certain percentage of deadline misses, we actually include some execution traces outside the original feasible region, or in other words, the feasible region is expanded. The expanded feasible region can be considered as a relaxed constraint set. The constraint similarity study discussed in Section 4 allows us to quantitatively compare the deviations of the changed constraint from its original set, and hence to select which constraint(s) to relax based on a quantitative measure.

5.2 Reducing Total Energy Consumption

As shown in Chantem et al. [2008], the problem of minimizing total energy consumption for the MPSoC is to minimize $\sum_{j \in J} \sum_{m \in M} P(j, m) \cdot EX(j, m) \cdot \delta(j, m)$ where

$$\delta(j, m) = \begin{cases} 1 & \text{if } j \text{ is assigned to } m \\ 0 & \text{otherwise.} \end{cases} \tag{42}$$

However, in our case, the actual execution time $ex(j, m)$ is not a constant value, and we assume it follows a certain probability distribution over the interval $(0, EX(j, m)]$. Therefore, the goal is to minimize the *expectation* of the total energy consumption and the objective function thus becomes

$$\textbf{minimize} \sum_{j \in J} \sum_{m \in M} P(j, m) \cdot E\left[ex(j, m)\right] \cdot \delta(j, m).$$

Next, we demonstrate through an example how to use the similar bound to reduce total energy consumption by relaxing timing constraints.

Example 6. Consider two tasks j_1 and j_2 with relative deadline constraints $d_{j_1} = d_{j_2} = 20$ms and synchronization constraints $|t(s_{j_1}) - t(s_{j_2})| \leq 5ms$. We thus have the following set of constraints:

$$\left\{ \begin{array}{ll} t(f_{j_1}) - t(s_{j_1}) \leq 20, & t(s_{j_1}) - t(s_{j_2}) \leq 5, \\ t(f_{j_2}) - t(s_{j_2}) \leq 20, & t(s_{j_2}) - t(s_{j_1}) \leq 5, \\ t(s_{j_1}) - t(f_{j_1}) \leq \epsilon, & t(s_{j_2}) - t(f_{j_2}) \leq \epsilon \end{array} \right\} \tag{43}$$

Here $t(s_{j_1}) - t(f_{j_1}) \leq \epsilon(\epsilon \to 0^-)$ guarantees causality. The normal form of the constraint set (indexed by $t(s_{j_1})$, $t(f_{j_1})$, $t(s_{j_2})$, $t(f_{j_2})$) is given by (44).

$$\left[\begin{array}{cccc} 0 & \epsilon & 5 & 5+\epsilon \\ 20 & 0 & 25 & 25+\epsilon \\ 5 & 5+\epsilon & 0 & \epsilon \\ 25 & 25+\epsilon & 20 & 0 \end{array} \right] \tag{44}$$

Now, consider the scheduling problem of task j_1 and j_2 on the following MPSoC with 4 cores. We have

$$\begin{array}{ccccc} & & 10W & 10W & \\ 20\text{ms} & & \boxed{m_1} & \boxed{m_2} & 20\text{ms} \\ 22\text{ms} & & \boxed{m_3} & \boxed{m_4} & 25\text{ms} \\ & & 7W & 5W & \end{array}$$

where $P(j_1, m_1) = P(j_2, m_1) = 10W$, $EX(j_1, m_1) = EX(j_2, m_1) = 20$ms, etc.

To satisfy the constraint set (43), j_1 and j_2 can only be assigned to m_1 and m_2, respectively. Assuming the actual execution time is uniformly distributed in the interval $(0, Ex(j_1, m_1)]$, the expected total energy consumption is $10W \times 10\text{ms} + 10W \times 10\text{ms} = 200W \cdot \text{ms}$.

If we are willing to compromise the timing constraints, the deadline constraint of j_1 can be relaxed to $d_{j_1} = 22$ms from 20ms, the new constraint set becomes

$$\left\{ \begin{array}{ll} t(f_{j_1}) - t(s_{j_1}) \leq 22, & t(s_{j_1}) - t(s_{j_2}) \leq 5, \\ t(f_{j_2}) - t(s_{j_2}) \leq 20, & t(s_{j_2}) - t(s_{j_1}) \leq 5, \\ t(s_{j_1}) - t(f_{j_1}) \leq \epsilon, & t(s_{j_2}) - t(f_{j_2}) \leq \epsilon \end{array} \right\} \tag{45}$$

with normal form

$$\left[\begin{array}{cccc} 0 & \epsilon & 5 & 5+\epsilon \\ 22 & 0 & 27 & 27+\epsilon \\ 5 & 5+\epsilon & 0 & \epsilon \\ 25 & 25+\epsilon & 20 & 0 \end{array} \right]. \tag{46}$$

Based on Theorem 4.2, the similarity between these two constraint sets is lower-bounded by $\left(\frac{20}{22}\right)^{4-1} \approx 75\%$. In other words, a system that satisfies the new constraints (45) has at least 75% guarantee of satisfying the initial system constraints

(43). The benefit of relaxing the constraint is that we can now use m_3 to shedule j_1 or j_2 and the expected total energy consumption is thus reduced to $177W\cdot$ ms, a 11.5% reduction.

Similarly, if we further relax the deadline constraint of j_2 to d_{j_2} =25ms, one can easily verify that the similarity between the original and the modified constraint sets is bounded by $[51.2\%, 1]$ $\left(\left(\frac{20}{25}\right)^{4-1} = 51.2\%\right)$. In other words, systems that satisfy the modified constraints still have at least 50% chance to satisfy the original one. However, with such deadline relaxation, we can now schedule tasks j_1 and j_2 on m_3 and m_4, respectively, with the corresponding expected total energy consumption reduced to $139.5W\cdot$ ms, a 30.25% reduction.

Suppose we now have another job j_3 with a relative deadline of 22ms. New constraints $t(f_{j_3}) - t(s_{j_3}) \leq 22$ms and $t(s_{j_3}) - t(f_{j_3}) \leq \epsilon$ need to be inserted into (43). Since j_3 has no timing relations with j_1 and j_2, based on Section 4.2, we partition the constraint set into twosmaller normal forms.

$$\begin{bmatrix} 0 & \epsilon & 5 & 5+\epsilon \\ 20 & 0 & 25 & 25+\epsilon \\ 5 & 5+\epsilon & 0 & \epsilon \\ 25 & 25+\epsilon & 20 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & \epsilon \\ 22 & 0 \end{bmatrix} \tag{47}$$

For (47), the most energy-efficient assignment is to assign j_1, j_2, and j_3 to m_1, m_2, and m_3, respectively, with a total expected energy consumption of $277W\cdot$ ms. If the deadlines for j_1 and j_3 are reduced to 22ms and 25ms, respectively, the corresponding normal forms are changed from (47) to (48).

$$\begin{bmatrix} 0 & \epsilon & 5 & 5+\epsilon \\ 22 & 0 & 27 & 27+\epsilon \\ 5 & 5+\epsilon & 0 & \epsilon \\ 25 & 25+\epsilon & 20 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & \epsilon \\ 25 & 0 \end{bmatrix} \tag{48}$$

We can then assign j_1, j_2, and j_3 to m_3, m_2, and m_4, respectively, reducing the total expected energy consumption to $239.5W\cdot$ ms, a 14% reduction. The similarity between (47) and (48) is bounded by $\inf\left\{\left(\frac{20}{22}\right)^{4-1}, \left(\frac{22}{25}\right)^{2-1}\right\} \approx 75\%$. In other words, we have at least 75% guarantee to satisfy the initial constraints with the relaxed constraint set.

The previous examples show that understanding the implication of constraint changes both from the system timing property and nontiming properties points of view plays a key role in conducting design trade-offs. The similarity metric provides a quantitative measure about this implication in terms of timing constraint satisfaction. Specifically, the similarity bound between the orignal constraint set and that the modified one quantifies the maximal timing constraint satisfaction compromise in order to achieve certain desired QoS improvements. It thus allows us to make well-found decisions.

For the preceding examples, we manually picked some timing constraints to relax and calculated the similarity between the resultant constraint set and the original one. Under the same setting given in Example 6, a more interesting problem is: suppose we are allowed to relax the predefined constraints by certain amounts, can we determine which constraints to relax and how to relax them in order to find an assignment that further reduces expected total energy consumption?

5.3 Determining Constraint Relaxations

As we have seen from Section 5.2, relaxing timing constraints can further reduce total energy consumption, and Theorem 4.2 gives the bound of similarity between the modified constraint set and the original one. However, for real systems with a large number of events and constraints, there are possibly infinite ways even to relax a single timing constraint, not to mention there are combinatorial choices of constraints to relax. Therefore, relaxing constraints through exhaustive search is not realistic. Next we consider one type of design problems and provide a systematic approach.

Given an application with both timing requirements and a desired QoS property, suppose that the design problem is formulated as an optimization of some QoS property under multiple types of constraints (including timing constraints). The goal is to find *appropriate* timing constraints and relax them to *appropriate* degrees so that the desired QoS property can be further improved while the initial timing constraints are still at least $P\%$ satisfied. We introduce the following steps for solving the problem.
Step 1: Based on given timing constraints, construct the corresponding timing constraint graph G. Partition G by strongly connected components. And for each strongly connected component, compute its normal form.
Step 2: Modify the original timing constraints such that each event pair of a constraint within a partition is constrained by a variable deadline (instead of the original deadline). Add new constraints to constrain the newly introduced deadline variables based on the specified similarity bound $P\%$.
Step 3: Solve the modified optimization problem using standard algorithms. The optimization solution contains the optimized value of the objective function which is the improved QoS property value, and the variable assignments which define the necessary timing constraint relaxations.

In the following, we illustrate the use of the preceding general steps through the example given in Section 5.2. More specifically, consider the specific example of assigning a set of five tasks $j_1, \ldots, j_5$ to the MPSoC illustrated under the following timing constraints.

(1) The relative deadlines of all tasks are 20ms, that is, $d_{j_1} = d_{j_2} = d_{j_3} = d_{j_4} = d_{j_5} =$ 20ms;
(2) There are synchronization constraints between j_1 and j_2, and between j_3 and j_4, that is, $|t(s_{j_1}) - t(s_{j_2})| \leq 5$ms and $|t(s_{j_3}) - t(s_{j_4})| \leq 5$ms;
(3) Task j_3 and j_4 should start no later than $10ms$ after t_5 finishes, that is, we have constraints $t(s_{j_3}) - t(f_{j_5}) \leq 10$ and $t(s_{j_4}) - t(f_{j_5}) \leq 10$.

Chantem et al. [2008] formulate the problem as an MILP to optimize expected total energy consumption as the following.

minimize

$$\sum_{j \in J} \sum_{m \in M} P(j,m) \cdot E\left[ex(j,m)\right] \cdot \delta(j,m) \tag{49}$$

subject to

$$\forall j \in J: \quad t(f_j) = t(s_j) + \sum_{m \in M} \delta(j,m) \cdot EX(j,m) \tag{50}$$

$$\forall j \in J: \quad \sum_{m \in M} \delta(j,m) = 1 \tag{51}$$

$$\forall e_i, e_j \in E: \quad t(e_i) - t(e_j) \leq d_{k_{i,j}} \tag{52}$$

Here $E = \{s_j, f_j | j \in J\}$, and (52) generalizes timing constraints to a pairwise form ($d_{k_{i,j}}$ are constants obtained from the original constraints, for events that are not constrained, $d_{k_{i,j}} = +\infty$).[6]

Solving the MILP gives the nonpreemptive schedule of tasks on the cores such that all timing constraints are met and the total energy consumption is minimized. Now, if we allow timing constraint relaxations but require a 75% constraint satisfaction guarantee, the original MILP needs to be modified based on the steps given earlier. In particular, we have what follows.

Step 1: For the constraint set given in (52), construct its corresponding constraint graph and partition the event set E into $E_1, \ldots, E_K$ based on the graph's strongly connected components. Only timing constraints *within* partitions are possible candidates for relaxations. Note that for any $j \in J$, s_j and f_j must be in the same partition since they are strongly connected by the relative deadline of j, that is, $t(f_j) - t(s_j) \leq d_j$ and $t(s_j) - t(f_j) \leq \epsilon$. Therefore, all relative deadlines are possible to be relaxed.

For $\forall k = 1, \ldots, K$, derive the constraint normal form $\mathbf{D}_k^*$ for constraints among E_k, that is, for $\forall e_i, e_j \in E_k$, $t(e_i) - t(e_j) \leq d^*_{k_{i,j}}$. For this example, we have partitions $E_1 = \{s_{j_1}, f_{j_1}, s_{j_2}, f_{j_2}\}$, $E_2 = \{s_{j_3}, f_{j_3}, s_{j_4}, f_{j_4}\}$, and $E_3 = \{s_{j_5}, f_{j_5}\}$. The constraint normal forms $\mathbf{D}_1^*$, $\mathbf{D}_2^*$, and $\mathbf{D}_3^*$ on these partitions are

$$\mathbf{D}_1^* = \mathbf{D}_2^* = \begin{bmatrix} 0 & \epsilon & 5 & 5+\epsilon \\ 20 & 0 & 25 & 25+\epsilon \\ 5 & 5+\epsilon & 0 & \epsilon \\ 25 & 25+\epsilon & 20 & 0 \end{bmatrix}, \mathbf{D}_3^* = \begin{bmatrix} 0 & \epsilon \\ 20 & 0 \end{bmatrix}, \tag{53}$$

respectively.

Step 2: For constraints within partitions, modify (52) in the MILP formulation to

$$\forall e_i, e_j \in E_k, k = 1, \ldots, K : t(e_i) - t(e_j) \leq d'_{k_{i,j}}, \tag{54}$$

$$\forall e_i, e_j \in E_k, k = 1, \ldots, K : d'_{k_{i,j}} \leq \left\lfloor \frac{d^*_{k_{i,j}}}{\sqrt[|E_k|-1]{P\%}} \right\rfloor, \tag{55}$$

where $d'_{k_{i,j}}$ is the newly introduced variable for constraint relaxations. In the modified MILP, (54) and (55) are responsible for the selection and relaxation of constraints. From (55), we have

$$\left(\frac{d^*_{k_{i,j}}}{d'^*_{k_{i,j}}}\right)^{|E_k|-1} \geq \left(\frac{d^*_{k_{i,j}}}{d'_{k_{i,j}}}\right)^{|E_k|-1} \geq P\%, \tag{56}$$

where $d'^*_{k_{i,j}}$ is the corresponding entry in the normal form of the relaxed constraints and thus $d'^*_{k_{i,j}} \leq d'_{k_{i,j}}$. According to Theorem 4.2 and Section 4.2, the probability that the

[6]Note that the constraints to guarantee that all tasks execute for their durations without overlap [Chantem et al. 2008] are omitted from the formulation for clarity of presentation.

system satisfying the relaxed constraint set also satisfies the original constraint set is no less than $P\%$.

For example, for constraint $t(s_{j_1}) - t(s_{j_3}) \leq 5$, we derive two constraints, that is, $t(s_{j_1}) - t(s_{j_3}) \leq d'_{s_{j_1}s_{j_3}}$ and $d'_{s_{j_1}s_{j_3}} \leq \left\lfloor 5/\sqrt[3]{0.75} \right\rfloor$; for constraint $t(s_{j_3}) - t(f_{j_5}) \leq 10$, since s_{j_3} and s_{j_5} belong to different partitions, the constraint is still in the modified MILP but cannot be relaxed. Specifically, (52) in the MILP is replaced by the following constraints.

$$\begin{array}{llll}
t(s_{j_1}) - t(f_{j_1}) \leq d'_{s_{j_1}f_{j_1}} & , & d'_{s_{j_1}f_{j_1}} \leq \left\lfloor \epsilon/\sqrt[3]{0.75} \right\rfloor \\
t(s_{j_1}) - t(s_{j_3}) \leq d'_{s_{j_1}s_{j_3}} & , & d'_{s_{j_1}s_{j_3}} \leq \left\lfloor 5/\sqrt[3]{0.75} \right\rfloor \\
\cdots & & \cdots \\
t(f_{j_5}) - t(s_{j_5}) \leq d'_{f_{j_5}s_{j_5}} & , & d'_{f_{j_5}s_{j_5}} \leq \left\lfloor 20/\sqrt[1]{0.75} \right\rfloor \\
t(s_{j_3}) - t(f_{j_5}) \leq 10 & , & t(s_{j_4}) - t(f_{j_5}) \leq 10
\end{array}$$

Step 3: Solve the modified MILP using an MILP solver (such as ILOG CPLEX®). The solution contains the minimum expected total energy consumption and the assigned value of $d'_{k_{i,j}}$ which are the new constraint values in the correspondingly relaxed constraints. In this example, solving the modified instance of the MILP formulation, we have an optimal solution of $416.5W\cdot$ ms, with $\delta(1,1) = 1$, $\delta(2,3) = 1$, $\delta(3,2) = 1$, $\delta(4,3) = 1$, and $\delta(5,4) = 1$. The corresponding schedule is to run j_1, j_2, and j_5 on core m_1, m_3, and m_4 from time 0, respectively, with their new relative deadlines being 20ms, 22ms, and 26ms, respectively. Since j_2 and j_4 are both assigned to core m_3, to void overlap, from time 22ms, j_3 and j_4 are scheduled to run on m_2 and m_3, with their new relative deadlines being 20ms and 22ms, respectively. The total execution time in this case is 44ms with all constrains satisfied. However, with the original MILP, we can only schedule all five tasks on m_1 and m_2, with a minimum total execution time of 60ms and expected energy consumption of $500W\cdot$ ms. Therefore, by compromising no more than 25% of satisfaction guarantees of the original constraints, we gain a reduction of expected energy consumption and total execution time by 16.7% and 26.7%, respectively.

5.4 Discussions

Through the previous example of reducing total energy consumption with constraint similarity guarantees, we have demonstrated that when we do not require 100% timing constraint satisfaction guarantees, which is often the case for soft real-time applications, the flexibility allowed can be used to improve system's other QoS properties. The similarity metric gives a quantitative design guidance of how much timing constraint satisfaction is to be compromised in order to bring the QoS gain. We have further illustrated the detailed steps in obtaining better system QoS properties while still maintaining the required level of system's timing behavior resemblance.

6. APPLICATION 2: PREDICTING TRACKING ERROR RATE BASED ON CONSTRAINT SIMILARITY

As mentioned in Section 1, real-world performances of real-time systems may deviate from their designs due to unpredictable factors such as the environment that they are deployed in. In this section, we use a simplified real-time target tracking system to illustrate how a timing constraint set in a real-world setting may differ from the

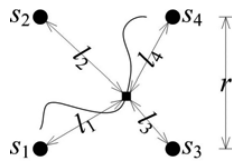


Fig. 9. Four sensors tracking a moving target.

one under ideal assumptions. We further illustrate that despite partially known and changing environments, the constraint similarity study given in Section 4 can be utilized to quantitatively infer the tracking error rate and thus improve the predictability of the system in real-world environments.

6.1 System Model

We consider a target tracking system presented in EnviroTrack [Abdelzaher et al. 2004]. To simplify our discussion, we assume a rectangular grid of sensors that periodically report to a control center their distances to a moving target. Each sensor s_i has a uniformly distributed bounded unknown delay, denoted by $t_i \in [0, d]$, from sensing the data to reporting the data. A target that moves in the area is detected by the sensors and the final coordinates of the target are decided by the control center aggregating the sensed data from them. The system's QoS is measured by the tracking error rate which is the ratio of inaccurately reported data due to inconsistent data from sensors to the total number of data reported. The tracking error rate is determined by the sensor data freshness and sensor data consistency.

In order to have fresh data, it is desirable for the control center to have short decision times. In addition, the sensor data aggregated at the control center should belong to the same sampling period to minimize data inconsistency among different sensors. More specifically, let $t(e_i(k))$ denote the time that sensor s_i reports to the control center of its distance to the target in the k'th detecting period, under the ideal assumption that a sensor locates the target at the beginning of the k'th detecting period. We have

$$t(e_i(k)) = kT + t_i, \tag{57}$$

where T is the detecting period for all sensors. Hence, for data consistency, we require $|t(e_i(k)) - t(e_j(k))| = |t_i - t_j| \leq d$, where $t_i, t_j \in [0, d]$.

To simplify our discussion, we restrict our attention to an $r \times r$ ($r = 1m$) square field with four ultrasonic sensors (whose detecting signal speed is V = 340m/sec) located on the corners of the field. The detecting radius of each sensor is assumed to be large enough to cover the entire field.

From time 0 and for every $T = 0.2$sec, all sensors try to measure their distances to a moving target, and after their distinct bounded unknown delays $t_i \in [0, d]$ ($d = 0.025$sec), the sensors report the distances to a control center which decides the coordinates of the target by aggregating the distance values from the four sensors. Under the ideal assumption that a sensor measures the distance at the beginning of the k'th detecting period, the time that the new distance value for sensor s_i is available for reporting is $t(e_i(k)) = kT + t_i$ as given in (57). As mentioned earlier, to guarantee the consistency of the reported data, the control center requires that the time distances between every two reporting events from two sensors be bounded by $\left|t(e_i(k)) - t(e_j(k))\right| = \left|t_i - t_j\right| \leq d$. This results in a constraint set C and its constraint matrix $\mathbf{D}$ is given in (58). In case of a constraint violation, that is, the control center does not receive data from one of the sensor(s) before the corresponding deadline, the

data received in previous periods will be used in the coordinate calculations[7].

$$\mathbf{D} = \begin{bmatrix} 0 & d & d & d \\ d & 0 & d & d \\ d & d & 0 & d \\ d & d & d & 0 \end{bmatrix} \tag{58}$$

6.2 Tracking Error and Tracking Error Rate Prediction

However, if we consider the signal transmission time, or possible objects in between the sensors and target, (57) should be changed into

$$t(e_i(k)) = kT + t_i + \delta_i(k), \tag{59}$$

where $\delta_i(k)$ is the time discrepancy caused by the Euclidean distance between the sensor s_i and the location of the target at the k'th period. The data consistency constraint thus becomes $\left|t(e_i(k)) - t(e_j(k))\right| \leq d + |\delta_i(k) - \delta_j(k)|$. In this example, if we take into account the travel time of ultrasonic distance measuring signals, the time of the event that sensor s_i reports the distance value is

$$t(e_i(k)) \approx kT + t_i + 2l_i/V, \tag{60}$$

where l_i is the distance from sensor s_i to the target[8]. For instance, when the target appears at the same site as sensor s_1, we have $l_1 = 0$, $l_2 = l_3 = r$ and $l_4 = \sqrt{2}r$, and the actual data consistency requirement on $t(e_i(k))$, $i = 1, \ldots, 4$, becomes C' whose constraint matrix is $\mathbf{D}' = \mathbf{D} + \Delta$, where

$$\Delta = \begin{bmatrix} 0 & \frac{2r}{V} & \frac{2r}{V} & \frac{2\sqrt{2}r}{V} \\ -\frac{2r}{V} & 0 & 0 & \frac{2(\sqrt{2}-1)r}{V} \\ -\frac{2r}{V} & 0 & 0 & \frac{2(\sqrt{2}-1)r}{V} \\ -\frac{2\sqrt{2}r}{V} & -\frac{2(\sqrt{2}-1)r}{V} & -\frac{2(\sqrt{2}-1)r}{V} & 0 \end{bmatrix}. \tag{61}$$

Therefore, if the control center uses constraint matrix (58) to monitor the events from the sensors, some timed data streams satisfying (58) may in fact correspond to inconsistent data, that is, distances sensed in previous periods. Moreover, different locations of the target in the field result in different actual data consistency constraint sets (similar to (61)). It is thus difficult for the control center to adjust the data consistent constraints.

[7]Note that if the target is restricted to move only within the square field, the control center will only need two distance values in order to decide the coordinates of the target. However, although four sensors bring some redundancy, there can still be cases where less than two distance values come before deadlines. In this case, the control center takes the values received in previous periods for approximation.

[8]We assume the speed of the moving target v is much smaller than the speed of the detecting signal V, that is, $v \ll V$.

Given the data consistency constraint matrix $\mathbf{D}$ in (58) under ideal assumptions, and a real-world deviation $\mathbf{D}'$ as the one in (61), some timed data streams that satisfy C' may violate C. These violations cause the imprecise coordinates (ones that have blatant regressions due to data taken from previous periods). They are circled in Figure 10(a), 10(c), and 10(e). From Theorem 4.2, the proportion of timed data streams satisfying C' that violate C is given $1 - (C \sim C')$. Again, $1 - (C \sim C')$ has the largest value when the target is at one of the four corners of the field, where

$$1 - (C \sim C') \in \left[0, 1 - \left(\frac{d}{d + 2\sqrt{2}r/V}\right)^{(4-1)}\right] \approx \left[0, 1 - (75\%)^3\right] \approx [0, 58\%]. \tag{62}$$

In fact, as can be observed from Figure 10, approximately 3 (marked with gray circles) of the entire 8 reported coordinates in Figure 10(a), 3 of the 9 coordinates in Figure 10(c), and 6 of the 17 coordinates in Figure 10(e) significantly deviate from the actual target location, indicating tracking error rates of 37.5%, 33.3%, and 35.3%, respectively. Note that these tracking error rates are not bounded by values indicated by the violation of any piece of individual timing constraint. The maximum constraint violation rate of individual timing constraint is only $1 - 75\% = 25\%$ in this example.

6.3 Discussions

Through the object tracking example presented in this section, we have shown that the performances of real-time systems may deviate from their designs when they are deployed in a real-world environment with some unanticipated physical factors ignored by the original designs. In these cases, the similarity metric can give quantitative estimations about the violation rates of the original timing constraint set, that is, how differently a system behaves from its original design.

It is worth pointing out that although the example only presents a tracking system consisting of 4 sensors, the methodology can be extended to large and dynamic systems. Because in any sampling period, only a small subset of all sensors are activated, and the similarity bound in Theorem 4.2 gives a quick and easy way to get an estimation of constraint similarity.

7. CONCLUSION

Real-world, real-time, and embedded systems may behave differently from specification in the time domain: some systems deviate from the designs due to unpredictable factors; some other soft real-time systems allow certain timing flexibilities that can often be utilized to improve other QoS properties of the systems. These deviations need to be exploited in a quantitative and predictable manner. Specifically, if a set of timing constraints are subject to imprecision or allowed to be modified, we need to measure how much the deviation from the origin constraint set. Based on this need, in this article, we introduce a quantitative metric to compare the similarity between two timing constraint sets. We based our study on feasible regions and proved that for a set of timing constraints, its feasible region is uniquely characterized by the constraint normal form. The similarity metric is then defined based on the common feasible region of the given two timing constraint sets to reflect their mutual satisfactions. Since directly calculating the similarity metric is computationally intractable, we give a similarity bound based on constraint set normal forms.

To demonstrate the capability of the new metric, we use an MPSoC system to illustrate how we may use the similarity metric to guide the design phases for reducing system energy consumption where some timing constraints are intentionally relaxed in order to improve the system's QoS properties; moreover, we apply the theory of timing constraint similarities to an object tracking system for predicting tracking

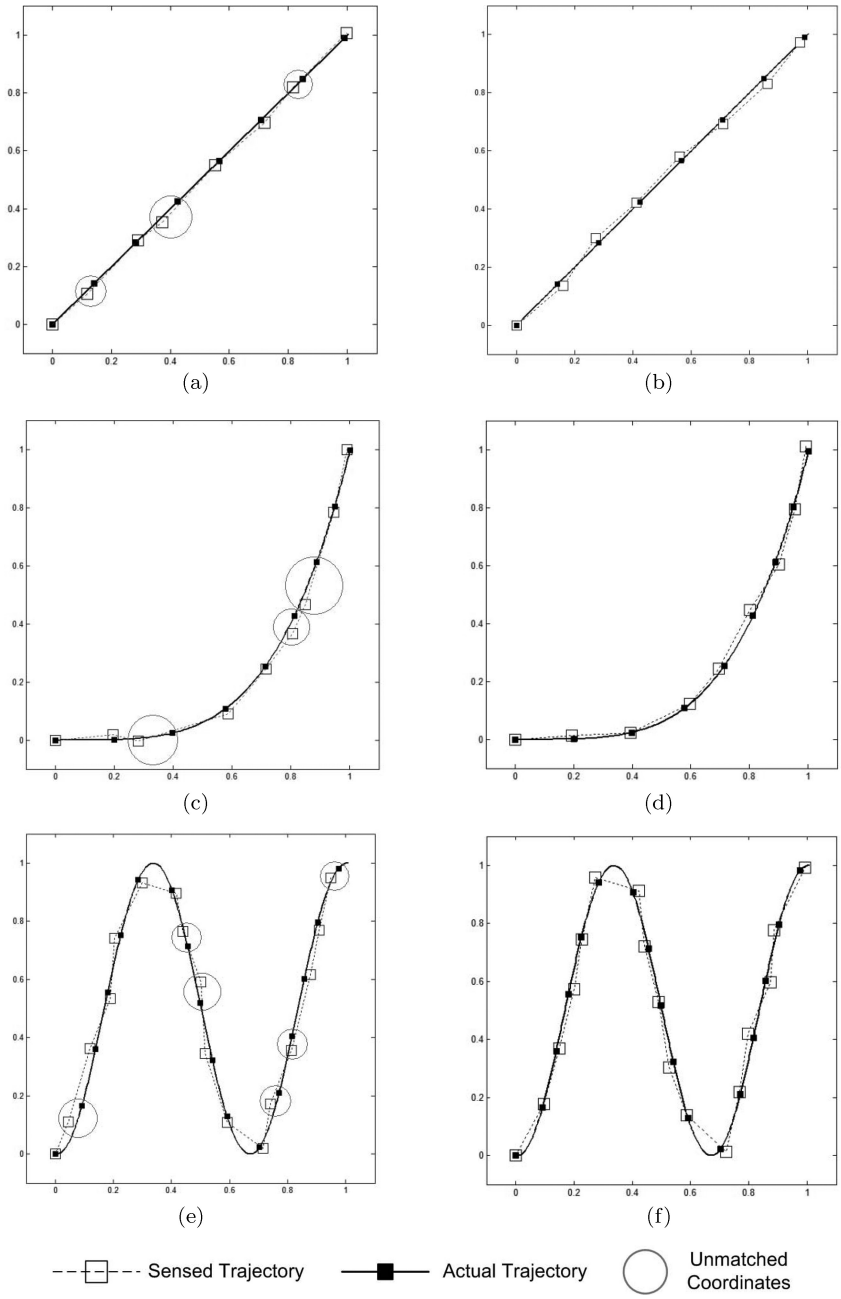


Fig. 10. Actual trajectories and sensed coordinates of the moving target before and after the data consistency constraints are modified.

error rates where some timing constraints are violated due to physical factors of the environment in which the system is deployed. In both cases, the similarity metric gives quantitative estimations about the satisfactions of the timing constraint set in the original design. These examples lead to a more general conclusion that the

similarity metric between timing constraint sets has a broad spectrum of applications whenever measuring of timing constraint satisfactions is needed.

However, it is worth pointing out that constraint satisfication problem itself is an NP-hard problem. The work presented in the article is not to solve the problem itself, rather to compare the similarities between two satisfiable regions defined by constraint sets and use the similarity matric to quantitatively measure the maximal divation between two constraint sets. It is not difficult to see from the similarity bound calculation formula (15) that the bound can be obtained in $O(n^2 \log n)$.

As future work, we plan to investigate the effect of nonuniformly distributed timed data streams on the evaluation of the similarity metric and its bound. Specifically, we will consider combining our earlier work on nonuniformly distributed interval-based events [Yu et al. 2008] with the computation of the similarity bounds. Intuitively, a set of interval-based events $\{I_1 = [\underline{I_1}, \overline{I_1}], \ldots, I_n = [\underline{I_n}, \overline{I_n}]\}$, where $\underline{I_1}, \ldots, \underline{I_n} \in \Re^+$, and $\overline{I_1}, \ldots, \overline{I_n} \in \Re^+$ are the lower and upper bounds of the corresponding time intervals, respectively, can be represented as a hypercube in the n-dimensional space whose density is determined by the joint distribution of all events. It will be revealing to understand the relationship between this hypercube with the hyperprism of a timing constraint set feasible region. This research is significant in deciding the satisfaction of timing constraints by events of a more practical model. Regarding the quality of the similarity bound, we realize that our bound may not be as tight, especially for higher-dimension cases. We will further examine and improve the quality of the similarity bound.

In the example given in Section 5, we mapped the timing constraint set relaxiation and QoS property trade-offs to an MILP problem. However, solving the MILP problem is time consuming and can limit the applicability of our approach to runtime adaptation when the problem size is large. Although how to improve the efficiency of solving the MILP is not the focus of this article, we believe it is possible that we can develop some type of heuristics to integrate the similarity metrics with a heuristic for solving the MILP problem and speed up the performance. We will explore different heuristics for MILP performance improvement in the future.

From the application perspective, we plan to apply the similarity metric to the design of fault-tolerant homogeneous/heterogeneous manycore processors. Effective defect tolerance techniques are essential to improve the yield of such complex integrated circuits. Previous attempts in this domain mainly focused on introducing microarchitecture-level redundancy and providing a logical topology that is always isomorphic to the topology of the target design so that from the viewpoint of the operating system and the programmers, they always see a unified logical topology regardless of the various underlying physical topologies [Shivakumar et al. 2003; Zhang et al. 2008]. Two challenging tasks for the topology reconfiguration is how to compare the performance of different logical topologies and how to effectively reconfigure them in order to find the best logical topology in terms of the metrics used for comparisons. Intuitively, from the timing perspective, one logical topology is better than the other if the temporal behavior of the former closely resembles the original specification than the later. This is consistent with our earlier observations in establishing the timing constraint set similarity metric. In the future, we plan to investigate the possibilities of applying the similarity metric in comparing alternative logical topologies and finding efficient heuristics for optimizing temporal resemblance.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their valuable comments. We would also like to thank T. Chantem for her help with ILOG CPLEX® while solving the MILP formulation in Section 5.3.

REFERENCES

ABDELZAHER, T., BLUM, B., CAO, Q., CHEN, Y., EVANS, D., GEORGE, J., GEORGE, S., GU, L., HE, T., KRISHNAMURTHY, S., LUO, L., SON, S., STANKOVIC, J., STOLERU, R., AND WOOD, A. 2004. Envirotrack: Towards an environmental computing paradigm for distributed sensor networks. In *Proceedings of the International Conference on Distributed Computing Systems*. 582–589.

ALUR, R. AND HENZINGER, T. A. 1989. A really temporal logic. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*. 164–169.

ALUR, R. AND DILL, D. L. 1994. A theory of timed automata. *Theor. Comput. Sci. 126,* 2, 183–235.

ALUR, R., FEDER, T., AND HENZINGER, T. A. 1991. The benefits of relaxing punctuality. In *Proceedings of the Symposium on Principles of Distributed Computing*. 139–152.

ARBAB, F. AND RUTTEN, J. 2002. A coinductive calculus of component connectors. In *Proceedings of the Workshop on Algebraic Development Techniques (WADT'02)*. Lecture Notes in Computer Science, Vol. 2755. Springer, 34–55.

AYDIN, H., MELHEM, R., MOSSE, D., AND MEJIA-ALVAREZ, P. 2001. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proceedings of the Real-Time Systems Symposium*. 95–105.

CHANTEM, T., DICK, R. P., AND HU, X. S. 2008. Temperature-Aware scheduling and assignment for hard real-time applications on mpsocs. In *Proceedings of the Design, Automation and Test in Europe (DATE'08)*. 288–293.

DASDAN, A. 1999. Timing analysis of embedded real-time systems. Tech. rep. UIUCDCS-R-99-2079, University of Illinois at Urbana-Champaign.

DASDAN, A. 2002a. Efficient algorithms for debugging timing constraint violations. In *Proceedings of the 8th ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*. ACM, New York, 50–56.

DASDAN, A. 2002b. Strongly polynomial-time algorithm for over-constraint resolution: Efficient debugging of timing constraint violations. In *Proceedings of the 10th International Symposium on Hardware/Software Codesign*. 127–132.

DASDAN, A. 2009. Provably efficient algorithms for resolving temporal and spatial difference constraint violations. *ACM Trans. Des. Autom. Electron. Syst. 14,* 1, 1–24.

DE ALFARO, L., MAJUMDAR, R., RAMAN, V., AND STOELINGA, M. 2007. Game relations and metrics. In *Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science (LICS'07)*. IEEE Computer Society, Los Alamitos, CA, 99–108.

DYER, M. E. AND FRIEZE, A. M. 1988. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput. 17,* 5, 967–974.

FANG, S.-C. AND PUTHEBNPURA, S. 1993. *Linear Optimization and Extensions: Theory and Algorithms*. Prentice-Hall.

GERBER, R., HONG, S., AND SAKSENA, M. 1995. Guaranteeing real-time requirements with resource-based calibration of periodic processes. *IEEE Trans. Softw. Engin. 21,* 7, 579–592.

GUPTA, V., JAGADEESAN, R., AND PANANGADEN, P. 2004. Approximate reasoning for real-time probabilistic processes. In *Proceedings of the 1st International Conference on the Quantitative Evaluation of Systems (QEST'04)*. 304–313.

HU, X. S., ZHOU, T., AND SHA, E. H.-M. 2001. Estimating probabilistic timing performance for real-time embedded systems. *IEEE Trans. VLSI Syst. 9,* 6, 833–844.

JACKSON, D., THOMAS, M., AND MILLETT, L. I. 2007. *Software for Dependable Systems: Sufficient Evidence?* The National Academies Press, Washington, D.C.

JAHANIAN, F. AND MOK, A. K.-L. 1987. A graph-theoretic approach for timing analysis and its implementation. *IEEE Trans. Comput. 36,* 8, 961–975.

JULIUS, A., GIRARD, A., AND PAPPAS, G. 2006. Approximate bisimulation for a class of stochastic hybrid systems. In *Proceedings of the American Control Conference*.

KALAVADE, A. AND MOGHÉ, P. 1998. A tool for performance estimation of networked embedded endsystems. In *Proceedings of the 35th Annual Conference on Design Automation*. ACM, 257–262.

LAWRENCE, J. 1991. Polytope volume computation. *Math. Comput. 57,* 195, 259–271.

LEE, E. A. 2005. Building unreliable systems out of reliable components: The real time story. Tech. rep. UCB/EECS-2005-5, EECS Department, University of California, Berkeley.

LI, Y. A. 1996. A probabilistic framework for estimation of execution time in heterogeneous computing systems. Ph.D. thesis, Major Professor-John K. Antonio, West Lafayette, IN.

LIAO, Y.-Z. AND WONG, C. April 1983. An algorithm to compact a vlsi symbolic layout with mixed constraints. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 2,* 2, 62–69.

MCMULLEN, P. AND SHEPARD, G. C. 1971. *Convex Polytopes and the Upper Bound Conjecture*. London Mathematical Society Lecture Notes Series 3, Cambridge University Press, London.

MOSCIBRODA, T., VON RICKENBACH, P., AND WATTENHOFER, R. 2006. Analyzing the energy-latency tradeoff during the deployment of sensor networks. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM'06)*, 1–13.

PAN, F., FREEH, V. W., AND SMITH, D. M. 2005. Exploring the energy-time tradeoff in high-performance computing. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium Workshop 11 (IPDPS'05)*. (Workshop 11) IEEE Computer Society, Los Alamitos, CA, 234.1.

PILLAI, P. AND SHIN, K. G. 2001. Real-Time dynamic voltage scaling for low-power embedded operating systems. *SIGOPS Oper. Syst. Rev. 35,* 5, 89–102.

PROVAN, J. S. 1994. Efficient enumeration of the vertices of polyhedra associated with network lp's. *Math. Program. 63,* 1, 47–64.

RAJU, S. C. V., RAJKUMAR, R., AND JAHANIAN, F. 1992. Monitoring timing constraints in distributed real-time systems. In *Proceedings of the IEEE Real-Time Systems Symposium*. 57–67.

SAEWONG, S. AND RAJKUMAR, R. 2003. Practical voltage-scaling for fixed-priority rt-systems. In *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'03)*. IEEE Computer Society, Los Alamitos, CA, 106.

SHIVAKUMAR, P., KECKLER, S. W., MOORE, C. R., AND BURGER, D. 2003. Exploiting microarchitectural redundancy for defect tolerance. In *Proceedings of the 21st International Conference on Computer Design (ICCD)*. 481–488.

SONG, L., DENG, Y., AND CAI, X. 2007. Towards automatic measurement of probabilistic processes. In *Proceedings of the 7th International Conference on Quality Software (QSIC'07)*. IEEE Computer Society, Los Alamitos, CA, 50–59.

THORSLEY, D. AND KLAVINS, E. 2008. Model reduction of stochastic processes using wasserstein pseudometrics. In *Proceedings of the American Control Conference*, 1374–1381.

TIA, T.-S., DENG, Z., SHANKAR, M., STORCH, M., SUN, J., WU, L.-C., AND LIU, J. W.-S. 1995. Probabilistic performance guarantee for real-time tasks with varying computation times. In *Proceedings of the Real-Time Technology and Applications Symposium*. 164.

WANG, F., NICOPOULOS, C., WU, X., XIE, Y., AND VIJAYKRISHNAN, N. 2007. Variation-aware task allocation and scheduling for mpsoc. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, Los Alamitos, CA, 598–603.

WOO, H., MOK, A. K., AND LEE, C.-G. 2006. A generic framework for monitoring timing constraints over uncertain events. In *Proceedings of the 27th IEEE International Real-Time Systems Symposium (RTSS'06)*. IEEE Computer Society, Los Alamitos, CA, 435–444.

YI HUANG, T. AND LIU, J. W. S. 1995. Predicting the worst-case execution time of the concurrent execution of instructions and cycle-stealing dma i/o operations. In *Proceedings of the ACM SIGPLAN Workshop on Languages, Compilers and Tools for Real-Time Systems*. 1–6.

YU, Y., REN, S., AND FRIEDER, O. 2008. Interval-Based timing constraints their satisfactions and applications. *IEEE Trans. Comput. 57,* 3, 418–432.

ZHANG, L., HAN, Y., XU, Q., AND LI, X. 2008. Defect tolerance in homogeneous manycore processors using core-level redundancy with unified topology. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'08)*. ACM, New York, 891–896.

Received March 2009; revised March 2010; accepted January 2011