Exact Approach to the Scheduling of F-shaped Tasks with Two and Three Criticality Levels

Antonin Novak[1,2], Premysl Sucha[1] and Zdenek Hanzalek[1,2]
[1]*Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, Prague, Czech Republic*
[2]*Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic*

Keywords: Scheduling, Mixed-criticality, Embedded Systems, Integer Linear Programming.

Abstract: The communication is an essential part of a fault tolerant and dependable system. Safety-critical systems are often implemented as time-triggered environments, where the network nodes are synchronized by clocks and follow a static schedule to ensure determinism and easy certification. The reliability of a communication bus can be further improved when the message retransmission is permitted to deal with lost messages. However, constructing static schedules for non-preemptive messages that account for retransmissions while preserving the efficient use of resources poses a challenging problem. In this paper, we show that the problem can be modeled using so-called F-shaped tasks. We propose efficient exact algorithms solving the non-preemptive message scheduling problem with retransmissions. Furthermore, we show a new complexity result, and we present computational experiments for instances with up to 200 messages.

1 INTRODUCTION

The communication buses in modern vehicles are an essential part of advanced driver assistants. Those systems depend on the data gather by sensors, such as LIDAR, cameras, and radars. The data describing the surrounding environment are communicated through communication buses to ECUs (*Electronic Control Units*) where they are processed, and appropriate actions are taken. For example, if an obstacle is detected in front of the vehicle, the car automatically activates breaks. Not only driver assistant systems rely on the communication. Different ECUs are responsible for running the car as a whole. The fuel is injected accordingly to the current combustion and outside conditions and cars with the drive-by-wire system steer via electronic signals.

The modern vehicle is considered as a fault-tolerant and dependable system. If one part of it breaks down or does not work as expected, the human life is in risk. Since the intra-vehicular communication is the key element of the car, it is subject to a safety certification. The safety certification is a process, where the manufacturer proves that his safety-critical systems such as autonomous driving are working correctly to a high degree of assurance. If manufacturers are not able to demonstrate the correct behavior of the central communication bus, then the whole certification process is not complete. According to *SSR Automotive Warranty & Recall Report 2016*, the number of software-related recalls in 2015 accounted for 15% of all recalls, marking the importance of the certification.

Traditionally, event-triggered communication protocols are commonly used in automotive industry. In the event-triggered environment, the actions are performed *on-demand*, i.e. triggered by some events. Even though, event-triggered protocols are flexible, their usage in modern cars is limited due to certification and verification issues. The response time analysis (i.e. the analysis of the behavior of the system) in real-life event-triggered communication systems including gateways and precedence relations is a very complex problem, therefore the safety certification of systems utilizing the event-triggered environment is a difficult task (Baruah and Fohler, 2011).

An alternative to event-triggered real-time systems is the time-triggered paradigm. Messages in time-triggered communication are transferred through the network at specific times prescribed by a static pre-computed static schedule. The schedule is constructed usually with tools known from *OR/Mathematical Optimization* field (Dvorak and Hanzalek, 2016; Kopetz et al., 2005) such that it re-

Novak A., Sucha P. and Hanzalek Z.
Exact Approach to the Scheduling of F-shaped Tasks with Two and Three Criticality Levels.
DOI: 10.5220/0006198101600170
In *Proceedings of the 6th International Conference on Operations Research and Enterprise Systems (ICORES 2017)*, pages 160-170
ISBN: 978-989-758-218-9

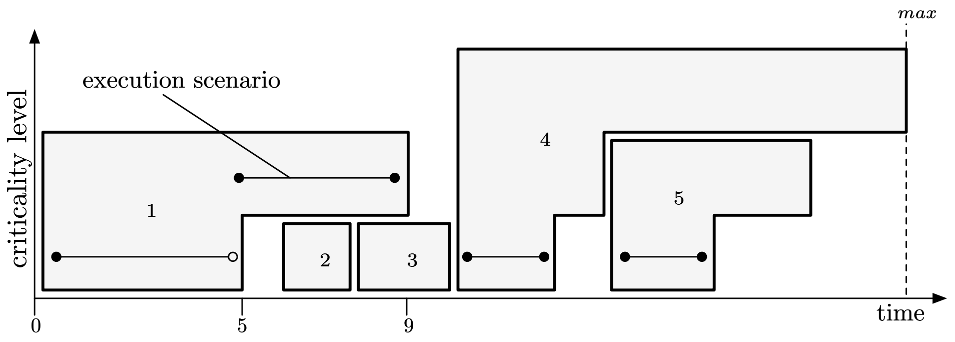


Figure 1: A feasible schedule of F-shaped tasks with a runtime execution scenario denoted by the black line.

spects the problem and safety constraints. Therefore, the certification of the system is achieved via showing feasibility of the produced communication schedule.

The determinism and verifiability of time-triggered communications led to the design of protocols that includes time-triggered communication for safety-critical systems. For example, FlexRay bus is used nowadays in the automotive industry (e.g. Porsche Panamera, Nissan Infinity Q50). One of the disadvantages of time-triggered protocols is their non-flexibility. For example, the static schedule does not take into consideration the message retransmission. The retransmission occurs when a highly critical message is not delivered e.g. due to EM noise. A possible solution how to enable message retransmissions in static time-triggered schedules is to allocate more processing time for each message to account for possible retransmissions. If no retransmission occurs during an actual execution, then the resource is idle until the start time of the next message. However, since retransmissions do not appear as frequently, the average resource utilization would be poor.

In this paper, we introduce a new solution to this problem. We build static schedules that allow retransmission of non-preemptive messages to some degree. An extra time needed for retransmissions is compensated by skipping less critical messages. The trick we use to solve the problem is to include a part of the problem solution to its formulation. Instead of scheduling *rectangles*, denoting the single exact processing time of a message, we schedule so-called *F-shapes*. This leads to an interesting combinatorial problem, where we are given a set of shapes that are aligned on the left side with the right side that is jagged (see an example in Fig. 1). The goal is to pack these shapes as tightly as possible so that they do not overlap.

The use of F-shapes is a very elegant solution that achieves efficient resource utilization. It modifies the traditional scheduling problem into the challenging one where the schedule has to assume alternatives based on the observed runtime scenario. Even though there is an exponential number of possible runtime scenarios, and for each of them, the static schedule is well-defined, we will show, that the introduced formalism makes the problem computationally tractable in practice.

1.1 Application to Automotive

Frequently, in the complex systems, functionalities with different criticality co-exist on a single bus. The adaptation of IEC 61508 safety standard (Bell, 2006), the *ASIL*, defines the application levels with a hazard assessment corresponding to three *Safety Integrity Levels*. Therefore the schedules with three criticality levels arise from the application in the automotive domain:

- messages with high criticality (criticality level 3) are used for safety-related functionalities (their failure may result in death or serious injury to people), such as steering and braking;
- messages with medium criticality (criticality level 2) are used for mission-related functionalities (their failure may prevent a goal-directed activity from being successfully completed), such as combustion engine control;
- messages with low criticality (criticality level 1) are used for infotainment functionalities, such as audio playback.

For the automotive application we assume, the criticality of a message corresponds to its maximum number of possible (re)transmissions and that messages are non-preemptive since the preemption is costly in communications. See an example of a static schedule that accounts for retransmissions in Fig. 1. The individual shapes correspond to messages scheduled on the communication bus. There T_2 and T_3 have low criticality, and no retransmissions are allowed. T_1 and T_5 correspond to messages with medium criticality; thus they can be retransmitted once. The most critical message is T_4, that can be retransmitted twice. The retransmission of the messages causes a prolongation of the processing time that is depicted in levels on the vertical axis. The top level of each message represents its WCET (*the worst case execution time*), i.e. the time that it takes to transmit the message under the most pessimistic conditions. The prolongations are compensated by skipping less critical messages. With this mechanism, the successful transmission of highly critical messages is guaranteed while in the average case runtime scenario the resource (i.e. communication bus) is efficiently utilized.

Scheduling of safety-critical non-preemptive messages on this time-triggered network can be modeled as the scheduling problem $1|mc = \mathcal{L}, mu|C_{\max}$ (Hanzalek et al., 2016). It represents the scheduling problem with one resource (a communication channel in the network) with non-preemptive mixed-criticality tasks with maximum $\mathcal{L}$ criticality levels, *mu* stands for the match-up of the execution scenario. The criterion is to minimize the maximal completion time $C_{\max}$.

A solution of the scheduling problem is given by a schedule that switches to the higher criticality level when a prolongation of a task occurs. After its successful completion, it matches-up with the original schedule. The trade-off between the safe and efficient schedules is achieved by skipping less critical messages when the prolongation of a more critical one takes place.

1.2 Paper Contribution and Outline

In this paper, we solve the scheduling problem of message retransmission in time-triggered environments. The objective is to find a non-preemptive static schedule that accounts for unforeseen message retransmissions while minimizing the length occupied by time-triggered communication. The uncertainty about the processing time is modeled using an abstraction based on F-shaped tasks. We show the relation between F-shaped tasks and the underlying probability distribution functions. Furthermore, we show a new complexity result that establishes the membership of the considered problem into $\mathcal{APX}$ complexity class, and we provide an approximation algorithm. We study the characterization of the set of optimal solutions for the problem with two criticality levels. Finally, we propose efficient exact algorithms for problems with two and three criticality levels, which solve instances with up to 200 tasks, beating the best-known method by a large margin.

The rest of the paper is organized as follows. In Sec. 2 we survey the related work. In Sec. 3 we show the relation between F-shaped tasks and discretization of cumulative probability distribution functions. In Sec. 5 we prove approximability of the problem. In Sec. 6 and 7 we show properties of the problem with two and three criticality levels and we propose efficient exact algorithms. Finally, in Sec. 8 we present computational results on the sythetic data as well as on the data inspired by a real-life embedded system of our industrial partner.

2 RELATED WORK

The exhaustive survey on mixed-criticality in real-time systems is presented by (Burns and Davis, 2013). This research is traditionally concentrated around event-triggered approach to scheduling. In the seminal paper (Vestal, 2007) proposed a method that assumes different WCETs (*the worst case execution time*) obtained for discrete levels of assurance. Apart from this proposition, the paper presents modified preemptive fixed priority schedulability analysis algorithms. However, the preemptive model is not suitable for communication protocols, and it significantly changes the scheduling problem. (Baruah et al., 2010) formulated the basic model of mixed-criticality systems. They study MC schedulability problem with two criticality levels under special restrictive cases in the event-triggered environment. (Theis et al., 2013) argued that mixed-criticality shall be pursued in time-triggered systems. (Baruah and Fohler, 2011)'s approach in the time-triggered environment assumed preemptive tasks with up to two criticality levels. It makes it unsuitable for communication protocols since the preemption would be costly. (Hanzalek et al., 2016) proposed the problem of non-preemptive mixed-criticality match-up scheduling motivated by scheduling messages on a highly used communication channel. They showed how a schedule with F-shaped tasks can be used to deal with a task disruption by skipping less critical tasks. They provide the relative order MILP model for $1|r_j, \tilde{d}_j, mc = \mathcal{L}, mu|C_{\max}$ scheduling problem, but it can deal with instances with only about 20 messages.

The concept of match-up scheduling was introduced by (Bean et al., 1991). In a case of a disruption, the goal is to construct a new schedule that matches the original one at some point in the future. This concept is mostly studied in the context of manufacturing problems (Qi et al., 2006).

Taking broader perspective, the problem can be viewed as a case of robust and stochastic optimization due to uncertainty about transmission times while satisfying safety requirements. (Bertsimas et al.,

2011) surveys robust versions of various optimization problems, but rather continuous than discrete ones. The field of stochastic optimization is reviewed by (Sahinidis, 2004). They state that integer variables introduced to stochastic programming complicate its solution, yielding suboptimal results even for small-sized problems.

As in our problem, some of the less critical messages are allowed to be skipped, the problem is related to the scheduling with a job rejection. (Shabtay et al., 2013) reviews offline scheduling with a job rejection. These approaches consider two criteria, a measure associated with schedule quality and the cost incurred by rejected jobs. The solution to this problem is a set of accepted jobs and a set of rejected jobs. However, rejected jobs cannot be executed in any execution scenario; thus this model is not suitable for communication protocols mentioned in our motivation.

To the best of our knowledge, the problem of offline non-preemptive mixed-criticality match-up scheduling was addressed by (Hanzalek et al., 2016) only, but it lacks an efficient solution method which is suggested in this paper.

3 NON-PREEMPTIVE MIXED-CRITICALITY SCHEDULING

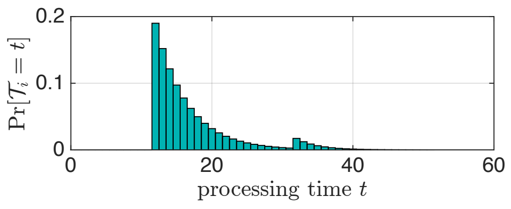
We assume that a set of communication messages is given to be scheduled on a single communication bus segment. For each message T_i, the criticality $\mathcal{X}_i \in \mathbb{N}$ is specified. It denotes the number of allowed transmissions. Each message (task) is specified by its *criticality levels*. For each *criticality level* $\ell \in \{1,\dots,\mathcal{X}_i\}$, we define the associated processing time with this level. See an example in Fig. 1. Here, T_1 has criticality $\mathcal{X}_1 = 2$; therefore it can be retransmitted once. The processing time at the first level is given by its BCET (*the best case execution time*) while the processing time at the second level is its WCET (*the worst case execution time*). During the run time execution, exactly one processing time of the message is realized; however, it is not known in advance which it will be.

We can view processing time prolongations as a retransmission of the whole message content. However, this mixed-criticality scheduling model is useful also for scheduling of computational tasks, where the exact computational time is not known in advance, but only a probability distribution is known. Let us consider the processing time of task T_i to be a random variable $\mathcal{T}_i$. Let us assume an arbitrary probability distribution over a discrete set of processing times from $\mathbb{N}$ for a particular task stating $\Pr[\mathcal{T}_i = t]$. The same information given by the probability distribution is captured by the CDF (*cumulative distribution function*) $\mathcal{F}_i$ giving the probability that processing time $\mathcal{T}_i$ is at most t. See Figs. 2a and 2b for such an example. Corresponding processing times $p_i^{(\ell)}$ for each criticality level ℓ are taken from $\mathcal{F}_i$ as $p_i^{(\ell)} = \mathcal{F}_i^{-1}(c_\ell)$, where $\mathcal{F}_i^{-1}$ is the quantile function. The criticality of a task is a user-defined parameter. For example, if we identify criticality levels with a safety standard IEC 61508 SIL (*Safety Integrity Levels*) (Bell, 2006), then the task criticality $\mathcal{X}_i$ is given by the SIL of the functionality carried out by the content and c_ℓ is defined as $1-$*probability of failure* defined by SIL ℓ.

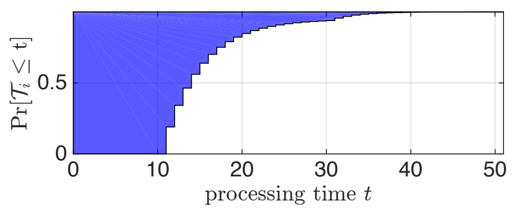
Processing times obtained according to criticality levels then form a single task like the one depicted in Fig. 2d. Since CDFs are non-decreasing functions, a set of processing times $p_i^{(\ell)}$ yields shapes like the F letter rather than ordinary rectangles, hence the name F-shape. See an example in Fig 2. There we see discretization for a task with criticality three at corresponding levels 1, 2 and 3 with the vertical axis on the logarithmic scale.

The solution to the scheduling problem is a feasible static schedule of the given set of F-shaped tasks. Consider a particular example of the schedule with tasks having up to three levels of criticality that is shown in Fig. 1. A feasible schedule with F-shaped tasks describes alternative schedules for any realization of the processing time of messages. Observed prolongations of more critical messages are compensated by skipping execution of less critical messages.

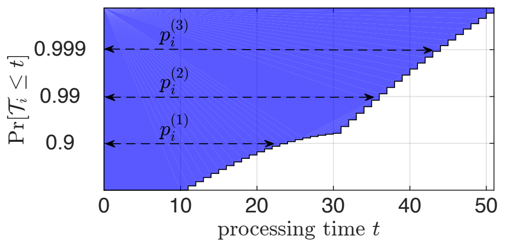
The black line denotes a scenario, where T_1 was disrupted once. The actual processing time of T_1 was 9 instead of 5 due to a disturbance. When the disruption occurred, the execution switched to the next higher criticality level. There, by the assumption, the execution was successful with a probability given by the $\ell = 2$ criticality level. After upon T_1 finished, the execution matched-up back with the lowest criticality level. In general, if a task T_i is prolonged to level ℓ, then all tasks T_j for which $s_i + p_i^{(1)} \le s_j < s_i + p_i^{(\ell)}$ are not executed. Therefore, in this execution scenario, after T_1 finished, T_4 was up next. Moreover, if we unify the F-shape from Fig. 2d with task T_4 in Fig. 1, then we can say that T_5 will be executed with very high probability of 0.99, but in rare cases, it won't be executed since T_4 is more critical and needs more time to complete. However, here the choice of 0.99 is just for the illustrative purposes only, since the levels of the probability are inputs to the problem and they can be set to any feasible value.



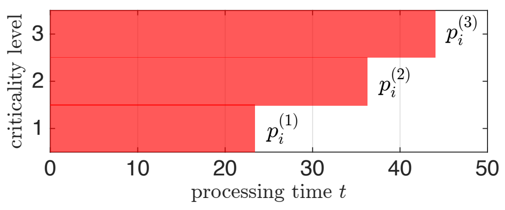
(a) A discrete probability distribution over a set of processing times.



(b) Corresponding cumulative distribution function.



(c) Discretization of a cumulative distribution function



(d) Resulting F-shaped task with $\mathcal{X}_i = 3$

Figure 2: Discretized cumulative distribution function forms an F-shaped task.

4 PROBLEM STATEMENT

We assume a set of non-preemptive F-shaped tasks $I_{\mathcal{MC}} = \{T_1, \ldots, T_n\}$ to be processed on a single machine. We define an F-shaped task and its criticality as follows:

Definition 1 (F-shape). *The F-shape T_i is a pair $(\mathcal{X}_i, \mathbf{P}_i)$ where $\mathcal{X}_i \in \{1, \ldots, \mathcal{L}\}$, $\mathcal{L} \in \mathbb{N}$ is the task criticality and $\mathbf{P}_i \in \mathbb{N}^{\mathcal{X}_i}$, $\mathbf{P}_i = (p_i^{(1)}, p_i^{(2)}, \ldots, p_i^{(\mathcal{X}_i)})$ is the vector of processing times such that*

$$p_i^{(1)} < p_i^{(2)} < \ldots < p_i^{(\mathcal{X}_i)}.$$

The F-shape is an abstraction for non-preemptive tasks with multiple different processing times. See for example T_4 in Fig. 1. It is F-shaped task with criticality $\mathcal{X}_4 = 3$; therefore it has 3 different processing times. Having a set $I_{\mathcal{MC}}$ of F-shaped tasks, we define the feasible schedule as follows:

Definition 2 (Feasible Schedule). *By the schedule for a set of F-shaped tasks $I_{\mathcal{MC}} = \{T_1, T_2, \ldots, T_n\}$ we refer to an assignment $(s_1, s_2, \ldots, s_n) \in \mathbb{N}^n$. We say that schedule $(s_1, s_2, \ldots, s_n)$ for $I_{\mathcal{MC}}$ is feasible if and only if $\forall i, j \in \{1, \ldots, n\}, i \neq j$:*

$$(s_i + p_i^{(\min\{\mathcal{X}_i, \mathcal{X}_j\})} \le s_j) \vee (s_j + p_j^{(\min\{\mathcal{X}_i, \mathcal{X}_j\})} \le s_i).$$

Feasibility of a schedule with F-shaped tasks requires that tasks do not overlap on any criticality level. For example in Fig. 1, since T_5 follows after T_4, it cannot start earlier than $s_4 + p_4^{(2)}$, since $\min\{\mathcal{X}_4, \mathcal{X}_5\} = 2$ is the highest common criticality level of T_4 and T_5.

We deal with the problem of finding a feasible schedule for a set of F-shaped tasks with criticality at most $\mathcal{L}$ such that the makespan (i.e. $\max s_i + p_i^{(\mathcal{X}_i)}$) is minimized. In the three-field Graham-Blazewicz notation it is denoted as $1|mc = \mathcal{L}, mu|C_{\max}$, where $mc = \mathcal{L}$ stands for the mixed-criticality aspect of tasks of maximal criticality $\mathcal{L}$ and mu stands for the match-up. This problem is known to be $\mathcal{NP}$-hard in the strong sense even for $mc = 2$ (two criticality levels) as shown by reduction from 3-Partition Problem in (Hanzalek et al., 2016).

5 GENERAL PROPERTIES

Since the problem $1|mc = 2, mu|C_{\max}$ is strongly $\mathcal{NP}$-hard, it does not admit FPTAS unless $\mathcal{P} = \mathcal{NP}$. However, we show that the problem is polynomial-time approximable within a constant multiplicative factor.

Proposition 1 (Approximability). *For any given fixed $\mathcal{L}$, the problem $1|mc = \mathcal{L}, mu|C_{max}$ is contained in $\mathcal{APX}$ complexity class.*

Proof. Suppose the algorithm *LCF (Least Criticality First)* that takes an input instance $I_{\mathcal{MC}}$ and schedules tasks in a non-decreasing sequence by their criticalities without waiting. Then the makespan of resulting schedule is

$$LCF(I_{\mathcal{MC}}) = \sum_{i|\mathcal{X}_i=1} p_i^{(1)} + \sum_{i|\mathcal{X}_i=2} p_i^{(2)} + \ldots + \sum_{i|\mathcal{X}_i=\mathcal{L}} p_i^{(\mathcal{L})}$$

A sum of processing times on a given criticality level over a set of tasks is a lower bound on the makespan.

Therefore we have

$$\max\{\sum_{i|X_i\geq 1} p_i^{(1)}, \sum_{i|X_i\geq 2} p_i^{(2)}, \dots, \sum_{i|X_i\geq \mathcal{L}} p_i^{(\mathcal{L})}\} \leq$$
$$\leq OPT(I_{\mathcal{MC}}) \leq LCF(I_{\mathcal{MC}}) \leq \mathcal{L}\cdot OPT(I_{\mathcal{MC}}).$$

where $OPT(I_{\mathcal{MC}})$ denotes the optimal makespan of $I_{\mathcal{MC}}$ problem instance. □

In fact, this result shows more than that there exists a polynomial-time algorithm producing schedules with a constant bounded quality. For example, for the problem with $\mathcal{L} = 2$ criticality levels, actually any left-shifted schedule will be at most twice as worse as the optimal makespan since *LCF* actually produces the worst ordering of tasks in terms of the makespan.

In the following sections, we present exact algorithms for the problem with 2 and 3 criticality levels. Due to the $C_{\max}$ criterion, it can be shown that the search for an optimal solution can be reduced to finding a permutation of tasks. Therefore, any optimal schedule is given by a permutation of tasks π. Hence we denote the makespan of the left-shifted schedule of permutation π by $C_{\max}(\pi)$. In Sec. 6 we give a characterization of the set of optimal permutations for problem $1|mc = 2, mu|C_{\max}$ and we introduce a MILP model utilizing it. In Sec. 7, we introduce an operator acting on F-shapes, and we show how the optimal solutions for problems with two and three criticality levels are related.

6 TWO CRITICALITY LEVELS

We showed that optimal solutions to $1|mc = \mathcal{L}, mu|C_{\max}$ are given by a permutation π of tasks. For the problem with two criticality levels, the optimal permutations can be characterized more precisely. Let us refer to tasks with criticality $X_i = 2$ as HI-tasks and tasks with criticality $X_j = 1$ as LO-tasks. The key structure of the optimal permutations are *covering blocks*:

Definition 3 (Covering Block)**.** *For any given feasible schedule $(s_1,\dots,s_n)$, a* HI*-task T_i and a* LO*-task T_j we say that T_j is covered by T_i, denoted as $T_i \in cov(T_j)$, if and only if $s_i + p_i^{(1)} \leq s_j < s_i + p_i^{(2)}$. The covering block B_i is then the* HI*-task T_i and the set of all* LO*-tasks covered by T_i.*

See an example in Fig. 1. There T_1 is covering T_2 and T_3. All these tasks form a covering block. Although the definition of covering block given above is meant for the problem with two criticality levels, the notion of *covering* can be generalized for more criticality levels. We assign a *length* to each covering block. The length is given as the maximum between the processing time $p_i^{(2)}$ of the HI-task T_i and the sum of processing times of tasks covered by T_i plus the processing time of T_i at the first level $p_i^{(1)}$.

Proposition 2 (Covering Block Length)**.** *Given the covering block B_i, its length defined as*

$$\max\{p_i^{(1)} + \sum_{T_j\,|\,T_i\in cov(T_j)} p_j^{(1)}, \quad p_i^{(2)}\}$$

is invariant with respect to the ordering of LO*-tasks T_j for which $T_i \in cov(T_j)$.*

Clearly, the ordering of LO-tasks T_j for which $T_i \in \mathrm{cov}(T_j)$ does not affect the block length since all LO-tasks are running without waiting. Furthermore, we say that task T_j is *fully covered* by the block B_i, if $B_i = p_i^{(2)}$ and $T_i \in \mathrm{cov}(T_j)$. If exists a task covered by the block B_i that is not fully covered, then we say that B_i is *saturated*. The makespan $C_{\max}$ of the schedule is given by a permutation of covering blocks. However, actually any permutation of covering blocks contributes to the makespan by the same amount; hence it is not subject to optimization.

Proposition 3 (Interchangebility)**.** *For every instance of the problem $1|mc = 2, mu|C_{max}$ there exists an optimal solution that is given by an arbitrary permutation of covering blocks.*

A characterization of optimal solutions for $1|mc = 2, mu|C_{\max}$ directly follows from Proposition 2 and 3:

Corollary 1. *The optimal solution for $1|mc = 2, mu|C_{max}$ is given by an assignment of* LO*-tasks to* HI*-tasks.*

6.1 Covering MILP Model for $1|mc = 2, mu|C_{\max}$

The following MILP model relies on Corollary 1. The model assigns LO-tasks to the HI-tasks in order to form covering blocks such that the sum of their lengths is the minimum. The decision variable x_{ij} indicates whether the LO-task T_j is covered by the HI-task T_i; therefore if $T_i \in \mathrm{cov}(T_j)$, then $x_{ij} = 1$. The makespan is then given by the sum of lengths of covering blocks and the sum of processing times of all LO-tasks that are not covered.

$$\min \sum_{i|X_i=2} B_i + \sum_{j|X_j=1} p_j^{(1)}(1 - \sum_{i|X_i=2} x_{ij}) \quad (6.1)$$

s.t.

$$B_i \geq p_i^{(1)} + \sum_{j|X_j=1} p_j^{(1)} x_{ij} \forall i \in I_{\mathcal{MC}}|_{X_i=2} \quad (6.2)$$

$$B_i \geq p_i^{(2)} \quad \forall i \in I_{\mathcal{MC}}|_{X_i=2} \quad (6.3)$$

$$\sum_{i|\mathcal{X}_i=2} x_{ij} \leq 1 \quad \forall j \in I_{\mathcal{MC}}|_{\mathcal{X}_j=1} \tag{6.4}$$

where

$$B_i \in \mathbb{Z}_0^+ \quad \forall i \in I_{\mathcal{MC}}|_{\mathcal{X}_i=2}$$
$$x_{ij} \in \{0,1\} \quad \forall i \in I_{\mathcal{MC}}|_{\mathcal{X}_i=2}, \forall j \in I_{\mathcal{MC}}|_{\mathcal{X}_j=1}$$

The main advantage of this model over the model proposed by (Hanzalek et al., 2016) is that it has much stronger linear relaxation. Further in Sec. 8 we demonstrate its ability to solve an order of magnitude larger instances.

7 THREE CRITICALITY LEVELS

Although two criticality levels are often sufficient for safety-critical application and this case is frequently studied in the field of mixed-critical systems (Burns and Davis, 2013), sometimes the application naturally contains three or more criticality levels. We capture the direct relation between problems with different maximum criticality levels by introducing a transformation given bellow. It is based on the observation that omitting some criticality levels provides an instance of the problem with less criticality level while maintaining a lower bound property. Furthermore, we introduce the *Bottom-up* algorithm that uses this observation. The algorithm is used then together with Covering MILP model for three criticality levels ($\mathcal{L} = 3$) shown in Sec. 7.2 to form an efficient solution method.

The transformation is defined as $h^{\pm}$ restrictions:

Definition 4 ($h^{\pm}$ restrictions)**.** *Given the mixed-criticality instance $I_{\mathcal{MC}}$ and a positive integer $h \in \mathbb{N}$, let $I_{\mathcal{MC}}^{h^-}$ and $I_{\mathcal{MC}}^{h^+}$ be sets defined as*

$$I_{\mathcal{MC}}^{h^-} = \{(\min\{h,\mathcal{X}_i\}, (p_i^{(1)}, \dots, p_i^{(\min\{h,\mathcal{X}_i\})})) \mid \forall i \in I_{\mathcal{MC}}\}$$
$$I_{\mathcal{MC}}^{h^+} = \{(\mathcal{X}_i - h + 1, (p_i^{(h)}, \dots, p_i^{(\mathcal{X}_i)})) \mid \forall i \in I_{\mathcal{MC}} : \mathcal{X}_i \geq h\}$$

We refer to $I_{\mathcal{MC}}^{h^-}$ ($I_{\mathcal{MC}}^{h^+}$) as h^- (h^+) restriction of the instance $I_{\mathcal{MC}}$.

The h^- restriction takes an F-shape and cuts off all criticality levels above level h. Similarly, given the set of F-shaped tasks, h^+ restriction drops all tasks with criticality below h, and for the rest, it cuts off criticality levels less than h. Restricting an $I_{\mathcal{MC}}$ instance yields to a mixed-criticality instance since omitting some of the criticality levels for an F-shape gives us an F-shape. The application of the restriction can be viewed as a relaxation the problem.

Proposition 4 (Two Lower Bounds on the Makespan)**.** *For the problem $1|mc = 3, mu|C_{max}$ expressions lb^-, lb^+ defined as*

$$lb^{\pm} = \min_{\pi \in \Pi(I_{\mathcal{MC}}^{2^{\pm}})} C_{max}(\pi)$$

are lower bounds on the makespan, where $\Pi(I_{\mathcal{MC}}^{2^+})$ and $\Pi(I_{\mathcal{MC}}^{2^-})$ denote the set of all permutations of elements $I_{\mathcal{MC}}^{2^+}$, $I_{\mathcal{MC}}^{2^-}$ respectively.

Proof. The lb^- is a lower bound on the makespan of $1|mc = 3, mu|C_{\max}$ since it relaxes on the overlapping condition at the third criticality level. Similarly, lb^+ is a lower bound on the makespan since it relaxes on the overlapping condition at the first criticality level. $\square$

7.1 Bottom-up Algorithm

We introduce a heuristic algorithm for the problem $1|mc = 3, mu|C_{\max}$. Let us refer to tasks with $\mathcal{X}_i = 3$ (i.e. criticality 3) as to GREAT-tasks. The *Bottom-up* algorithm is based on the idea of constructing the schedule in two stages. In the first stage, the relaxed problem is solved up to the optimality, which minimizes a lower bound on the optimal makespan of the original problem. The second stage takes the relaxed solution and constructs a locally optimal solution for the original problem.

The first stage of the algorithm solves 2^- restriction of the given problem instance; hence it is an instance of $1|mc = 2, mu|C_{\max}$ problem that can be solved with the model described in Section 6. It assigns LO-tasks to HI-tasks and GREAT-tasks; therefore it forms covering blocks. In the second stage, the algorithm defines a new problem instance $I'_{\mathcal{MC}}$ of the problem $1|mc = 2, mu|C_{\max}$. The instance is constructed as follows. It contains LO-tasks with processing time equal to the length of covering blocks from the stage one. LO-tasks that are not part of any covering block are assigned to an arbitrary covering block. The assignment of LO-tasks to 2^- restricted GREAT-tasks from the first stage defines HI-tasks in the new instance $I'_{\mathcal{MC}}$. Then, the $I'_{\mathcal{MC}}$ instance is solved once again as an instance of $1|mc = 2, mu|C_{\max}$ problem. See the complete description of the *Bottom-up* algorithm in Alg. 1.

In general, the *Bottom-up* algorithm produces suboptimal solutions even though they are provably bounded by a factor of 3 from the optimal solution, as stated by Proposition 1. However, there are cases when we can verify if the produced schedule is optimal. This is achieved by the concept of *critical paths* that captures the cause of achieved makespan.

Algorithm 1: Bottom-up.

```
 1: $\pi \leftarrow$ solve $I^{2^-}_{\mathcal{MC}}$ restriction by Covering MILP 6.1
 2: $I'_{\mathcal{MC}} \leftarrow \emptyset$
 3: for each covering block $B_i$ in the left-shifted solution $\pi$ do
 4:     if $X_i = 2$ in $I_{\mathcal{MC}}$ then
 5:         $\mathbf{P}_i \leftarrow (B_i)$
 6:         $I'_{\mathcal{MC}} \leftarrow I'_{\mathcal{MC}} \cup \{(1, \mathbf{P}_i)\}$
 7:     else if $B_i < p_i^{(3)}$ then
 8:         $\mathbf{P}_i \leftarrow (B_i, p_i^{(3)})$
 9:         $I'_{\mathcal{MC}} \leftarrow I'_{\mathcal{MC}} \cup \{(2, \mathbf{P}_i)\}$
10:     else
11:         ▷ block $B_i$ is saturated, it contributes by a constant term to the makespan of $I'_{\mathcal{MC}}$
12:     end if
13: end for
14: $\pi \leftarrow$ solve $I'_{\mathcal{MC}}$ by Covering MILP 6.1
```

Definition 5 (Critical Path). *Given the left-shifted schedule $(s_1, \ldots, s_n)$ of the permutation π, the critical path is $\mathcal{CP} \subseteq \{1, \ldots, |\tilde{\pi}|\} \times \{1, \ldots, \mathcal{L}\}$ for some $\tilde{\pi} \subseteq \pi$ such that $\forall (i, \ell) \in \mathcal{CP}, i < |\tilde{\pi}| : s_{\tilde{\pi}(i)} + p^{(\ell)}_{\tilde{\pi}(i)} = s_{\tilde{\pi}(i+1)}$ where $\sum_{(i,\ell)\in\mathcal{CP}} p^{(\ell)}_{\tilde{\pi}(i)} = C_{max}(\pi) = C_{max}(\tilde{\pi})$.*

Essentially, for any given left-shifted schedule, the critical path is a subset of tasks and their criticality levels such that $\forall (i, \ell) \in \mathcal{CP}$ holds that if the processing time $p^{(\ell)}_{\tilde{\pi}(i)}$ is increased by some $\varepsilon > 0$, then the makespan of the same schedule is also increased by ε.

Proposition 5 (Sufficient Optimality Conditions). *If one of following conditions holds, then the schedule produced by the Bottom-up algorithm is optimal for problem $1|mc = 3, mu|C_{max}$.*

1. *There exists a critical path going through the first and the second levels only.*
2. *Every* Lo*-task is fully covered by the second criticality level.*

When none of the optimality conditions is satisfied, e.g. a critical path is coming through every criticality level, we get back to the MILP model 7.2 for the problem $1|mc = 3, mu|C_{\max}$ in order to find an optimal solution or for the proof that the current solution is the optimal one. The solver is supplied with the initial solution and a lower bound obtained by the *Bottom-up* algorithm. The computational time of *Bottom-up* algorithm is dominated by lines 1 and 14. The total computational times are reported in Tab. 2.

7.2 Covering MILP Model for $1|mc = 3, mu|C_{\max}$

The Covering MILP model for three criticality levels uses a similar idea as the model for $1|mc = 2, mu|C_{\max}$. It assigns Lo-tasks to covering blocks and covering blocks to the Great-tasks. The model utilizes the idea that optimal solutions are made of blocks (in this case formed by Great-tasks that cover less critical tasks) whose order is interchangeable within a solution. It assigns Lo-tasks to the Hi-tasks and to 2^- restriction of Great-tasks to form covering blocks. Blocks are assigned to the Great-tasks in order to create a solution. The big M constant is as large as the number of Lo-tasks contained in the problem instance.

$$\min \sum_{i|X_i=3} p_i + \sum_{j|X_j=2} P_{j,\emptyset} + \sum_{k|X_k=1} p_k^{(1)} x_{\emptyset,\emptyset,k} \tag{7.1}$$

s.t.

$$p_i \geq p_i^{(3)} \quad \forall i \in I_{\mathcal{MC}}|_{X_i=3} \tag{7.2}$$

$$M y_{i,j} \geq \sum_{k|X_k=1} x_{i,j,k}$$
$$\forall i \in I_{\mathcal{MC}}|_{X_i=3} \cup \emptyset, \forall j \in I_{\mathcal{MC}}|_{X_j=2} \tag{7.3}$$

$$P_{j,i} \geq p_j^{(2)} y_{i,j}$$
$$\forall i \in I_{\mathcal{MC}}|_{X_i=3} \cup \emptyset, \forall j \in I_{\mathcal{MC}}|_{X_j=2} \tag{7.4}$$

$$P_{j,i} \geq p_j^{(1)} y_{i,j} + \sum_{k|X_k=1} p_k^{(1)} x_{i,j,k}$$
$$\forall i \in I_{\mathcal{MC}}|_{X_i=3} \cup \emptyset, \forall j \in I_{\mathcal{MC}}|_{X_j=2} \tag{7.5}$$

$$p_i \geq p_i^{(2)} + \sum_{j|X_j=2} P_{j,i} \quad \forall i \in I_{\mathcal{MC}}|_{X_i=3} \tag{7.6}$$

$$p_i \geq p_i^{(1)} + \sum_{j|X_j=2} P_{j,i} + \sum_{k|X_k=1} p_k^{(1)} x_{i,\emptyset,k}$$
$$\forall i \in I_{\mathcal{MC}}|_{X_i=3} \tag{7.7}$$

$$\sum_{i|X_i=3\cup\emptyset} \sum_{j|X_j=2\cup\emptyset} x_{i,j,k} \geq 1$$
$$\forall k \in I_{\mathcal{MC}}|_{X_k=1} \tag{7.8}$$

$$\sum_{i|X_i=3\cup\emptyset} y_{i,j} \geq 1 \quad \forall j \in I_{\mathcal{MC}}|_{X_j=2} \tag{7.9}$$

where

$$y_{i,j} \in \{0,1\} \forall i \in I_{\mathcal{MC}}|_{X_i=3} \cup \emptyset, \forall j \in I_{\mathcal{MC}}|_{X_j=2}$$
$$x_{i,j,k} \in \{0,1\}$$
$$\forall i \in I_{\mathcal{MC}}|_{X_i=3} \cup \emptyset, \forall j \in I_{\mathcal{MC}}|_{X_j=2} \cup \emptyset,$$
$$\forall k \in I_{\mathcal{MC}}|_{X_k=1} \cup \emptyset : k \neq \emptyset \vee (i = \emptyset \wedge j = \emptyset)$$
$$p_i \in \mathbb{Z}_0^+ \quad \forall i \in I_{\mathcal{MC}}|_{X_i=3}$$
$$P_{j,i} \in \mathbb{Z}_0^+ \quad \forall i \in I_{\mathcal{MC}}|_{X_i=3} \cup \emptyset, \forall j \in I_{\mathcal{MC}}|_{X_j=2}$$

When *Bottom-up* fails to prove optimality, it goes back to this model while supplying the lb^- lower bound and the initial solution. The reason for executing *Bottom-up* ahead solving MILP model 7.2 is two-fold. First, we have observed the solver struggles to prove optimality when the solution is clearly optimal regarding the critical path. The other observation is that if the problem instance contains the majority of tasks with criticality one and two, then solving its 2^- restriction frequently yields optimal solution since the highest criticality levels are not likely to be utilized. The same holds for the instances with a large number of tasks with higher criticality. Furthermore, solving $2^\pm$ restrictions of $I_{\mathcal{MC}}$ is cheap compared to the solving the whole MILP model 7.2 as it can be seen in Tab. 1.

8 COMPUTATIONAL EXPERIMENTS

For the problem $1|mc=2,mu|C_{\max}$ we have randomly generated sets of 20 instances with n tasks for each $n \in \{10,\dots,200\}$. Criticalities of tasks were distributed uniformly. The processing time of a task at level 1 is sampled from the uniform distribution $\mathcal{U}(1,11)$. For tasks with the criticality of 2, the prolongation at level 2 is sampled from uniform distribution $\mathcal{U}(1,10)$.

For the problem $1|mc=3,mu|C_{\max}$ we have randomly generated sets of 20 instances with n tasks for each $n \in \{10,\dots,80\}$. For each n, the set contains instances with different splits of tasks' criticalities and different distributions for prolongation (e.g. $\mathcal{U}(1,10)$ and $\mathcal{U}(1,7)$ for the second level, $\mathcal{U}(1,10)$ and $\mathcal{U}(1,14)$ for the third level, etc.) in order to generate instances of various properties. We have investigated the impact of different processing time distributions to the overall performance. We have observed that the the proposed approach is not sensitive to the choice of particular distributions, but rather to the difference between combined processing times allocated to each criticality level. Therefore, in our experiments, we have used uniform distributions with parameters that represent challenging instances.

The column *avg t* (*max t*) in Tab. 1 and 2 denotes the average (maximal) computational time for instances that were solved within the time limit of 300 s. The column *unsl* contains the percentage of instances that were not solved within the time limit and *avg gap* denotes average optimality gap proven by the solver for the unsolved instances. Results were obtained with two Intel Xeon E5-2620 v2 @ 2.10 GHz processors using Gurobi Optimizer 6.5 with the algorithms implemented in Python 3.4.

In Tab. 1 it can be seen that our model is able to solve about an order of the magnitude larger problem instances. The *Relative Order* model proposed by (Hanzalek et al., 2016) consistently fails to narrow optimality gap for instances with more than 40 tasks. In Tab. 2 it is shown that the combination of *Bottom-up* heuristic and MILP 7.2 is able to solve reliably instances with 60 tasks up to the optimality and almost all instances with 80 tasks. Moreover, the proven gap is much smaller than for the *Relative Order* model; therefore it shows that our model has stronger linear relaxation.

To put our algorithms into the another test, we tested them on data obtained from our automotive industrial partner. The data comes from a real-life automotive system consisting of a communication messages between 23 ECUs. From this instance, we constructed a probabilistic model that corresponds to the given instance. We are interested in scheduling messages inside the basic period (10 ms); therefore those are messages occurring in every communication cycle. The aim is to minimize $C_{\max}$ to maximize remaining space for other messages with larger periods.

The instance divides messages into three categories. The lowest critical are debug and development messages which do not use any form of a checksum. More critical messages are secured by a parity check. The most critical messages are secured by CRC8 code. The message criticalities are distributed according $\Pr[X_i = 1] = 0.48$, $\Pr[X_i = 2] = 0.48$, $\Pr[X_i = 3] = 0.04$. The length of each message is drawn from distribution $\mathcal{U}(8,12)$. The prolongation on the second and the third criticality level is sampled from $\mathcal{U}(8,16)$ to model the message retransmission and an extra overhead.

The real-life industrial dataset was created by generating 20 instances of the problem $1|mc=3,mu|C_{\max}$ according to distributions mentioned above for each of $n \in \{50,100,150,200\}$, where n is the number of messages. The results are reported in Tab. 3. The computational times were obtained under the same circumstances as described above.

The results for real-life industrial data are quantitatively better compared to those obtained in Tab. 2. The reason is likely that the data contain relatively a few GREAT-tasks and the range of lengths of LO-tasks is relatively narrow. Therefore, many of them are identical, and the solver might be able to exploit this symmetry even though it was not supplied to it. With the Covering MILP model, the solver scales well even for larger instances.

Table 1: Computational results for the problem $1|mc = 2, mu|C_{\max}$.

	Covering MILP 6.1				Relative Order MILP (Hanzalek et al., 2016)			
n tasks	avg t $[s]$	max t $[s]$	unsl $[\%]$	avg gap $[\%]$	avg t $[s]$	max t $[s]$	unsl $[\%]$	avg gap $[\%]$
10	> 0.01	0.03	0	—	13.07 ($\pm$44.93)	200.22	0	—
15	> 0.01	0.03	0	—	49.67 ($\pm$49.38)	127.09	60	27.32 ($\pm$12.54)
20	0.01 ($\pm$0.01)	0.03	0	—	—	—	100	40.09 ($\pm$15.27)
40	0.09 ($\pm$0.17)	0.81	0	—	—	—	100	77.66 ($\pm$6.23)
60	1.37 ($\pm$4.33)	19.71	0	—	—	—	100	84.23 ($\pm$2.90)
80	0.38 ($\pm$0.45)	1.94	0	—	—	—	100	90.72 ($\pm$1.77)
100	1.28 ($\pm$1.38)	5.05	0	—	—	—	100	93.38 ($\pm$0.76)
150	11.77 ($\pm$24.29)	93.01	0	—	—	—	100	96.02 ($\pm$0.24)
200	22.69 ($\pm$61.24)	281.04	0	—	—	—	100	97.33 ($\pm$0.13)

Table 2: Computational results for the problem $1|mc = 3, mu|C_{\max}$.

	Bottom-up w/ Covering MILP 7.2				Relative Order MILP (Hanzalek et al., 2016)			
n tasks	avg t $[s]$	max t $[s]$	unsl $[\%]$	avg gap $[\%]$	avg t $[s]$	max t $[s]$	unsl $[\%]$	avg gap $[\%]$
10	0.02 ($\pm$0.01)	0.04	0	—	0.09 ($\pm$0.07)	0.28	0	—
20	0.16 ($\pm$0.36)	1.66	0	—	—	—	100	28.71 ($\pm$16.62)
30	0.17 ($\pm$0.17)	0.66	0	—	—	—	100	63.28 ($\pm$7.35)
40	0.69 ($\pm$1.02)	3.61	0	—	—	—	100	72.85 ($\pm$6.14)
50	2.40 ($\pm$7.42)	33.56	0	—	—	—	100	80.61 ($\pm$2.96)
60	6.71 ($\pm$11.97)	44.67	0	—	—	—	100	84.30 ($\pm$2.70)
70	11.30 ($\pm$22.31)	79.38	10	0.38 ($\pm$0.19)	—	—	100	89.34 ($\pm$1.43)
80	37.92 ($\pm$68.82)	224.86	20	0.34 ($\pm$0.13)	—	—	100	91.09 ($\pm$1.40)

Table 3: Computational results for the real-life instances.

	Bottom-up w/ Covering MILP 7.2	
n tasks	avg t $[s]$	avg gap $[\%]$
50	0.08 ($\pm$0.14)	—
100	1.17 ($\pm$3.33)	—
150	3.67 ($\pm$7.23)	0.26 ($\pm$0.00)
200	5.34 ($\pm$15.51)	—

9 CONCLUSION

In this paper, we have proposed two exact approaches for the problem of non-preemptive mixed-criticality match-up scheduling for solving the problem of message retransmission in time-triggered communication protocols. We investigated the fundamental properties of F-shapes to obtain efficient models of the problem. Our algorithms outperform recently proposed approach by a large margin. Furthermore, we showed the membership of $1|mc = \mathcal{L}, mu|C_{\max}$ problem in $\mathcal{APX}$ complexity class for an arbitrary fixed $\mathcal{L}$.

ACKNOWLEDGEMENT

This work was supported by the Grant Agency of the Czech Republic under the Project GACR P103-16-23509S. Furthermore, this work was supported by the US Department of the Navy Grant N62909-15-1-N094 SALTT issued by Office Naval Research Global. The United States Government has a royalty-free license throughout the world in all copyrightable material contained herein.

REFERENCES

Baruah, S. and Fohler, G. (2011). Certification-cognizant time-triggered scheduling of mixed-criticality systems. In *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, pages 3–12. IEEE.

Baruah, S., Li, H., and Stougie, L. (2010). Towards the design of certifiable mixed-criticality systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*, pages 13–22. IEEE.

Bean, J. C., Birge, J. R., Mittenthal, J., and Noon, C. E. (1991). Matchup scheduling with multiple resources, release dates and disruptions. *Operations Research*, 39(3):470–483.

Bell, R. (2006). Introduction to iec 61508. In *Proceedings of the 10th Australian workshop on Safety critical systems and software-Volume 55*, pages 3–12. Australian Computer Society, Inc.

Bertsimas, D., Brown, D. B., and Caramanis, C. (2011). Theory and applications of robust optimization. *SIAM review*, 53(3):464–501.

Burns, A. and Davis, R. (2013). Mixed criticality systems-a review. *Department of Computer Science, University of York, Tech. Rep.*

Dvorak, J. and Hanzalek, Z. (2016). Using two independent channels with gateway for FlexRay static segment scheduling. *IEEE Transactions on Industrial Informatics*, article in press.

Hanzalek, Z., Tunys, T., and Sucha, P. (2016). Non-preemptive mixed-criticality match-up scheduling problem. *Journal of Scheduling, doi: 10.1007/s10951-016-0468-y*.

Kopetz, H., Ademaj, A., Grillinger, P., and Steinhammer, K. (2005). The time-triggered ethernet (TTE) design. In *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*, pages 22–33. IEEE.

Qi, X., Bard, J. F., and Yu, G. (2006). Disruption management for machine scheduling: the case of spt schedules. *International Journal of Production Economics*, 103(1):166–184.

Sahinidis, N. V. (2004). Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering*, 28(6):971–983.

Shabtay, D., Gaspar, N., and Kaspi, M. (2013). A survey on offline scheduling with rejection. *Journal of scheduling*, 16(1):3–28.

Theis, J., Fohler, G., and Baruah, S. (2013). Schedule table generation for time-triggered mixed criticality systems. *Proc. WMC, RTSS*, pages 79–84.

Vestal, S. (2007). Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, pages 239–243. IEEE.