An Optimization Modulo Theories-Based Approach to Cumulative Scheduling with Delays

Antton Kasslin[1,*], Jeremias Berg[1,*]

[1]Department of Computer Science, University of Helsinki, Finland

Abstract

Scheduling problems arise in many applications and are commonly solved with automated reasoning and constraint-based methods. We study a variant of the so-called resource-constrained cumulative scheduling problem, with a flavor of flow allocation, in which a set of workers must send workloads to a shared processing facility. The goal is to schedule the workload sent from each worker at each time step without exceeding the facility's maximum processing capacity or the storage capacities of the workers. In a central instantiation of the problem, the workers are industrial sources that produce wastewater that needs to be transferred to a treatment plant. The problem of computing a feasible schedule in a similar setting has previously been studied under the name of the wastewater treatment plant problem.

More precisely, we study the applicability of optimization modulo theories (OMT) and mixed integer programming (MIP) to solving real-world benchmarks of this scheduling problem. As our main contributions, we: i) extend a previously proposed constraint model for computing feasible schedules by e.g., allowing more flexible scheduling of the released water, ii) extend the model with several different optimization criteria, including optimizing for evenness of the workload arriving at the facility, and minimizing the makespan of the schedule, and iii) collect a new open-source dataset based on real-world wastewater flow quantities. We evaluate our model on two differently-sized time spans of the new dataset as well as on previously used benchmarks using state-of-the-art solvers in both paradigms. Our evaluation demonstrates that optimization criteria related to makespan minimization do not need to slow down the run time of the solvers. In contrast, the criteria on workload evenness are, in general, more difficult to handle.

Keywords

Optimization Modulo Theories, OMT, Mixed Integer Linear Programming, MILP, Preemptive Cumulative Scheduling, Flow Evenness, Time Delay

1. Introduction

The resource-constrained scheduling problem (RCSP) [1, 2] is a well-known and general formalization of a problem setting in which tasks need to be scheduled over a discrete timeframe in a way that respects the resource requirements of the individual tasks and any possible precedence constraints between them. Many variants of RCSP are known to be NP-hard [3], but the strong prevalence of scheduling problems in different domains has motivated research into different variants of RCSP (see e.g. [4] for a recent survey) and effective methods for computing both feasible and optimal schedules. Resource-constrained scheduling problems can roughly be divided into disjunctive and cumulative scheduling problems based on the portion of tasks that can be scheduled in parallel, and further into preemptive and non-preemptive scheduling based on whether tasks can be interrupted once started. We focus on a variant of preemptive cumulative scheduling that, arguably, has received less attention than many other variants of RCSP [2] and in doing so contribute to the well-established field of constraint-based approaches across various constraint paradigms for solving scheduling problems [2, 5, 6, 7, 8, 9, 10, 11].

More precisely, we study a problem setting in which a set of workers must access the limited resources of a shared processing facility. At each time step, all workers release separate tasks that each require some amount of the facility's processing capacity. Each worker can then send a part of the task's

SMT 2025: 23rd International Workshop on Satisfiability Modulo Theories, August 10–11, 2025, Glasgow, UK

*Corresponding author.

antton.kasslin@helsinki.fi (A. Kasslin); jeremias.berg@helsinki.fi (J. Berg)

https://www.jeremiasberg.com (J. Berg)

0009-0000-3021-8746 (A. Kasslin); 0000-0001-7660-8061 (J. Berg)

total workload directly to the facility, and temporarily store the remainder. The goal is to schedule the workload sent from each worker at each timestep without exceeding either the maximum intake capacity of the facility or the storage capacities of individual workers. The general formulation highlights that our model could be applicable in many settings; the processing facility could be a storage facility or a vaccination center that needs to vaccinate parts of the population. In our experiments, we focus on a central concrete instantiation of the problem called the wastewater treatment problem in which the workers are pumping stations [12, 13] that, at each timestep, receive some amount of water that needs to be transported to a wastewater treatment plant for processing. Each station can temporarily store a limited amount of water, and the treatment plant can only receive a limited amount at each timestep. While a model for computing feasible schedules has been proposed in [13], to the best of our understanding, we present the first study into constraint models for optimal schedules. While presented through the lens of scheduling in previous work, this problem is also closely related to flow allocation in networks [14, 15].

More precisely, as the main contribution of this paper, we develop a constraint model for computing schedules to the general formulation of the wastewater treatment problem. Our model is designed based on consultation from the Helsinki Region Environmental Services Authority (HSY). Specifically, we allow each worker to send only part of the new workload for processing at a time. Our model also allows a time delay between a worker releasing workload and that workload arriving at the facility. We study the computational feasibility of finding both feasible and optimal schedules under a number of different optimization criteria. We show how to minimize the makespan of the schedule and how to compute schedules in which the amount of workload (e.g. wastewater) arriving at the processing facility is as even as possible. Especially the latter criterion is motivated by the discussions with HSY; high wastewater influx rates are undesirable as they necessitate feeding the input faster through the limited-capacity treatment process, leading to worse output water quality. We detail an optimization modulo theories (OMT) [16, 17, 18], and a mixed integer programming (MIP) [19] model for this problem.

As an additional contribution, we collect real-world wastewater data from HSY and use it to form a new benchmark set containing altogether 624 OMT and 624 MIP benchmarks. We report on an extensive evaluation of state-of-the-art solvers in these two constraint paradigms on this problem. In addition to the new benchmarks that we collected, we also evaluate solvers on the data used in [13]. In the evaluation, we study different variants of the problem and demonstrate, e.g., the effect of enforcing all workloads to integer values. Our results indicate that computing optimal schedules incurs a negligible amount of overhead compared to computing feasible ones, and that the MIP solver Gurobi is particularly effective in solving these problems.

The rest of the paper is structured as follows: in Section 2, we detail our problem setting and the constraint paradigms of interest. In Section 3, we describe the constraint model for computing optimal schedules in this setting under different optimization criteria. In Section 4, we describe the new open-source benchmarks we collected as part of this work, and in Section 5, we report on the experimental evaluation of these state-of-the-art solvers in all paradigms on these and previously established benchmarks. The paper is concluded in Section 6.

2. Preliminaries

We detail preliminaries on the variant of preemptive cumulative scheduling that we study, as well as the constraint paradigms in which we model the problem.

2.1. Cumulative Resource-Constrained Scheduling with Delays

We study a variant of the resource-constrained cumulative scheduling problem in which each worker can subdivide and temporarily store their tasks. For some intuition on our problem setting, consider a set of pumping stations that pump wastewater to a treating facility that can process a limited amount of wastewater at each time step. Each pumping station receives new wastewater at each time step and needs to decide how much to pump forward for treatment and how much to store in its temporary

storage with a limited capacity. The task is to schedule the amount of water pumped forward at each timestep while ensuring that neither the treating facility's processing capacity, nor any pumping station's storage capacity is exceeded. Additionally, each pumping station has an upper limit on the amount of water that can be pumped per timestep, and it takes some time for the water to arrive from a pumping station to the facility. A similar problem was studied under the name of the wastewater treatment plant problem in [13].

More abstractly, a problem instance in our setting specifies a finite and discretized time horizon over `#t` timesteps under which `#w` different workers need the limited resources of a single shared facility that can receive a total workload of `maxCapacity` on each time step. The input instance specifies for each worker w and timestep t the new incoming resource requirement $\texttt{task}_{w,t}$ that w will need from the facility within the time horizon. Since the facility can receive a limited quantity of workload per timestep, each worker can temporarily store up to $\texttt{storageCapacity}_w$ of workload. The total quantity of storage capacity at each worker is constant for all time steps; the quantity stored at time t takes from the remaining storage space available at subsequent timesteps until the workload is sent to the facility. Finally, we assume that the sent workload takes $\texttt{d}_w$ timesteps to arrive from w at the facility and that the maximal rate of workload output to the facility for each worker is limited by $\texttt{maxOutput}_w$.

The goal in our setting is to schedule the amount of workload sent from each worker at each timestep to the facility. More precisely, a *schedule* specifies for each worker w a storage output $\texttt{Sout}(w,t)$ as the amount of the currently stored workload to send for processing at timestep t, and the part $\texttt{P}(w,t)$ of $\texttt{task}_{w,t}$ released at timestep t to immediately send for processing. The rest of the task released at t then needs to be temporarily stored. A schedule is feasible if neither the maximum capacity of the facility nor the maximum storage or output capacities of any worker are exceeded at any timestep.

In preemptive scheduling, the processing of each task can be paused and continued later. Since a worker in our setting can subdivide its task in a way that momentarily sends zero workload for processing, our problem formulation captures a preemptive setting as long as the storage capacities of the individual workers are high enough to enable this. As we focus on an optimization setting, we also only consider instances in which the time horizon, the `maxCapacity` of the facility, and the $\texttt{maxOutput}_w$ and $\texttt{storageCapacity}_w$ values of the workers are large enough for feasible schedules to exist.

To summarize, we say that a schedule is *feasible* if the total workload from all workers combined arriving at the facility at any timestep does not not exceed `maxCapacity`, and the following three statements hold for each timestep t and worker w: (i) the workload temporarily stored at w does not exceed $\texttt{storageCapacity}_w$, (ii) the total quantity of workload released from w, $\texttt{Sout}(w,t) + \texttt{P}(w,t)$ does not exceed the maximum output rate $\texttt{maxOutput}_w$, (iii) all workloads arrive at the facility within the time horizon.

For a concrete example, Table 1 details a feasible schedule for a problem instance where the shared facility is a wastewater treatment plant (wwtp), and there are two pumping stations with flexible forward-pumping capacities up to $\texttt{maxOutput}_1 = 6000\ \text{m}^3/\text{h}$, and $\texttt{maxOutput}_2 = 10000\ \text{m}^3/\text{h}$, both located 0 hours away from the facility (i.e. $\texttt{d}_1 = \texttt{d}_2 = 0$). The maximum capacity of the facility is $\texttt{maxCapacity} = 15\,000\ \text{m}^3$ incoming wastewater per hour, and the workers have individual storage capacities of $\texttt{storageCapacity}_1 = 6000\,\text{m}^3$ and $\texttt{storageCapacity}_2 = 10000\,\text{m}^3$, respectively. The displayed schedule is for $\texttt{task}_{1,1} = 4000$, $\texttt{task}_{1,2} = 5000$, $\texttt{task}_{2,1} = 2000$ and $\texttt{task}_{2,2} = 5000\ \text{m}^3$ and assuming initial storage levels of $\texttt{startLevel}_1 = \texttt{Storage}(1,0) = 3000$ and $\texttt{startLevel}_2 = \texttt{Storage}(2,0) = 5000\ \text{m}^3$.

2.1.1. Optimal Schedules

In addition to the problem of computing feasible schedules, we consider several different optimization extensions. The first three are motivated by a desire to find schedules that reduce exceptionally high or low levels in the hourly processing requirements placed on the facility. In the context of wastewater treatment, the motivation for computing schedules that have an even flow of wastewater arriving at the treatment plant is motivated by the treatment processes and discussions we had with HSY [12] . With

Table 1
Example of a feasible schedule for wastewater pumping for two stations (workers) with maximum outputs $\texttt{maxOutput}_1 = 6000$ and $\texttt{maxOutput}_2 = 10000$ per timestep t, and a treatment facility max capacity $\texttt{maxCapacity} = 15000$ for each t. The **Storage** column contains the storage level of worker w at the end of timestep t, measured after the possible storage level changes during t.

t	Worker 1				Worker 2				Total to wwtp
	task	Sout	P	Storage	task	Sout	P	Storage	
0	0	0	0	3000	0	0	0	5000	0
1	4000	2000	4000	1000	2000	4000	2000	1000	12000
2	5000	1000	5000	0	5000	1000	5000	0	12000

high influx rates, water has to be fed through the limited-capacity process faster, resulting in a shorter retention time for the water in each successive part of the process, reducing the overall treatment level. Additionally, optimization of the treatment process itself is facilitated by a steady influx of wastewater.

In more detail, we focus on three separate optimization criteria related to minimizing fluctuations in the quantity of workload that arrives at the facility. In the following, let $\texttt{Total}(t)$ be the total workload arriving at the facility at timestep t. The MAXMIN objective maximizes $\texttt{MinWorkload}$, i.e., maximizes the minimum workload arriving at the facility over all timesteps. Analogously, the MINMAX objective minimizes the maximum $\texttt{Total}(t)$ over all timesteps, resulting in the smallest possible maximum workload $\texttt{MaxWorkload}$ received at the facility during the time horizon. Finally, the MINDIFF objective minimizes the difference $\texttt{MaxWorkload} - \texttt{MinWorkload}$, essentially preferring schedules with more even incoming hourly workloads. Under this objective, the problem can be seen as an example of a resource leveling problem [4]. Table 1 shows an optimal schedule with the MINDIFF objective, where $\texttt{MaxWorkload} - \texttt{MinWorkload} = 12000 - 12000 = 0$.

Additionally, we present results on two objectives that align more closely with previous work on scheduling problems. The MAKESPAN objective minimizes the makespan of the schedule, i.e. the last timestep at which the facility receives any workload. Finally, the MSTORAGE objective minimizes the total sum of storage levels over all workers w and all timesteps t. As an interesting side remark, we note that the schedules optimal under MSTORAGE are also optimal under the MAKESPAN objective. Because a decrement of $\texttt{Storage}(w, t)$ by a quantity of $\texttt{Sout}(w, t) > 0$ also decreases the sum of $\texttt{Storage}(w, t+1) + \texttt{Storage}(w, t+2) + \cdots + \texttt{Storage}(w, \#\texttt{t})$, MSTORAGE leads to emptying of all storages at the earliest possible timestep, which is the objective of MAKESPAN.

To end this section, we note that our problem setting is closely related to the well-studied network flow allocation problems [14, 15] that seek to assign a flow value to each edge in a directed graph, typically in a way that maximizes the total amount of flow between a specified source and sink node. The two central differences between the max-flow problem and our problem setting are that we allow the workers (i.e. the nodes in a graph) to temporarily store workload (i.e., the flow in the graph), and that the three main optimization criteria we study do not correspond to maximizing the flow through the network.

2.2. OMT and MIP

We assume familiarity with propositional logic and satisfiability and recount some basics of optimization modulo theories (OMT) with arithmetics over integer and real numbers using the standard interpretation of the symbols $+, -, =, \leq$ and $<$.

A term T is a sum or difference of integer or real variables. A theory atom a is a comparison $(\mathrm{T}_1 < \mathrm{T}_2)$ or $\mathrm{T}_1 = \mathrm{T}_2$ between terms. A literal ℓ is either a $\{0, 1\}$-variable x, a theory atom a, or their negation $\neg x$ and $\neg a$. A clause C is a disjunction $(\vee)$ of literals, and a formula F is a conjunction $(\wedge)$ of clauses. When convenient, we view a clause C as a set of its literals and a formula F as a set of its clauses.

An assignment α maps variables to their domains. A theory atom a is assigned to 1 by α (i.e. $\alpha(a) = 1$) if assigning the variables in the terms results in a true comparison. The semantics of assignments are extended to literals, clauses and formulas in the standard way: $\alpha(\neg\ell) = 1 - \alpha(\ell)$,

Table 2
Constants used in the OMT model of the scheduling problem.

Constant	Description
`#w`	number of workers
`#t`	number of timesteps
`maxCapacity`	maximum workload the facility can receive at each time step
$\texttt{task}_{w,t}$	total workload of the task released at worker w at time t
$\texttt{storageCapacity}_w$	maximum workload storage capacity of w
$\texttt{startLevel}_w$	workload stored at worker w at the beginning of timestep 1
$\texttt{d}_w$	number of timesteps for workload from w to arrive at the facility
$\texttt{maxOutput}_w$	maximum workload a worker can send for processing at any timestep

Table 3
Variables for the OMT model of the scheduling problem

Variable	Description
$\texttt{P}(w,t)$	workload released at w at time t immediately sent to the facility
$\texttt{Sout}(w,t)$	workload sent from storage of w at timestep t to the facility
$\texttt{Storage}(w,t)$	workload in the temporary storage of w at the end of timestep t
$\texttt{Total}(t)$	total workload arriving at the facility at t

$\alpha(C) = \max\{\alpha(\ell) \mid \ell \in C\}$, and $\alpha(F) = \min\{\alpha(C) \mid C \in F\}$. We say that an assignment α for which $\alpha(F) = 1$ is a solution to F. The satisfiability modulo linear integer and rational arithmetic problem is to decide the existence of a solution to a given formula. An OMT instance (over the same theory) (F, T) consists of a formula F and a term T. The task is to compute a solution α of F that minimizes T. If one wishes to maximize T, this is achieved by minimizing $-\mathrm{T}$.

In addition to OMT, we consider mixed integer programming (MIP) where the goal is to minimize an objective $O \equiv \sum_i c_i z_i + \sum_i c_i r_i$ where z_i are integer variables, r_i are real variables, and c_i are real constants, subject to a set of linear inequalities.

3. Constraint Models for Cumulative Scheduling with Delays

In this section, we detail our constraint models for the scheduling problem. In Section 3.1, we detail the OMT model and how it relates to the model for computing feasible schedules proposed in [13]. In Section 3.2 we describe the MIP model.

3.1. OMT Model for Cumulative Scheduling

Table 2 details the constants defined by an instance of the scheduling problem that our models use. The number of workers in the problem instance and the number of time steps in the time horizon are denoted by `#w` and `#t`, respectively. The constant `maxCapacity` is the maximum workload that can arrive at the facility at each timestep. The workload required from the facility by the task released (created) at timestep t at worker w is denoted by $\texttt{task}_{w,t}$. The maximum workload that w can have in storage at any given timestep is $\texttt{storageCapacity}_w$. We allow for some workload to be stored already at the beginning of the time horizon; we denote the amount stored at w in the beginning by $\texttt{startLevel}_w$. Denoting delay, $\texttt{d}_w$ is the number of timesteps it takes for any workload sent from w to arrive at the storage facility. The maximum quantity of workload that can be sent from the worker w for processing at any timestep is $\texttt{maxOutput}_w$. We will assume that the maximum amount of workload that can be added to the storage of w at any timestep is at least equal to $\texttt{storageCapacity}_w$; a storage may be filled in a single timestep.

Table 3 details the variables in the OMT model. The variable $\texttt{P}(w,t)$ is the (sub)quantity of the workload of $\texttt{task}_{w,t}$ which is immediately sent to the facility from worker w at timestep t. Analogously, $\texttt{Sout}(w,t)$ is the amount of workload sent for processing from the storage of w at timestep t. For

notational convenience, we let $\texttt{P}(w,t) = \texttt{Sout}(w,t) = 0$ whenever $t \leq 0$. $\texttt{Storage}(w,t)$ is the amount of workload in the temporary storage of w at the end of timestep t. Finally, $\texttt{Total}(t)$ is the total workload arriving at the facility at time t, from all workers combined.

3.1.1. Constraints in the OMT Model

Note that while the timesteps, t, range from 1 to $\texttt{\#t}$, we denote $\texttt{startLevel}_w$ for all workers w by $\texttt{Storage}(w,0)$.

The first constraints define the domains of the main variables. The amount of workload immediately sent for processing by worker w is at least 0 and at most $\texttt{task}_{w,t}$; the constraint

$$(0 \leq \texttt{P}(w,t)) \wedge (\texttt{P}(w,t) \leq \texttt{task}_{w,t}) \tag{1}$$

is included for all timesteps t in the range of 1 to $\texttt{\#t} - \texttt{d}_w$. Note that for any feasible schedules to exist $\texttt{task}_{w,t} = 0$ has to hold for the last $\texttt{d}_w$ timesteps.

For some intuition on the domains of $\texttt{Sout}(w,t)$ and $\texttt{storageCapacity}_w$, note that in any feasible schedule, the storage of worker w needs to be emptied at the latest at the timestep $\texttt{\#t} - \texttt{d}_w$ and not increased after it so that all of the workload from w can arrive at the facility by the timestep $\texttt{\#t}$. In our model, the $\texttt{Sout}(w,t)$ variable is only included for $t \in [1, \texttt{\#t} - \texttt{d}_w]$ during which its domain is set to be between 0 and $\texttt{storageCapacity}_w$; the constraint

$$(0 \leq \texttt{Sout}(w,t)) \wedge (\texttt{Sout}(w,t) \leq \texttt{Storage}(w,t-1)) \tag{2}$$

is added for all timesteps $t \in [1, \texttt{\#t} - \texttt{d}_w]$. Our setting also requires that the total quantity of workload sent from w at each time step is at most $\texttt{maxOutput}_w$; the constraint

$$(\texttt{Sout}(w,t) + \texttt{P}(w,t) \leq \texttt{maxOutput}_w) \tag{3}$$

is included for all workers and timesteps in the range of 1 to $\texttt{\#t} - \texttt{d}_w$. The domain of $\texttt{Storage}(w,t)$ is set between 0 and $\texttt{storageCapacity}_w$ for all timesteps in the range of 0 to $\texttt{\#t} - \texttt{d}_w - 1$ and to equal 0 for the final $\texttt{d}_w + 1$ timesteps (note that $\texttt{Storage}(w,t)$ denotes the storage level at the *end* of timestep t, after possible emptying or filling at t); the constraint

$$(0 \leq \texttt{Storage}(w,t)) \wedge (\texttt{Storage}(w,t) \leq \texttt{storageCapacity}_w) \tag{4}$$

is included for $t \in [0, \texttt{\#t} - \texttt{d}_w - 1]$, and $(\texttt{Storage}(w,t) = 0)$ for $t \in [\texttt{\#t} - \texttt{d}_w, \texttt{\#t}]$.

Finally, the total workload arriving for processing at t is the sum of storage-derived and immediately sent workloads arriving from each worker at t; the constraint

$$\texttt{Total}(t) = \textstyle\sum_{w=1}^{\texttt{\#w}} \texttt{Sout}(w, t - \texttt{d}_w) + \texttt{P}(w, t - \texttt{d}_w) \tag{5}$$

is added for all timesteps and all defined $\texttt{Sout}(w,t)$ values. With these variables, the constraint that enforces that the maximum capacity of the facility is not exceeded is

$$(\texttt{Total}(t) \leq \texttt{maxCapacity}) \tag{6}$$

which is added for all timesteps $t \in [1, \texttt{\#t}]$.

The final sets of constraints encode the semantics of $\texttt{Storage}(w,t)$ and link the workload immediately sent for processing with the changes in the amounts stored. For some intuition on these, note that the amount of new workload stored at worker w on time t is $\texttt{task}_{w,t} - \texttt{P}(w,t)$. With this intuition, the amount of workload stored at worker w is increased on each iteration by $\texttt{task}_{w,t} - \texttt{P}(w,t)$ and decreased by $\texttt{Sout}(w,t)$. Stated as a constraint, this is

$$\texttt{Storage}(w,t) = \texttt{Storage}(w,t-1) - \texttt{Sout}(w,t) + \texttt{task}_{w,t} - \texttt{P}(w,t) \tag{7}$$

which is added for all $w \in [1, \texttt{\#w}]$ and $t \in [1, \texttt{\#t} - \texttt{d}_w]$.

The last constraints link the changes in the amount of workload stored with the amount of workload immediately sent for processing. In essence, the constraints state that the amount of new workload $\texttt{task}_{w,t}$ is equal to the amount of workload immediately sent for processing and any positive change in the amount stored at w. If the change in the amount stored is negative (or zero), indicating that some workload (or none) was also sent from the storage, then the workload equal to $\texttt{task}_{w,t}$ should be sent for processing immediately. As constraints, with $\Delta\texttt{S}(w,t) = \texttt{Storage}(w,t) - \texttt{Storage}(w,t-1)$ as notational shorthand, this is:

$$\neg(0 < \Delta\texttt{S}(w,t)) \vee (\texttt{task}_{w,t} = \texttt{P}(w,t) + \Delta\texttt{S}(w,t)) \tag{8}$$

$$\neg(\Delta\texttt{S}(w,t) \leq 0) \vee (\texttt{task}_{w,t} = \texttt{P}(w,t)) \tag{9}$$

i.e. $0 < \Delta\texttt{S}(w,t) \implies \texttt{task}_{w,t} = \texttt{P}(w,t) + \Delta\texttt{S}(w,t)$ and $\Delta\texttt{S}(w,t) \leq 0 \implies \texttt{task}_{w,t} = \texttt{P}(w,t)$. The constraints 8 and 9 are added for all $w \in [1, \#\texttt{w}]$, $t \in [1, \#\texttt{t} - \texttt{d}_w]$, for which $\texttt{task}_{w,t} > 0$.

We summarize the model for computing feasible schedules as follows. Let $F^{\texttt{schedule}}$ be an SMT formula consisting of Constraints 1-9 as specified in this section. Then any solution α of $F^{\texttt{schedule}}$ sets the $\texttt{P}(w,t)$, $\texttt{Sout}(w,t)$, and $\texttt{Storage}(w,t)$ variables in a way that corresponds to a feasible schedule.

3.1.2. Optimal Schedules with OMT

To extend the constraint model from feasibility to optimization, we next describe how to encode each of the five optimization criteria o discussed in Section 2.1.1 as a term T^o such that the optimal solutions to the OMT instance $(F^{\texttt{schedule}}, \text{T}^o)$ correspond to optimal schedules under o. Here $F^{\texttt{schedule}}$ is an SMT formula consisting of the Constraints 1-9 detailed in 3.1.1.

To encode the MAXMIN and MINMAX optimization criteria we add the real variables `MinWorkload` and `MaxWorkload` to the instance as well as the constraints $(\texttt{MinWorkload} \leq \texttt{Total}(t)) \wedge (\texttt{Total}(t) \leq \texttt{MaxWorkload})$ for all $t \in [1, \#\texttt{t}]$ to define them. Then maximizing `MinWorkload` over the solutions that satisfy the $F^{\texttt{schedule}}$ results in a schedule that maximizes the minimum workload arriving at the facility at each time step (i.e. the MAXMIN criteria). Analogously, minimizing `MaxWorkload` obtains a schedule that minimizes the maximum workload (i.e., the MINMAX criteria). Finally, minimizing $\texttt{MaxWorkload} - \texttt{MinWorkload}$ obtains a schedule in which the largest absolute difference in arriving workload is as small as possible across all timesteps, i.e. the MINDIFF optimization criteria. The bounds for `MinWorkload` and `MaxWorkload` are initialized to 0 and `maxCapacity`.

The MSTORAGE objective is encoded by minimizing the sum of all $\texttt{Storage}(w,t)$ variables. Finally, the MAKESPAN objective is encoded by minimizing the integer `LastWorkIn` variable defined to equal the last time step on which the facility receives any workload. This definition is enforced with the constraints $\neg(\texttt{Total}(t) > 0) \vee (t \leq \texttt{LastWorkIn})$ added for each timestep.

3.1.3. Relation to Previous Work

We briefly detail the main differences between our OMT model for cumulative scheduling presented in Section 3.1.1, and the SMT approach to computing feasible schedules for the wastewater treatment plant problem proposed by Bofill, Muñoz and Murillo in [13] that we will call the BMM model from now on. The BMM model was studied in a setting where the workers are industrial facilities that produce integer quantities of wastewater to send to a wastewater treatment plant. The BMM model assumes no delay for the workers, i.e $\texttt{d}_w = 0$ for all w, and that the storages of each worker are empty at the beginning of the timeframe, i.e. that $\texttt{startLevel}_w = 0$ for all w. It also enforces a limit $\texttt{TankFlow}_w$ only on $\texttt{Sout}(w,t)$, instead of $\texttt{Sout}(w,t) + \texttt{P}(w,t)$, i.e., the amount of wastewater that can be released from the storage of each worker at any given timestep. The BMM model also includes the constraints $(\texttt{Sout}(w,t) = 0) \vee (\texttt{Sout}(w,t) = \min\{\texttt{TankFlow}_w, \texttt{Storage}(w,t-1)\})$ that enforce that the workload released from the storage of any worker is either zero or the maximum possible, and the constraints $(\texttt{P}(w,t) = 0) \vee (\texttt{P}(w,t) = \texttt{task}_{w,t})$ that enforce for each $\texttt{task}_{w,t}$ that either all of $\texttt{task}_{w,t}$ is directly sent for processing, or none is.

Our constraint model extends the problem of computing feasible schedules to optimization and allows more flexible workload sending from workers. Our model allows, e.g., for sending all of the new workload $\texttt{task}_{w,t}$ at t together with a part of the workload in storage. Constraint 3 in our model limits the total quantity of workload sent from each worker rather than just the quantity released from the storage of each worker; this decision was made based on discussion with the local wastewater agency HSY where we confirmed that all wastewater at a given station is pumped forward using the same pumps regardless of possible prior temporary storing.

3.2. A MIP Model for Cumulative Scheduling with Delays

The MIP model we propose uses all constants and variables listed in Tables 2 and 3 as well as a $\{0,1\}$-variable $\texttt{SMinus}(w,t)$ for each worker w and timestep t. $\texttt{SMinus}(w,t)$ is an indicator for w emptying (some of) its storage at timestep t. In other words, a solution to the MIP model that assigns $\texttt{SMinus}(w,t) = 1$ corresponds to a schedule in which $\texttt{Sout}(w,t) > 0$.

3.2.1. Constraints in the MIP Model

Our MIP model includes the Constraints 1-7 detailed in Section 3.1.1 as well as a "Big-M" encoding of the implications in Constraints 8-9. The constant $\texttt{M}$ was set as the maximum $\texttt{task}_{w,t} + \texttt{maxOutput}_w$ over all timesteps and workers, and ϵ was set to 10^{-14}, which is the smallest possible *positive* storage decrement for any worker and timestep.

$$\Delta\texttt{S}(w,t) \leq -\epsilon \cdot \texttt{SMinus}(w,t) + \texttt{M} \cdot (1 - \texttt{SMinus}(w,t)) \tag{10}$$
$$\Delta\texttt{S}(w,t) \geq -\texttt{M} \cdot \texttt{SMinus}(w,t) \tag{11}$$
$$\texttt{P}(w,t) \geq \texttt{task}_{w,t} - \texttt{M} \cdot (1 - \texttt{SMinus}(w,t)) \tag{12}$$
$$\texttt{P}(w,t) \leq \texttt{task}_{w,t} + \texttt{M} \cdot (1 - \texttt{SMinus}(w,t)) \tag{13}$$
$$\texttt{P}(w,t) + \Delta\texttt{S}(w,t) \geq \texttt{task}_{w,t} - \texttt{M} \cdot \texttt{SMinus}(w,t) \tag{14}$$
$$\texttt{P}(w,t) + \Delta\texttt{S}(w,t) \leq \texttt{task}_{w,t} + \texttt{M} \cdot \texttt{SMinus}(w,t) \tag{15}$$

where $\Delta\texttt{S}(w,t) = \texttt{Storage}(w,t) - \texttt{Storage}(w,t-1)$. Constraint 10 requires that a non-negative $\Delta\texttt{S}(w,t)$ results in $\texttt{SMinus}(w,t) = 0$. Constraint 11 requires that a negative $\Delta\texttt{S}(w,t)$ results in $\texttt{SMinus}(w,t) = 1$. Thus, $\texttt{SMinus}(w,t)$ is 1 if the storage of worker w is decremented at timestep t, and 0 otherwise. Constraints 12-13 ensure that if the storage of w at timestep t is being partially emptied, the entire task $\texttt{task}_{w,t}$ is directly sent to the processing facility. If instead there is no storage decrement, then constraints 14-15 split the workload as $\texttt{task}_{w,t} = \texttt{P}(w,t) + \Delta\texttt{S}(w,t)$.

3.2.2. Optimal Schedules with MIP

For the MIP model, the MAXMIN, MINMAX, MINDIFF and MSTORAGE objectives were encoded precisely as described in Section 3.1.2. In order to encode MAKESPAN, we introduce for each t a binary variable $\texttt{TotalPositive}(t)$ as an indicator for the total workload arriving at the facility being greater than zero. With this variable, the implication $\texttt{Total}(t) > 0 \implies \texttt{LastWorkIn} \geq t$ that defines the $\texttt{LastWorkIn}$ to be minimized for the MAKESPAN objective is encoded with the following constraints: $\texttt{Total}(t) \leq \texttt{maxCapacity} \cdot \texttt{TotalPositive}(t)$, and $\texttt{LastWorkIn} \geq t \cdot \texttt{TotalPositive}(t)$. These ensure that if $\texttt{TotalPositive}(t) = 0$ then $\texttt{Total}(t) = 0$ and if $\texttt{Total}(t) > 0$ then $\texttt{TotalPositive}(t) = 1$.

4. New Open-Source Benchmarks

Before presenting the results of our experimental evaluation, we briefly detail the new open-source benchmark set for OMT and MIP that we collected for this work.

We obtained hourly real-world data on the quantities of wastewater pumped and wastewater surface levels in 2023 and 2024 at two pumping stations maintained by HSY. The two pumping stations have

maximum outputs of $\texttt{maxOutput}_1 = 30000$ m^3/h and $\texttt{maxOutput}_2 = 5543$ m^3/h, respectively. The maximum capacity of the treatment plant is $\texttt{maxCapacity} = 29166$ m^3/h. The storage capacities of the workers were estimated from the data. For the first station, we knew the theoretical total volume of the storage. However, in consultation with HSY, we learned that storing water amounts even close to the maximum volume would risk flooding of associated underground structures and be highly undesirable. Thus, we instead set $\texttt{storageCapacity}_1 = 168241$ m^3 based on the maximum water level realized in 2023 and 2024. For the second station, we did not know the maximum volume of the storage. Instead, we assumed that the largest drops in surface level correspond to situations with maximal output of the pumps coinciding with comparatively negligible volume of incoming water. We averaged 10 largest drops in surface levels during 2023 and 2024 to get an estimate for the ratio of surface level change (in cm) to output water volume, and based on the highest observed storage surface level in this two-year period then set the maximum storage volume as $\texttt{storageCapacity}_2 = 20468$ m^3. The maximum output $\texttt{maxOutput}_1 = 30000$ m^3/h for the first station was obtained from the agency, while $\texttt{maxOutput}_2$ was set to 5542 m^3/h based on the maximum value in the data. The default delay $\texttt{d}_1$ and $\texttt{d}_2$ of both stations is 0 hours.

With these values for the constants, we create two sets of benchmarks: the **24-hour local set** and the **1104-hour local set**. The 24-hour set is based on the timespan from 2024-11-16 00:00 to 2024-11-16 23:00, and the 1104-hour set on 2024-11-16 00:00 to 2024-12-31 23:00. This specific timeframe for our evaluation was chosen due to the convenience of the data not containing any missing values in these ranges. With the starting point of the timespan decided, the starting storage levels for the two stations could be read directly from the dataset as $\texttt{startLevel}_1 = 58182$ m^3 and $\texttt{startLevel}_2 = 379$ m^3, respectively. In line with [13] we experiment with a range of different maximum capacities of the processing facility. As our focus is on optimization, we did some preliminary testing in order to determine capacities high enough to guarantee the existence of feasible schedules. For the 1104-hour set, we consider $\texttt{maxCapacity}$ values in the range 12000 to 30000 m^3 with intervals of 2000 m^3 and for the 24-hour set we used 18000 to 30000 m^3 with intervals of 1000. The ranges start from the lowest satisfiable $\texttt{maxCapacity}$ divisible by 1000 common for the all-zero and artificially generated $\texttt{d}_w$ value choices. In both sets, we also include benchmarks with $\texttt{maxCapacity}$ set to the maximum value of 50000 m^3 to observe the solving times on a still higher maximum capacity. In total, the 24-hour set includes 88 and the 1104-hour set 120 feasibility instances. Considering the five optimization criteria described in Section 3.1.2, the full sets include 1248 benchmarks, of which 208 are feasibility instances.

5. Experimental Evaluation

We report on an experimental evaluation of the constraint models described in Section 3. After detailing the setup of the experiments in Section 5.1, we present results on the relative hardness of computing optimal schedules under different optimization criteria, as well as the effect of enforcing the workloads to be integer and having delays between the workers and the facility in Section 5.2.

5.1. Setup

We describe the benchmarks, solvers and the hardware we use. The open-source benchmarks (including the script for generating them) are available online [20].

Benchmarks

In addition to the local benchmark sets described in Section 4 we use the dataset called real (which we call B24) from [13] since our preliminary tests showed it to be the most challenging of the datasets in [13]. The dataset contains information about the volume of wastewater produced each hour over a 24-hour period by eight industrial facilities. We use 2500, 3000, 1500, 1000, 3000, 1500, 1000, and 2500 L/h from [13] as as the $\texttt{maxOutput}_w$ values for the 8 workers in our benchmarks. These are the same as used in the previous work for restricting the amount of water sent from the storages at each

Table 4
Summary of the MIP and OMT benchmark sets. The number of variables and constraints are reported for the OMT instances.

name	short	# mcapacity	# benchmarks	#constraints	#variables
24-hour local	L24	11	44	413 - 433	145 - 157
1104-hour local	L1104	15	60	18773 - 18793	6625 - 6637
BMM-real	B24	12	48	1377 - 1486	577 - 673

time step. We also consider `maxCapacity` values in the range of 5000 to 6000 L/h with intervals of 100 L/h, together with 50000 L/h. The lowest value was chosen based on the feasibility checks in our preliminary experiments, while the step of 100 L is the same as in previous work [13].

In our experiments, we study the effect of two variations of the benchmarks being solved. The first is whether all variables in the models are restricted to real values or integral values. The motivation for studying the integer variants is two-fold. On the one hand, the abstract problem setting could be applicable in settings where the workload cannot be divided, e.g., storing discrete wares or treating people. On the other, some solvers do not support floating-point values, and those that do can suffer from rounding errors when dealing with them.

The second variation of benchmarks we study is whether there is a delay between the workers and the facility. The default delays for all workers in the local data is 0 based on the physical locations of the stations and the processing facility. The BMM dataset does not specify delays for the industrial facilities. Thus, we consider the default values of the workers in the BMM set to be 0 as well. To demonstrate the effect that delays have on the overall solving time, we consider a variant of the benchmarks with synthetic delay values. For the local sets, the synthetic variant puts the delay of the second worker to 2. For B24, we randomly assigned delay values 2, 2, 5, 2, 6, 5, 3, and 4 for the eight facilities, respectively.

Table 4 summarizes the feasibility benchmarks we use in our evaluation. The number of constraints and variables reported are for the OMT variants of the benchmarks. The total number of benchmarks reported is obtained as the number of different maximum capacity values considered, plus the four configurations following from the two variants (delays and integer variables) that we just described. For example, for the 24-hour local dataset with 11 different `maxCapacity` values, this translates to 44 benchmarks of feasibility problems. The total number of unique combinations over all the datasets is thus 44+60+48=152 for OMT and MIP models, encompassing all combinations of `maxCapacity`, choice of integer vs. real-valued workload, and all-zero or non-zero delay values. Combined with the 5 optimization criteria and runs without optimization we end up with 912 runs for each OMT and MIP solver ($6 \cdot (44 + 60) = 624$ runs with the local sets, and 288 with B24 data).

Solvers and Hardware

As OMT solvers, we consider OptiMathSAT (OMath) version 1.7.3 [17] (obtained from https://optimathsat.disi.unitn.it/) and Z3 [21] (from https://github.com/Z3Prover/z3). For MIP, we use Gurobi version 12.0.0 [22] (from https://www.gurobi.com/).

The OMT benchmarks in SMT-LIBv2-format were produced using the Python API of Z3 [21], and the MIP benchmarks in MPS format with the Python API of Gurobi [22]. All evaluations were performed single-threaded on 2.50-GHz Intel Xeon Gold 6248 machines with 381-GB RAM in RHEL under a per-instance 32-GiB memory limit and 30-minute time limit.

5.2. Results

Table 5 overviews the effect of different optimization criteria as well as enforcing all workload values to integers on the overall solving time of the MIP and OMT solvers. Each cell in the matrix corresponds to 76 runs, half of the total 152 feasibility instances we use.

We observe that neither the distinction between integral or real values nor the choice between different optimization criteria influence the running time of Gurobi or OptiMathSAT that much. In

Table 5
Percentage of benchmarks solved and the mean running time under different optimization criteria with workloads as integers (Type = Int) or as reals (Type = Real). Each cell corresponds to 76 runs. The column **Feas.** corresponds to no optimization criteria.

Solver	Type	Metric	Feas.	MSTORAGE	MAKESPAN	MAXMIN	MINMAX	MINDIFF
Gurobi	Real	solved	100%	100%	100%	100%	100%	100%
		mean	4.4s	5.6s	6.3s	6.0s	6.4s	5.8s
	Int	solved	100%	100%	100%	100%	100%	100%
		mean	4.4s	5.6s	6.3s	6.3s	6.2s	6.1s
Z3	Real	solved	100%	61.8%	100%	80.3%	60.5%	60.5%
		mean	16.6s	707.6s	191.2s	455.1s	717.3s	717.1s
	Int	solved	100%	93.4%	100%	100%	100%	78.9%
		mean	25.3s	559.7s	24.1s	52.4s	232.8s	519.7s
OMath.	Real	solved	100%	60.5%	60.5%	60.5%	60.5%	60.5%
		mean	176.8s	713.0s	712.8s	712.7s	712.4s	712.8s
	Int	solved	100%	60.5%	60.5%	60.5%	60.5%	55.3%
		mean	95.0s	713.4s	712.7s	713.0s	712.9s	809.0s

Table 6
Percentage of all benchmarks solved when the delays are synthetic (Type=synthetic) or all zero (Type=all-zero) for the OMT and MIP solvers. Each cell has 76 runs. The column **Feas.** corresponds to no optimization criteria.

Solver	Type	Metric	Feas.	MSTORAGE	MAKESPAN	MAXMIN	MINMAX	MINDIFF
Gurobi	synthetic	solved	100%	100%	100%	100%	100%	100%
		mean	4.5s	5.4s	6.3s	6.5s	6.5s	6.0s
	all-zero	solved	100%	100%	100%	100%	100%	100%
		mean	4.4s	5.8s	6.3s	5.8s	6.1s	6.0s
Z3	synthetic	solved	100%	75.0%	100%	100%	80.3%	78.9%
		mean	18.5s	660.6s	106.4s	108.8s	483.1s	519.2s
	all–zero	solved	100%	80.3%	100%	80.3%	80.3%	60.5%
		mean	23.4s	606.7s	108.8s	398.7s	467.0s	717.6s
OMath.	synthetic	solved	100%	60.5%	60.5%	60.5%	60.5%	60.5%
		mean	129.8s	713.2s	713.1s	712.7s	712.7s	713.2s
	all-zero	solved	100%	60.5%	60.5%	60.5%	60.5%	55.3%
		mean	142.0s	713.2s	712.4s	713.0s	712.6s	808.6s

fact, all Gurobi runs finished within 32 seconds. For Z3, we observe that enforcing the MSTORAGE optimization criteria leads, in general, to higher running times and fewer solved instances compared to enforcing MAKESPAN. We also observe that Z3 generally solves the integer-restricted benchmarks more efficiently, and that optimizing the evenness of the flow to the processing plant is in general, more challenging than only minimizing the makespan of the schedule, as witnessed by the lower number of instances solved and higher running times of Z3 under MINMAX, MAXMIN and MINDIFF when compared to MAKESPAN. For a more fine-grained view of the results for the OMT solvers, we note that OptiMathSAT solved all feasibility benchmarks, but was not able to solve any optimization benchmarks from the L1104 dataset within the time limit, while Z3 solved all feasibility instances and 58.3 % of the optimization instances from L1104.

As an interesting side note, we observe that when enforcing the constraint $(\mathrm{P}(w,t) = 0) \vee (\mathrm{P}(w,t) = \mathtt{task}_{w,t})$ for all workers and timesteps that was used in the previous model of a similar problem [13] (recall Section 3.1.3), we observed significant increases in solving time for both MIP and OMT. In fact, with this additional constraint, the performance of the OMT solvers was much more comparable to the performance of Gurobi, an observation in line with the one made in [13]. Removing this constraint led to overall decreases in solving time in both paradigms, albeit more significant decreases for Gurobi.

Table 6 demonstrates the effect of non-zero delays on the overall solving time of MIP and OMT

solvers. We observe that delays between the workers and the processing facility do not, in general, influence the running time of any solvers significantly.

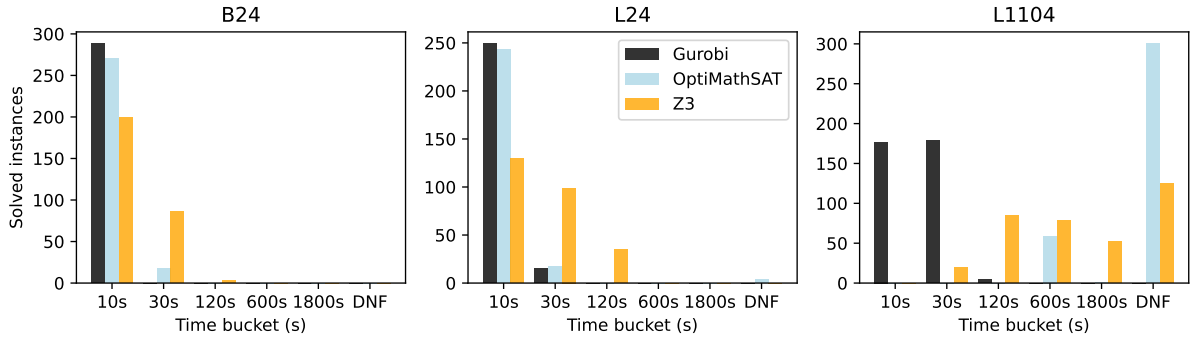

Figure 1: The number of solved instances in each exclusive time range ($[0, 10]$ s, $]10, 30]$, $]30, 120]$, $]120, 600]$, $]600, 1800]$, DNF=not solved in 30 minutes) for Gurobi (leftmost bar for each range), OptiMathSAT and Z3 over all variations of benchmarks and optimization criteria, for all three datasets (288 benchmarks for B24, 264 for L24 and 360 for L1104, total 912 benchmarks for each solver).

Figure 1 provides a more detailed breakdown of the hardness of the benchmarks used in this paper. The plot details a breakdown of solving times for all combinations of feasibility benchmarks and optimization criteria. We observe that Gurobi solves almost all benchmarks in 30 seconds or less and that the L1104 dataset is generally more challenging for all solvers, more noticeably for Z3 and OptiMathSAT, the latter of which solves less than a quarter of the L1104 benchmarks in thirty minutes. Finally, we note that while L24 has the same timespan but fewer workers than B24, it still seems to result in more difficult benchmarks; we hypothesize that this is due to the significantly larger values in the data.

6. Conclusions

We studied a variant of cumulative scheduling under different optimization criteria, focusing on a concrete instantiation in which a set of wastewater sources schedule pumping the water for treatment while not overloading the facility's max capacity or their individual (temporary storage) capacities. We presented the first constraint model for computing optimal schedules for this problem, collected a new real-world dataset, and used it to evaluate the performance of state-of-the-art solvers in MIP and OMT in this setting. Our results demonstrate the effectiveness of MIP for the setting and the feasibility of OMT. Interesting future work includes extending the model to settings where not all workers are *directly* connected to the processing facility; multiple workers may link together prior to the facility, as is common for wastewater treatment infrastructure.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

Acknowledgements

The work is supported by the Research Council of Finland under the grant 362987. The authors wish to thank the Finnish Computing Competence Infrastructure (FCCI) for computational and data storage resources, and Helsinki Region Environmental Services Authority (HSY) for the Finnish wastewater treatment data.

References

[1] Y. Caseau, F. Laburthe, Cumulative scheduling with task intervals, in: JICSLP, MIT Press, 1996, pp. 363–377.

[2] P. Baptiste, C. L. Pape, Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems, Constraints An Int. J. 5 (2000) 119–139.

[3] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.

[4] S. Hartmann, D. Briskorn, An updated survey of variants and extensions of the resource-constrained project scheduling problem, Eur. J. Oper. Res. 297 (2022) 1–14.

[5] E. P. K. Tsang, Constraint based scheduling: Applying constraint programming to scheduling problems, J. Sched. 6 (2003) 413–414.

[6] E. Demirovic, N. Musliu, F. Winter, Modeling and solving staff scheduling with partial weighted maxsat, Ann. Oper. Res. 275 (2019) 79–99.

[7] C. A. Floudas, X. Lin, Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications, Ann. Oper. Res. 139 (2005) 131–162.

[8] S. S. Craciunas, R. S. Oliver, M. Chmelík, W. Steiner, Scheduling real-time communication in IEEE 802.1qbv time sensitive networks, in: RTNS, ACM, 2016, pp. 183–192.

[9] X. Jin, C. Xia, N. Guan, P. Zeng, Joint algorithm of message fragmentation and no-wait scheduling for time-sensitive networks, IEEE CAA J. Autom. Sinica 8 (2021) 478–490.

[10] G. Patti, L. L. Bello, L. Leonardi, Deadline-aware online scheduling of TSN flows for automotive applications, IEEE Trans. Ind. Informatics 19 (2023) 5774–5784.

[11] D. Shen, T. Zhang, J. Wang, Q. Deng, S. Han, X. S. Hu, Qos guaranteed resource allocation for coexisting embb and URLLC traffic in 5g industrial networks, in: RTCSA, IEEE, 2022, pp. 81–90.

[12] Ontario Onsite Wastewater Association, Flow balancing and flow equalization, 2020. URL: https://www.oowa.org/wp-content/uploads/2022/05/FINAL-OOWA_GD_Flow-Balancing-07162020.pdf.

[13] M. Bofill, V. Muñoz, J. Murillo, Solving the wastewater treatment plant problem with SMT, CoRR abs/1609.05367 (2016).

[14] T. Nixon, R. M. Curry, P. A. B., Mixed-integer programming models and heuristic algorithms for the maximum value dynamic network flow scheduling problem, Comput. Oper. Res. 175 (2025) 106897.

[15] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network flows - theory, algorithms and applications, Prentice Hall, 1993.

[16] P. Trentin, Optimization Modulo Theories with OptiMathSAT, Ph.D. thesis, University of Trento, Italy, 2019.

[17] R. Sebastiani, P. Trentin, Optimathsat: A tool for optimization modulo theories, J. Autom. Reason. 64 (2020) 423–460. URL: https://doi.org/10.1007/s10817-018-09508-6. doi:10.1007/S10817-018-09508-6.

[18] C. W. Barrett, R. Sebastiani, S. A. Seshia, C. Tinelli, Satisfiability modulo theories, in: Handbook of Satisfiability, volume 336 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2021, pp. 1267–1329.

[19] D. Chen, R. G. Batson, Y. Dang, Applied Integer Programming: Modeling and Solution, Wiley, 2009. URL: http://dx.doi.org/10.1002/9781118166000. doi:10.1002/9781118166000.

[20] A. Kasslin, J. Berg, Benchmark repository for "An Optimization Modulo Theories-based Approach to Cumulative Scheduling with Delays", 2025. URL: https://doi.org/10.5281/zenodo.15741417. doi:10.5281/zenodo.15741417.

[21] L. M. de Moura, N. S. Bjørner, Z3: an efficient SMT solver, in: C. R. Ramakrishnan, J. Rehof (Eds.), Proc TACAS, volume 4963 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 337–340. URL: https://doi.org/10.1007/978-3-540-78800-3_24. doi:10.1007/978-3-540-78800-3_24.

[22] Gurobi Optimization, LLC, Gurobi Optimizer Reference Manual, 2024. URL: https://www.gurobi.com.