On Solving Non-preemptive Mixed-criticality Match-up Scheduling Problem with Two and Three Criticality Levels

Antonin Novak[1,2], Premysl Sucha[2], Zdenek Hanzalek[1,2]

[1] Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, CZ
[2] Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, CZ

Abstract. In this paper we study an $\mathcal{NP}$-hard problem of a single machine scheduling minimizing the makespan, where mixed-critical tasks with uncertain processing time are scheduled. This scheduling problem arises from the problem of the synthesis of a safe schedule for control messages on a communication channel with the application in the automotive industry. We study the structure of problems with two and three criticality levels, we propose efficient exact algorithms solving the problem and we present computational experiments for instances with up to 200 tasks. Moreover, we show that the considered problem is approximable within a constant multiplicative factor.

1 Introduction

Many applications have mixed-criticality nature. Such applications contain tasks with different criticality in a sense that by following a schedule, the execution of a task with a lower criticality may be skipped if a higher critical task requires more time to complete. This paper deals with the scheduling problem in a mixed-criticality environment where the exact processing time of a task is not known in advance. Instead of that, we assume that for every task a positive integer denoting its criticality and a finite set of alternative processing times are given. The criticality of the task is given by its importance in the considered application and alternative processing times are derived as an approximation of the corresponding probability distribution function. It allows us to define a single schedule for any possible realization of the processing times. Every execution alternative complies with the requirements given by the tasks' criticalities – regardless of any possibility, a prolongation of a task with lower criticality will never reject the execution of a higher critical task.

1.1 Motivation

Our model is motivated by safety-critical applications, such as autonomous cars, using so called time-triggered communication networks (e.g. *FlexRay* bus, *TT-Ethernet* [1,2]), where the nodes have synchronized clocks and messages are

transmitted at the moments defined by the offline schedule. Disturbances, such as electromagnetic noise, affect the reliability of the bus since messages can get lost and, subsequently, degrade the functionality of the car. Since driving a car is a safety-critical application, we improve the reliability of the communication by the retransmission of lost messages. The need for retransmission is rare, but it leads to a prolongation of the communication message at runtime. For this automotive application, the criticality of a message corresponds to its maximum number of possible (re)transmissions.

Scheduling of safety-critical messages on this time-triggered network can be modeled as a scheduling problem where tasks having different criticality levels represent messages and the resource is a communication channel in the network. A typical case in a car assumes three different levels of criticality:

- tasks with high criticality (three transmissions: 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 (e.g. T_4 in Fig. 1);
- tasks with medium criticality (two transmissions: 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 (e.g. T_1 in Fig. 1);
- tasks with low criticality (one transmission: criticality level 1) are used for infotainment functionalities, such as MP3 player (e.g. T_2 in Fig. 1).

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 tasks, when the prolongation of the more critical one takes place. The problem of the offline mixed-criticality match-up scheduling can be represented by F-shapes [3].

An example of a schedule with F-shaped tasks is illustrated in Fig. 1. There five F-shaped tasks forms an offline schedule. An online realized execution scenario for this schedule is depicted by the black line. While realizing it, the actual processing time of T_1 was 9 instead of 5 due to a disturbance. The execution policy states that tasks with smaller start time than the realized completion time of preceding tasks are skipped, i.e. T_2 and T_3. After a task is completed, the execution matches-up with the schedule at the lowest criticality. Therefore, in this example, after T_1 finishes, T_4 is up next. In another possible online execution scenario, where the processing time of T_1 would be 5, T_2 would follow.

Schedules with three criticality levels arises from the application in the automotive domain. The adaptation of IEC 61508 standard [4], *ASIL*, defines the application levels with a hazard assessment corresponding to three *Safety Integrity Levels*. However, the study of the problem with two criticality levels is interesting also, since as we show, it acts as a relaxation to the problems with the greater number of criticality levels. Furthermore, our proposed heuristic can be easily lifted to the arbitrary number of criticality levels.

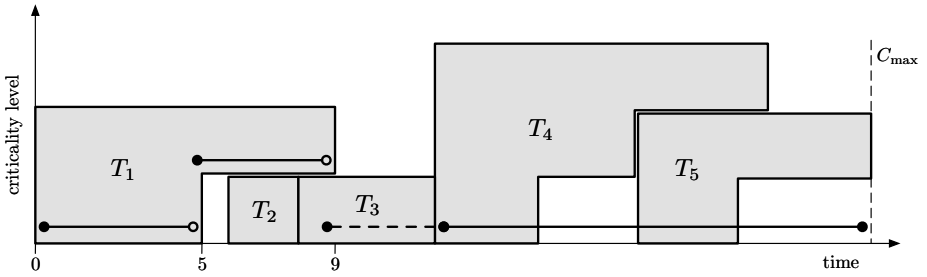

Fig. 1: A feasible schedule of F-shaped tasks with an online execution scenario denoted by the black line.

1.2 Contribution and Paper Outline

In this paper we show the relation between F-shaped tasks and the underlying probability distribution functions. Furthermore, we show that considered problem is polynomial-time approximable within a constant multiplicative factor and we give a 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 we evaluate on instances with up to 200 tasks.

The rest of the paper is organized as follows. In Sect. 2 we survey the related work. In Sect. 3 we show the relation between F-shaped tasks and discretization of cumulative probability distribution functions. In Sect. 6 and 7 we show properties of problem with two and three criticality levels and we propose efficient exact algorithms. Finally, in Sect. 8 we present computational results.

2 Related Work

The most of the existing work on mixed-criticality scheduling deals with the problem by designing scheduling policies that take place in a case of a disruption. Aytug et al. [5] surveys among others a number of reactive approaches. That is, when a disruption is observed during an online execution scenario, a new schedule is computed as the reaction to the occurred situation. However, reactive approaches are not suitable in the case of embedded systems (e.g. cars). An online recomputation of the schedule is undesirable since these systems are typically equipped with a low computational power.

The concept of match-up scheduling was introduced by Bean et al. [6]. 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 context of manufacturing problems [7]. Hanzalek et al. [3] 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 a relative order MILP model for $1|r_j, \tilde{d}_j, mc = \mathcal{L}, mu|C_{\max}$.

Taking farther perspective, the problem can be viewed as a case of robust and stochastic optimization due to an uncertainty about transmission times while satisfying safety requirements. Bertsimas et al. [8] surveys robust versions of various optimization problems, but rather continuous than discrete ones. The field of stochastic optimization is reviewed by Sahinidis in [9]. 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. [10] 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 [3] only, but it lacks of an efficient solution method which is suggested in this paper.

3 Mixed-criticality Match-up Scheduling

3.1 Derivation of F-shaped Tasks

We 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 *cumulative distribution function* (CDF) $\mathcal{F}_i$ giving the probability, that processing time $\mathcal{T}_i$ is at most t. See Figs. 2 and 3 for such an example.

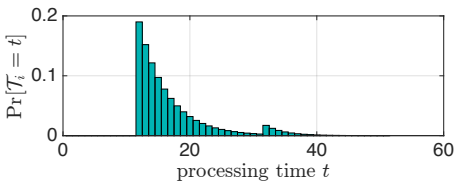

Fig. 2: A discrete probability distribution (PD) over a set of processing times.

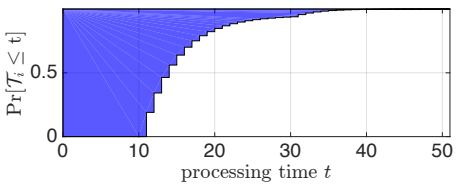

Fig. 3: Corresponding cumulative distribution function (CDF).

We assume that the *criticality* (a positive integer) of each task is given as a part of the problem instance. Criticality levels are associated with a certain required probability (of task execution) and these are used for discretization of the cumulative distribution function $\mathcal{F}_i$ associated with the particular task.

Corresponding processing times $p_i^{(\ell)}$ for each criticality level ℓ are sampled from $\mathcal{F}_i$. The situation for a task with *criticality* 3 with corresponding levels 1, 2 and 3 is depicted in Fig. 4 with vertical axis on the logarithmic scale. Processing

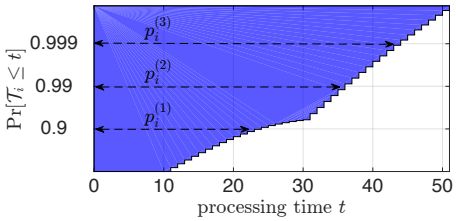

Fig. 4: Discretization of a CDF.

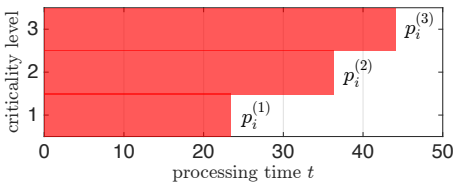

Fig. 5: Corresponding F-shaped task with three criticality levels.

times sampled according to criticality levels then forms a single task like the one depicted in Fig. 5. Since CDFs are non-decreasing functions, processing times $p_i^{(\ell)}$ yield shapes like the F letter rather than ordinary rectangles, hence the name F-shape.

4 Problem Statement

We assume a set of 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 $\mathcal{X}_i \in \{1, \ldots, \mathcal{L}\}$ 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 $p_i^{(1)} < p_i^{(2)} < \ldots < p_i^{(\mathcal{X}_i)}$.*

Having the set $I_{\mathcal{MC}}$ of F-shaped tasks, we define a feasible schedule of $I_{\mathcal{MC}}$ 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 iff $\forall i, j \in \{1, \ldots, n\}, i \neq j$:*

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

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 $\min\{\mathcal{X}_4, \mathcal{X}_5\} = 2$ which 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 makespan (i.e. $\max s_i + p_i^{(\mathcal{X}_i)}$) is minimized. In the three-field scheduling notation it is denoted as $1|mc = \mathcal{L}, mu|C_{\max}$,

where $mc = \mathcal{L}$ stands for 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 [3].

5 General Properties

Although even the problem $1|mc = 2, mu|C_{\max}$ is strongly $\mathcal{NP}$-hard, we show that in some sense finding a reasonable solution is not difficult. 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 $\mathcal{APX}$.*

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

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

□

In fact, this shows that e.g. for the problem with $\mathcal{L} = 2$ criticality levels, any left-shifted schedule will be at most 2 times worse than the optimal makespan since LCF actually produces the worst ordering of tasks in terms of the makespan. In the following sections we focus on 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. We denote the makespan of the left-shifted schedule of permutation π by $C_{\max}(\pi)$. In Sect. 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.

6 Two Criticality Levels

Following the terminology of [3], in this section we refer to tasks with criticality 2 as *hi-tasks* and to tasks with criticality 1 as *lo-tasks*. We start by defining a structure that appears inside optimal solutions.

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

See an example in Fig 1. There it is e.g. T_1 covering T_2 and T_3. All of these tasks form a covering block. From the perspective of the length of such a block, the ordering of lo-tasks inside the block does not matter. This is stated by Prop. 2.

Proposition 2 (Lo-task Commutation) *Given the covering block* B_i*, its length* $|B_i| = \max\{p_i^{(1)} + \sum_{j \in cov(T_i)} p_j^{(1)}, \quad p_i^{(2)}\}$ *is the same for any ordering of lo-tasks* $T_j \in cov(T_i)$*.*

Further we show, that optimal solutions are made of covering blocks.

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

Proof. See Appendix.

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

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

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

The following MILP model is based on Prop. 2 and 3. It assigns lo-tasks to the hi-tasks to form covering blocks such that the sum of their lengths is minimal. The decision variable x_{ij} indicates whether the lo-task T_j is covered by the hi-task T_i. These are then used to compute the length of covering blocks, which sum together with other uncovered lo-tasks to form the makespan of the schedule.

$$\min \sum_{i|\mathcal{X}_i=2} B_i + \sum_{j|\mathcal{X}_j=1} p_j^{(1)}(1 - \sum_{i|\mathcal{X}_i=2} x_{ij}) \tag{1}$$

subject to

$$B_i^{\text{lo}} = p_i^{(1)} + \sum_{j|\mathcal{X}_j=1} x_{ij} p_j^{(1)} \quad \forall i \in I_{\mathcal{MC}}|_{\mathcal{X}_i=2} \tag{2}$$

$$B_i \geq B_i^{\text{lo}} \quad \forall i \in I_{\mathcal{MC}}|_{\mathcal{X}_i=2} \tag{3}$$

$$B_i \geq p_i^{(2)} \quad \forall i \in I_{\mathcal{MC}}|_{\mathcal{X}_i=2} \tag{4}$$

$$\sum_{i|\mathcal{X}_i=2} x_{ij} \leq 1 \quad \forall j \in I_{\mathcal{MC}}|_{\mathcal{X}_j=1} \tag{5}$$

where

$$B_i, B_i^{\text{lo}} \in \mathbb{Z}_0^+ \quad \forall i \in I_{\mathcal{MC}}|_{\mathcal{X}_i=2} \tag{6}$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I_{\mathcal{MC}}|_{\mathcal{X}_i=2}, \forall j \in I_{\mathcal{MC}}|_{\mathcal{X}_j=1} \tag{7}$$

7 Three Criticality Levels

For purposes of this section, we refer to tasks with the highest criticality (i.e. 3) as to hi-tasks, tasks with criticality 2 as to mid-tasks and to tasks with criticality 1 as lo-tasks.

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

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

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

The h^- restriction cuts off all criticality levels above level h. Similarly, h^+ restriction drops all tasks with criticality below h and for the rest of tasks 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 result into an F-shape. A restriction of an instance can be viewed as a relaxation of the problem.

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

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

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

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

7.1 Bottom-up Algorithm

First we introduce a heuristic algorithm for the problem $1|mc = 3, mu|C_{\max}$. The algorithm at first solves 2^- restriction of the given problem instance. Then it creates a new problem instance containing lo-tasks corresponding to covering blocks from the previous solution. Lo-tasks that are not part of any covering block are assigned to an arbitrary covering block. Hi-tasks of covering blocks that have criticality 3 in original instance then form tasks with criticality 2 by including their top criticality level from original instance. Then, the new instance is solved once again as an instance of $1|mc = 2, mu|C_{\max}$ problem. In general, *Bottom-up* algorithm produces suboptimal solutions. However, there are cases when we can verify if the produced schedule is optimal.

Algorithm 1 Bottom-up

$\hat{\pi} \leftarrow$ solve $I^{2^-}_{\mathcal{MC}}$ restriction by Covering MILP 6.1
$I'_{\mathcal{MC}} \leftarrow \emptyset$
for each covering block B_i in the left-shifted solution $\hat{\pi}$ **do**
    if $\mathcal{X}_i = 2$ in $I_{\mathcal{MC}}$ **then**
        $\mathbf{P}_i \leftarrow (|B_i|)$
        $I'_{\mathcal{MC}} \leftarrow I'_{\mathcal{MC}} \cup \{(1, \mathbf{P}_i)\}$
    else if $|B_i| < p_i^{(3)}$ **then**
        $\mathbf{P}_i \leftarrow (|B_i|, p_i^{(3)})$
        $I'_{\mathcal{MC}} \leftarrow I'_{\mathcal{MC}} \cup \{(2, \mathbf{P}_i)\}$
    else
        ▷ block B_i is *saturated*, it contributes by a constant term to the makespan of $I'_{\mathcal{MC}}$
    end if
end for
$\hat{\pi} \leftarrow$ 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 sequence of processing times at associated levels of criticality such that increasing any of them would increase the makespan by the same amount.

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

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

Proof. See Appendix.

When none of optimality conditions is satisfied, e.g. a critical path is coming through every criticality level, we fallback 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.

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

Proposed MILP model for problem $1|mc = 3, mu|C_{\max}$ is based on a similar idea as the model for $1|mc = 2, mu|C_{\max}$ — to form covering blocks and align these under tasks with the highest criticality.

$$\min \sum_{i \in I_{\mathcal{MC}}|x_i=3} p_i + \sum_{j \in I_{\mathcal{MC}}|x_j=2} P_{j,\emptyset} + \sum_{k \in I_{\mathcal{MC}}|x_k=1} p_k^{(1)} x_{\emptyset,\emptyset,k} \tag{8}$$

subject to

$$p_i \geq p_i^{(3)} \quad \forall i \in I_{\mathcal{MC}}|x_i=3 \tag{9}$$

$$My_{i,j} \geq \sum_{k \in I_{\mathcal{MC}}|x_k=1} x_{i,j,k} \quad \forall i \in I_{\mathcal{MC}}|x_i=3 \cup \emptyset, \forall j \in I_{\mathcal{MC}}|x_j=2 \tag{10}$$

$$P_{j,i} \geq p_j^{(2)} y_{i,j} \quad \forall i \in I_{\mathcal{MC}}|x_i=3 \cup \emptyset, \forall j \in I_{\mathcal{MC}}|x_j=2 \tag{11}$$

$$P_{j,i} \geq p_j^{(1)} y_{i,j} + \sum_{k|x_k=1} p_k^{(1)} x_{i,j,k} \quad \forall i \in I_{\mathcal{MC}}|x_i=3 \cup \emptyset, \forall j \in I_{\mathcal{MC}}|x_j=2 \tag{12}$$

$$p_i \geq p_i^{(2)} + \sum_{j \in I_{\mathcal{MC}}|x_j=2} P_{j,i} \quad \forall i \in I_{\mathcal{MC}}|x_i=3 \tag{13}$$

$$p_i \geq p_i^{(1)} + \sum_{j|x_j=2} P_{j,i} + \sum_{k|x_k=1} p_k^{(1)} x_{i,\emptyset,k} \quad \forall i \in I_{\mathcal{MC}}|x_i=3 \tag{14}$$

$$\sum_{i \in I_{\mathcal{MC}}|x_i=3 \cup \emptyset} \sum_{j \in I_{\mathcal{MC}}|x_j=2 \cup \emptyset} x_{i,j,k} \geq 1 \quad \forall k \in I_{\mathcal{MC}}|x_k=1 \tag{15}$$

$$\sum_{i \in I_{\mathcal{MC}}|x_i=3 \cup \emptyset} y_{i,j} \geq 1 \quad \forall j \in I_{\mathcal{MC}}|x_j=2 \tag{16}$$

where

$$y_{i,j} \in \{0,1\} \quad \forall i \in I_{\mathcal{MC}}|x_i=3 \cup \emptyset, \forall j \in I_{\mathcal{MC}}|x_j=2 \tag{17}$$

$$x_{i,j,k} \in \{0,1\} \quad \forall i \in I_{\mathcal{MC}}|x_i=3 \cup \emptyset, \forall j \in I_{\mathcal{MC}}|x_j=2 \cup \emptyset, \forall k \in I_{\mathcal{MC}}|x_k=1 \cup \emptyset : k \neq \emptyset \vee (i = \emptyset \wedge j = \emptyset) \tag{18}$$

$$p_i \in \mathbb{Z}_0^+ \quad \forall i \in I_{\mathcal{MC}}|x_i=3 \tag{19}$$

$$P_{j,i} \in \mathbb{Z}_0^+ \quad \forall i \in I_{\mathcal{MC}}|x_i=3 \cup \emptyset, \forall j \in I_{\mathcal{MC}}|x_j=2 \tag{20}$$

The model utilizes the idea, that optimal solutions are made of blocks (in this case formed by tasks with critiality level 3 that cover less critical tasks) which commute within a solution. It assigns lo-tasks to the mid-tasks and to 2^- restriction of hi-tasks to form covering blocks. Blocks are assigned to the hi-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.

When Bottom-up fails to prove optimality, it fallbacks 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 in terms of the critical path. The other observation is that if the problem instance contains majority of tasks with criticality of one and two, then solving its 2^- restriction frequently yields optimal solution since the highest criticality levels

are not likely to be utilized. Similar holds for the instances with the large number of tasks of a higher criticality. Furthermore, solving $2^{\pm}$ restrictions of $I_{\mathcal{MC}}$ is cheap compared to the solving the whole MILP model 7.2.

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 $n \in \{10, \ldots, 200\}$. Criticalities of tasks were distributed uniformly. Processing time of a task at level 1 is sampled from the uniform distribution $\mathcal{U}(1, 11)$. For tasks with 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 $n \in \{10, \ldots, 80\}$. 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. The

	Covering MILP Sect. 6.1			Relative Order MILP [3]		
n tasks	avg $[s]$	unsl [%]	avg gap [%]	avg $[s]$	unsl [%]	avg gap [%]
10	> 0.01	—	—	13.07 ($\pm$44.93)	—	—
15	> 0.01	—	—	49.67 ($\pm$49.38)	60	27.32 ($\pm$12.54)
20	0.01 ($\pm$0.01)	—	—	—	100	40.09 ($\pm$15.27)
40	0.09 ($\pm$0.17)	—	—	—	100	77.66 ($\pm$6.23)
60	1.37 ($\pm$4.33)	—	—	—	100	84.23 ($\pm$2.90)
80	0.38 ($\pm$0.45)	—	—	—	100	90.72 ($\pm$1.77)
100	1.28 ($\pm$1.38)	—	—	—	100	93.38 ($\pm$0.76)
150	11.77 ($\pm$24.29)	—	—	—	100	96.02 ($\pm$0.24)
200	22.69 ($\pm$61.24)	—	—	—	100	97.33 ($\pm$0.13)

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

	Bottom-up w/ Covering MILP Sect. 7.2			Relative Order MILP [3]		
n tasks	avg $[s]$	unsl [%]	avg gap [%]	avg $[s]$	unsl [%]	avg gap [%]
10	0.02 ($\pm$0.01)	—	—	0.09 ($\pm$0.07)	—	—
20	0.16 ($\pm$0.36)	—	—	—	100	28.71 ($\pm$16.62)
30	0.17 ($\pm$0.17)	—	—	—	100	63.28 ($\pm$7.35)
40	0.69 ($\pm$1.02)	—	—	—	100	72.85 ($\pm$6.14)
50	2.40 ($\pm$7.42)	—	—	—	100	80.61 ($\pm$2.96)
60	6.71 ($\pm$11.97)	—	—	—	100	84.30 ($\pm$2.70)
70	11.30 ($\pm$22.31)	10	0.38 ($\pm$0.19)	—	100	89.34 ($\pm$1.43)
80	37.92 ($\pm$68.82)	20	0.34 ($\pm$0.13)	—	100	91.09 ($\pm$1.40)

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

column *avg* in Tab. 1 and 2 denotes the average computational time for instances that were solved within the time limit of 300 s. The column *unsl* contains the fraction 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.

9 Conclusion

In this paper we have proposed two exact approaches for the problem of non-preemptive mixed-criticality match-up scheduling. They have shown to outperform the approach recently proposed in [3]. We further show the membership of $1|mc = \mathcal{L}, mu|C_{\max}$ problem in $\mathcal{APX}$ complexity class for an arbitrary fixed $\mathcal{L}$. We suggest that future work shall focus on a proper generalization of covering approach for an arbitrary number of criticality levels and on developing approximation algorithms with provably better ratios.

References

1. Jan Dvorak and Zdenek Hanzalek. Multi-variant time constrained flexray static segment scheduling. In *Factory Communication Systems (WFCS), 2014 10th IEEE Workshop on*, pages 1–8. IEEE, 2014.
2. Hermann Kopetz, Astrit Ademaj, Petr Grillinger, and Klaus Steinhammer. 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, 2005.
3. Hanzalek Z., Tunys T., and Sucha P. Non-preemptive mixed-criticality match-up scheduling problem. *Journal of Scheduling, doi: 10.1007/s10951-016-0468-y*, 2016.
4. Ron Bell. 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., 2006.
5. Haldun Aytug, Mark A Lawley, Kenneth McKay, Shantha Mohan, and Reha Uzsoy. Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, 161(1):86–110, 2005.
6. James C Bean, John R Birge, John Mittenthal, and Charles E Noon. Matchup scheduling with multiple resources, release dates and disruptions. *Operations Research*, 39(3):470–483, 1991.
7. Xiangtong Qi, Jonathan F Bard, and Gang Yu. Disruption management for machine scheduling: the case of spt schedules. *International Journal of Production Economics*, 103(1):166–184, 2006.
8. Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.
9. Nikolaos V Sahinidis. Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering*, 28(6):971–983, 2004.
10. Dvir Shabtay, Nufar Gaspar, and Moshe Kaspi. A survey on offline scheduling with rejection. *Journal of scheduling*, 16(1):3–28, 2013.

Appendix

Proposition 3 (Covering Blocks Commutation)

Proof. By the exchange argument we have that at least one optimal schedule is starting with a hi-task. Moreover, we can achieve the same makespan by rearranging uncovered lo-tasks (i.e. the ones that are not part of any covering block) to the end. Therefore, the makespan of an optimal solution is given by two parts — one follows from the makespan achieved by some particular ordering of covering blocks and the other one that is given by the sum of processing time of lo-tasks that are not part of any block.

Now, actually any permutation of blocks yields the same makespan. Suppose two blocks adjacent in a solution — B_i and B_j. Suppose that B_i is starting at time t. If B_j goes directly after B_i, it starts at time $t + |B_i| = t + \max\{p_i^{(1)} + \sum_{k \in \mathrm{cov}(T_i)} p_k^{(1)}, \quad p_i^{(2)}\}$ in a left-shifted schedule, since B_j cannot start earlier (otherwise it would overlap with T_i or with some lo-task that is part of B_i block). Therefore B_j finishes at $t + |B_i| + |B_j|$. If B_i can start at time t, then B_j can start at t also. Thus, swapping B_i and B_j does not influence the makespan. Therefore, the makespan of the whole schedule is given by the sum of lengths of covering blocks and by the sum of the rest of lo-tasks. □

Proposition 5 (Sufficient Optimality Conditions)

Proof. Suppose a schedule satisfying Cond. 1 produced by Bottom-up algorithm. Since no overlapping constraint associated with the highest level between any pair of hi-tasks is active, the makespan of the schedule after the second stage is the same as in the first, i.e. $\min_{\pi \in \Pi(I^{2-}_{\mathcal{MC}})} C_{\max}(\pi)$, which is by Prop. 4 optimal.

Now suppose a schedule produced by Bottom-up algorithm satisfying Cond. 2. Since in the first stage of Bottom-up algorithm no lo-task is a part of a covering block which length is greater than corresponding hi-task in $I^{2-}_{\mathcal{MC}}$, then $I'_{\mathcal{MC}} = I^{2+}_{\mathcal{MC}}$. Therefore the makespan of produced schedule is $\min_{\pi \in \Pi(I^{2+}_{\mathcal{MC}})} C_{\max}(\pi)$ which is by Prop. 4 optimal. □