A Metric for Judicious Relaxation of Timing Constraints in Soft Real-Time Systems

Yue Yu and Shangping Ren
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
{yyu8, ren}@iit.edu

Xiaobo Sharon Hu
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
shu@nd.edu

***Abstract*—For soft real-time systems, timing constraints are not as stringent as those in hard real-time systems: some constraint violations are permitted as long as the amount of violation is within a given limit. The allowed flexibility for soft real-time systems can be utilized to improve system's other quality-of-service (QoS) properties, such as energy consumption. One way to enforce constraint violation limit is to allow an expansion of timing constraint feasible region, but restrict the expansion in such a way that the relaxed constraint feasible region sufficiently *resembles* the original one. In this paper, we first introduce a new metric, *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 [1], we instead introduce an efficient method for estimating its bound. We further discuss how this metric can be exploited for evaluating trade-offs between timing constraint compromises and system's other QoS property gains. We use energy consumption reduction as an example to show the application of the proposed metric.**

I. Introduction

Real-time and embedded systems often face trade-offs between time and limited resources such as energy. 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 system's other QoS properties, such as reduce 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 objectives, e.g., energy consumption bound. Though some researchers have studied problems that are somewhat related to the above problems (to be discussed in the next section), we contend that there exists no systematic approach for tackling these challenges.

In this paper, we propose a framework for measuring timing constraint satisfaction which can be used to address the above two challenges. Specifically, we introduce a novel metric, i.e., *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 modified for, e.g., 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 guide a design process, we discuss 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 example serves as a demonstration to show that the similarity metric provides an effective tool to measure and guide the trade-offs between different QoS properties.

The rest of this paper is organized as follows. Next section provides a motivating example and reviews related work. Section III introduces a timing constraint set normal form. It is used to establish the constraint similarity metric. Section IV presents the similarity metric that quantifies how much one set of timing constraints resembles another. Section V applies the theory of timing constraint similarities to an MPSoC system to reduce its total energy consumption with minimal changes to the satisfaction of original timing constraints. Finally, we conclude and point out future work in Section VI.

II. Motivation and Related Work

To be able to quantify the level at which a timing constraint is satisfied in a soft real-time system has several important implications. It provides a systematic way to compare different system implementations when none of them can strictly meet the given timing constraints. In addition, it allows studies of "what if" scenarios where certain timing constraints are relaxed to some extent to improve other QoS properties. Further, 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 power[1] $10W$, $7W$, 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 question above 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 above 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 non-timing properties (e.g., power consumption) are satisfied, but the timing property change is minimal, or the timing property is the most similar (closest) to the original one? The goal of this paper 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. [2] propose a way to find the probability of a single task meeting its timing constraint, referred to as task feasibility probability. Kalavade et al. [3] 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. point out in [4] 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 [4], Wang et al. [5] 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 consider the problem of measuring the level of timing constraint satisfaction when the original timing constraints cannot be satisfied or are intentionally modified.

Our study, on the other hand, focuses on a more generalized constraint model where correlations between tasks are treated as linear timing constraints. The model is similar to Real-Time Logic [6] in that the focus is on timing constraints between event pairs. More specifically, we study similarities between 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. Note that some of the researches on more expressive constraint types such as Linear Real-Time Logic [7] and their feasibility results can be used in combination with our proposed approach.

Many notions on similarities have been defined in the literature for process models. Gupta et al. [8] give a pseudometric analogue of bisimulation for generalized semi-Markov processes and show that two metrically similar processes have similar observable quantitative properties. Thorsley et al. [9] 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, e.g., in [10], [11], [12]. 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 non-equivalent processes or constraints, the metrics introduced in this paper not only measures the resemblance between two sets of timing constrains, but also provides a quantitative design guidance in deciding the trade-offs between timing 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 [13]; 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 [14]. In fact, for real-time and

[1]The peak power is the maximum level of energy measured during an observation period.

embedded system, dynamic voltage scaling techniques, which reduce system supply voltage for lower operation frequencies, has been extensively used in various power management schemes [15], [16], [17]. However, to our best knowledge, there is no quantitative study of trading timing constraint satisfaction in soft real-time systems for other QoS properties.

III. 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 IV.

In our system model, we take a commonly used approach in that system behaviors (or computations) are represented as *data streams*, i.e., a sequence of event occurrences $(e_1, e_2, \ldots, e_n)$ [18], and a *timed data stream* is formed by pairing each event e_i with its corresponding occurrence time $t(e_i)$, as defined below [19]:

Definition 1 (Timed Data Stream): A timed data stream (TDS) is a sequence $((e_1, t(e_1)), (e_2, t(e_2)), \ldots, (e_n, t(e_n)))$ 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 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 paper. Feasible regions are the key in comparing timing constraint sets and we illustrate them in Example 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 Fig. 1 (shaded area)

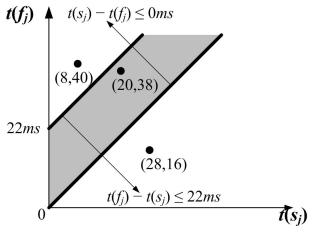


Fig. 1. The feasible region of constraint $0ms < t(f_j) - t(s_j) \leq 22ms$.

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 violates causality $t(s_j) - t(f_j) < 0$ and deadline $t(f_j) - t(s_j) \leq 22$, respectively. □

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

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

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 Fig. 2 with its boundaries marked as bold lines.

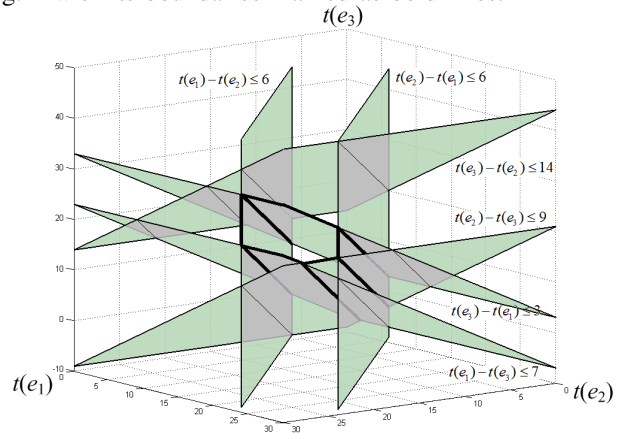


Fig. 2. The feasible region of a constraint set (1).

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, i.e., each point $(t(e_1), t(e_2), t(e_3))$ in the region satisfies constraint set (1). □

From Example 2 and 3, we can see that a feasible region characterizes all valid execution time traces, i.e., 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. Therefore, 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 so that the feasible region comparisons can be directly made upon the algebraic abstractions. This algebraic representation builds upon the concept of the most stringent constraints, which we explain by again using Example 3. Examine the feasible region of Example 3 shown in Fig. 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.

It is worth noticing the conceptual differences between the most stringent timing constraint between an event pair implied from a given set of timing constraints and a feasible region that satisfies the given set of timing constraints. As in the above example, in order to satisfy constraints $t(e_3) - t(e_1) \leq 3$ and $t(e_1) - t(e_2) \leq 6$ in the given constraint set (1), we must satisfy $t(e_3) - t(e_2) \leq 9$, which is more stringent than the one $(t(e_3) - t(e_2) \leq 14)$ given in (1). As we later show that the feasible region can be characterized by the most stringent constraints among all event pairs.

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 [6]. The most stringent constraint set has an important property which is summarized in the following lemma.

Lemma 1: 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.

Proof: *The proof is given in our technical report [20].* □

An important implication of Lemma 1 is that the shape of the feasible region is determined solely by the most stringent timing constraints between all *pairs* of events. 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 2 (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

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

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*. □

With Definition 2, the inclusion relation of two feasible regions defined by two timing constraint sets can be validated by comparing the constraint sets' normal forms.

Theorem 1: Given two sets of real-time constraints C and C' 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 $\mathbf{D}^* \geq \mathbf{D'}^*$, i.e., $\forall i, j : d^*_{i,j} \geq d'^*_{i,j}$.

Proof: *The proof is given in our technical report [20].* □

From Theorem 1, we have the following:

$$\begin{aligned} \mathbf{D}^* = \mathbf{D'}^* &\Leftrightarrow \mathbf{D}^* \geq \mathbf{D'}^* \wedge \mathbf{D'}^* \geq \mathbf{D}^* \\ &\Leftrightarrow \text{feasible region of } C \text{ include that of } C' \\ &\quad \wedge \text{feasible region of } C' \text{ include that of } C \\ &\Leftrightarrow \text{feasible regions of } C \text{ and } C' \text{ are identical} \end{aligned}$$

In other words, there is a one-to-one correspondence between a timing constraint normal form and a feasible region. Therefore, 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 relationship between feasible regions of two different constraint sets by studying the constraint normal forms.

[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.

IV. Similarities between Timing Constraint Sets

Example 1 has shown that timing constraint changes often affect system's other QoS properties, i.e., there are trade-offs between the stringency of timing constraints and other QoS properties. It is hence important to know how much the timing behavior compromise is in order to bring QoS benefits.

A. 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 III. 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 and leave the discussion of non-uniformity to subsection IV-B.

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 Fig. 3.

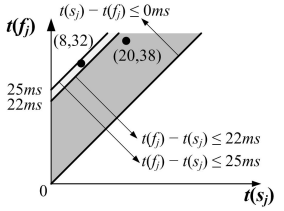


Fig. 3. The feasible regions satisfying constraint $0 < t(f_j) - t(s_j) \leq 22$ and $0 < t(f_j) - t(s_j) \leq 25$.

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:

$$\left\{ \begin{array}{ll} t(e_1) - t(e_2) \leq 5, & t(e_2) - t(e_1) \leq 7, \\ t(e_1) - t(e_3) \leq 5, & t(e_3) - t(e_1) \leq 2, \\ t(e_2) - t(e_3) \leq 10, & t(e_3) - t(e_2) \leq 5 \end{array} \right\} \quad (3)$$

The relationship between feasible regions satisfying constraint sets (1) and (3) is illustrated in Fig.4, where bold lines, light lines, and the shaded region represent constraint sets (1), (3), and the intersection between their feasible regions, respectively.

From Fig.4, we can see that although feasible regions satisfying constraint sets (1) and (3) are not identical, they share some common region. Hence, we can expect that they have some timing behaviors in common. □

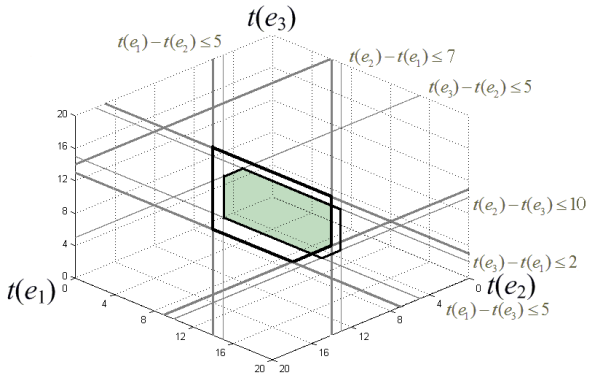


Fig. 4. The feasible regions satisfying constraint sets (1) (bold lines) and (3) (light lines), and their intersection (the shaded region).

Generalizing the above discussions, we define the similarity between two timing constraint sets as the following:

Definition 3 (Constraint Set Similarity): Let $\mathcal{S}(C)$ denote the size of the feasible region of a timing constraint set C. Given two timing constraint sets C, C', the similarity relation is defined as $C \sim C' = \frac{\mathcal{S}(C^{\cap})}{\mathcal{S}(C')}$, where $C^{\cap}$ is the intersection of C and C'. □

Intuitively, if $C \sim C' = P\%$, i.e., 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 ($\mathcal{S}(C)$ in our context) has been shown to be $\#P$-hard [1], and thus directly calculating the proportions of the intersection in any of the feasible regions, i.e., the similarity metric, by comparing their sizes is costly. To overcome the computational hurdle of evaluating directly the constraint set similarity between two constraint sets, we resort to finding a lower bound on the constraint set similarity such that it is easily computable and is tight. The following theorem defines such a bound.

Theorem 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, i.e., $\mathbf{D}^* \not\geq \mathbf{D'}^*$, then the similarity is bounded by:

$$\left(\min_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{i,j} \leq d'^*_{i,j}}} \left\{ \frac{d^*_{i,j}}{d'^*_{i,j}} \right\} \right)^{|E|-1} \leq C \sim C' < 1 \qquad (4)$$

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, i.e., $\mathbf{D}^* \geq \mathbf{D'}^*$.

Proof: *The proof is given in our technical report [20].* □

From Theorem 2, one can see that the similarity bound can be calculated easily once the normal forms of the constraint sets are available. Comparing similarities of different constraint sets then can be indirectly achieved through evaluating their similarity bounds. Before discussing various implications of using the similarity bound in Section IV-B, we demonstrate the use of Theorem 2 on the constraint sets given in Example 4. From Theorem 2, the ratio of the common region between (1) and (3) to the feasible region of (3) is bounded by $\left[\frac{36}{49}, 1\right)$ where $\frac{36}{49} = \left(\min\left\{\frac{6}{7}, \frac{9}{10}\right\}\right)^{3-1}$. Therefore, assuming a uniform distribution of the event timing behavior in the feasible regions, Theorem 2 guarantees that at least $\frac{36}{49} = 73\%$ timed data streams that satisfy (3) also satisfy (1). This gives us a quantitative measure of the resemblance between systems constrained by (1) and (3), respectively. Actually, as shown in [20], the exact ratio of the common region between (1) and (3) to the feasible region of (3) is $\frac{73}{80} = 91.25\%$.

B. Discussions

Timed data stream distribution in the feasible region

In the above discussions, we assume that timed data streams are uniformly distributed in the feasible region of the constraint set. The bound given in Theorem 2 is based on such an assumption. However, the definition of constraint feasible region similarities can be extended to non-uniform 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 Fig. 5(a) and 5(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 Fig. 5(c). Therefore, the concept $\mathcal{S}(C)$ in Definition 3 are to be extended to weighted sizes.

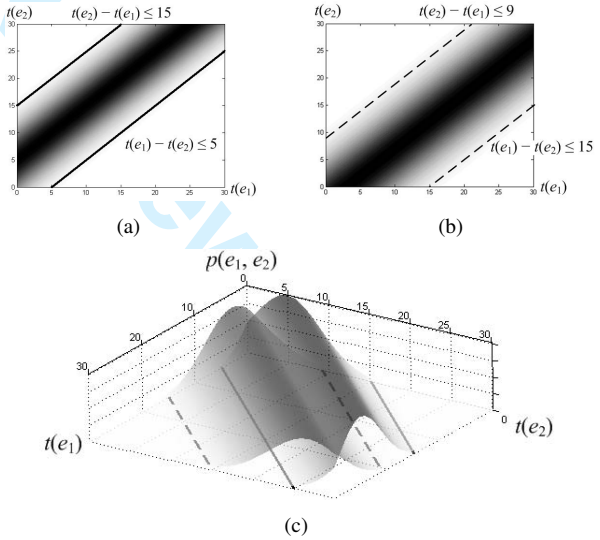


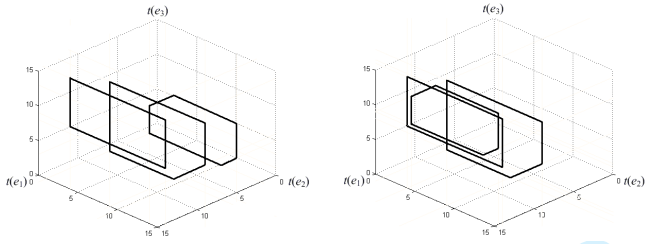
Fig. 5. Feasible region similarities of non-uniformly distributed TDS's.

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, e.g., [2], [3], [21], [22]. 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 paper.

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 3, 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 6 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 Fig. 6(a) and Fig. 6(b). However, depending on the positions from which C'' similar to C', C and C'' can be either very similar (as shown in Fig. 6(b)) or very dissimilar (as shown in Fig. 6(a)).



(a) (b)

Fig. 6. Similarity relation is not transitive.

The tightness of the similarity bound

From Theorem 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.

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 2 still applies, but it approaches to 0 ($C_1 \sim C_2 = \min\left\{\frac{22}{25}, \frac{5}{+\infty}\right\} \rightarrow 0^+$), 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 IV-A, 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 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.

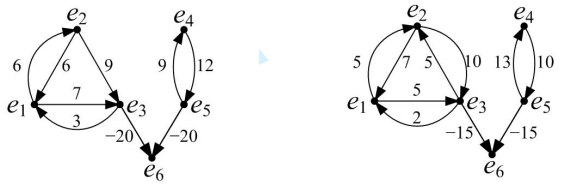
II. Let $E_1, \ldots, E_K$ denote the K partitions and $C_1, \ldots, C_K$ and $C'_1, \ldots, C'_K$ denote the constraints of C and C' within the partitions, respectively. Then $C \sim C'$ is bounded by

$$C \sim C' \geq \min_{k=1,\ldots K} \{C_k \sim C'_k\} \tag{5}$$

$$\geq \min_{k=1,\ldots K} \left\{ \min_{\substack{i,j=1,\ldots,n,\\ i\neq j,\ d^*_{k_{i,j}} \leq d'^*_{k_{i,j}}}} \left\{ \frac{d^*_{k_{i,j}}}{d'^*_{k_{i,j}}} \right\}^{|E_k|-1} \right\} \tag{6}$$

By partitioning events as well as the constraints among them, we reduce the dimensions of feasible regions of a 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 7 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 $\min\{\frac{36}{49}, \frac{9}{13}, 1\} \approx 69\%$.



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

Fig. 7. Similarity between general timing constraint sets.

V. 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 [23]. In particular, in this section, we will demonstrate (a) given the similarity metric and its bound (Section IV), calculate the probability guarantee that the original timing constraints are still satisfied by the 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 reduces energy consumption.

It is worth pointing out that reducing energy consumption is used only as an example 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 which involve timing and limited resources.

A. 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 when running 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 [23]. 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 IV 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.

B. Reducing Total Energy Consumption

As shown in [23], 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

$$\delta(j,m) = \begin{cases} 1 & \text{if } j \text{ is assigned to } m \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

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

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

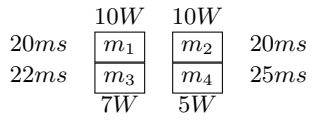
Example 5: Consider two tasks j_1 and j_2 with relative deadline constraints $d_{j_1} = d_{j_2} = 20ms$ and synchronization constraints $|t(s_{j_1}) - t(s_{j_2})| \le 5ms$. We thus have the following set of constraints:

$$\left\{ \begin{array}{ll} t(f_{j_1}) - t(s_{j_1}) \le 20, & t(s_{j_1}) - t(s_{j_2}) \le 5, \\ t(f_{j_2}) - t(s_{j_2}) \le 20, & t(s_{j_2}) - t(s_{j_1}) \le 5, \\ t(s_{j_1}) - t(f_{j_1}) \le \epsilon, & t(s_{j_2}) - t(f_{j_2}) \le \epsilon \end{array} \right\} \tag{8}$$

where $t(s_{j_1}) - t(f_{j_1}) \le \epsilon (\epsilon \to 0^-)$ 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 (9).

$$\begin{bmatrix} 0 & \epsilon & 5 & 5+\epsilon \\ 20 & 0 & 25 & 25+\epsilon \\ 5 & 5+\epsilon & 0 & \epsilon \\ 25 & 25+\epsilon & 20 & 0 \end{bmatrix} \tag{9}$$

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



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

To satisfy the constraint set (8), 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 $10W \times 10ms + 10W \times 10ms = 200W \cdot ms$.

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

$$\left\{ \begin{array}{ll} t(f_{j_1}) - t(s_{j_1}) \le 22, & t(s_{j_1}) - t(s_{j_2}) \le 5, \\ t(f_{j_2}) - t(s_{j_2}) \le 20, & t(s_{j_2}) - t(s_{j_1}) \le 5, \\ t(s_{j_1}) - t(f_{j_1}) \le \epsilon, & t(s_{j_2}) - t(f_{j_2}) \le \epsilon \end{array} \right\} \tag{10}$$

with normal form

$$\begin{bmatrix} 0 & \epsilon & 5 & 5+\epsilon \\ 22 & 0 & 27 & 27+\epsilon \\ 5 & 5+\epsilon & 0 & \epsilon \\ 25 & 25+\epsilon & 20 & 0 \end{bmatrix} \tag{11}$$

Based on Theorem 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 (10) has at least 75% guarantee of satisfying the initial system constraints (8). 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 $[51.2\%, 1]$ ($\left(\frac{20}{25}\right)^{4-1} = 51.2\%$). 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 $t(f_{j_3}) - t(s_{j_3}) \leq 22ms$ and $t(s_{j_3}) - t(f_{j_3}) \leq \epsilon$ need to be inserted into (8). Since j_3 has no timing relations with j_1 and j_2, based on Section IV-B, we partition the constraint set into two smaller normal forms.

$$\begin{bmatrix} 0 & \epsilon & 5 & 5+\epsilon \\ 20 & 0 & 25 & 25+\epsilon \\ 5 & 5+\epsilon & 0 & \epsilon \\ 25 & 25+\epsilon & 20 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & \epsilon \\ 22 & 0 \end{bmatrix} \tag{12}$$

For (12), 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 (12) to (13)

$$\begin{bmatrix} 0 & \epsilon & 5 & 5+\epsilon \\ 22 & 0 & 27 & 27+\epsilon \\ 5 & 5+\epsilon & 0 & \epsilon \\ 25 & 25+\epsilon & 20 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & \epsilon \\ 25 & 0 \end{bmatrix} \tag{13}$$

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 $239.5W \cdot ms$, 14% reduction. The similarity between (12) and (13) is bounded by $\min\left\{\left(\frac{20}{22}\right)^{4-1}, \left(\frac{22}{25}\right)^{2-1}\right\} \approx 75\%$. In other words, we have at least 75% guarantee to satisfy the initial constraints with the relaxed constraint set. □

The above examples show that understanding the implication of constraint changes both from the system timing property and non-timing properties points of view plays a key role in conducting design tradeoffs. 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 above 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 5, 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?

C. Determining Constraint Relaxations

As we have seen from Section V-B, relaxing timing constraints can further reduce total energy consumption, and Theorem 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. Below, 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 above general steps through the example given in Section V-B. 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 $20ms$, i.e., $d_{j_1} = d_{j_2} = d_{j_3} = d_{j_4} = d_{j_5} = 20ms$;
2) There are synchronization constraints between j_1 and j_2, and between j_3 and j_4, i.e., $|t(s_{j_1}) - t(s_{j_2})| \leq 5ms$ and $|t(s_{j_3}) - t(s_{j_4})| \leq 5ms$;
3) Task j_3 and j_4 should start no later than $10ms$ after t_5 finishes, i.e., we have constraints $t(s_{j_3}) - t(f_{j_5}) \leq 10ms$ and $t(s_{j_4}) - t(f_{j_5}) \leq 10ms$.

Chantem et al. [23] formulate the problem as an MILP to optimize expected total energy consumption as following:

minimize

$$\sum_{j \in J} \sum_{m \in M} P(j,m) \cdot E\left[ex(j,m)\right] \cdot \delta(j,m) \tag{14}$$

subject to

$$\forall j \in J: \quad t(f_j) = t(s_j) + \sum_{m \in M} \delta(j,m) \cdot EX(j,m) \tag{15}$$

$$\forall j \in J: \quad \sum_{m \in M} \delta(j,m) = 1 \tag{16}$$

$$\forall e_i, e_j \in E: \quad t(e_i) - t(e_j) \leq d_{k_{i,j}} \tag{17}$$

where $E = \{s_j, f_j | j \in J\}$, and (17) generalizes timing constraints to a pairwise form ($d_{k_{i,j}}$ are constants obtained from

the original constraints, for events that are not constrained, $d_{k_{i,j}} = +\infty$).[3]

Solving the MILP gives the non-preemptive 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 above. In particular,

Step 1: For the constraint set given in (17), 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, i.e., $t(f_j) - t(s_j) \leq d_j$ 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, i.e., 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

$$\mathbf{D}_1^* = \mathbf{D}_2^* = \begin{bmatrix} 0 & \epsilon & 5 & 5+\epsilon \\ 20 & 0 & 25 & 25+\epsilon \\ 5 & 5+\epsilon & 0 & \epsilon \\ 25 & 25+\epsilon & 20 & 0 \end{bmatrix}, \mathbf{D}_3^* = \begin{bmatrix} 0 & \epsilon \\ 20 & 0 \end{bmatrix} \tag{18}$$

respectively.

Step 2: For constraints within partitions, modify (17) in the MILP formulation to

$$\forall e_i, e_j \in E_k, k = 1, \ldots, K : t(e_i) - t(e_j) \leq d'_{k_{i,j}} \tag{19}$$

$$\forall e_i, e_j \in E_k, k = 1, \ldots, K : d'_{k_{i,j}} \leq \left\lfloor \frac{d^*_{k_{i,j}}}{\sqrt[|E_k|-1]{P\%}} \right\rfloor \tag{20}$$

where $d'_{k_{i,j}}$ is the newly introduced variable for constraint relaxations. In the modified MILP, (19) and (20) are responsible for the selection and relaxation of constraints. From (20), we have

$$\left(\frac{d^*_{k_{i,j}}}{d'^*_{k_{i,j}}}\right)^{|E_k|-1} \geq \left(\frac{d^*_{k_{i,j}}}{d'_{k_{i,j}}}\right)^{|E_k|-1} \geq P\% \tag{21}$$

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 2 and Section IV-B, the probability that the 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, i.e., $t(s_{j_1}) - t(s_{j_3}) \leq d'_{s_{j_1}s_{j_3}}$ and $d'_{s_{j_1}s_{j_3}} \leq \lfloor 5/\sqrt[3]{0.75} \rfloor$; 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, (17) in the MILP is replaced by the following constraints

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

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 is 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.

Through the above example of reducing total energy consumption with constraint similarity guarantees, we have demonstrated that when we do not require 100% 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. We have further illustrate the detailed steps in obtaining better system QoS properties while still maintaining the required system's timing behavior resemblance. It is worth pointing out that the process of generating timing constraints from system specifications and the above steps for relaxing these constraints can all be automated, and thus will not be prohibitive when studying real-world systems. Step 1 and 2 for formulating the constraint relaxation MILP requires polynomial time. Solving the MILP in Step 3 requires exponential time, and is thus the most computationally expensive part. Currently, we have not found effective heuristics for runtime constraint relaxations when system specification parameters could change. However, the constraint similarity metric itself is not restricted to offline analysis; in fact, the metric can be used to measure timing behavior deviations for system parameter changes at runtime.

VI. Conclusion

Soft real-time systems allow certain timing flexibilities that can often be utilized to improve QoS properties of the systems. However, this flexibilities need to be exploited in

[3]Note that the constraints to guarantee that all tasks execute for their durations without overlap [23] are omitted from the formulation for clarity of presentation.

a quantitative and predictable manner. Specifically, if a set of timing constraints are allowed to be modified, we need to measure how much the relaxation differs from the origin set. Based on this need, in this paper, 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, and reflects their mutual satisfactions. Since directly calculating the similarity metric is computationally intractable, we give a similarity bound based on the normal form. We used an MPSoC system to illustrate how we may use the similarity metric to guide the design phases for reducing system energy consumption. This example leads to a more general conclusion that the similarity metric between timing constraint sets can be used to guide the trade-offs between different QoS properties.

As future work, we plan to investigate the effect of non-uniformly distributed timed data streams on the evaluation of the similarity metric and its bound. Specifically, we will consider combining our earlier work on non-uniformly distributed interval-based events [24] with the computation of the similarity bounds. Intuitively, a set of interval-based events $\{I_1 = [min(I_1), max(I_1)], \ldots, I_n = [min(I_n), max(I_n)]\}$, 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.

ACKNOWLEDGMENT

The authors would like to thank Thidapat Chantem for her help with ILOG CPLEX® while solving the MILP formulation in Section V-C. We are also grateful to referees for their valuable comments and suggestions.

This work is supported in part by NSF CNS-0834180, CNS-0720457, and CNS-0746643.

REFERENCES

[1] M. E. Dyer and A. M. Frieze, "On the complexity of computing the volume of a polyhedron," *SIAM J. Comput.*, vol. 17, no. 5, pp. 967–974, 1988.

[2] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J. W.-S. Liu, "Probabilistic performance guarantee for real-time tasks with varying computation times," in *Proceedings of the Real-Time Technology and Applications Symposium*, Washington, DC, USA, 1995, p. 164.

[3] A. Kalavade and P. Moghé, "A tool for performance estimation of networked embedded end-systems," in *Proceedings of the 35th annual conference on Design automation*. ACM, 1998, pp. 257–262.

[4] X. S. Hu, T. Zhou, and E. H.-M. Sha, "Estimating probabilistic timing performance for real-time embedded systems," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 9, no. 6, pp. 833–844, 2001.

[5] F. Wang, C. Nicopoulos, X. Wu, Y. Xie, and N. Vijaykrishnan, "Variation-aware task allocation and scheduling for mpsoc," in *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design*. Piscataway, NJ, USA: IEEE Press, 2007, pp. 598–603.

[6] F. Jahanian and A. K.-L. Mok, "A graph-theoretic approach for timing analysis and its implementation," *IEEE Transactions on Computers*, vol. 36, no. 8, pp. 961–975, 1987.

[7] S. Andrei and A. M. K. Cheng, "Verifying linear real-time logic specifications," in *RTSS '07: Proceedings of the 28th IEEE International Real-Time Systems Symposium*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 333–342.

[8] V. Gupta, R. Jagadeesan, and P. Panangaden, "Approximate reasoning for real-time probabilistic processes," in *First International Conference on the Quantitative Evaluation of Systems (QEST04), 00:304C313*. IEEE Press, 2004, pp. 304–313.

[9] D. Thorsley and E. Klavins, "Model reduction of stochastic processes using wasserstein pseudometrics," *American Control Conference, 2008*, pp. 1374–1381, June 2008.

[10] L. de Alfaro, R. Majumdar, V. Raman, and M. Stoelinga, "Game relations and metrics," in *LICS '07: Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 99–108.

[11] L. Song, Y. Deng, and X. Cai, "Towards automatic measurement of probabilistic processes," in *QSIC '07: Proceedings of the Seventh International Conference on Quality Software*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 50–59.

[12] A. Julius, A. Girard, and G. Pappas, "Approximate bisimulation for a class of stochastic hybrid systems," *American Control Conference, 2006*, pp. 6 pp.–, June 2006.

[13] T. Moscibroda, P. von Rickenbach, and R. Wattenhofer, "Analyzing the energy-latency trade-off during the deployment of sensor networks," *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1–13, April 2006.

[14] F. Pan, V. W. Freeh, and D. M. Smith, "Exploring the energy-time tradeoff in high-performance computing," in *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 11*. Washington, DC, USA: IEEE Computer Society, 2005, p. 234.1.

[15] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Dynamic and aggressive scheduling techniques for power-aware real-time systems," *Proceedings of the Real-Time Systems Symposium.*, pp. 95–105, 2001.

[16] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 89–102, 2001.

[17] S. Saewong and R. Rajkumar, "Practical voltage-scaling for fixed-priority rt-systems," in *RTAS '03: Proceedings of the The 9th IEEE Real-Time and Embedded Technology and Applications Symposium*. Washington, DC, USA: IEEE Computer Society, 2003, p. 106.

[18] H. Woo, A. K. Mok, and C.-G. Lee, "A generic framework for monitoring timing constraints over uncertain events," in *RTSS '06: Proceedings of the 27th IEEE International Real-Time Systems Symposium*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 435–444.

[19] F. Arbab and J. Rutten, "A coinductive calculus of component connectors," in *WADT'02*, ser. LNCS, vol. 2755, 2002, pp. 34–55.

[20] Y. Yu and S. Ren, "A new metric for quantifying similarity between timing constraint sets in soft real-time systems," Department of Computing Science, Illinois Institute of Technology, Tech. Rep., 2009. [Online]. Available: http://dijkstra.cs.iit.edu/code/TechReports/CS-115-01-02-2009.pdf

[21] T. yi Huang and J. W. S. Liu, "Predicting the worst-case execution time of the concurrent execution of instructions and cycle-stealing dma i/o operations," in *Proc. of ACM SIGPLAN Workshop on Languages, Compilers and Tools for Real-Time Systems*, 1995, pp. 1–6.

[22] Y. A. Li, "A probabilistic framework for estimation of execution time in heterogeneous computing systems," Ph.D. dissertation, West Lafayette, IN, USA, 1996, major Professor-John K. Antonio.

[23] T. Chantem, R. P. Dick, and X. S. Hu, "Temperature-aware scheduling and assignment for hard real-time applications on mpsocs," *Design, Automation and Test in Europe (DATE '08)*, pp. 288–293, March 2008.

[24] Y. Yu, S. Ren, and O. Frieder, "Interval-based timing constraints their satisfactions and applications," *IEEE Transactions on Computers*, vol. 57, no. 3, pp. 418–432, 2008.