An Evaluation of Constraint-Based Approaches to Cumulative Scheduling with Delays

Antton Kasslin[1]
University of Helsinki, Helsinki, Finland

Jeremias Berg
University of Helsinki, Helsinki, Finland

Abstract

Scheduling problems arise in many applications and are commonly solved with constraint-based methods. We study a variant of the so-called resource-constrained cumulative scheduling problem 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 e.g. 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 delays between the workers and the facility, 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 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.

2012 ACM Subject Classification Applied computing, Operations research

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

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

The resource-constrained scheduling problem (RCSP) [5, 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. 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, 20, 9, 10, 7, 13, 17].

In our problem setting, 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 total workload directly to the facility, and temporarily store the remainder. The

[1] Corresponding author

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 [16, 4] that, at each timestep, receive some amount of water that needs to be transported to a wastewater treatment plant (wwtp) 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. 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 [15, 1].

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 Helsinki Region Environmental Services Authority (HSY). Specifically, we allow a time delay between a worker releasing workload and that workload arriving at the facility. In contrast to previous work on a similar problem [16, 4], our model also allows each worker to send only part of the new workload for processing at a time. 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 evened out. 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) [19, 18, 3], and a mixed integer programming (MIP) [6] model for this problem.

Further, we collect real-world wastewater data from HSY to form a new set containing 1248 benchmarks for OMT and MIP. We report on an extensive evaluation of state-of-the-art solvers in these two constraint paradigms on this problem. In addition, we evaluate solvers on the data used in [4]. 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 0...42x overhead when compared to computing feasible ones and that the MIP solver Gurobi is especially effective in solving these problems.

2 Preliminaries

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 example, 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 limited-capacity temporary storage. 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, the output capacity (volume per timestep) of each pumping station is limited, and it takes some time for the water to arrive from a pumping station to the facility.

More abstractly, a problem instance in our setting specifies a finite and discretized time

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 wwtp $\texttt{maxCapacity} = 15000$ for each t. $\texttt{Storage}(w,t)$ denotes 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

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; 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 contrast to the model presented here, a previous study [4] considered solely feasibility with strictly zero-hour delays, all-or-none storage of incoming tasks and unlimited output for $\texttt{P}(w,t)$. Further, as our focus is on optimization instead, we 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.

For a concrete example, Table 1 details a feasible schedule for a problem instance with a wastewater treatment plant (wwtp) as the shared facility, and two pumping stations located 0 hours away from the facility (i.e. $\texttt{d}_1 = \texttt{d}_2 = 0$). The workers have individual storage capacities of $\texttt{storageCapacity}_1 = 6000$ m^3 and $\texttt{storageCapacity}_2 = 10000$ m^3, respectively. The displayed schedule assumes initial storage levels of $\texttt{startLevel}_1 = \texttt{Storage}(1,0) = 3000$ and $\texttt{startLevel}_2 = \texttt{Storage}(2,0) = 5000$ m^3.

2.1.1 Optimal Schedules

We introduce three objectives motivated by a desire to find schedules that reduce exceptionally high or low levels in the hourly processing requirements placed on the facility. The motivation for even input to a wwtp is motivated by the treatment processes [16] and discussions we had with the local agency. With 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. Objective `MAXMIN` maximizes `MinWorkload`, i.e., maximizes the minimum workload arriving at the facility over all timesteps. Analogously, `MINMAX` minimizes the maximum $\texttt{Total}(t)$, resulting in the smallest possible maximum workload `MaxWorkload` received at the facility during the time horizon. Finally, `MINDIFF` 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 [12]. 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 seen in previous work on scheduling problems. Objective `MAKESPAN` minimizes the makespan of the schedule, i.e. the last timestep at which the facility receives workload, and objective `MSTORAGE` minimizes the sum of $\texttt{Storage}(w,t)$ over all w and t. As an interesting side note, optimal schedules under `MSTORAGE` lead to the achievement of 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 $\texttt{Storage}(w,t+1) + \cdots + \texttt{Storage}(w,\texttt{\#t})$, `MSTORAGE` empties all storages at the earliest possible timesteps. Thus, solutions that are optimal under `MSTORAGE` are also optimal under `MAKESPAN`.

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.

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 $\mathrm{T}_1 = \mathrm{T}_2$. A literal ℓ is either a $\{0,1\}$-variable x or a theory atom a. A clause C is a disjunction ($\vee$) of literals, and a formula F is a conjunction ($\wedge$) of 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: $\alpha(\neg\ell) = 1 - \alpha(\ell)$, $\alpha(C) = \max\{\alpha(\ell) \mid \ell \in C\}$, 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.

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 The OMT Model

3.1 Constraints in the OMT and MIP Models

The amount of workload immediately sent for processing by worker w is at most $\texttt{task}_{w,t}$; the constraints

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

are 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.

To simplify notation, let $\texttt{Storage}(w, 0) = \texttt{startLevel}_w$ for all w. In any feasible schedule, the storage of worker w must be emptied at the latest at $\texttt{\#t} - \texttt{d}_w$ 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. 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 amount of new workload stored at worker w on time t is $\texttt{task}_{w,t} - \texttt{P}(w, t)$. Thus, 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 $\texttt{Sout}(w, t)$:

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

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

The 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 as $\texttt{P}(w, t)$:

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

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

i.e. $0 < \Delta\texttt{S}(w, t) \implies \texttt{task}_{w,t} = \texttt{P}(w, t) + \Delta\texttt{S}(w, t)$ and $\Delta\texttt{S}(w, t) \leq 0 \implies \texttt{task}_{w,t} = \texttt{P}(w, t)$, where $\Delta\texttt{S}(w, t) = \texttt{Storage}(w, t) - \texttt{Storage}(w, t-1)$ is used only as a notational shorthand. 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.

For the MIP model, we further introduce a $\{0,1\}$-variable $\texttt{SMinus}(w,t)$ as an indicator for w emptying (some of) its storage at t, i.e. for $\texttt{Sout}(w,t) > 0$. Constraints 1-7 remain as for the OMT model, while constraints 8-9 are replaced with the following "Big-M" encoding:

$$\Delta\texttt{S}(w,t) \leq -\epsilon \cdot \texttt{SMinus}(w,t) + \texttt{M} \cdot (1 - \texttt{SMinus}(w,t)) \tag{10}$$

$$\Delta\texttt{S}(w,t) \geq -\texttt{M} \cdot \texttt{SMinus}(w,t) \tag{11}$$

$$\texttt{P}(w,t) \geq \texttt{task}_{w,t} - \texttt{M} \cdot (1 - \texttt{SMinus}(w,t)) \tag{12}$$

$$\texttt{P}(w,t) \leq \texttt{task}_{w,t} + \texttt{M} \cdot (1 - \texttt{SMinus}(w,t)) \tag{13}$$

$$\texttt{P}(w,t) + \Delta\texttt{S}(w,t) \geq \texttt{task}_{w,t} - \texttt{M} \cdot \texttt{SMinus}(w,t) \tag{14}$$

$$\texttt{P}(w,t) + \Delta\texttt{S}(w,t) \leq \texttt{task}_{w,t} + \texttt{M} \cdot \texttt{SMinus}(w,t) \tag{15}$$

where the constant ϵ is set to 10^{-14} as the smallest possible *positive* storage decrement.

3.2 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 $(F^{\texttt{schedule}}, \mathrm{T}^o)$ correspond to optimal schedules under o. Here $F^{\texttt{schedule}}$ is an SMT formula consisting of Constraints 1-9 as concluded in 3.1.

To encode `MAXMIN` and `MINMAX`, we add real variables `MinWorkload` and `MaxWorkload` as well as constraints $(\texttt{MinWorkload} \leq \texttt{Total}(t)) \wedge (\texttt{Total}(t) \leq \texttt{MaxWorkload})$ for all $t \in [1, \texttt{\#t}]$. Then maximizing `MinWorkload` over the solutions that satisfy the $F^{\texttt{schedule}}$ maximizes the minimum workload arriving at the facility at each time step (`MAXMIN`). Analogously, minimizing `MaxWorkload` obtains a schedule that minimizes the maximum workload (`MINMAX`). 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. `MINDIFF`. 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.

4 Experimental Evaluation

Benchmarks and Setup

We establish two sets of benchmarks with `maxCapacity` values between 5000 and 50000 depending on the set: the **24-hour local set** and the **1104-hour local set** based on the timespans from 2024-11-16 00:00 to 2024-11-16 23:00 and to 2024-12-31 23:00, respectively. For the five optimization criteria, the full sets include 1248 benchmarks, including 208 feasibility instances. We also use the real-life dataset (B24) from [4] which includes the volumes of wastewater produced each hour over a 24-hour period by eight industrial facilities.

As OMT solvers, we consider OptiMathSAT (OMath.) version 1.7.3 [18] (obtained from `https://optimathsat.disi.unitn.it/`) and Z3 [8] (from `https://github.com/Z3Prover/z3`). For MIP, we use Gurobi version 12.0.0 [11] (from `https://www.gurobi.com/`). The solvers and benchmarks (including the script for generating them) are available at [14].

Table 2 Solve percentages and mean running without (**Feas.**) or with optimization criteria with workloads as floats (Type = Float) or as integer (Type = Int). Each cell corresponds to 76 runs.

Solver	Type	Metric	Feas.	MSTORAGE	MAKESPAN	MAXMIN	MINMAX	MINDIFF
Z3	Float	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.	Float	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

We study the effects of (1) handling exclusively either floating point or integer values, and (2) different delay values on benchmark solving times. The real-life delay for all workers in the local data is 0. The problem setting of B24 dataset from [4] did not consider delays. Thus, we consider the default delays in the BMM set to be 0 as well. To demonstrate the effect of delays, 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 the BMM set, we randomly assigned the delay values 2,2,5,2,6,5,3 and 4 hours for the 8 facilities.

The total number of benchmarks is obtained as the number of different maximum capacity values considered, plus the four configurations following from the two variants (delays and integer workloads). The total number of unique combinations over all the datasets is 44+60+48=152 for OMT and MIP models, encompassing all combinations of `maxCapacity`, integer vs. floating-point workloads, and all-zero or synthetic delays. Combined with the 5 optimization criteria and runs without optimization we end up with 912 runs for each solver.

The OMT benchmarks in SMT-LIBv2-format were produced using the Python API of Z3 [8], and the MIP benchmarks in MPS format with the Python API of Gurobi [11]. 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.

4.1 Results

Most notably, all Gurobi runs finished within 32 seconds, whereas both Z3 and Optimathsat had instances for the `MSTORAGE`, `MAXMIN`, `MINMAX` and `MINDIFF` objectives that were not solved in 30 minutes. Table 2 overviews the effect of different optimization criteria as well as enforcing all workload values to integers on the overall solving time of the OMT solvers.

We observe that neither the distinction between integral or floating point values nor the choice between different optimization criteria influence the running time OptiMathSAT that much. For Z3, 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 (`MINDIFF`, `MINMAX`, `MAXMIN`) is in general more challenging than enforcing `MAKESPAN`. While Z3 solved all feasibility instances and 58.3 % of the optimization instances of the L1104 dataset, OptiMathSAT was not able to solve any optimization benchmarks from the L1104 set in thirty minutes.

As an interesting side note, we observe that when enforcing the constraint that requires either all-new wastewater produced to be directly sent for processing or then none of it

Table 3 Percentage of all benchmarks solved when the delays are synthetic (Type=synthetic) or all zero (Type=all-zero). Each cell corresponds to 76 runs.

Solver	Type	Metric	Feas.	MSTORAGE	MAKESPAN	MAXMIN	MINMAX	MINDIFF
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

similar to the one used in [4], we observed significant increases in solving time for both MIP and OMT. Removing this constraint led to overall decreases in solving time in both paradigms, albeit more significant decreases for Gurobi.

For Z3, MINDIFF and MAXMIN benchmarks with all-zero delays are more difficult to solve than those with synthetic delays, while other large differences are not observed (table 3).

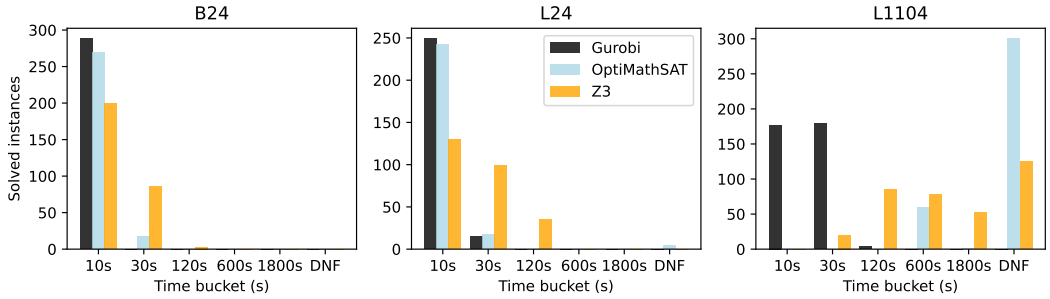

Figure 1 The number of solved instances within $[0, 10]$, $]10, 30]$, $]30, 120]$, $]120, 600]$ and $]600, 1800]$ s, DNF=not solved in 30 min, for the three solvers over all variations of benchmarks and optimization criteria, for the datasets (288 benchmarks for B24, 264 for L24, 360 for L1104).

In Figure 1, we observe that Gurobi solves almost all benchmarks in 30 seconds or less and that the 1104-hour dataset is more challenging for all solvers, especially for Z3 and OptiMathSAT which solves less than a quarter of the L1104 benchmarks in thirty minutes.

5 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 and the feasibility of OMT for the setting. 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.

References

1. Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.
2. Philippe Baptiste and Claude Le Pape. Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems. *Constraints An Int. J.*, 5(1/2):119–139, 2000.
3. Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1267–1329. IOS Press, 2021.
4. Miquel Bofill, Víctor Muñoz, and Javier Murillo. Solving the wastewater treatment plant problem with SMT. *CoRR*, abs/1609.05367, 2016.
5. Yves Caseau and François Laburthe. Cumulative scheduling with task intervals. In *JICSLP*, pages 363–377. MIT Press, 1996.
6. Der-San Chen, Robert G. Batson, and Yu Dang. *Applied Integer Programming: Modeling and Solution*. Wiley, December 2009. URL: `http://dx.doi.org/10.1002/9781118166000`, `doi:10.1002/9781118166000`.
7. Silviu S. Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. Scheduling real-time communication in IEEE 802.1qbv time sensitive networks. In *RTNS*, pages 183–192. ACM, 2016.
8. Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Proc TACAS*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008. `doi:10.1007/978-3-540-78800-3\_24`.
9. Emir Demirovic, Nysret Musliu, and Felix Winter. Modeling and solving staff scheduling with partial weighted maxsat. *Ann. Oper. Res.*, 275(1):79–99, 2019.
10. Christodoulos A. Floudas and Xiaoxia Lin. Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Ann. Oper. Res.*, 139(1):131–162, 2005.
11. Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL: `https://www.gurobi.com`.
12. Sönke Hartmann and Dirk Briskorn. An updated survey of variants and extensions of the resource-constrained project scheduling problem. *Eur. J. Oper. Res.*, 297(1):1–14, 2022.
13. Xi Jin, Changqing Xia, Nan Guan, and Peng Zeng. Joint algorithm of message fragmentation and no-wait scheduling for time-sensitive networks. *IEEE CAA J. Autom. Sinica*, 8(2):478–490, 2021.
14. Antton Kasslin and Jeremias Berg. Benchmark repository for "An Optimization Modulo Theories-based Approach to Cumulative Scheduling with Delays", June 2025. `doi:10.5281/zenodo.15741417`.
15. Tanner Nixon, Robert M. Curry, and Phanuel Allaissem B. Mixed-integer programming models and heuristic algorithms for the maximum value dynamic network flow scheduling problem. *Comput. Oper. Res.*, 175:106897, 2025.
16. 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`.
17. Gaetano Patti, Lucia Lo Bello, and Luca Leonardi. Deadline-aware online scheduling of TSN flows for automotive applications. *IEEE Trans. Ind. Informatics*, 19(4):5774–5784, 2023.
18. Roberto Sebastiani and Patrick Trentin. Optimathsat: A tool for optimization modulo theories. *J. Autom. Reason.*, 64(3):423–460, 2020. URL: `https://doi.org/10.1007/s10817-018-09508-6`, `doi:10.1007/S10817-018-09508-6`.
19. Patrick Trentin. *Optimization Modulo Theories with OptiMathSAT*. PhD thesis, University of Trento, Italy, 2019.
20. Edward P. K. Tsang. Constraint based scheduling: Applying constraint programming to scheduling problems. *J. Sched.*, 6(4):413–414, 2003.