Computing Schedules for Time-Triggered Control using Genetic Algorithms

Truong Nghiem [*] **Georgios E. Fainekos** [**]

[*] *Univ. of Pennsylvania, Philadelphia, USA (nghiem@seas.upenn.edu)*
[**] *Arizona State University, Tempe, USA (fainekos@asu.edu)*

Abstract: In this work, we study the optimal scheduling problem for embedded control systems. Given a number of numerical computation tasks that must be executed on an embedded computer, the goal is to find a periodic schedule that minimizes the error of the implementation with respect to the ideal system. In this paper, we pose *two open problems*: the decidability of the optimal scheduling problem for timed-triggered controllers (OSTTC) and the time complexity of the corresponding bounded version of the problem. We conjecture that the former is undecidable and that the latter is NP-Complete. In view of the previous conjectures, we present a Genetic Algorithm (GA) for the OSTTC problem. Initial experimental results of GA-OSTTC are very promising when compared to random search and exhaustive bounded search.

Keywords: Scheduling; Digital control; Genetic algorithms;

1. INTRODUCTION

Digital control theory (Aström and Wittenmark (1997)) has traditionally focused on the design and analysis of computer control systems under the assumption that the computer can perform all required computations within a single sampling step. This assumption is valid for certain applications, such as in industrial control systems where a dedicated powerful computer can operate within the required timing constraints. However, it might fail in real time systems and, in particular, in embedded control systems where a microprocessor might not be powerful enough and/or might have other tasks to execute, too. In such cases, the control tasks are not executed and updated simultaneously, but they are executed according to a schedule and only certain control inputs and outputs are updated after a task is completed. Obviously, this control scheduling approach results in a degradation of the performance of the control system with respect to the ideal implementation. The goal of this paper is to compute control schedules such that the reduction in system performance is minimized.

This work builds upon previous research on time-triggered implementations of control systems (Yazarel et al. (2005); Nghiem et al. (2006)). Time-triggered architectures offer opportunities for a more predictable mapping of control models (Kopetz and Bauer (2003)). In a time-triggered implementation, instead of mapping control blocks to periodic tasks, the compiler can allocate well-defined time slots to control blocks. Given a mapping of all the control blocks to the time slots, one can precisely define the trajectories of the implementation and quantify the error with respect to the model-level semantics as indicated in (Yazarel et al. (2005); Nghiem et al. (2006)).

In turn, the quantification of the error defines a metric on the performance of the implementation. The smaller the error is, the closer the implementation is to the ideal system behavior. Searching for the optimal schedule for the implementation reduces to the problem of minimizing the error of the implementation over the space of all possible schedules. We refer to this problem as the Optimal Scheduling problem for Timed-Triggered Controllers (OSTTC). Currently, it is an open problem whether there exists an algorithm which can compute a schedule whose implementation error is arbitrarily close to the implementation error of the optimal schedule. In this paper, we conjecture that the implementation error of the optimal schedule cannot be approximated by an algorithm, in general. Also of interest is the bounded version B-OSTTC of OSTTC. Namely, can we compute the optimal schedule within the set of all schedules up to some length? Obviously, the latter problem is decidable; however, its time complexity is unknown. We conjecture that the approximability of the B-OSTTC is NP-Complete.

In view of the aforementioned potentially negative results and due to the importance of the problem, we present a new Genetic Algorithm (GA) (Sumathi et al. (2008)) for the OSTTC problem. The GA-OSTTC metaheuristic is the main contribution of this paper. Initial experimental results are very promising and demonstrate the feasibility of the proposed approach to this difficult problem. We also demonstrate the applicability of our method to the design of control schedules for a Computerized Numerical Control (CNC) milling machine.

This work falls within the general context of implementing feedback control loops by software (Caspi and Maler (2005)). The problem of generating code from hybrid automata has been considered in Alur et al. (2003) and Hur et al. (2004), but the focus has been on choosing the sampling period so as to avoid errors due to switching and communication. Automaton-based scheduling of control systems was introduced in Alur and Weiss (2008). However, a quantitative measure of the performance was not considered. Recently, Mazo and Tabuada (2008) presented event-triggered methods for real-time scheduling of stabilizing controllers where the scheduling is online and determined by predefined events. Evolutionary and Genetic Algorithms have been applied in the past to the scheduling problem, for example, in non-preemptive hard-real time scheduling (Pico and Wainwright (1994)). In control engineering, EA/GA have been used for optimization in controller design and in nonlinear model predictive control (Fleming and Purshouse (2002)).

2. PROBLEM DEFINITION AND PRELIMINARIES

2.1 Feedback Control Model

A feedback control model is a tuple $\mathcal{M} = \langle \mathcal{M}_P, \mathcal{M}_C \rangle$ consisting of a *plant model* $\mathcal{M}_P$ and a *controller model* $\mathcal{M}_C$. In this paper, we consider linear time-invariant (LTI) plant models

$$\mathcal{M}_P : \begin{cases} \dot{x}(t) = A_p x(t) + B_p u(t) \\ y(t) = C_p x(t) \end{cases} \qquad x(0) \in \mathbb{R}^n$$

where $x \in \mathbb{R}^n$ is the plant state vector, $y \in \mathbb{R}^p$ is the output, $u \in \mathbb{R}^m$ is the control input, $A_p \in \mathbb{R}^{n\times n}$, $B_p \in \mathbb{R}^{n\times m}$ and $C_p \in \mathbb{R}^{p\times n}$.

A controller model $\mathcal{M}_C = (\mathcal{B}_1, \ldots, \mathcal{B}_r)$ consists of r control blocks. Each block describes the dynamics of some controller state z or computes the values of some control input u in terms of observable plant outputs, controller states, and other control inputs. The control blocks are assumed to be well-defined so that given the plant state at time t_0 and the plant outputs for time $t \geq t_0$, the plant state variables and controller state variables at time $t \geq t_0$ are uniquely determined. The interconnection of the control blocks defines a set of differential and algebraic equations (see Nghiem et al. (2006) for details). In this paper, we consider controller models which are linear

$$\mathcal{M}_C : \begin{cases} \dot{z}(t) = A_c z(t) + B_c y(t), \qquad z(0) = 0 \\ u(t) = K_P y(t) + K_I z(t) + K_D \dot{y}(t) + L_c u(t) \end{cases}$$

where $z \in \mathbb{R}^q$ is the state of the controller, $A_c \in \mathbb{R}^{q\times q}$, $B_c \in \mathbb{R}^{q\times p}$, $K_P \in \mathbb{R}^{m\times p}$, $K_I \in \mathbb{R}^{m\times q}$, and $L_c \in \mathbb{R}^{m\times m}$. We assume that the dependence between control inputs is acyclic, i.e., L_c is a lower triangular matrix. In the special case where $A_c = 0$, $B_c = I$ and $L_c = 0$, we obtain the familiar equation $u(t) = K_P y(t) + K_I \int_0^t y(\tau)d\tau + K_D \frac{dy}{dt}(t)$ of the proportional-integral-derivative (PID) controllers. This model also captures more general observer-based controllers, as shown in Nghiem et al. (2006).

2.2 Model Level Semantics

Given a feedback control model $\mathcal{M} = \langle \mathcal{M}_P, \mathcal{M}_C \rangle$, the *continuous-time semantics* of the feedback control model, denoted by $[\![\mathcal{M}]\!]$, is defined as the collection of all trajectories $(x(t), y(t), u(t), z(t))$ which for all $t \geq 0$ satisfy the differential and algebraic equations of $\mathcal{M}_P$ and $\mathcal{M}_C$. We assume that the feedback composition is well-posed, meaning that for any initial plant state $x(0)$ the equations have unique solutions. The continuous-time semantics is *implementation independent* semantics that is used for the mathematical analysis and design of controllers that achieve desired performance specifications on the output $y(t)$ using a variety of techniques from control theory.

2.3 Implementation Level Semantics

The ideal model-level semantics assumes that all control blocks of $\mathcal{M}_C$ are "computed" *instantaneously* and *simultaneously.* Of course, any software implementation of $\mathcal{M}_C$ will violate both assumptions. Moreover, ideal differentiation and integration computations assumed by the model-level semantics cannot be satisfied by the implementation, where approximation algorithms must be used.

The control blocks of controller $\mathcal{M}_C = (\mathcal{B}_1, \ldots, \mathcal{B}_r)$ are usually defined during the process of designing the controller, using a design tool such as SIMULINK. Typically, the designer may choose among several structures of $\mathcal{M}_C = (\mathcal{B}_1, \ldots, \mathcal{B}_r)$ to express the same controller model. In this paper, the structure considered is the same as in Nghiem et al. (2006), in which a block $\mathcal{B}_I$ integrates the controller state z_i and m blocks $\mathcal{B}_1, \ldots, \mathcal{B}_m$ compute the control inputs u_j respectively.

We assume that the implementation is on a time-triggered platform in which time can be allotted in fixed-size slots of *length* δ, and each control block is assigned to a fixed number of such slots. The computation of each block $\mathcal{B}_j$ (or $\mathcal{B}_I$) consists of reading the relevant plant output variables using sensors, updating the control input u_j (or controller state z), and finally writing the computed control value to the actuators at the end of its allotted time. The computation time of each block is captured by a *timing function* $\tau : \{\mathcal{B}_I, \mathcal{B}_1, \ldots, \mathcal{B}_m\} \to \mathbb{N}$ which associates to each block the number of time slots needed to execute it. To model the order in which the control blocks are executed we consider a *dispatch sequence* (or *schedule*) ρ, which is an infinite string over the set $\Sigma_{\mathcal{B}} = \{\mathcal{B}_0, \mathcal{B}_I, \mathcal{B}_1, \ldots, \mathcal{B}_m\}$. Here, $\mathcal{B}_0$ is used to model idling from the viewpoint of the controller (e.g., allocation of a time slot to activities other than the computation of control outputs). Typically, ρ will be periodic, and will be specified by a finite string that repeats. For example, given a controller $\mathcal{M}_C = (\mathcal{B}_I, \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3)$, a possible dispatch sequence is the nonuniform sequence $(\mathcal{B}_I\mathcal{B}_1\mathcal{B}_2\mathcal{B}_I\mathcal{B}_1\mathcal{B}_3\mathcal{B}_0)^\omega$ that includes idling. Note that a dispatch sequence contains only ordering information, and is thus independent of the processing speed of the platform.

Given a feedback control model $\mathcal{M} = \langle \mathcal{M}_P, \mathcal{M}_C \rangle$, a dispatch sequence ρ, a timing function τ and a time slot length δ, we can define the *implementation semantics* associated with $\mathcal{M}$, denoted as $[\![\mathcal{M}]\!]_{(\rho,\tau,\delta)}$, to be the set of trajectories obtained by executing the control blocks of controller $\mathcal{M}_C$ according to the dispatch sequence ρ, where the number of slots of length δ for each control block are chosen according to the timing function τ. The details can be found in Section 2.3 of Nghiem et al. (2006).

2.4 Problem Definition

The main goal of this paper is to find a schedule for the computation such that the behavior of the implementation is as close as possible to the ideal system. In order to solve the optimal scheduling problem, we must first quantify the quality of the controller implementation for a particular dispatch sequence ρ, timing function τ and time slot length δ. Having defined both the ideal platform-independent semantics, and the platform-dependent semantics, we can directly define the error of the implementation as a function of the initial plant state $x(0)$ simply as

$$(x(t), y(t), u(t), z(t)) = [\![\mathcal{M}]\!](x(0))$$
$$(\tilde{x}(t), \tilde{y}(t), \tilde{u}(t), \tilde{z}(t)) = [\![\mathcal{M}]\!]_{(\rho,\tau,\delta)}(x(0))$$
$$e_{\mathcal{M}}(\rho, \tau, \delta, x(0)) = \int_0^{+\infty} \|y(t) - \tilde{y}(t)\|_2^2 dt \qquad (1)$$

We are therefore measuring the implementation error in the L_2 sense. Note that the implementation error is measured on the output variables of the overall closed loop system, rather than the error on the controller variables.

Given a feedback control model $\mathcal{M}$ and a set of initial plant states X_0, we will say that the implementation $(\rho_1, \tau_1, \delta_1)$ is more accurate than the implementation $(\rho_2, \tau_2, \delta_2)$ (noted $(\rho_1, \tau_1, \delta_1) \preceq_{\mathcal{M}} (\rho_2, \tau_2, \delta_2)$) if the implementation error of $(\rho_1, \tau_1, \delta_1)$ is smaller than that of $(\rho_2, \tau_2, \delta_2)$ for all initial states:

$$\forall x(0) \in X_0 \quad e_{\mathcal{M}}(\rho_1, \tau_1, \delta_1, x(0)) \leq e_{\mathcal{M}}(\rho_2, \tau_2, \delta_2, x(0)).$$

Note that the relation $\preceq_{\mathcal{M}}$ is a preorder on the set of implementations. Now, we are in position to define the problem of Optimal Schedules for Timed-Triggered Control (OSTTC).

Problem 1. (OSTTC). Given a feedback control model $\mathcal{M}$, a set of initial plant states X_0, a timing function τ and a time slot length δ, find a dispatch sequence $\rho' \in \Sigma_{\mathcal{B}}^+$ such that for any other dispatch sequence $\rho \in \Sigma_{\mathcal{B}}^+$, $(\rho', \tau, \delta) \preceq_{\mathcal{M}} (\rho, \tau, \delta)$.

In other words, given a particular platform (which defines τ and δ), find a dispatch sequence that minimizes the implementation error. Recall that $\Sigma_{\mathcal{B}}^+ = \cup_{i=1}^{\infty} \Sigma_{\mathcal{B}}^i$ denotes the set of all finite length strings that are built from symbols from $\Sigma_{\mathcal{B}}$. Note that OSTTC does not impose any restrictions on the length of the dispatch sequence. For pragmatic reasons, a control schedule executed on a real platform must also satisfy certain length limitations. Therefore, we also consider the bounded version of OSTTC, i.e., computation of optimal dispatch sequences up to some length l. Here, $\Sigma_{\mathcal{B}}^{\leq l} = \Sigma_{\mathcal{B}} \cup \Sigma_{\mathcal{B}}^2 \cup \ldots \cup \Sigma_{\mathcal{B}}^l$.

Problem 2. (B-OSTTC). Given a feedback control model $\mathcal{M}$, a set of initial states X_0, a timing function τ, a time slot length δ and an integer $l > 0$, find a periodic dispatch sequence $\rho' \in \Sigma_{\mathcal{B}}^{\leq l}$ such that for any other periodic dispatch sequence $\rho \in \Sigma_{\mathcal{B}}^{\leq l}$, $(\rho', \tau, \delta) \preceq_{\mathcal{M}} (\rho, \tau, \delta)$.

The contributions of this paper are the following. In Section 3, we pose two new open problems. Namely, the approximability of the OSTTC and B-OSTTC problems. We conjecture that there does not exist an algorithm that can solve the approximation version of the OSTTC problem. Similarly, we conjecture that the approximation version of B-OSTTC cannot be solved in polynomial time. Finally, in Section 4, we present a metaheuristic evolutionary algorithm which can provide good dispatch sequences to either problem.

We will use a Computerized Numerical Control (CNC) milling machine as a running example in this paper.

Example 3. (CNC machine). Consider a 2-axis CNC machine where each axis is driven by a DC servo motor. The CNC controller for each axis is a cascade feedback control structure with command feedforward (Tsai et al. (2004)). The inner loop uses a PI controller $(K_{P,v} + K_{I,v}s)$ to regulate the motor speed. Position control along the corresponding axis is performed by the outer loop with a P rule (K_P). Precomputed trajectories in position P, velocity V and acceleration A are generated by the CNC computer. The objective of CNC control is to track the position command as closely as possible. To reduce the tracking error, a command feedforward controller $(K_a,\ K_{v_1},\ K_{v_2})$ is utilized with the reference commands V and A. Given a precomputed position trajectory $P(t) = [x(t)\ y(t)]^T$ and an actual position trajectory $\tilde{P}(t) = [\tilde{x}(t)\ \tilde{y}(t)]^T$ for a time horizon $T > 0$, $0 \leq t \leq T$, where x and y are the positions on the two axes, the tracking error is computed by

$$e(P, \tilde{P}) = \int_{t=0}^{T} \left[(x(t) - \tilde{x}(t))^2 + (y(t) - \tilde{y}(t))^2 \right] \mathrm{d}t \qquad (2)$$

The controllers for both axes are implemented on the same computer using the time-triggered architecture. We consider three different implementation schemes:

- Scheme 1: there is only one control block which performs all the control functions of both axis controllers.
- Scheme 2: there are two control blocks: $\mathcal{B}_x$ controls the x-axis motor and $\mathcal{B}_y$ controls the y-axis motor.
- Scheme 3: there are two control blocks for each axis: $\mathcal{B}_x^1$ ($\mathcal{B}_y^1$) reads the current position on the x-axis (y-axis) and computes the set-point value for the speed control loop, $\mathcal{B}_x^2$ ($\mathcal{B}_y^2$) obtains the current velocity, computes the control value, then actuates the motor for the x-axis (y-axis).

For schemes 2 and 3, it is necessary to design a schedule for the control blocks so as to reduce the tracking error (2).

2.5 Review of Error Analysis

In this section, the main result for the computation of the implementation error $e_{\mathcal{M}}(\rho, \tau, \delta, x(0))$ is reviewed. In detail, Theorem 2 in Nghiem et al. (2006) allows the exact computation of $e_{\mathcal{M}}(\rho, \tau, \delta, x(0))$.

Theorem 4. The implementation error is exactly equal to

$$e_{\mathcal{M}}(\rho, \tau, \delta, x(0)) = \bar{\psi}(0)^T \hat{\mathcal{O}} \bar{\psi}(0) \qquad (3)$$

where $\bar{\psi}(0) = Hx(0)$ with $H = [\ I\ 0\ I\ 0\ 0\ 0\ 0\ 0\ ]^T$, and $\hat{\mathcal{O}}$ is the solution of the nested Lyapunov equations

$$\hat{\mathcal{O}} = \hat{G}_0^T \hat{Q} \hat{G}_0 + \hat{E}_0^T \mathcal{O} \hat{E}_0, \quad \mathcal{O} = \hat{E}^T \mathcal{O} \hat{E} + \hat{G}^T \hat{Q} \hat{G} \qquad (4)$$

for implementation-dependent matrices $\hat{G}_0$, $\hat{Q}$, $\hat{E}_0$, $\hat{E}$ and $\hat{G}$ which are presented in Section 3 of Nghiem et al. (2006).

Theorem 4 states that the L_2-norm error is a global quadratic function of the initial state x_0. In implementing the controller, we are only interested in implementations that follow the ideal system, in the sense that the corresponding implementation error is finite for all $x(0) \in \mathbb{R}^n$. From the Lyapunov equation (4), it is straightforward to see that this can only be achieved if matrix $\hat{E}$ is stable (Rugh (1996)), as stated in the following corollary.

Corollary 5. The implementation error $e_{\mathcal{M}}(\rho, \tau, \delta, x(0))$ for a given implementation (ρ, τ, δ) is finite for all $x(0) \in \mathbb{R}^n$ if and only if matrix $\hat{E}$ is stable, that is all eigenvalues of $\hat{E}$ are inside the unit circle on the complex plane.

3. DESIGN OF DISPATCH SEQUENCES

When designing a time-triggered implementation of a given control model $\mathcal{M}$, the main problem is choosing a dispatch sequence ρ that gives good performance while conforming to certain requirements, e.g., the minimum ratio of idling time. For a specific implementation (ρ, τ, δ) of a control model $\mathcal{M}$, Theorem 4 provides a quantitative measure of its quality by allowing us to compute the implementation error as a quadratic function of the initial state. Intuitively, the smaller the error is, the better the implementation is. If $\|H^T \hat{\mathcal{O}} H\|_2$ is used as the performance measure of an implementation, the optimal scheduling problem is equivalent to solving the minimization $\rho^\star = \arg\min_{\rho \in \Pi} \|H^T \hat{\mathcal{O}}(\rho) H\|_2$, where Π is either $\Sigma_{\mathcal{B}}^+$ or $\Sigma_{\mathcal{B}}^{\leq l}$ and $\hat{\mathcal{O}}(\rho)$ depends on ρ (see Nghiem et al. (2006)).

In practice, a system usually starts from a bounded set of initial conditions X_0, i.e., $x(0) \in X_0$. In this case, we are trying to solve the following optimization problem:

$$\sigma^\star = \min_{\rho \in \Pi} \max_{x \in X_0} x^T H^T \hat{\mathcal{O}}(\rho) H x,$$
$$\rho^\star = \arg\min_{\rho \in \Pi} \max_{x \in X_0} x^T H^T \hat{\mathcal{O}}(\rho) H x.$$

Given a dispatch sequence ρ, if X_0 is a compact convex set then $\max_{x \in X_0} x^T H^T \hat{\mathcal{O}}(\rho) H x$ is a quadratic program and it can be efficiently solved using quadratic programming techniques (Boyd and Vandenberghe (2004)).

In the following discussion, we will use the notion of (K, L)-approximation algorithms. Assume the existence of an algorithm that on the input of Problem 1, it returns a number $\tilde{\sigma}$. Following Tsitsiklis and Blondel (1997), we say that an algorithm is a (K, L)-approximation algorithm if for any problem instance $(\mathcal{M}, X_0, \tau, \delta)$, we have $|\tilde{\sigma} - \sigma^\star| \leq K + L\sigma^\star$. The parameter $K \geq 0$ models the absolute error and the parameter $L \in [0, 1)$ the relative error.

Let us consider approximation versions of Problems 1 and 2. OSTTC is a combinatorial optimization problem over a countable set since $\Sigma_{\mathcal{B}}^+$ contains the set of all possible dispatch sequences. An approximation version of OSTTC can be stated as following.

Problem 6. (A-OSTTC). Given a feedback control model $\mathcal{M}$, a set of initial states X_0, a timing function τ and a time slot length δ, does there exist a (K, L)-approximation algorithm for computing $\sigma^\star$ when $\Pi = \Sigma_{\mathcal{B}}^+$?

Currently, it is unknown whether such an approximation algorithm exists. The main difficulty in providing an answer to Problem 6 lies in the very complex matrix definitions and manipulations in the computation of $e_{\mathcal{M}}$ (Theorem 4). We conjecture that there exists no (K, L)-approximation algorithm for computing $\sigma^\star$ when $\Pi = \Sigma_{\mathcal{B}}^+$.

Clearly, if we restrict the search space of OSTTC, the decision version B-OSTTC becomes a decidable problem (on a machine that can manipulate real numbers). Then, the question we need to investigate is whether we can find an efficient algorithm, i.e., a polynomial-time algorithm, that solves the optimal scheduling problem. Consider the following approximation version of B-OSTTC.

Problem 7. (A-B-OSTTC). Given a feedback control model $\mathcal{M}$, a set of initial plant states X_0, a timing function τ, a time slot length δ and an integer $l > 1$, does there exist a polynomial-time (K, L)-approximation algorithm for computing $\sigma^\star$ when $\Pi = \Sigma_{\mathcal{B}}^{\leq l}$?

By Corollary 5, we conjecture that the complexity of A-B-OSTTC is related to the complexity of determining the existence of a stable finite-length product of matrices from a finite set of matrices. The latter can be shown to be NP-complete (based on a straightforward modification of the proof of Theorem 2 in Blondel and Tsitsiklis (1997) for the k-Mortality problem). Thus, we conjecture that there exists no polynomial-time (K, L)-approximation algorithm for computing $\sigma^\star$ when $\Pi = \Sigma_{\mathcal{B}}^{\leq l}$ unless $P = NP$.

4. A GENETIC ALGORITHM FOR OSTTC

In this work, we utilize Evolutionary Algorithms (Sumathi et al. (2008)) and, in particular, we formulate a Genetic Algorithm in order to solve the optimal scheduling problem (GA-OSTTC). Briefly, genetic algorithms are based on the concept of evolutionary development of species. The algorithm starts with an initial population which is evaluated for its aptness with respect to a fitness function. The best members of the population are selected and combined giving rise to new offspring which hopefully have better characteristics than the previous generation. This basic operation essentially exploits promising regions of the search space. The metaheuristic algorithm is enhanced with other genetic operations such as mutation, which mimics the natural process of random mutation of genes. Mutation helps in the exploration of the state space.

The first step in providing the details for the GA-OSTTC is to define the representation of the population. In the OSTTC problem, a population is a subset of the set of all possible dispatch sequences. We model each dispatch sequence as a sequence of integers, in which an integer i represents control block $\mathcal{B}_i$. For example, the dispatch sequence $(\mathcal{B}_1, \mathcal{B}_3, \mathcal{B}_2, \mathcal{B}_4)^\omega$ will be encoded as $(1, 3, 2, 4)$. Note that the theoretical results (schema theorem in Sumathi et al. (2008)) for the validity of GA as search algorithms use binary encodings.

Next, we describe the mechanisms used in GA-OSTTC.

Fitness function. One important ingredient of the GA is the fitness function f, which computes a measure of how good a member is. In GA-OSTTC, this measure is the performance of the individual dispatch sequence ρ, computed as one of the following three values:

- $e_{\mathcal{M}}(\rho, \tau, \delta, x(0))$ for an initial state $x(0)$,
- $\sup_{x \in X_0} e_{\mathcal{M}}(\rho, \tau, \delta, x)$ for a set X_0 of initial states,
- $\|H^T \hat{\mathcal{O}}(\rho) H\|_2$ (see section 3).

In practice, it might be preferable to limit the length of the dispatch sequences so that they are not excessively long. To this purpose, the fitness function can be selected so that it imposes a penalty on the length of the sequence. Let $f_e(\rho)$ be the performance measure of a dispatch sequence ρ and $l(\rho)$ be its length. Depending on how strict the length constraint should be, an appropriate fitness function can be defined. For example, the fitness function $f(\rho) = f_e(\rho)\,(1 + \lambda\,(\max\{l(\rho), l_{max}\} - l_{max}))$, for $\lambda > 0$, imposes a loose constraint on $l(\rho)$ being at most l_{max}, while a stringent constraint is imposed by

$$f(\rho) = f_e(\rho) e^{\lambda(\max\{l(\rho), l_{max}\} - l_{max})}, \quad \lambda > 0. \qquad (5)$$

Initialization. We initialize N_R different populations, each with N_P members. Each population can evolve entirely independently from the other populations. Therefore, they can be executed in parallel. During the initialization, each member (dispatch sequence) is randomly generated with uniform distribution and its fitness value is evaluated.

Crossover. The main operation of GA is the crossover. At this stage, we pick α pairs of individuals (dispatch sequences) from each population P_j at random by weighting their fitness values. Then, for each such pair, we pick a position uniformly at random from the longer one, split the sequences and exchange them. For example, consider the individuals $\rho_1 = (2, 1, 3, 2)$ and $\rho_2 = (1, 1, 4, 1, 2, 2)$ and assume that we split them at position 2, the offspring will be $\rho'_1 = (2, 1, 4, 1, 2, 2)$ and $\rho'_2 = (1, 1, 3, 2)$. If we split at position 5, which exceeds the length of ρ_1, then $\rho'_1 = (2, 1, 3, 2, 2)$ and $\rho'_2 = (1, 1, 4, 1, 2)$. Finally, we evaluate the offspring and add them to the population.

Mutation. As opposed to crossover where the goal is to mix the best members of a population, mutation is an operation that takes place upon individuals. The goal of mutation is to diversify the populations and increase the coverage of the search space. At each generation i, we choose β individuals at random and mutate each of them using one of the following methods:

(M1) Add a random block to a random position.
(M2) Remove one block from a random position.
(M3) Change the block at a random position to another random block.
(M4) Inverse the sequence.
(M5) Duplicate the sequence.

The algorithm selects one of these rules with certain probabilities, specifiable by users as parameters to the algorithm. Let $P_M = (P_{M1}, P_{M2}, P_{M3}, P_{M4}, P_{M5})$ where $P_{Mi} \geq 0$ is the probability of choosing rule Mi, for $i = 1 \dots 5$, and $\sum_{i=1}^{5} P_{Mi} = 1$. Through experiments, we found a good set of values for P_M to be $P_M = (0.2778, 0.1389, 0.4167, 0.0694, 0.0972)$. However, different probabilities can be chosen for different applications of the GA-OSTTC.

5. NUMERICAL EXPERIMENTS

The GA-OSTTC algorithm was applied on several numerical examples. The Global Optimization Toolbox of MatLab was used for the implementation. In the following, N_G is the total number of generations and N_E is the number of generations between exchange of populations.

5.1 PID Controller

Consider the example in Nghiem et al. (2006), in which a PID controller controls a plant with dynamics $\dot{x}(t) = A_P x(t) + B_P u(t)$ and $y(t) = C_P x(t)$, where $x(t) \in \mathbb{R}^4$, $u(t) \in \mathbb{R}^2$, $y(t) \in \mathbb{R}^2$,

$$A_P = \begin{bmatrix} -1020 & -156.3 & 0 & 0 \\ 128 & 0 & 0 & 0 \\ 0 & 0 & -10.2 & -2.002 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad B_P = \begin{bmatrix} 8 & 0 \\ 0 & 0 \\ 0 & 0.5 \\ 0 & 0 \end{bmatrix},$$

and $C_P = \left[\begin{smallmatrix} 0 & 4.8828 & 0 & 0 \\ 0 & 0 & 0 & 0.4 \end{smallmatrix}\right]$. The parameters of the PID controller are

$$K_P = \begin{bmatrix} -116 & 0 \\ 0 & -250 \end{bmatrix}, K_I = \begin{bmatrix} -480 & 0 \\ 0 & -30 \end{bmatrix}, K_D = \begin{bmatrix} -0.2 & 0 \\ 0 & -20 \end{bmatrix}$$

Table 1. Results of Example 1 ($\delta = 0.001$s).

Algorithm	(1)	(2)	Best Cost	Ave. Time
GA-OSTTC (strict)	23	0	0.0139	11.1s
GA-OSTTC (loose)	0	34	0.0114	15.52s
RS ($l_{max} = 10$)	2	0	0.0139	15.64s
RS ($l_{max} = 20$)	0	3	0.0132	27.59s

The time-triggered implementation of this PID controller has three control blocks: $\mathcal{B}_I$ computes the integration, $\mathcal{B}_1$ computes u_1, and $\mathcal{B}_2$ computes u_2.

To illustrate the advantage of GA-OSTTC, we compare it to the brute-force search and the random search (RS) algorithms. The brute-force algorithm searches the entire space of all possible dispatch sequences up to a certain length l_{max} and selects the best one. The RS algorithm chooses a length less than l_{max} at random and selects each block in the dispatch sequence randomly with uniform distribution. It terminates after a certain number N_1 of iterations without improvement or after a maximum number $N_2 > N_1$ of iterations, and returns the best dispatch sequence that it found. The norm $\|H^T \hat{\mathcal{O}}(\rho) H\|_2$ was used as the performance measure of dispatch sequence ρ in all experiments.

First, δ was set to 0.001s and the length up to 10. The optimal sequence $\rho_1^\star = (\mathcal{B}_1\mathcal{B}_1\mathcal{B}_1\mathcal{B}_1\mathcal{B}_2\mathcal{B}_2\mathcal{B}_1\mathcal{B}_2\mathcal{B}_I\mathcal{B}_2)^\omega$ with optimal cost 0.0139 was found by the brute-force search algorithm after 252 seconds on our computer. GA-OSTTC was executed with $N_R = 2$, $N_P = 30$, $N_G = 100$, and $N_E = 10$. Two experiments were performed with two different fitness functions: one is strict on constraining the length of dispatch sequences to $l_{max} = 10$ (Eq. (5) with $\lambda = 2$) and the other is loose (Eq. (5) with $\lambda = 0.01$). The RS algorithm was also carried out with $N_1 = 5000$ and $N_2 = 10^5$, and for $l_{max} = 10$ and $l_{max} = 20$.

Each experiment was repeated 40 times and the results are reported in Table 1. In column (1) are the numbers of times $\rho_1^\star$ was found during the 40 executions. Loosening the constraint on $l(\rho)$ allows GA-OSTTC to search for dispatch sequences of length longer than $l_{max} = 10$, which possibly are better than $\rho_1^\star$. The numbers of such dispatch sequences found by each algorithm are reported in column (2). The best performance measures $\|H^T \hat{\mathcal{O}}(\rho) H\|_2$ are reported in the fourth column, and the average execution times are in the last column. Obviously, GA-OSTTC was not only faster than the RS algorithm, but also found more and better dispatch sequences. Compared to the brute-force search algorithm, GA-OSTTC was a magnitude faster and produced as good, or even better, dispatch sequences in most cases.

To test these algorithms to the limit, we raised the time slot length δ to 0.002s, at which there are only 167 stable dispatch sequences in 3^{10} sequences of length up to 10. The optimal sequence found by brute-force search was $\rho_2^\star = (\mathcal{B}_I\mathcal{B}_I\mathcal{B}_2\mathcal{B}_I\mathcal{B}_I\mathcal{B}_I\mathcal{B}_I\mathcal{B}_1\mathcal{B}_2\mathcal{B}_1)^\omega$ with optimal cost 1.7856. The search took 240 seconds. The above experiments were repeated; this time GA-OSTTC has $N_P = 40$. The results are reported in Table 2. Again, GA-OSTTC performed much better than the RS algorithm.

Table 2. Results of Example 1 ($\delta = 0.002$s).

Algorithm	(1)	(2)	Best Cost	Ave. Time
GA-OSTTC (strict)	11	0	1.7856	10.84s
GA-OSTTC (loose)	0	21	0.2107	29.71s
RS ($l_{max} = 10$)	1	0	1.7856	13.48s
RS ($l_{max} = 20$)	0	6	1.5489	25.95s

Table 3. Experimental results of Example 2.

Controller	Tracking Error $e(P(t), \tilde{P}(t))$
Ideal controller	3.415×10^{-3}
Scheme 1 ($\delta_1 = 2$ms)	Unstable
Scheme 2 (with $\rho_2^\star$, $\delta_2 = 1.5$ms)	3.909×10^{-3}
Scheme 3 (with $\rho_3^\star$, $\delta_3 = 1$ms)	3.842×10^{-3}

5.2 CNC Control

In this experiment, we consider the 2-axis CNC machine from Example 3. We used the system parameters of the example in Tsai et al. (2004). Let δ_1, δ_2, δ_3 be the worst-execution times of the control blocks (and also the time slot lengths) of scheme 1, scheme 2 and scheme 3, respectively. Obviously, $\delta_1 > \delta_2 > \delta_3$. Using basic linear system theory, we computed the maximum allowable value for δ_1 to be approximately 1.61ms, i.e., the CNC system is unstable if $\delta_1 > 1.61$ms. However, it might still be stable using either scheme 2 or scheme 3, provided that δ_2 and δ_3 are small enough. Since we divide the control computation into smaller control blocks, we are able to update it faster and focus on the sub-systems with faster dynamics (i.e., the two speed-control loops). In fact, we found that schemes 2 and 3 are feasible if δ_2 and δ_3 do not exceed approximately 1.5ms.

Assume that $\delta_2 = 1.5$ms and $\delta_3 = 1$ms. Using GA-OSTTC, we computed a good dispatch sequence for scheme 2 to be $\rho_2^\star = (\mathcal{B}_y\mathcal{B}_x\mathcal{B}_y)^\omega$ and for scheme 3 to be $\rho_3^\star = \left(\mathcal{B}_y^1\mathcal{B}_y^2\mathcal{B}_x^1\mathcal{B}_y^2\mathcal{B}_y^2\mathcal{B}_x^2\right)^\omega$. These two schedules were compared on how well the CNC machine tracked a reference circle with radius 100mm. The reference profiles for velocity and acceleration were computed using the motion planning method in Park (1996), with a desired speed of 200mm/s and a peak acceleration of 700mm/s^2. The tracking errors between the reference circular trajectory and the actual trajectory (Eq. (2)) yielded by the ideal continuous-time controller and by the three time-triggered implementation schemes are given in Table 3. Note that with time slot length $\delta_1 = 2$ms, implementation scheme 1 is unstable. Both implementations 2 and 3 perform well and their tracking errors are close to that of the ideal controller. Moreover, scheme 3 is better than scheme 2, which is intuitive because $\rho_3^\star$ prioritizes the speed control blocks while $\rho_2^\star$ does not.

6. CONCLUSIONS

In this paper, we have stated two open problems regarding the computation of optimal schedules for time-triggered control architectures. In order to tackle these two difficult problems, we have introduced a Genetic Algorithm (GA-OSTTC). The experimental results indicate that GA-OSTTC provides better solutions than random search and that it is order of magnitudes faster than exhaustive search. Future research will focus on proving the conjectures for the open problems in Section 3.

REFERENCES

Alur, R., Ivancic, F., Kim, J., Lee, I., and Sokolsky, O. (2003). Generating embedded software from hierarchical hybrid models. In *Proc. of the ACM Conf. on Languages, Compilers, and Tools for Embedded Systems*, 171–182.

Alur, R. and Weiss, G. (2008). Regular specifications of resource requirements for embedded control software. In *Proc. of the 2008 IEEE Real-Time and Embedded Technology and Applications Symposium*, 159–168. IEEE Computer Society.

Aström, K. and Wittenmark, B. (1997). *Computer-controlled systems: Theory and Design.* Prentice Hall.

Blondel, V.D. and Tsitsiklis, J.N. (1997). When is a pair of matrices mortal? *Information Processing Letters*, 63(5), 283–286.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press.

Caspi, P. and Maler, O. (2005). From control loops to real-time programs. In *Handbook of Networked and Embedded Control Systems.* Birkhäuser.

Fleming, P. and Purshouse, R. (2002). Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice*, 10(11), 1223–1241.

Hur, Y., Kim, J., Lee, I., and Choi, J. (2004). Sound code generation from communicating hybrid models. In *Hybrid Systems: Computation and Control, Proc. of the 7th International Workshop*, LNCS 2993, 432–447.

Kopetz, H. and Bauer, G. (2003). The time triggered architecture. *Proc. of the IEEE*, 91(1), 112–126.

Mazo, M. and Tabuada, P. (2008). On event-triggered and self-triggered control over sensor/actuator networks. In *Proc. 47th IEEE Conf. Decision and Control*, 435–440.

Nghiem, T., Pappas, G.J., Alur, R., and Girard, A. (2006). Time-triggered implementations of dynamic controllers. In *Proc. of the 6th ACM & IEEE International conference on Embedded software*, 2–11. ACM.

Park, J.S. (1996). Motion profile planning of repetitive point-to-point control for maximum energy conversion efficiency under acceleration conditions. *Mechatronics*, 6(6), 649–663.

Pico, C. and Wainwright, R. (1994). Dynamic scheduling of computer tasks using genetic algorithms. In *IEEE Conf. on Evolutionary Computation*, 829–833.

Rugh, W.J. (1996). *Linear System Theory.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, second edition.

Sumathi, S., Hamsapriya, T., and Surekha, P. (2008). *Evolutionary intelligence: an introduction to theory and applications with Matlab.* Springer.

Tsai, M.C., Chiu, I.F., and Cheng, M.Y. (2004). Design and implementation of command and friction feedforward control for cnc motion controllers. *IEE Proc. - Control Theory and Applications*, 151(1), 13–20.

Tsitsiklis, J.N. and Blondel, V.D. (1997). The lyapunov exponent and joint spectral radius of pairs of matrices are hard - when not impossible - to compute and to approximate. *Mathematics of Control, Signals, and Systems*, 10(1), 31–40.

Yazarel, H., Girard, A., Pappas, G.J., and Alur, R. (2005). Quantifying the gap between embedded control models and time-triggered implementations. In *Proc. of the 26th IEEE Real-Time Systems Symposium*, 111–120.