A Dual-Mode Strategy for Performance-Maximisation and Resource-Efficient CPS Design

XIAOTIAN DAI, WANLI CHANG, SHUAI ZHAO, and ALAN BURNS, University of York, UK

The emerging scenarios of cyber-physical systems (CPS), such as autonomous vehicles, require implementing complex functionality with limited resources, as well as high performances. This paper considers a common setup in which multiple control and non-control tasks share one processor, and proposes a dual-mode strategy. The control task switches between two sampling periods when rejecting (coping with) a disturbance. We create an optimisation framework looking for the switching sampling periods and time instants that maximise the control performance (indexed by settling time) and resource efficiency (indexed by the number of tasks that are schedulable on the processor). The latter objective is enabled with schedulability analysis tailored for the dual-mode model. Experimental results show that (i) given a set of tasks, the proposed strategy improves the control performances whilst retaining schedulability; and (ii) given requirements on the control performances, the proposed strategy is able to schedule more tasks.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; *Real-time systems.*

Additional Key Words and Phrases: cyber-physical systems, dual-mode scheduling, resource dimensioning, switching systems, schedulability analysis, non-convex optimisation

ACM Reference Format:
Xiaotian Dai, Wanli Chang, Shuai Zhao, and Alan Burns. 2019. A Dual-Mode Strategy for Performance-Maximisation and Resource-Efficient CPS Design. *ACM Trans. Embedd. Comput. Syst.* 18, 5s, Article 85 (October 2019), 20 pages. https://doi.org/10.1145/3358213

1 INTRODUCTION

Cyber-physical systems (CPS) often involve a mixture of control tasks, which interact with the physical dynamics in closed loops, and non-control tasks, which perform operations mainly on the cyber side. The emerging application scenarios of CPS, such as autonomous vehicles and intelligent robots, are implementing ever more complex functionality. On one hand, the requirements on performances are high. In addition, the conventionally used indirect metrics like quadratic cost, which usually enable closed-form analysis, do not satisfy the practical needs; instead, timing-related metrics such as settling time are demanded. On the other hand, the resources on the implementation platform are limited. This is a particularly important aspect when mass production and deployment are targeted.

This work considers a common setup that multiple control and non-control tasks share one processor. The non-control tasks are modelled as periodic tasks. The perturbations to the control

This article appears as part of the ESWEEK-TECS special issue and was presented at the International Conference on Embedded Software (EMSOFT) 2019.
Authors' address: Xiaotian Dai, xiaotian.dai@york.ac.uk; Wanli Chang, wanli.chang@york.ac.uk; Shuai Zhao, shuai.zhao@york.ac.uk; Alan Burns, alan.burns@york.ac.uk, University of York, Department of Computer Science, Heslington, York, UK.

1539-9087/2019/10-ART85 $15.00
https://doi.org/10.1145/3358213

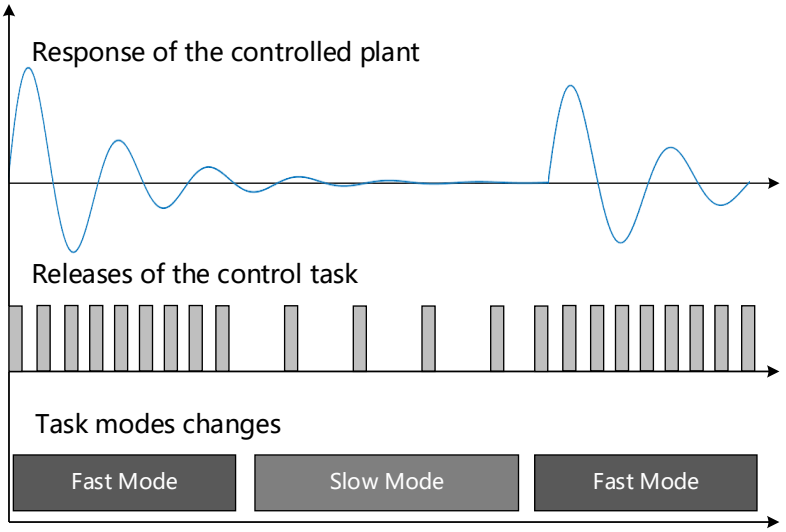

Fig. 1. An illustration of the dual-mode model

plant arrive sporadically. All tasks are under fixed-priority preemptive scheduling. The *general idea of this paper* is that, we exploit a dual-mode model, where the control task switches between two sampling periods when rejecting (coping with) a disturbance, as illustrated in Figure 1, for better control performance and resource utilisation.

Main contributions: The novelty of this work largely lies on developing a switching strategy considering both control systems performance and scheduling resources utilisation simultaneously. The existing works on switching systems in the control theory literature have mainly focused on guaranteeing stability of arbitrary switching, using theoretical tools such as common quadratic Lyapunov functions (CQLF) and switched Lyapunov functions (SLF) [20]. They generally do not consider the implementation resources. In this work, while guaranteeing control stability, we optimise the two switching sampling periods and the switching time instant in the dual-mode model, towards minimising the settling time, which is a direct time-domain measurement of the control performance, and maximising the number of tasks that are schedulable on the processor, which reflects the resource efficiency. We also develop schedulability analysis for the dual-mode model, to be used in the above optimisation framework. The entire strategy is analysable and determined during the design phase.

This is the first work that targets both timing-related control performance and resource dimensioning within a switching system. We make a significant step from the control-scheduling co-design literature. Generally, first, they can only treat simple indirect control performance metrics like quadratic cost, which facilitate optimisation but counter-intuitive; second, they consider schedulability as a utilisation constraint to satisfy, but do not optimise the number of tasks that are schedulable on a given platform, i.e., resource dimensioning. More detailed discussion will be provided in the next section.

Organisation of the paper: Section 2 discusses the related work. The technical background is described in Section 3, including the dual-mode task model. Two motivational examples are given in Section 4, showing that by exploiting the dual-mode model, control performance and resource efficiency can both be improved. Section 5 and 6 report the schedulability analysis and the optimisation framework, respectively. Experimental results are presented in Section 7, and Section 8 contains concluding remarks.

2 RELATED WORK

Traditionally, control systems are designed independently from task scheduling, assuming that the underlying operating platform will provide the necessary sampling periods. The idea of cooperatively designing a control system with real-time scheduling is introduced in [25], to overcome the limitation of such independent assumption and improve task utilisation. It is later shown in [2] that designing a control system separately without considering the underlying real-time facilities could result in an unexpected control performance. The methods discussed for better scheduling control tasks include flexible task models, timing compensation and feedback scheduling. In [3], the authors introduce further scheduling issues that reduce control performance to motivate control-scheduling co-design, e.g., sampling jitter, input-output delay, interferences from higher-priority tasks and non-determinism from general-purpose hardware and off-the-shelf operating systems. None of the above consider a switching strategy.

Approaches to dynamically adjust the sampling periods have been reported [11, 12, 18], resulting in switching systems. However, these works have three main issues. First, the analysis on the stability of arbitrary switching is insufficient, if there is any. Second, quadratic cost is used as the objective, and the proposed approaches are not able to handle complicated and practically relevant control performance metrics. Third, they satisfy schedulability as a constraint, and do not address the problem of resource dimensioning, i.e., how many tasks can be scheduled on the given hardware platform, or almost equivalently, how many implementation resources are required to schedule all given tasks. On the contrary, our work guarantees stability, minimises the settling time, and maximises the number of schedulable tasks.

There are other works leading to switching systems, but also with different issues. For instance, an event-based controller is implemented in [1] as an alternative to the traditional time-triggered control paradigm. As a consequence, the corresponding task of the controller will be released aperiodically. The authors claim that this method could lead to large reduction in the CPU utilisation, while making minor control performance degradation. However, while stability of an event-based controller may be guaranteed, it is hard to verify the control performance against the requirement as well as predict the scheduling performance.

Another relevant direction of work is the (m, k)-firm scheduling [19]. Instead of executing every job instance of the task, the scheduler only needs to schedule at least m job instances out of any k consecutive releases. This is feasible as occasional misses of the output update can be tolerated by most control applications. In [24], the (m, k)-firm model is used as a less stringent guarantee than the hard deadline requirement. It is shown how to achieve graceful degradation when a system is overloaded. The fundamental difference from our work is that, the (m, k)-firm scheduling is a passive methodology reacting to unforeseen but bounded circumstances and keeping the degradation of performances to an acceptable level, while we are implementing an active methodology pursuing high performances and schedulability.

Our work also connects to the optimal sampling period assignment [8, 13]. The optimal sampling problem is tackled in [8], where the sampling instants and control inputs are selected to minimise a given function of the system state and control input. In particular, a necessary condition for the optimality of a set of sampling instants is derived and a quantization-based sampling strategy is proposed to be computationally tractable. However, the proposed method is only applied in the LQR (linear quadratic regulation) problem, which is relatively simple to analyse due to its quadratic cost function. The implementation resources are not considered.

Online optimal sampling period assignment is investigated in [13] to maximise the control performance. A feedback scheduler is developed to periodically assign new sampling periods based on the current plant states. It is shown that most computation can be done online and stored in a

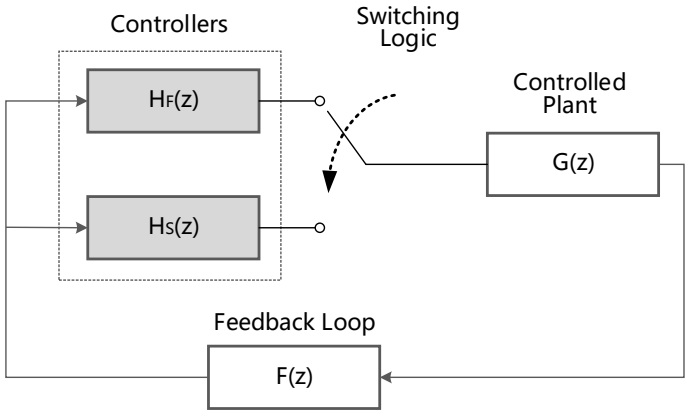

Fig. 2. An illustration of the switched control model

look-up table. Again, only the quadratic cost function is considered. Besides, since the switching is not fixed, yet occurs depending on the plant states in real time, stability cannot be guaranteed.

3 BACKGROUND

This section gives the technical background on the control systems under investigation and the dual-mode task model, including its implementation. Further details on real-time task scheduling and digital control systems can be found in [10, 14, 15].

3.1 Modelling of the control system

A discrete-time dynamic system can be mathematically modelled using a difference equation. In this work, we use a transfer function in the z-domain, $G(z)$, to describe the input-output dynamics of a discrete-time linear time-invariant (LTI) single-input-single-output (SISO) control system [22]:

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}}{a_0 + a_1 z^{-1} + \cdots + a_n z^{-n}} \tag{1}$$

where $X(z)$ represents the system input, $Y(z)$ represents the system output response, both in the z-domain, n is the system order (the order of $G(z)$), $\{a_i \mid i \in \{0, 1, \ldots, n\}\}$ and $\{b_i \mid i \in \{0, 1, \ldots, m\}\}$ are constant coefficients. There is often a constraint on $X(z)$ that has to be satisfied, known as control input saturation.

Assuming the transfer function of the controller to be $H(z)$, and the feedback path to be $F(z)$, the closed-loop transfer function can be written as:

$$G_c(z) = \frac{G(z)H(z)}{1 + G(z)H(z)F(z)} \tag{2}$$

The controller $H(z)$ is first designed in continuous time and then discretized. Depending on the control task mode, i.e., the fast or the slow mode, two versions of controllers, $H_F(z)$ and $H_S(z)$ are used mutually exclusively. Both controllers are designed using the same control law and ensured to be stable, but with different time constants and parameters according to the sampling period associated with the mode. A diagrammatic representation that demonstrates the overall feedback loop is shown in Figure 2. The switching between modes is controlled by the switching logic, and only one of the controllers is activated at any instant of time.

To specify the performance of a controller, the settling time [14, 15] is used. It is defined as the time taken for a controlled plant to get within a certain range, normally 2% or 5%, of the new

equilibrium, i,e, the steady-state, without subsequently deviating from it. In this work, the unit step response is evaluated and the settling time is obtained by simulating the control system. In the dual-mode task model that will be elaborated in the following subsection, the settling time varies depending on the switching sampling periods and the switching time instant.

3.2 The dual-mode task model

Consider a task set Γ with n control tasks $\Gamma_c = \{\tau_{c(1)}, \tau_{c(2)}, \cdots, \tau_{c(n)}\}$ and m non-control tasks $\Gamma_{nc} = \{\tau_{nc(1)}, \tau_{nc(2)}, \cdots, \tau_{nc(m)}\}$. The task set is hosted on a fixed-priority preemptive scheduler using deadline monotonic priority assignment. A non-control task in Γ_{nc} is defined as: $\tau_i \equiv (C_i, T_i, D_i = T_i)$, where C_i is the worst-case execution time, T_i is the task period for periodic tasks, or the minimal inter-arrival time for sporadic tasks, D_i is the deadline which is set equal to T_i. For a control task τ_i in Γ_c under the dual-mode model:

$$\tau_i \equiv (\mathcal{M}_i, C_i, T_i^H, T_i^L, \alpha_i, D_i, T_i^\Gamma) \tag{3}$$

where $\mathcal{M}_i = \{\mathcal{M}_S, \mathcal{M}_F\}$ is the task mode, in which $\mathcal{M}_S$ stands for the slow mode, and $\mathcal{M}_F$ for the fast mode; C_i is the worst-case execution time; T_i^H and T_i^L are the fast-mode sampling period and slow-mode sampling period, respectively; α_i is a ratio that determines the time instant after which a mode switch is performed, where α_i is defined in the region of (0,1); D_i is the deadline, and can be set equal to T_i^H. This ensures that, as long as the task is schedulable, no sampling period will be missed. T_i^Γ is the minimum inter-arrival time of disturbances from sporadic physical events. We assume that the new disturbance will not arrive until the current one is fully rejected.

In this dual-mode model, the switching is performed bidirectionally. A control task initially executes with the sampling period T_i^H. A switching from T_i^H to T_i^L is then made at the relative time $t = \alpha_i T_i^\Gamma$. As the change of task mode will not take effect until the next job release, the exact switching point t_S occurs at:

$$t_S = \left\lceil \frac{\alpha_i T_i^\Gamma}{T_i^H} \right\rceil \cdot T_i^H \tag{4}$$

and

$$T_i = \begin{cases} T_i^H, & \text{if } t \le t_S \\ T_i^L, & \text{if } t > t_S \end{cases} \tag{5}$$

The task will switch back to T_i^H when a new disturbance is experienced at $t \ge T_i^\Gamma$, and t will be reset to 0. The task will stay in the slow mode if no disturbance occurs.

Generally speaking, a longer sampling period produces worse control performance and even causes instability, but it can also lead to an improved system schedulability. We define an upper bound T_i^+, beyond which the control system output response becomes unacceptable, and a lower bound T_i^-, which should keep the utilisation of the task τ_i not exceeding the maximum allowed value. A simple and extreme value can be obtained from $C_i/T_i^- \le 1$. Overall we have $T_i^- \le T_i^H < T_i^L \le T_i^+$. Since T_i^- is a fixed value for a given control task, its priority is independent of the chosen switching sampling periods and remains unchanged at run-time. Note that the choices of T_i^+ and T_i^- will not be able to guarantee stability, which will be explained later in this section.

The task utilisation in this dual-mode model can be computed as

$$U_i = \alpha_i \frac{C_i}{T_i^H} + (1 - \alpha_i)\frac{C_i}{T_i^L}, \ \forall \tau_i \in \Gamma_C \tag{6}$$

or to be more exact

$$U_i = \frac{t_S C_i}{T_i^\Gamma T_i^H} + (1 - \frac{t_S}{T_i^\Gamma})\frac{C_i}{T_i^L},\ \forall \tau_i \in \Gamma_C \tag{7}$$

Note that this is the worst-case utilisation. The task might not switch back to $\mathcal{M}_F$ immediately after T_i^Γ, in which case the actual utilisation is smaller than U_i.

Implementing such a dual-mode task requires a run-time monitor and a software timer. When a disturbance in the control system output is observed by the monitor (a threshold is crossed), t_i^H is set and the timer starts. After the switching point t_S, t_i^L is set. The routine described above can be either executed in the kernel mode or integrated into the scheduler and invoked from the scheduler handler. The run-time overhead is negligible in terms of both memory and computational resources.

At the end of this section, we would like to make a note on the switching stability of this dual-mode model. As discussed earlier in the paper, there have been theoretical tools developed to ensure stability of arbitrary switching. In this model, the task only switches once with the switching sampling periods and switching time instant fixed during the design phase. Therefore, as long as both controllers corresponding to the two sampling periods are stable, there is no issue of switching instability, given that there is enough separation between the two consecutive disturbances, which follows the assumption on the minimum inter-arrival time.

4 MOTIVATIONAL EXAMPLES

In this section, we show the potential of the dual-mode strategy with two motivational examples. First, for a given set of tasks, the control performance can be improved while the schedulability is maintained. A control task that cannot meet the performance requirement with a uniform sampling period is able to meet it when taking the dual-mode model. Second, while satisfying the requirements on the control performance, more tasks can be scheduled on the processor.

4.1 Example One: Improving control performance

The control performance depends on how often a controller responds to the plant dynamics, i.e., sampling periods. For a control task with a uniform sampling period, in order to achieve high performance, generally the period needs to be short. However, there is a bound on the sampling period coming from the utilisation, which has been discussed in Section 3. In this example, we consider a non-control task with the period $T_j = 0.6s$, the worst-case execution time $C_j = 0.3s$, and its deadline equals to the period. Therefore, the utilisation of the control task cannot exceed 0.5 due to the existence of this non-control task.

We study a second-order plant, whose continuous-time dynamics is given in the s-domain by:

$$G(s) = \frac{15}{s^2 - 0.2s + 25.01} \tag{8}$$

To control this plant, a Proportional-Integral-Derivative (PID) controller is designed using the following parameters[1]: $K_p = 26.35$, $K_i = 66.09$, $K_d = 2.06$. The transfer function of the PID controller is:

$$H(s) = \frac{2.06s^2 + 26.35s + 66.09}{s} \tag{9}$$

To run this PID controller on a computer system, it needs to be discretized according to a sampling period. We assume the worst-case execution time of the task $C_i = 10ms$ and as mentioned above,

[1]This is done with the commonly used MATLAB PIDTuner. A description is available at: https://uk.mathworks.com/help/control/ref/pidtuner.html.

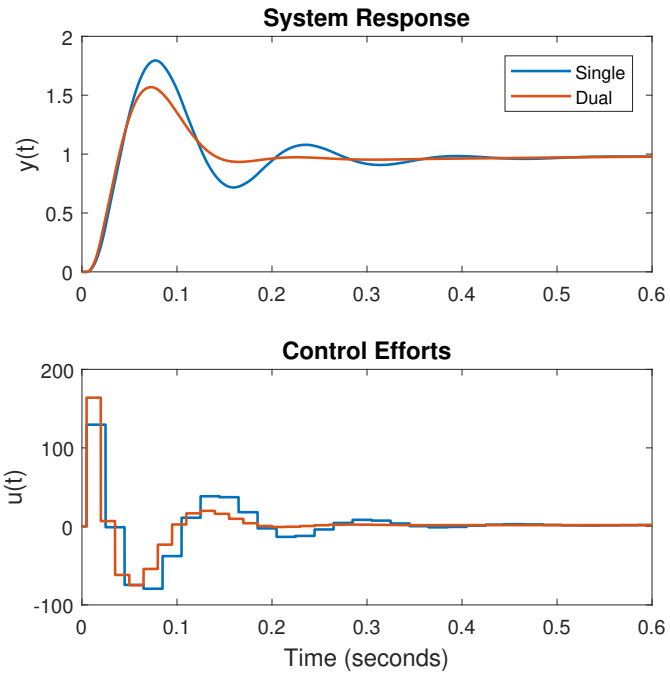

Fig. 3. Control system output response and input for (i) a uniform sampling period; (ii) the dual-mode strategy

the utilisation $U_i \leq 0.5$. Therefore, the minimum allowed sampling period is $T_i = 20ms$. This results in a discrete-time PID controller with the following transfer function in the z-domain[5]:

$$H(z) = \frac{10.3z^2 + 5.8z - 2.88}{z - 1} \tag{10}$$

The plant dynamics is then also discretized, leading to a discrete-time closed-loop transfer function in the z-domain, as shown in Section 3.

However, this controller produces unsatisfactory output response, as illustrated in Figure 3. The settling time TS is $0.35s$ and does not satisfy the requirement $\hat{TS} \leq 0.3s$. A shorter sampling period may improve the control performance and meet its requirement, but will violate the constraint on the utilisation, making the system unschedulable.

When a dual-mode model is applied, the control task can initially run at $T_i^H = 15ms$, and then switch to $T_i^L = 35ms$ after $t_S = 30ms$. The schedulability is maintained and the timing of task execution is illustrated in Figure 4. The control task has a higher priority. A formal and exact schedulability analysis for a task set with dual-mode tasks will be given later in Section 5. The controller in (9) needs to be discretized into two versions with sampling periods $15ms$ and $35ms$, respectively: $H_F(z)$ and $H_S(z)$. Under this dual-mode model, the settling time is reduced to $0.18s$, satisfying the control performance requirement. The system output response is plotted in Figure 3. In both cases, the uniform sampling period and the dual-mode model, the constraint on the control input is respected. Intuitively, the initial phase with a larger disturbance is more critical than the later phase when the system has more or less settled down. Therefore, the dual-mode model responding to the plant dynamics according to its needs is preferred.

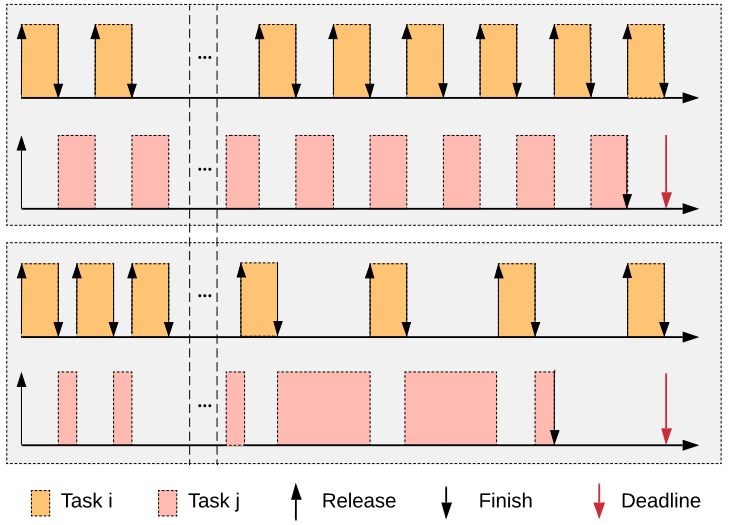

Fig. 4. An illustration of the scheduling when the control task τ_i is under (i) the uniform sampling period, $T_i = 20ms$ (the top diagram); (ii) the dual-mode model, $T_i^H = 15ms$ and $T_i^L = 35ms$ (the bottom diagram). Some repetitive task instances are omitted due to space limit. The control task τ_i has a higher priority.

4.2 Example Two: Improving system schedulability

This example demonstrates that the system schedulability can be improved when a dual-mode model is applied. We assume four tasks with details described in Table 1. τ_1 is a control task and the other three are non-control tasks. This task set is ordered by the priority p_i with 0 as the highest. All tasks are released at the same time.

As shown in Figure 5 (the top half), this task set is initially unschedulable as the task τ_4 (with the lowest priority) finishes after its deadline. In fact, the total task utilisation is $\sum U = 1.1$, exceeding the maximum utilisation of a single processor. Further examining the data set identifies that the control task has an utilisation of 0.4 and contributes most of the interference to τ_4. In order to make the task set schedulable, τ_4 has to meet its deadline, by fitting the additional $\Delta C_i = 6ms$ (the block in red) into the schedule.

Table 1. Parameters of the example task set. τ_1 is a control task. τ_2, τ_3, τ_4 are non-control tasks. p_i is task priority

Task	p_i	C_i (ms)	T_i (ms)	D_i (ms)	Schedulable?
τ_1	0	4	10	10	✓
τ_2	1	2	12	12	✓
τ_3	2	2	14	14	✓
τ_4	3	20	50	50	×

We apply the dual-mode model and switch the sampling period of the control task τ_1 (with the highest priority), from $10ms$ to $20ms$ after $t = 14ms$. It can be seen in the bottom half of Figure 5, the available time slots for τ_4 increase due to the longer sampling period of τ_1 after the switch. This additional capacity makes the task set schedulable. Again, the schedulability analysis will be

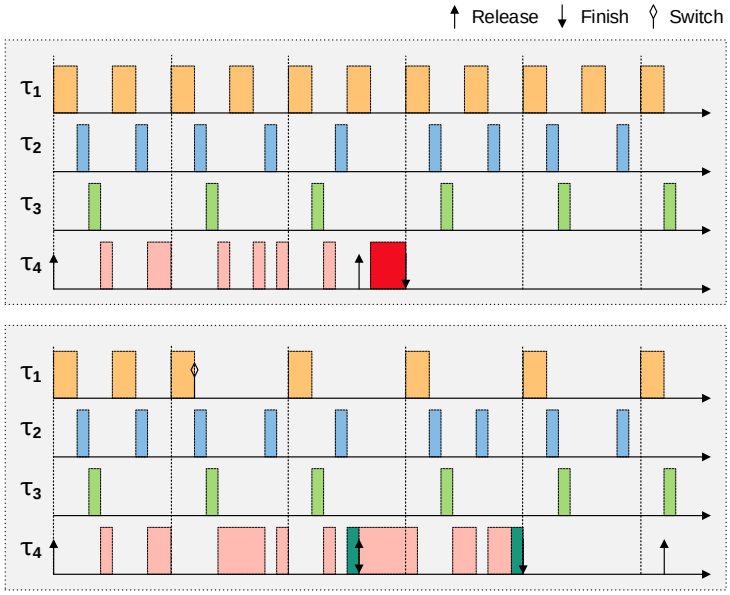

Fig. 5. Converting τ_1 into a dual-mode task makes an unschedulable task set (the top diagram, note the red block) schedulable (the bottom diagram). All tasks are released at the same time.

provided later. Since the control system in this example settles to the steady state before $t = 14ms$, this dual-mode model does not introduce any penalty to the control performance.

5 SCHEDULABILITY ANALYSIS FOR THE DUAL-MODE MODEL

For the fixed-priority scheduling, the schedulability of a task can be checked through response-time analysis. We assume that the tasks are independent without blocking due to sharing of other resources. The worst-case response time R_i is calculated by adding the task worst-case execution time C_i and the interference time due to preemptions from higher-priority tasks I_j:

$$\begin{aligned} R_i &= C_i + \sum I_j \\ &= C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \end{aligned} \tag{11}$$

in which $\lceil \cdot \rceil$ is the ceiling function and $hp(i)$ specifies the indices of all tasks that have higher priorities than the task i. This equation indicates that the response time analysis of the dual-mode task with two switching periods is no different from a single-period task, as the response time of a task with a constrained (or implicit) deadline is independent of its own period. The worst-case response time is compared against the deadline (i.e., the shorter period as explained in Section 3) to check the schedulability. If the dual-mode task is schedulable with its shorter period, it will remain schedulable with the longer period and during switching [7]. For the tasks with higher priorities than the dual-mode task, the response time analysis remains unchanged, as the dual-mode task is a lower-priority task for them and causes no interference, thus playing no role in the response time analysis.

However, for the tasks with lower priorities than the dual-mode task, their response time is influenced as the interference pattern in the dual-mode needs to be modelled — the response time

analysis in (11) needs to be extended. Following the models introduced in Section 3, we consider a lower-priority task τ_i and a dual-mode task τ_j, with $j \in hp(i)$. The critical instant (maximum interference) is when τ_j arrives at the same time as τ_i, runs with the shorter period first, switches to the longer period at t_S, but then switches back to the shorter period as early as possible (this pattern being repeated). This follows from the fact that the worst-case execution time, C_j, remains the same in both modes, and hence the maximum interference occurs when the task executes as frequently as allowed by the model. Depending on the finishing time of the task τ_i, there are three cases to consider for calculating the response time of τ_i:

(1) If $R_i < t_S$, the task τ_j always executes with T_j^H, and the resultant worst-case response time of τ_i is no different from the case if τ_j is a normal periodic task with the period T_j^H:

$$R_i = C_i + \left\lceil \frac{R_i}{T_j^H} \right\rceil C_j$$

(2) If $t_S \leq R_i \leq T_j^\Gamma$, after the switching, the task τ_j will use T_j^L which reduces the number of interferences:

$$R_i = C_i + \left(\frac{t_S}{T_j^H} + \left\lceil \frac{R_i - t_S}{T_j^L} \right\rceil \right) C_j \tag{12}$$

(3) If $R_i > T_j^\Gamma$, the worst-case behaviour is that the dual-mode task immediately switches back to the fast mode after T_j^Γ. We let the total interference time on the task τ_i during T_j^Γ be denoted by I_Γ, which can be derived from (12), and n indicate the number of T_j^Γ that τ_i experiences:

$$R_i = C_i + \begin{cases} nI_\Gamma + \left\lceil \frac{R_i - nI_\Gamma}{T_j^H} \right\rceil C_j, & \text{if } (R_i \bmod T_j^\Gamma) < t_S \\ nI_\Gamma + \left(\frac{t_S}{T_j^H} + \left\lceil \frac{R_i - nI_\Gamma - t_S}{T_j^L} \right\rceil \right) C_j, & \text{if } (R_i \bmod T_j^\Gamma) \geq t_S \end{cases} \tag{13}$$

in which $n = \lfloor R_i / T_j^\Gamma \rfloor$.

To generalise for all cases, we introduce the *max()* and *min()* functions which output the larger or lower value of two parameters, respectively. Overall, we have:

$$R_i = C_i + nI_\Gamma + \left(\left\lceil \frac{\min(t_S, R_i - nI_\Gamma)}{T_j^H} \right\rceil + \left\lceil \frac{\max(0, R_i - nI_\Gamma - t_S)}{T_j^L} \right\rceil \right) C_j \tag{14}$$

This models the interference from a single dual-mode task. For each additional such task, a similar interference item I_j is added. The resulting response time equation can be solved in the usual way (using fixed-point interaction on a recurrence relation). If all tasks are computed to have response times no greater than their deadlines, then the set of control and non-control tasks is schedulable.

At the end of this section, we would like to point [6] to interested readers, where schedulability analysis of mixed-criticality tasks with different periods is discussed. Contrary to our work that all deadlines are expected to be met, deadline misses of low-criticality tasks can be tolerated in [6]. In addition, earliest-deadline-first scheduling is assumed in [6], while we take fixed-priority scheduling.

6 OPTIMISATION FRAMEWORK

As illustrated by the motivational examples in Section 4, the proposed dual-mode task model can provide better control performance indexed by settling time and resource efficiency reflected by the number of tasks that are schedulable. However, the success of this dual-mode model depends on sensible parameters, which are challenging to select. These model parameters include the two

switching periods and the switching time instant for each control task, and will be referred to as decision variables.

First, there is no analytical method to compute the settling time from the decision variables mentioned above. Second, the relationship between the decision variables and the objectives to optimise (settling time and utilisation) as well as the constraints (control input saturation and schedulability) is complex and non-convex. Third, the decision space has a large number of dimensions — for a given system with n control tasks, the dimension of the decision space to search is $3n$. Fourth, these decision variables are not independent. For instance, the decision variables of the same task jointly contribute to the settling time. The decision variables of different tasks collectively influence the system schedulability. In addition, as intuitively discussed earlier, there exists a trade-off between the control performance and the resource efficiency, where in general longer sampling periods can lead to reduced processor utilisation and thus better schedulability, but may have negative impact on the settling time.

From the above analysis, we have to deploy heuristics to search the decision space for solutions that respect all the constraints and optimise the objectives. Heuristic search has been adopted in embedded real-time systems for searching feasible task parameter settings [4, 16, 17]. In particular, genetic algorithms have been successfully applied for searching feasible task allocation, priority ordering and resource sharing solutions of real-time systems in the presence of either network communications or blocking [21, 23, 27].

In this section, we report an optimisation framework looking for the parameters in the dual-mode model based on genetic algorithms. There are two scenarios coming from the practical demands: (i) given a set of tasks, we would like to maximise the control performances while maintaining schedulability; (ii) we would like to maximise the number of tasks that are schedulable on a processor while satisfying control performance requirements. Before presenting the problem formulation and explaining the solution approach, we will first discuss the parameter encoding.

6.1 Parameter encoding

Following the models in Section 3, the optimisation framework considers a set Γ with n plants to be controlled (forming Γ_c) and m non-control tasks (forming Γ_{nc}). This problem of parameter tuning (i.e., searching for the decision variables) under study can be effectively mapped to a typical mixed-integer programming optimisation problem [26] with nonlinear inequality constraints, where a correlation exists between the tuning parameters. For n control tasks in the dual-mode, the tuning parameters are encoded as one sequence

$$\Psi_\Gamma = \{T_1^H, T_1^L, \alpha_1, T_2^H, T_2^L, \alpha_2, ..., T_n^H, T_n^L, \alpha_n\}$$

where Ψ_Γ denotes the chromosome encoding of the task set Γ. For each control task τ_i in Γ, its tuning decision variables should comply with the following value bounds and constraints, as previously discussed in Section 3.2:

$$\textbf{value bounds:} \begin{cases} T_i^- > 0 \\ T_i^+ < T_i^\Gamma \\ 0 \le \alpha_i \le 1 \end{cases}$$

$$\textbf{constraint:} \quad T_i^- \le T_i^H < T_i^L \le T_i^+$$

where T_i^- and T_i^+ are predefined constants as mentioned earlier, providing value bounds for the tuning parameters T_i^L and T_i^H. This encoding method is simple but effective, as there is often a limited number of control tasks in the system, and hence the encoding sequence will not be long.

6.2 Problem formulation

The control performance of τ_i, where $\tau_i \in \Gamma_c$, is denoted as $P_c(i)$, and

$$P_c(i) = \frac{\hat{TS}_i - TS_i}{\hat{TS}_i} \tag{15}$$

where TS_i is the settling time and $\hat{TS}_i$ is the requirement. The overall control performance of a system λ_C is computed by summing up the control performances of individual control tasks and then taking the average:

$$\lambda_C = \frac{1}{n} \sum_{i=1}^{n} P_c(i) \tag{16}$$

With the above fitness function on λ_C, we can formulate the optimisation problem for the first scenario that aims to maximise the control performance:

$$\begin{array}{ll} \text{Given:} & \text{a fixed task set} \\ \text{Maximise} & \lambda_C \\ \text{On} & \{\alpha_i, T_i^H, T_i^L \mid \tau_i \in \Gamma_c\} \\ \text{s.t.} & \forall i \in \Gamma_c : \\ & 0 < \alpha_i \leq 1 \\ & T_i^- \leq T_i^H < T_i^L \leq T_i^+ \\ & P_c(i) \geq 0 \\ & \forall \tau_j \in \Gamma : \\ & R_j \leq D_j \end{array} \tag{17}$$

In addition to the value bounds and constraint on the decision variables, there are two other constraints to be respected: (i) the task set is schedulable; (ii) the control performance requirement is satisfied. The schedulability test can be conducted using the schedulability analysis discussed in Section 5. A solution point (a set of decision variables) resulting in a schedulable system (i.e., all the tasks are able to meet their deadlines) is considered as feasible and assigned with a positive fitness value based on the fitness function. For the solution points resulting in unschedulable systems, the fitness value is 0, i.e., the globally minimum fitness value.

For each control task $\tau_i \in \Gamma_c$, there exists a control performance requirement on the settling time, denoted by $\hat{TS}_i$ as discussed above. The settling time TS_i must be shorter than or equal to this value, which is equivalent to $P_c(i) \geq 0$. In addition, the constraint on the control input needs to be respected, which can be checked together with the settling time when simulating the control system. Solution points that fail to meet this requirement are also assigned with the minimum fitness value (i.e., 0). By assigning 0 to the infeasible solution points, the genetic algorithm solver is given a clear searching direction towards feasible space.

The problem formulation for the second scenario aiming to maximise the number of tasks that are schedulable on the processor is similar to the one above, except for a different fitness function. We use λ_U measuring the idle utilisation of the whole system, including both control and non-control tasks, as the fitness function

$$\lambda_U = 1 - \sum_{\tau_i \in \Gamma} U_i \tag{18}$$

Once the optimisation is completed, we will deploy a simple exhaustive algorithm to allocate as many more (non-control) tasks to the processor as possible. During this process, schedulability of the entire system is always ensured. We cannot directly use the number of tasks that are schedulable

as the fitness function, as its differentiability is poor. Many solution points will then have the same fitness value, making it difficult for the genetic algorithm solver to conduct an effective search.

Based on the above discussion on the constraints and fitness functions, in the genetic algorithm solver, the assignment of the fitness value (denoted as λ with a range of $[0, 1]$) for any solution point under both scenarios with different fitness functions is summarised as follows

$$\lambda = \begin{cases} 0, \text{ if } (\exists \tau_i \in \Gamma_c,\ TS_i \geq \hat{TS}_i) \ \vee \ (\exists \tau_j \in \Gamma,\ R_j > D_j) \\ \lambda_C, \text{ if optimising control performance} \\ \lambda_U, \text{ otherwise} \end{cases} \tag{19}$$

6.3 Execution of the genetic algorithm solver

With the parameter encoding, fitness functions and constraints constructed, the optimisation problem under study can be fed to a general-purpose genetic algorithm solver. This solver starts with a population of randomly generated but feasible (i.e., complying with all the constraints) solution points (sets of decision variables) of a given system, based on the encoding approach discussed earlier in Section 6.1. For each generation, the genetic algorithm evaluates every individual point via one of the fitness functions given in Section 6.2, depending on the optimising scenario. A limited number of individuals that have the best fitness values will be passed directly into the next generation (i.e., elitism). Then, with tournament selection, the genetic algorithm produces the next generation via generic crossover and mutation. With the above procedure, the genetic algorithm continuously produces new generations, and intuitively, new solution points with better quality. The genetic algorithm solver is terminated when either the ideal point (i.e., the upper bound for the given fitness function) is achieved or the maximum number of generations is reached.

We would like to make a note that, in this work, we compute the precise control performance (settling time) with simulation, since we aim to provide hard guarantees as demanded in CPS. Approximations are possible to improve the computational efficiency, especially for certain classes of control systems. In addition, those individuals that have been computed will not be evaluated again in the next iterations, if no mutations/crossovers have been performed.

7 EXPERIMENTAL RESULTS

In this section, experiments are conducted to evaluate the proposed dual-mode strategy. First, we examine its properties as compared with the uniform sampling period case. Then we demonstrate both scenarios that maximise the control performance and the resource efficiency, respectively. We also show the effectiveness of our optimisation solver, which is conducted and can be reproduced via the genetic algorithm interface `ga()` provided by Matlab R2017b Global Optimization Toolbox v3.4.3[2].

MATLAB/Simulink is adopted in the optimisation framework for simulating the control system. A customised fixed-priority scheduler, namely the *FPS-DUAL* module, is integrated into the simulation model, as shown in Figure 6. The *FPS-DUAL* simulates the timing and scheduling sequences of all tasks. It should be noted that the behaviour of the control systems depends on the timing and scheduling that can be provided by the *FPS-DUAL* module. This implementation of co-simulation is made publicly available for reproducibility[3].

[2]https://uk.mathworks.com/products/global-optimization.html
[3]https://github.com/automaticdai/research-dual-period

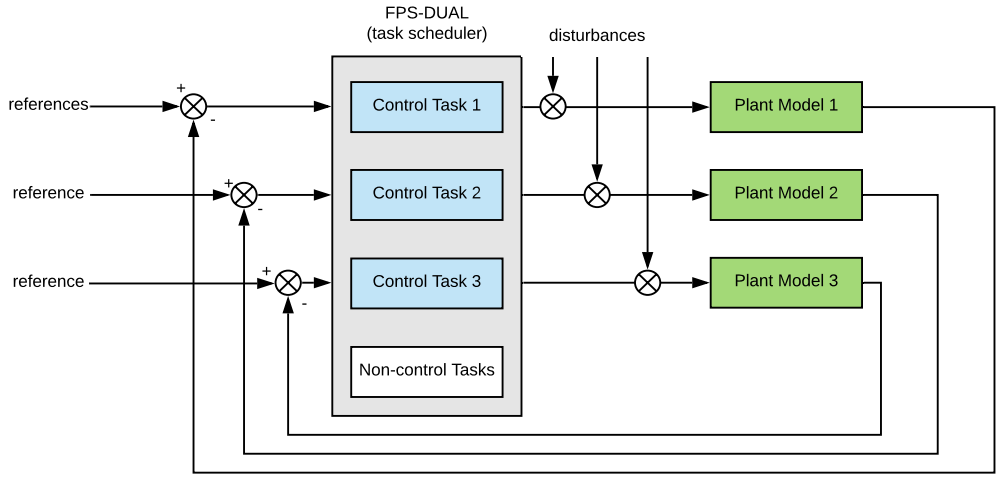

Fig. 6. The conceptual Simulink model with the *FPS-DUAL* module adopted for co-simulation of control systems and scheduling under the dual-mode model

7.1 Effectiveness of the dual-mode strategy

In this experiment, we demonstrate the effectiveness of the proposed dual-mode model by examining its properties in comparison with uniform sampling periods. Specifically, we compare the control performance of a plant that is controlled by a task with a uniform sampling period (SINGLE) to a dual-mode task (DUAL) with equivalent utilisation referring to equation (6). The control performance is expressed in the form of fitness for consistence (higher is better). Due to the properties of DUAL, for each utilisation, there exists a number of possible solutions for the task parameters. For the illustration purpose, we fix α to a median value, i.e., 0.5 and show all the feasible combinations of periods using a simple linear programming method. The produced results are shown in a box-plot of Figure 7.

From the figure, it can be seen that in most of the cases the control performance of dual-mode tasks (box plots) outperforms the uniform sampling period (solid line with diamond markers). DUAL and SINGLE have similar performances when the sampling period is small ($< 18ms$ in this case). For larger periods, however, DUAL starts to outperform SINGLE for most of the parameters. The overlapped fitness happens when $T_i^H = T_i^L$, and in this case the dual-mode task behaves exactly the same as a uniform sampling period task. Another observation is that the variation of all possible parameters for DUAL can be significant, which indicates the importance of searching for the right set of parameters.

7.2 Optimising control performance

This experiment considers a control system with three identical internally unstable plants. Each plant is controlled by a periodic control task (with one or more sampling periods). Other non-control functionalities in the system are also modelled as periodic tasks, and we assume five non-control task. To validate the optimisation solution method (i.e., the searching method), we compare the best results of the genetic algorithm with random search. To validate the efficiency of the solution candidates using DUAL, we also compare the results with SINGLE, i.e., the uniform sampling period without switching. In comparison, we use the best period for single-period tasks, which is found by exhaustively exploring the whole period parameters space of all tasks. Task sets are generated

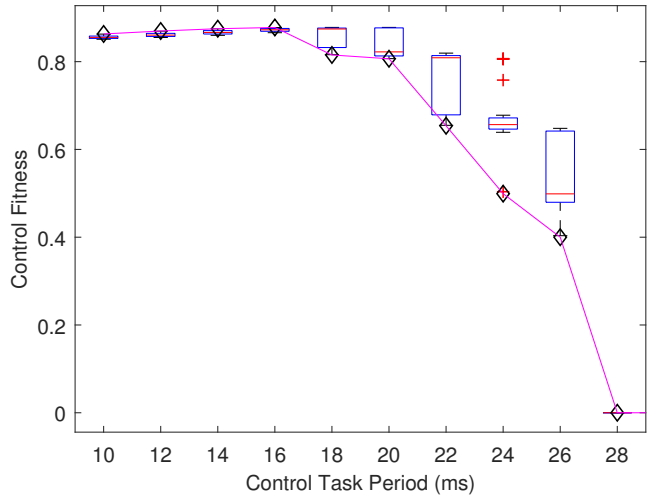

Fig. 7. Demonstrating the control performance of tasks with uniform sampling periods (solid line with diamond markers) and dual-mode tasks (box plots)

by the *UUnifast* algorithm [9]. The total utilisation for control tasks is bounded to $[0.60, 0.99]$, and the dual-mode limits are $T_i^- = 15ms$ and $T_i^+ = 25ms$ for each control task. The experiment is conducted via sweeping the total utilisation of non-control tasks, simulating an increased pressure on schedulability.

Figure 8 presents the overall control performance of this system (measured by fitness function λ_C) obtained by (i) the dual-mode strategy with the proposed optimisation framework, (ii) the dual-mode strategy with random search and (iii) the traditional uniform sampling period model. As shown in the figure, the uniform sampling period model yields the best control performance when the utilisation of non-control tasks $U_{nc} \leq 0.13$. With such utilisation settings, the pressure on schedulability is low and the benefit of a dual-mode model is not shown. However, by further increasing the utilisation U_{nc} and thus the scheduling pressure, the proposed dual-mode strategy outperforms the uniform sampling period model in most cases with obvious control performance improvement. This observation again confirms the effectiveness of the proposed dual-mode strategy. We would like to make a note that such improvement is significant especially considering that many CPS domains are cost-sensitive, such as the automotive.

As for the optimisation problem solving methods (genetic algorithm against random search), they return similar results when $U_{nc} \leq 0.16$ (i.e., low schedulability pressure). However, results obtained by the genetic algorithm demonstrate clearly better overall control performance when U_{nc} is larger than 0.16. In these cases, it becomes more difficult to schedule the system so that the performance of random search is decreased significantly due to the reduced range of feasible (i.e., schedulable) solutions. Finally, with $U_{nc} \geq 0.37$, no feasible solution can be found by any strategies due to system overload.

Based on the above observations, the proposed optimisation framework yields the best control performance in most cases. This indicates that the studied parameter searching problem has been successfully mapped to an effective genetic algorithm optimisation problem. This is also confirmed by Figure 9, which demonstrates the searching process of both the genetic algorithm and random search in each iteration with $U_{nc} = 0.19$ and $U_{nc} = 0.31$. In both cases, although random search

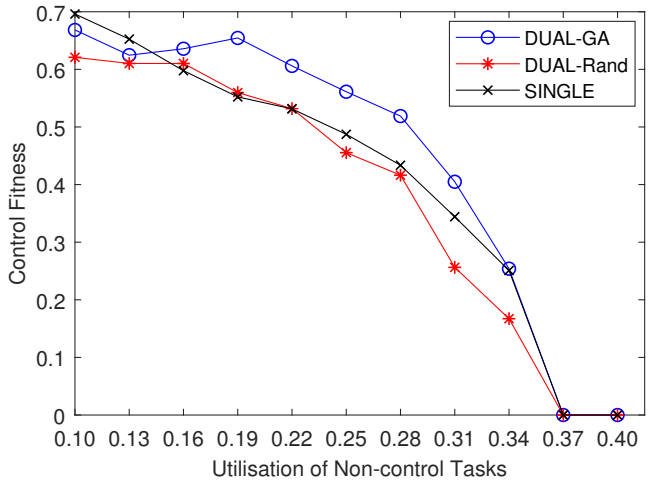

Fig. 8. Control performance comparison between the dual-mode genetic algorithm, dual-mode random search and uniform sampling period

Table 2. Utilisation fitness λ_U of single- and dual-mode tasks with varied U_{nc}

U_{nc}	**0.10**	**0.13**	**0.16**	**0.19**	**0.22**	**0.25**
Single	0.129	0.099	0.069	0.039	0	0
Dual	0.229	0.204	0.204	0.151	0.117	0
Δ	0.100	0.105	0.136	0.112	0.117	0

obtains better results in early iterations, it gets barely improved during the rest of iterations. In contrast, the genetic algorithm advances the control performance continuously towards the ideal point during the entire optimisation process. We can thus conclude that while the solution approach based on genetic algorithms is a heuristic and does not guarantee global optimality, it is effective in searching for the optimal decision variables.

7.3 Optimising resource efficiency

This experiment investigates the second scenario: optimising resource efficiency as reflected by the number of tasks that are schedulable. The utilisation is used as a direct fitness function. We consider a control system that contains a given number of control tasks and non-control tasks. Similar to the previous experiment, we initially have three control tasks and five non-control tasks, and use the genetic algorithm to search for feasible parameters with maximised resource efficiency. In the dual-mode model, the total utilisation of the control tasks is bounded to [0.6, 0.99] with $T_i^- = 15ms$ and $T_i^+ = 25ms$. The utilisation of those non-control tasks U_{nc} is swept from 0.1 to 0.25. The fitness function λ_U is adopted in this experiment with constraints on the control performance set to 0.6.

We first demonstrate improved schedulability robustness (in terms of increased free system utilisation). As given in Table 2, the dual-mode strategy outperforms the uniform sampling period with better utilisation fitness value λ_U obtained for each tested U_{nc}. In particular, with $U_{nc} = 0.22$, the dual-mode strategy with the proposed optimisation framework can still obtain feasible solutions

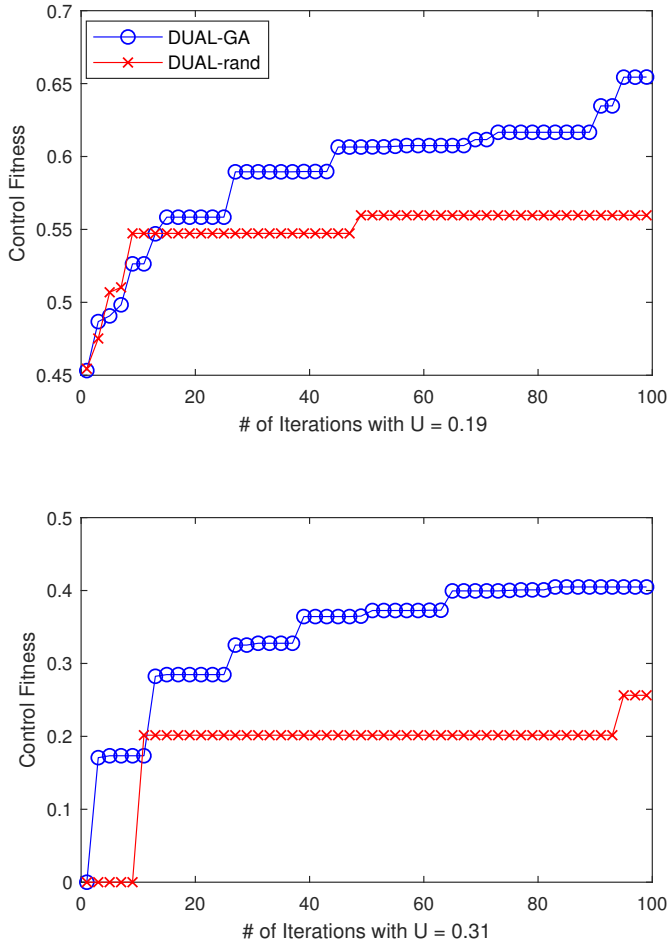

Fig. 9. Control performance optimisation progress comparison between the genetic algorithm and the random search

Table 3. Number of additional schedulable tasks of all non-control task candidates

	Single			Dual		
U_{nc}	min	mean	max	min	mean	max
0.10	6	8.4	11	9	10.9	13
0.13	6	7.3	10	8	10.2	13
0.16	5	6.3	9	8	10.2	13
0.19	3	4.6	7	7	8.9	11
0.22	-	-	-	6	7.8	10
0.25	-	-	-	-	-	-

(i.e., satisfies both control and schedulability requirements), but the uniform sampling period leads

to an unschedulable system (i.e., with fitness value 0). This observation confirms that, under the dual-mode model, the proposed optimisation framework yields better resource efficiency in general than the uniform sampling period tasks.

Besides improving schedulability robustness, better resource efficiency also allows the possibility for integrating additional functionalities by fitting more non-control tasks, without jeopardising system schedulability. Table 3 considers the application scenario where there exists a set of additional non-control tasks (i.e., additional functionalities) that could be integrated into the system. The optimisation objective is to maximise the number of such tasks that can be fitted into the system while maintaining both control performance requirements and system schedulability. In total, 10 task sets are generated randomly by the *UUifast* algorithm, where each task set contains 10 tasks with total utilisation of 1. For each task set, a simple task allocation algorithm exhaustive in nature is applied to fit as many tasks as possible to the systems obtained in the above experiment for each U_{nc}. The system schedulability and control performances are evaluated each time a new task is added into system.

As shown in Table 3, the number of additional non-control tasks that can be integrated into the system (without jeopardising schedulability) obtained by the dual-mode strategy outperforms the uniform sampling period (the single mode) for all test cases. In particular, with $U_{nc} = 0.19$, the dual-mode model strictly dominates the single mode, where the minimum number of additional schedulable tasks obtained by the dual mode equals the best performance of the single mode. With $U_{nc} = 0.22$, the single mode fails to deliver feasible solutions while the dual mode still demonstrates strong resource efficiency with 7.8 additional tasks schedulable on average. This experiment again demonstrates that the proposed dual-mode model and the optimisation framework are effective and provide better resource efficiency than the traditional uniform sampling period model.

Summarising the above, in this section, we have demonstrated the effectiveness of the proposed dual-mode strategy with the optimisation framework via a set of experiments. Compared with the traditional uniform sampling period model, the dual-mode strategy yields better overall control performance and resource efficiency in general. The optimisation framework behaves well in providing feasible solutions for those dual-mode control tasks, which satisfy both schedulability and control performance requirements. In addition, the proposed framework provides effective optimisation towards either control performance or resource efficiency depending on the application scenario.

At the end of this section, we would like to discuss the computation efforts and scalability of the solution approach based on genetic algorithms. As an example, running the solver with an initial population of 100 for 100 iterations on an Intel i7-8750H at 2.2GHz takes 1568s, which is about 25 minutes. This is sufficient for up to 10 control tasks and acceptable as an offline computation task. Practically, it is highly unlikely to have more than 10 control tasks running on a uniprocessor system. However, if indeed more control tasks are needed, the scalability becomes an issue, since each additional control task adds three more dimensions to the decision space. Adding non-control tasks makes negligible impact.

8 CONCLUDING REMARKS

In this paper, we have introduced a dual-mode strategy for resource-efficient and performance-maximisation in CPS. This strategy consists of a dual-mode task model, which switches between the fast and slow modes depending on the control stages; and an optimisation framework, which uses the genetic algorithm to solve constrained non-convex optimisation problems to find feasible task parameters.

We first gave two motivational examples to demonstrate the rational behind the dual-mode model. We showed with equivalent utilisation, our dual-mode strategy could produce better control

performances compared to a uniform sampling period model. In the second example, we turned an unschedulable task set into a schedulable one, by changing the control task to follow the dual-mode model. In Section 5, the response time analysis for a set of dual-mode tasks is given. It shows that the proposed task model is analysable, which makes it more appropriate when compared to other discussed methods, e.g., state-aware feedback scheduling, and event-based controllers.

To use this model for improving resource utilisation and maximising control performance, we formulated an optimisation problem, and proposed a complete solver based on the genetic algorithm. Finally, the effectiveness of this strategy was evaluated by a number of experiments. These experiments show that the dual-mode strategy with the optimisation framework can improve i) overall control performance; and ii) system schedulability by being able to accommodate more non-control tasks whilst guaranteeing the required control performances.

Further to the proposed dual-mode strategy, some potential extensions can be explored. One obvious extension is to adapt this model to other scheduling schemes, e.g., Earliest Deadline First (EDF). In additional to dual modes, increasing the number of modes would also be possible to achieve a better resource utilisation and control performance. As for the optimisation framework, the two independent objectives can be integrated into a multi-objective problem. The control performance of each control task can be an individual objective. The system schedulability can be a different one. Trade-offs between control and scheduling can be explored with such a multi-objective problem setting.

ACKNOWLEDGMENTS

This work has been partially funded by UK EPSRC project EP/N023641/1, Layers for Structuring Trustworthy Ambient Systems (STRATA).

REFERENCES

[1] Karl-Erik Årzén. 1999. A simple event-based PID controller. In *Proc. 14th IFAC World Congress*, Vol. 18. 423–428.
[2] Karl-Erik Årzén, Bo Bernhardsson, Johan Eker, Anton Cervin, Klas Nilsson, Patrik Persson, and Lui Sha. 1999. Integrated control and scheduling. *Department of Automatic Control, Lund Institute of Technology, Sweden, Tech. Rep. ISRN LUTFD2/TFRT–7586–SE* (1999).
[3] Karl-Erik Årzén, Anton Cervin, Johan Eker, and Lui Sha. 2000. An introduction to control and scheduling co-design. In *Proceedings of the 39th IEEE Conference on Decision and Control, 2000.*, Vol. 5. IEEE, 4865–4870.
[4] Giuseppe Ascia, Vincenzo Catania, and Maurizio Palesi. 2006. A Multi-objective Genetic Approach to Mapping Problem on Network-on-Chip. *J. UCS* 12, 4 (2006), 370–394.
[5] Karl J Åström and Björn Wittenmark. 2013. *Computer-controlled systems: theory and design.* Courier Corporation.
[6] Sanjoy Baruah. 2016. Schedulability analysis of mixed-criticality systems with multiple frequency specifications. In *2016 International Conference on Embedded Software (EMSOFT)*. IEEE, 1–10.
[7] S.K. Baruah and A. Burns. 2006. Sustainable Schedulability Analysis. In *Proc. of IEEE Real-Time Systems Symposium (RTSS)*. 159–168.
[8] Enrico Bini and Giuseppe Buttazzo. 2014. The optimal sampling pattern for linear control systems. *IEEE Trans. Automat. Control* 59, 1 (2014), 78–90.
[9] Enrico Bini and Giorgio C Buttazzo. 2005. Measuring the performance of schedulability tests. *Real-Time Systems* 30, 1-2 (2005), 129–154.
[10] A. Burns and A. J. Wellings. 2016. *Analysable Real-Time Systems Programmed in Ada.* ISBN: 9781530265503.
[11] R. Castane, P. Marti, M. Velasco, A. Cervin, and D. Henriksson. 2006. Resource management for control tasks based on the transient dynamics of closed-loop systems. In *18th Euromicro Conference on Real-Time Systems (ECRTS)*.
[12] Anton Cervin, Johan Eker, Bo Bernhardsson, and Karl-Erik Årzén. 2002. Feedback-feedforward scheduling of control tasks. *Real-Time Systems* 23, 1–2 (2002), 25–53.
[13] Anton Cervin, Manel Velasco, Pau Martí, and Antonio Camacho. 2011. Optimal online sampling period assignment: Theory and experiments. *IEEE Transactions on Control Systems Technology* 19, 4 (2011), 902–910.
[14] Wanli Chang and Samarjit Chakraborty. 2016. Resource-aware automotive control systems design: A cyber-physical systems approach. *Foundations and Trends in Electronic Design Automation* 10, 4 (2016), 249–369.
[15] Richard C Dorf and Robert H Bishop. 2011. *Modern control systems.* Pearson.

[16] Paul Emberson and Iain Bate. 2008. Extending a task allocation algorithm for graceful degradation of real-time distributed embedded systems. In *2008 Real-Time Systems Symposium*. IEEE, 270–279.
[17] Paul Emberson and Iain Bate. 2010. Stressing search with scenarios for flexible solutions to real-time task allocation problems. *IEEE Transactions on Software Engineering* 36, 5 (2010), 704–718.
[18] Luca Greco, Daniele Fontanelli, and Antonio Bicchi. 2011. Design and stability analysis for anytime control via stochastic scheduling. *IEEE Trans. Automat. Control* 56, 3 (2011), 571–585.
[19] Moncef Hamdaoui and Parameswaran Ramanathan. 1995. A dynamic priority assignment technique for streams with (m, k)-firm deadlines. *IEEE transactions on Computers* 44, 12 (1995), 1443–1451.
[20] Hai Lin and Panos Antsaklis. 2009. Stability and stabilizability of switched linear systems: A survey of recent results. *IEEE Trans. Automat. Control* 54, 2 (2009), 308–322.
[21] Paris Mesidis and Leandro Soares Indrusiak. 2011. Genetic mapping of hard real-time applications onto NoC-based MPSoCs - a first approach. In *6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*. IEEE, 1–6.
[22] Charles L Phillips and H Troy Nagle. 2007. *Digital control system analysis and design*. Prentice Hall Press.
[23] Adrian Racu and Leandro Soares Indrusiak. 2012. Using genetic algorithms to map hard real-time on NoC-based systems. In *7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*. IEEE, 1–8.
[24] P. Ramanathan. 1997. Graceful degradation in real-time control applications using (m, k)-firm guarantee. *Proceedings of IEEE 27th International Symposium on Fault Tolerant Computing* (1997), 132–141.
[25] Danbing Seto, John P Lehoczky, Lui Sha, and Kang G Shin. 1996. On task schedulability in real-time control systems. In *Real-Time Systems Symposium, 1996., 17th IEEE*. IEEE, 13–21.
[26] Takao Yokota, Mitsuo Gen, and Yin-Xiu Li. 1996. Genetic algorithm for non-linear mixed integer programming problems and its applications. *Computers & industrial engineering* 30, 4 (1996), 905–917.
[27] Shuai Zhao. 2018. *A FIFO Spin-based Resource Control Framework for Symmetric Multiprocessing*. Ph.D. Dissertation. University of York.