Systems & Control Transactions

Original Research Article
Peer Reviewed Conference Proceeding
ESCAPE 36 - European Symposium on Computer Aided Process Engineering
Sheffield, UK. 21-24 June 2026

PSE PRESS

Set-based Formulations for the State Task Network Scheduling Problem

David A. Liñán[a], Georgia Stinchfield[b], Carl D. Laird[b], and Jan Kronqvist[a]*

[a] KTH Royal Institute of Technology, Department of Mathematics, Stockholm, SE-10044, Sweden
[b] Carnegie Mellon University, Department of Chemical Engineering, Pittsburgh, PA-15213, USA
* Corresponding Author: jankr@kth.se.

ABSTRACT

The state task network (STN) representation is a widely used modeling approach for optimal multipurpose batch production scheduling. In practice, STNs have been traditionally formulated as mixed-integer programming (MIP) problems and solved using general-purpose MIP solvers relying on branch-and-bound and branch-and-cut. In the meantime, alternative modeling and solution paradigms for optimization have been developed, enabling the incorporation of alternative variable types and optimization algorithms. Specifically, this work relies on the Hexaly software, which introduced set-based models and their solution through general-purpose hybrid algorithms, i.e., methods that combine traditional MIP with constraint programming, local search, large neighborhood search, among other tools. So far, Hexaly has shown promising results when tackling optimal scheduling problems, however, set-based models and solution approaches for STN optimization have not been studied in the literature. Aiming to fill this gap, this work introduces the first set-based models for the STN problem and performs preliminary benchmarking tests with Hexaly's general-purpose hybrid algorithms. Results from two case studies suggest that set-based models may perform better than the traditional MIP STN formulation when dealing with a simple STN with sequential connectivity and a long-term scheduling instance of a STN with fixed integer batching variables. Overall, this work establishes the foundations for advancing research on set-based approaches for network scheduling, opening new directions beyond traditional MIP optimization.

Keywords: Scheduling, Optimization, Modelling, Process Operations, Batch Systems

INTRODUCTION

Scheduling problems represent some of the most prominent applications of mathematical optimization [1]. In particular, the state task network (STN) [2] scheduling model is a seminal mixed-integer programming (MIP) formulation that led to the development of a variety of network scheduling MIP formulations such as the resource task network (RTN) [3] model, among other extensions [4, 5]. The main practical feature of network scheduling models is their flexibility in supporting complex tasks with multiple inputs and outputs, enabling material mixing and splitting, i.e., the STN and RTN models do not require materials to be handled sequentially [6]. This modeling flexibility has led to a variety of real-world applications, e.g., optimal scheduling in a vegetal oil refinery [7], multipurpose batch plant scheduling in the chemical-pharmaceutical industry [8], electricity scheduling for cement plants [9], among others [10, 11]. Despite several extensions and variants are available in the literature, this work concentrates on the standard STN formulation.

Traditional mathematical programming has powered the generation of optimal STN scheduling solutions in chemical engineering. Most works in the literature rely on MIP modeling and algorithms, e.g., branch-and-bound and branch-and-cut [10]. This is because STN scheduling applications are usually linear and only require binary or integer variables combined with continuous variables. Nonetheless, the generalized disjunctive programming (GDP) modeling formalism has also been proposed to derive STN formulations, through the incorporation of Boolean variables, disjuncts and logic relationships [12, 13]. In addition to MIP and GDP, a higher-level modeling formalism available in the literature is set-based modeling, which not only enables all the modeling features from MIP and GDP, but also introduces the idea of using sets as

variables. Set-based optimization has resulted in compact formulations yielding computational advantages compared to MIP in a variety of scheduling applications, e.g., job-shop, open-shop, resource-constrained, among others [14]. However, the field of optimal set-based STN scheduling has not yet been explored.

This work brings novelty by deriving and testing the first set-based formulations for the STN scheduling problem. Set-based models introduce three key variable types, namely set variables, list variables and interval variables. In this work, we aim to contribute novel mixed-interval disjunctive program (MIDP) and mixed-interval-list disjunctive program (MILDP) STN models. Our secondary objective is to evaluate the computational performance of commercial software Hexaly given these set-based formulations. How we choose to model the STN directly contributes to the efficiency with which the optimization problem is solved. However, deriving the best set-based model (e.g., intervals, intervals + sets, intervals + lists) is not straightforward. While Hexaly exploits this type of modeling, the progress made by their proprietary algorithms and heuristics is not directly visible to the practitioner. Thus, observing how different formulations of the STN interact with Hexaly is vital for determining which combinations of set-based modeling paradigms are most efficient.

This study is organized as follows: first we provide modeling background. Then, we describe the proposed MIDP and MILDP STN models. Two case studies are then used to illustrate the performance of the proposed set-based formulations. Concluding remarks and future work are provided at the end.

BACKGROUND

Background on STN MIP modeling is provided first, followed by background on set-based modeling.

Mixed-integer program (MIP)

Optimal STN scheduling relies on four key fixed sets that are used to define a problem: tasks $i \in I$ (e.g., $i \in$ {manufacture product A, separate A from B, ...}); units $j \in J$ available to perform these tasks (e.g., $j \in$ {reactor 1, reactor 2, distillation column, ...}); states $k \in K$ that act as initial, intermediate or final storage defining the feasible interconnection between tasks, and a scheduling horizon that is discretized into $n_T + 1$ time points $t \in T = \{0, 1, 2, \dots, n_T\}$. Based on these four sets, additional fixed sets are used to specify the interconnections between tasks, units and states, namely

$$IJ^{\times} = \{(i,j) \in I \times J |\ i \text{ can be performed in } j\}, \quad (1)$$

$$IK^{+} = \{(i,k) \in I \times K |\ i \text{ feeds } k\}, \quad (2)$$

$$IK^{-} = \{(i,k) \in I \times K |\ k \text{ feeds } i\}. \quad (3)$$

Generally, two variable types need to be optimized in a STN problem: discrete timing variables and continuous batch size variables. In a traditional MIP model, timing variables $x_{i,j,t} \in \{0,1\}$ decide if task i is executed in unit j starting at time t. Non-overlapping constraints are needed to ensure that a given unit can only perform one task at a time, i.e.,

$$\sum_{(i,j)\in IJ^{\times}} \sum_{t-\tau_{i,j}+1 \le t' \le t} x_{i,j,t'} \le 1, \forall j \in J, \forall t \in T, \quad (4)$$

where $\tau_{i,j}$ is the number of discrete time points needed to execute task i in unit j, i.e., a discrete processing time.

Batch size variables $b_{i,j,t} \in [0, \beta_j^{max}] \subset \mathbb{R}$ define the amount of material processed in task i by unit j at time t, where $\beta_{i,j}^{max}$ and $\beta_{i,j}^{min}$ are the fixed maximum and minimum capacities of unit j when executing task i. Variable $b_{i,j,t}$ is bounded by $\beta_{i,j}^{max}$ and $\beta_{i,j}^{min}$ when a task is executed, as defined by the following capacity constraints:

$$\beta_{i,j}^{min} x_{i,j,t} \le b_{i,j,t} \le \beta_{i,j}^{max} x_{i,j,t}, \forall (i,j) \in IJ^{\times}, \forall t \in T. \quad (5)$$

STN problems stand out due to the consideration of states k, representing storage facilities in which incoming materials from previous task executions can be stored until they are released to subsequent tasks. This means that there is a mass balance inherently embedded within the scheduling problem. These mass balances are needed to ensure that, at a given time point, enough materials are available for the execution of subsequent tasks and storage facilities do not overflow. Mathematically, storage modeling requires intermediate storage states $s_{k,t} \in \mathbb{R}$ to define the mass balance constraints in (6) and (7), as well as the state tracking constraints in (8).

$$s_{k,0} = \gamma_k^{init} - \sum_{\substack{(i,j)\in IJ^{\times} \\ (i,k)\in IK^{-}}} \rho_{i,k}^{-} b_{i,j,0}, \forall k \in K, \quad (6)$$

$$s_{k,t} = s_{k,t-1} + \sum_{\substack{(i,j)\in IJ^{\times} \\ (i,k)\in IK^{+}, t-\tau_{i,j} \ge 0}} \rho_{i,k}^{+} b_{i,j,t-\tau_{i,j}} - \sum_{\substack{(i,j)\in IJ^{\times} \\ (i,k)\in IK^{-}}} \rho_{i,k}^{-} b_{i,j,t}, \forall k \in K, \forall t \in T \backslash \{0\}, \quad (7)$$

$$\gamma_k^{min} \le s_{k,t} \le \gamma_k^{max}, \forall k \in K, \forall t \in T. \quad (8)$$

These constraints introduce parameter γ_k^{init} denoting the initial storage in state k; the maximum (γ_k^{max}) and minimum (γ_k^{min}) storage capacity of state k, and the fraction of material in state k consumed ($\rho_{i,k}^{-}$) or produced ($\rho_{i,k}^{+}$) by task i.

A common completion constraint is that all executed tasks must have finished by the end of the scheduling horizon, which can be expressed as

$$x_{i,j,t} = 0, \forall (i,j) \in IJ^{\times}, \forall t \in \{n_T - \tau_{i,j} + 1, \dots, n_T\}. \quad (9)$$

The constraints in (4)-(9) are standardly used to model STN problems, however, additional requirements are often imposed depending on specific application

requirements. Other modeling features considered in the case study section include constraint (10) to ensure that every task is executed at most once, or constraint (11) to ensure that the number of times task i is executed in unit j matches integer batching parameter $\psi_{i,j}$.

$$\sum_{\substack{j\in J\\(i,j)\in IJ^{\times}}}\sum_{t\in T}x_{i,j,t}\leq 1,\forall i\in I,\forall t\in T, \tag{10}$$

$$\sum_{t\in T}x_{i,j,t}=\psi_{i,j},\forall(i,j)\in IJ^{\times}. \tag{11}$$

Set-based modeling

Set-based modeling relies on three variable types that are not used in MINLP or GDP modeling defined as:

Interval variables

> Let $n,m\in\mathbb{Z}$ be fixed numbers with $m\geq n$ and $(\!(n,m)\!)=\{(n',m')\in\mathbb{Z}^2|\ n'\geq n,m'\leq m,m'\geq n'\}$. We define an *interval variable* $(x,y)\in(\!(n,m)\!)$, i.e., (x,y) can take tuples in $(\!(n,m)\!)$ as values.

For instance, take $n=0$ and $m=2$. Then, it is possible to define interval variable $(x,y)\in(\!(n,m)\!)$, where $(\!(n,m)\!)$ is equal to $\{(0,1),(0,2),(1,2),(0,0),(1,1),(2,2)\}$. Note that interval variables are herein defined as tuples, but they resemble intervals in the sense that the second entry y (the "end") is always greater than or equal to the first entry x (the "start"). By convention, an interval variable is said to be empty when its start and end points match.

Set variables

> Consider some fixed set A and take $\mathbb{P}(A)$ as the power set of A, i.e., the set of subsets of A. We define a *set variable* $X\in\mathbb{P}(A)$, i.e., X can take sets in $\mathbb{P}(A)$ as values.

For example, take $A=\{0,1\}$. Then, it is possible to define set variable $X\in\mathbb{P}(A)$ with $\mathbb{P}(A)=\{\{0\},\{0,1\},\{1\},\{\}\}$. Note that sets $\{0,1\}$ and $\{1,0\}$ are the same, i.e., set variables do not distinguish between different orderings of elements. List variables are similar to set variables, except that these account for permutations.

List variables

> Consider some fixed set A and its power set $\mathbb{P}(A)$. Also, take $\mathbb{S}(A)=\bigcup_{X\in\mathbb{P}(A)}\mathcal{S}(X)$, where $\mathcal{S}(X)$ is the set of lists obtained by permuting the elements of X. We define a *list variable* $y\in\mathbb{S}(A)$, i.e., y can take lists in $\mathbb{S}(A)$ as values.

Again, take $A=\{0,1\}$ as an example. Then, it is possible to define list variable $y\in\mathbb{S}(A)$ with $\mathbb{S}(A)=\{\langle 0\rangle,\langle 0,1\rangle,\langle 1,0\rangle,\langle 1\rangle,\langle\rangle\}$. Note that set and list variables may take the empty set $\{\}$ or empty list $\langle\rangle$ as valid values.

In the previous definitions, A is called a "fixed set" in the sense that it is not a set variable. In fact, Hexaly only supports A being a finite range set of the form $\{0,1,2,\ldots,n-1\}$. However, Hexaly supports establishing one-to-one relationships between A and other sets of the same size, even if they contain variables, e.g., a one-to-one function between A and a set of interval variables that needs to be shuffled through a list variable. For notation simplicity, this work avoids one-to-one functions and allows A being a finite set of variables explicitly.

MATHEMATICAL MODELS

In this section, we introduce the constraints needed to formulate MIDP and MILDP set-based STN models.

Mixed-interval disjunctive program (MIDP)

We propose a STN MIDP model that replaces timing variables $x_{i,j,t}\in\{0,1\}$ by timing interval variables. To do that, we define execution subindex $q\in Q_{i,j}$, where $Q_{i,j}$ is the fixed set of all potential executions of task i in unit j defined as

$$Q_{i,j}=\{1,2,\ldots,\lfloor n_T/\tau_{i,j}\rfloor\},\forall(i,j)\in IJ^{\times}, \tag{12}$$

with $\lfloor\bullet\rfloor$ being the floor operator. This allows to define interval variables $(\mathfrak{x}_{i,j,q}^{start},\mathfrak{x}_{i,j,q}^{end})\in(\!(0,n_T)\!),\forall(i,j)\in IJ^{\times},\forall q\in Q_{i,j}$. Interval $(\mathfrak{x}_{i,j,q}^{start},\mathfrak{x}_{i,j,q}^{end})$ represents the potential execution q of task i in unit j, starting at time point $\mathfrak{x}_{i,j,q}^{start}$ (inclusive) and finishing at time point $\mathfrak{x}_{i,j,q}^{end}$ (exclusive).

For some $(i,j)\in IJ^{\times}$, not every interval $q\in Q_{i,j}$ is necessarily going to be executed. Therefore, we use the convention that the length of $(\mathfrak{x}_{i,j,q}^{start},\mathfrak{x}_{i,j,q}^{end})$ is 0 (i.e., $\mathfrak{x}_{i,j,q}^{end}-\mathfrak{x}_{i,j,q}^{start}=0$) when interval q is not executed. Otherwise, the interval is executed and its length is equal to the processing time of the task-unit pair $(\mathfrak{x}_{i,j,q}^{end}-\mathfrak{x}_{i,j,q}^{start}=\tau_{i,j})$, i.e., we impose the following interval presence constraints:

$$\left[\mathfrak{x}_{i,j,q}^{end}-\mathfrak{x}_{i,j,q}^{start}=\tau_{i,j}\right]\vee\left[\mathfrak{x}_{i,j,q}^{end}-\mathfrak{x}_{i,j,q}^{start}=0\right],$$
$$\forall(i,j)\in IJ^{\times},\forall q\in Q_{i,j}, \tag{13}$$

where $\vee$ is the or operator. Now, we introduce a new operator defined as $\mathbf{if}(\Omega)=\{1\text{ if }\Omega=\text{True},0\text{ if }\Omega=\text{False}\}$, where Ω is a logical condition. Hexaly supports this operator for a variety of logical conditions. This operator allows us to define non-overlapping constraints as

$$\sum_{\substack{i\in I\\(i,j)\in IJ^{\times}}}\sum_{q\in Q_{i,j}}\mathbf{if}\left(t\geq\mathfrak{x}_{i,j,q}^{start}\right)-\mathbf{if}\left(t\geq\mathfrak{x}_{i,j,q}^{end}\right)\leq 1,$$
$$\forall j\in J,\forall t\in T. \tag{14}$$

Next, we introduce MIDP capacity constraints as

$$\mathbf{if}\left(\bigvee_{q\in Q_{i,j}}\begin{bmatrix}\mathfrak{x}_{i,j,q}^{start}=t\\ \mathfrak{x}_{i,j,q}^{end}-\mathfrak{x}_{i,j,q}^{start}\neq 0\end{bmatrix}\right)\beta_{i,j}^{min}\leq b_{i,j,t},$$

$$b_{i,j,t} \leq \mathbf{if}\left(\bigvee_{q\in Q_{i,j}}\begin{bmatrix}\mathfrak{x}^{start}_{i,j,q} = t\\ \mathfrak{x}^{end}_{i,j,q} - \mathfrak{x}^{start}_{i,j,q} \neq 0\end{bmatrix}\right)\beta^{max}_{i,j},$$

$$\forall (i,j) \in IJ^{\times}, \forall t \in T, \tag{15}$$

which link timing interval variables with continuous batch size variables that remain unchanged with respect to the traditional MIP STN model. Note that $x_{i,j,t}$ is equivalent to the **if** operator multiplying $\beta^{min}_{i,j}$ and $\beta^{max}_{i,j}$ in (15), i.e., if $x_{i,j,t} = 1$, then there is some interval $q \in Q_{i,j}$ of non-zero length starting at time t.

Since batch size variables $b_{i,j,t}$ remain unchanged in this formulation, mass balance and state tracking constraints (6)-(8) remain unchanged with respect to the MIP formulation. Furthermore, MIP completion constraints in (9) are readily enforced by $(\mathfrak{x}^{start}_{i,j,q}, \mathfrak{x}^{end}_{i,j,q}) \in (\!(0, n_T)\!)$, i.e., the end of an interval $\mathfrak{x}^{end}_{i,j,q}$ is bounded above by n_T. In summary, MIP constraints (4), (5) and (9) are the counterpart of MIDP constraints (13)-(15), while both MIP and MIDP share constraints (6)-(8).

After defining the main components of the MIDP STN model, we now examine how the additional MIP modeling features in (10) and (11) can be incorporated into the MIDP formulation. An equivalent version of constraint (10) in the MIDP model to ensure that each task is executed at most once is of the form

$$\sum_{\substack{I\in I\\ (i,j)\in IJ^{\times}}} \sum_{q\in Q_{i,j}} \mathbf{if}\left(\mathfrak{x}^{end}_{i,j,q} - \mathfrak{x}^{start}_{i,j,q} > 0\right) \leq 1, \forall i \in I. \tag{16}$$

Regarding constraint (11), major modifications are needed if the number of times task i is executed in unit j is known and equal to $\psi_{i,j}$. Firstly, set $Q_{i,j}$ in (12) must be redefined as

$$Q_{i,j} = \{1, 2, \dots, \psi_{i,j}\}, \forall (i,j) \in IJ^{\times}, \tag{17}$$

So that only $\psi_{i,j}$ intervals are considered in the formulation. Secondly, these intervals must be scheduled, meaning that interval presence constraints in (13) simplify to the following interval length constraints:

$$\mathfrak{x}^{end}_{i,j,q} - \mathfrak{x}^{start}_{i,j,q} = \tau_{i,j}, \forall (i,j) \in IJ^{\times}, \forall q \in Q_{i,j}. \tag{18}$$

Lastly, all intervals have length $\tau_{i,j}$, thus the condition within the **if** operator in (15) simplifies and capacity constraints reduce to

$$\mathbf{if}\left(\bigvee_{q\in Q_{i,j}} \left[\mathfrak{x}^{start}_{i,j,q} = t\right]\right)\beta^{min}_{i,j} \leq b_{i,j,t},$$

$$b_{i,j,t} \leq \mathbf{if}\left(\bigvee_{q\in Q_{i,j}} \left[\mathfrak{x}^{start}_{i,j,q} = t\right]\right)\beta^{max}_{i,j},$$

$$\forall (i,j) \in IJ^{\times}, \forall t \in T. \tag{19}$$

Mixed-interval-list disjunctive program (MILDP)

The interval variables from the MIDP formulation can be included in sets that define list variables, so that some constraints of the MIDP formulation can be reformulated in terms of list variables in a MILDP model. To do that, we create the following sets with $Q_{i,j}$ defined in (12):

$$A_j = \left\{\left(\mathfrak{x}^{start}_{i,j,q}, \mathfrak{x}^{end}_{i,j,q}\right) \in (\!(0, n_T)\!) \,|(i,j) \in IJ^{\times}, q \in Q_{i,j}\right\},$$

$$\forall j \in J, \tag{20}$$

which for every unit j contain all the intervals (i.e., tasks) that can be potentially performed in that unit. To decide which intervals are going to be scheduled and their order, we define list variables $y_j \in \mathbb{S}(A_j)$. We denote $y_j = \langle y_j(1), y_j(2), \dots, y_j(|y_j|)\rangle$, where the components $y_j(p), \forall p \in \{1, 2, \dots, |y_j|\}$ provide ordered access to the individual intervals in y_j. With this new variable, interval presence constraints take the following form:

$$\mathfrak{x}^{end}_{i,j,q} - \mathfrak{x}^{start}_{i,j,q} = \mathbf{if}\left(\left(\mathfrak{x}^{start}_{i,j,q}, \mathfrak{x}^{end}_{i,j,q}\right) \in y_j\right)\tau_{i,j},$$

$$\forall (i,j) \in IJ^{\times}, \forall q \in Q_{i,j}. \tag{21}$$

Also, non-overlapping constraints can now be compactly formulated as

$$y_j(p) < y_j(p-1), \forall p \in \{2, 3, \dots, |y_j|\}, \forall j \in J, \tag{22}$$

where the strict $<$ constraint between two intervals means that the second entry of the right-hand-side interval tuple is less than or equal to first entry of the left-hand-side interval tuple. Note that having $|y_j|$ as variable in (22) is an accepted cardinality expression in Hexaly's set-based modeling platform.

Capacity constraints remain unchanged with respect to those defined for the MIDP model in (15). Also, mass balance and state tracking constraints (6)-(8) remain unchanged in the MILDP formulation. Similarly to the MIDP model, completion constraints of tasks within the scheduling horizon are not needed because these are readily satisfied by interval variables. In summary, MIP constraints (4), (5) and (9) are the counterpart of MILDP constraints (15), (21) and (22), while both MIP and MILDP share constraints (6)-(8).

Now that the main MILDP constraints have been defined, we explain how constraints (10) and (11) are translated from the MIP formulation to the MILDP model. On the one hand, constraint (10) takes the form of constraint (16) from the MIDP model. On the other hand, imposing a constraint equivalent to (11) in the MILDP model requires defining set $Q_{i,j}$ as in (17) instead of (12), imposing interval length constraint (18) instead of (21), imposing capacity constraints (19) instead of (15), and enforcing the following constraint on the size of lists:

$$|y_j| = \sum_{\substack{i\in I\\ (i,j)\in IJ^{\times}}} \psi_{i,j}, \forall j \in J. \tag{23}$$

CASE STUDY

We assess the performance of the proposed MIDP and MILDP formulations in comparison with the traditional MIP formulation through two case studies. The objective functions and parameters defining these case studies are introduced below.

Case Study 1

This case study was proposed by Hirn and Grapendal [15], who proposed a STN model for the optimal scheduling of electromagnetic compatibility tests with limited scheduling time and equipment. That is, 15 devices of different sizes are scheduled to be tested in 3 testing chambers with different capacities. For this case study, each task represents a device that needs to be tested once; therefore, special constraint (10) is enforced. Additionally, a profit maximization objective is considered. For the MIP formulation the objective function is defined as

$$\max \sum_{(i,j)\in IJ^{\times}} \sum_{t\in T} \upsilon_i x_{i,j,t}, \tag{24}$$

where υ_i is the fixed profit generated by one execution of task i, which readily considers the difference between revenue and task execution costs. Objective function (24) translates into

$$\max \sum_{(i,j)\in IJ^{\times}} \sum_{q\in Q_{i,j}} \upsilon_i \mathbf{if}\left(\mathfrak{t}_{i,j,q}^{end} - \mathfrak{t}_{i,j,q}^{start} > 0\right), \tag{25}$$

for set-based formulations. The resulting optimization problem is shown in Table 1, while all the fixed parameters are available as supplementary material (Appendix A).

Table 1: MIP, MIDP and MILDP problem formulations.

MIP	MIDP	MILDP
Eq. (24), s.t. Eqs. (4)-(10).	Eq. (25), s.t. Eqs. (6)-(8), (12)-(15), (16).	Eq. (25), s.t. Eqs. (6)-(8), (12), (15), (16), (21), (22).

Case Study 2

This case study was adapted from the work by Velez and Maravelias [16], who studied MIP formulations for long-term STN scheduling. It considers the well-studied Kondili et al. [2] STN, which includes one mixing task performed in a separation unit, three reaction tasks scheduled in two reactors and one separation task with its corresponding separation unit. This STN has three inputs for which purchasing costs are considered and two products that generate revenue. For this case study, it is assumed that the number of times each task is executed in each unit is known, thus, special constraint (11) is enforced. In this case, the profit maximization objective takes the following form:

$$\max \sum_{k\in K} \omega_k s_{k,n_T} - \sum_{(i,j)\in IJ^{\times}} \alpha_{i,j} \psi_{i,j}, \tag{26}$$

where ω_k is the revenue from selling the material in state k and $\alpha_{i,j}$ is the cost of running task i in unit j. Altogether, the optimization problem considered is summarized in Table 2, while all the fixed parameters are available in the supplementary material (Appendix B).

Table 2: MIP, MIDP and MILDP problem formulations.

MIP	MIDP	MILDP
Eq. (26), s.t. Eqs. (4)-(9), (11).	Eq. (26), s.t. Eqs. (6)-(8), (14), (17)-(19).	Eq. (26), s.t. Eqs. (6)-(8), (17)-(19), (22), (23).

RESULTS AND DISCUSSION

This section evaluates the performance of the MIP, MIDP, and MILDP models by solving multiple instances of case studies 1 and 2 across varying levels of complexity. All optimization runs were performed in the Python API of Hexaly 14.0 with a time limit of 3600 seconds on a 1.7 GHz, 64 GB, Intel Core® Ultra™ 7 165U, Dell Inc. Latitude 7350 laptop.

Case Study 1

Different instances of this case study were generated by testing different time horizons n_T from 5 to 17, while keeping the scheduling time step fixed. Since tasks can be executed only once, increasing n_T results in increased flexibility to accommodate more tasks within the scheduling horizon. The performance plots obtained with solution time and objective gap as performance metrics are shown in Figures 1 and 2, respectively.

In the y-axis, Figure 1 shows the fraction of problems solved to global optimality within the solution time shown in the x-axis. Similarly, Figure 2 shows the fraction of problems solved to either feasibility or global optimality within the percentual objective gap in the x-axis. From Figure 1, the MIDP formulation clearly outperforms the MIP and MILDP formulations for this case study. Within the time limit, the MIDP formulation can solve 11 out of the 13 tested instances to global optimality. On the contrary, the MIP and MILDP formulations are unable to close the objective gap to satisfactory levels (see Figure 2).

Figure 2 shows that the MILDP model achieves better effectiveness at closing the optimality gap reported by the solver compared to the standard MIP, despite not performing as well as the MIDP formulation. Furthermore, all formulations reached the same primal objective within the imposed time limit. This outcome suggests that the set-based formulations used enable Hexaly's optimization algorithm to compute tighter objective lower bounds for this test problem. Based on publicly available documentation, the observed increased performance of set-

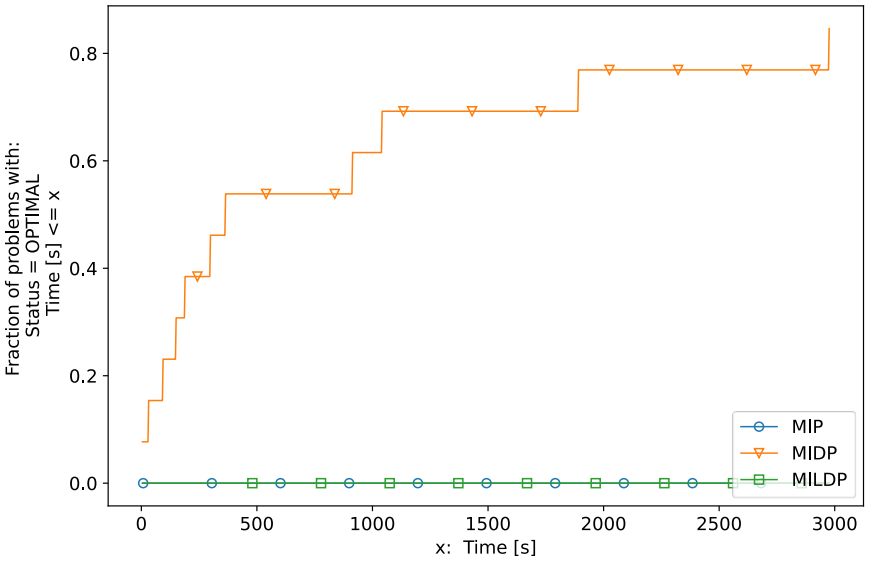

Figure 1. Solution time performance profile for case 1.

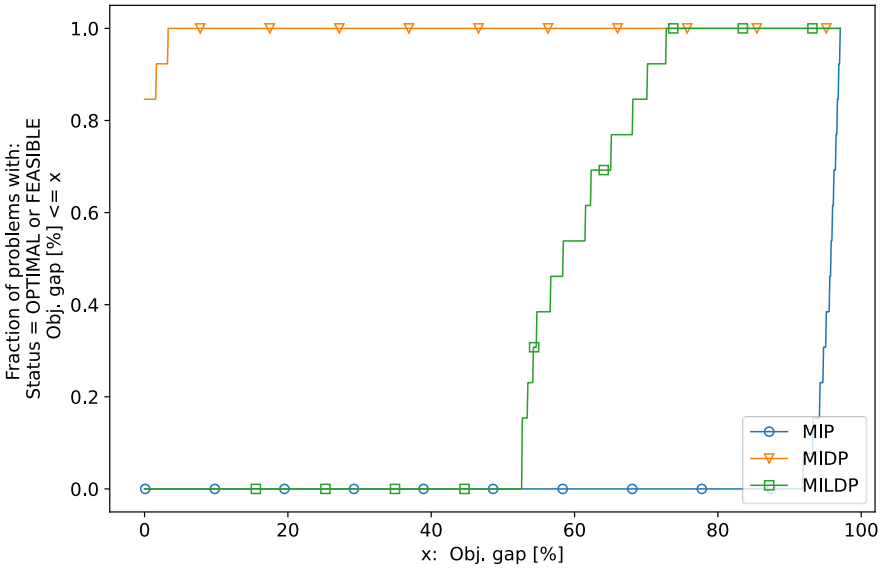

Figure 2. Objective gap performance profile for case 1.

based formulations at solving the dual problem and finding better objective bounds is attributed to a combination of column generation, column elimination and/or branch-cut-price strategies tailored to list and interval variables. Particularly, the energetic relaxation implemented in Hexaly has been shown to be effective for deriving lower bounds in scheduling problems [17].

The results obtained in this work align with the previous findings by Hirn and Grapendal [15], who highlighted how increasing the time horizon affects the number of nodes that a standard MIP algorithm (GLPK) must explore to close the optimality gap. As the time horizon n_T increases, more devices can be scheduled, which adds combinatorial complexity and forces GLPK to move from exploring roughly 1 node to about 1×10^7 nodes before finding the optimal solution, until all devices can finally be scheduled at n_T=16. This is in line with the best-performing set-based model (MIDP), which only struggles to close the optimality gap for the instances with n_T=14 and n_T=15, reporting gaps of 3.21% and 1.61%, respectively. In contrast, the MIP formulation solved with Hexaly consistently reports optimality gaps larger than 90%, further highlighting the computational benefits of the proposed MIDP model.

Case Study 2

A total of 20 different instances of this case study were generated by adjusting the fineness (or accuracy)

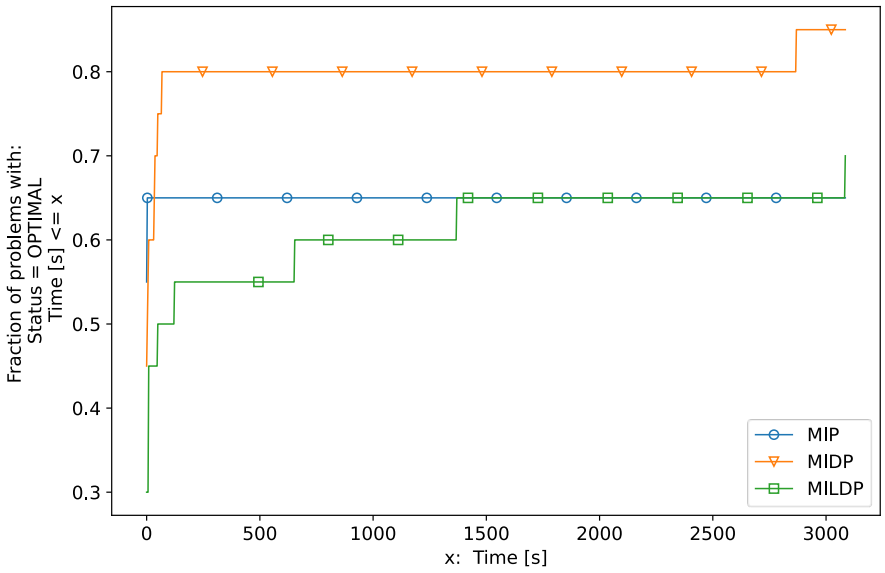

Figure 3. Solution time performance profile for case 2.

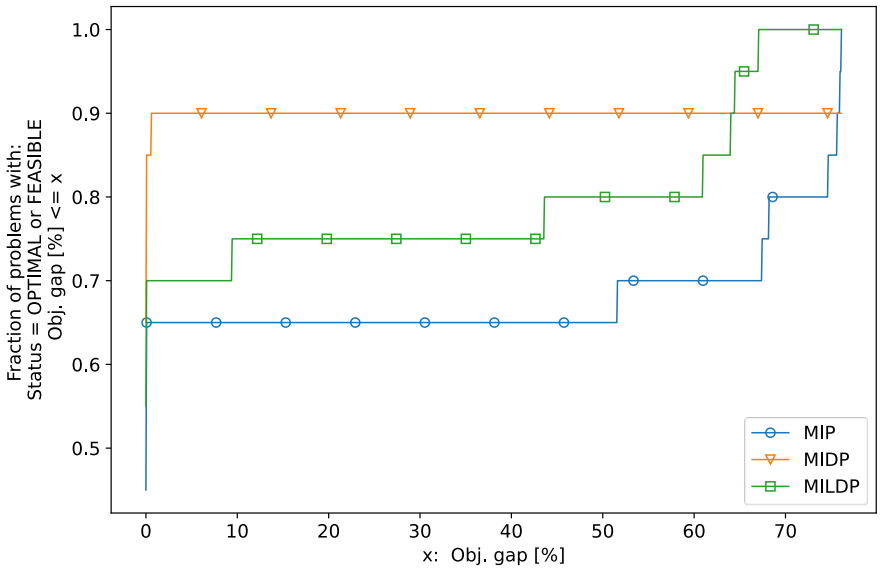

Figure 4. Objective gap performance profile for case 2.

for time discretization, ranging from 2 (i.e., $n_T = 1$), to 121 (i.e., $n_T = 120$) discretization points. The remaining parameters of the formulation were adjusted to maintain a physical scheduling horizon of 120 hours. The performance plots obtained with solution time and objective gap as performance metrics are shown in Figures 3 and 4, respectively.

Figure 3 illustrates improvements in the performance of the MIP formulation relative to the previous case study. Specifically, the MIP formulation outperforms the MILDP formulation for solution times below 1500 seconds. However, at higher time limits the MILDP model exhibits a slight increase in performance, solving one additional instance to global optimality within the 3600 seconds limit. In contrast, the MIDP formulation outperforms both the MIP and MILDP models, solving more than 80% of the test problems to global optimality.

Similarly to the previous case study, both the MIDP and MILDP models outperform the original MIP model in closing the objective gap (see Figure 4). However, not all formulations reach the same primal solution in this case study. The MILDP model consistently identified the best-known primal solution across all instances. This indicates that the advantage of set-based formulations arises not only from dual strategies that yield tighter objective lower bounds, but also from primal heuristics that quickly generate high-quality feasible solutions. Specifically, Hexaly leverages local search and large neighborhood search

strategies that operate directly on interval and list variable domains, eliminating the need for binary reformulations. For example, consider a task-unit pair (i, j). The MIDP and MILDP formulations avoid introducing one binary variable $x_{i,j,t}$ for each time point $t \in \{0, 1, 2, \ldots, n_T\}$, and instead replace them with $\psi_{i,j}$ interval variables, where $\psi_{i,j}$ is a fixed integer. In terms of variable reduction, the best-case scenario in the set-based formulations compared to the MIP formulation would occur when $\psi_{i,j} = 1$ and $n_T = 120$. In that case, 121 binary variables in the traditional MIP formulation are replaced by a single interval variable in the MIDP and MILDP formulations. At solving time, Hexaly does not reformulate interval and list variables as binary variables, but applies powerful primal heuristics combined with exact algorithms to the set-based representation directly. Thus, this study shows that exploiting set-based formulations rather than relying on standard binary variable models enables the primal solver within Hexaly to scale better with large scheduling instances and get good quality solutions faster.

The results discussed through this and the previous section highlight instances where at least one of the proposed MIDP and/or MILDP formulations outperform a traditional MIP formulation when using Hexaly. Nevertheless, set-based formulations are not always expected to bring computational improvements. To illustrate this, we have performed a computational experiment that is a modified version of the current case study, with the only difference being that integer batching variables are now left unfixed, i.e., constraint (11) is removed from the MIP. In the MIDP formulation, this translates to using definition (12) instead of (17), constraint (13) instead of (18), and constraint (15) instead of (19). Similarly, the MILDP formulation needs to be modified by using definition (12) instead of (17), constraint (21) instead of (18), constraint (15) instead of (19), and removing constraint (23).

Assuming a 300-second time limit, the solving-time performance profiles of this modified case study are shown in Figure 5. Recall that, within 300 seconds, the MIDP formulation solves 80% of the test problems to optimality, compared to 65% solved by the MIP model in the original version of this case study (see Figure 3). In this modified version, the MIP performance-profile line lies above both the MIDP and MILDP lines in Figure 5, indicating that the MIP formulation is now outperforming the set-based formulations, e.g., the MIP formulation solves 55% of the tested instances to optimality within 40 seconds, compared to 25% solved to optimality by the MIDP formulation. These results highlight that STN optimal scheduling benefits from the proposed MIDP and MILDP formulations when dealing with problems with fixed integer batching variables; however, this outcome does not generalize to problems with unfixed integer batching variables for the case study under consideration.

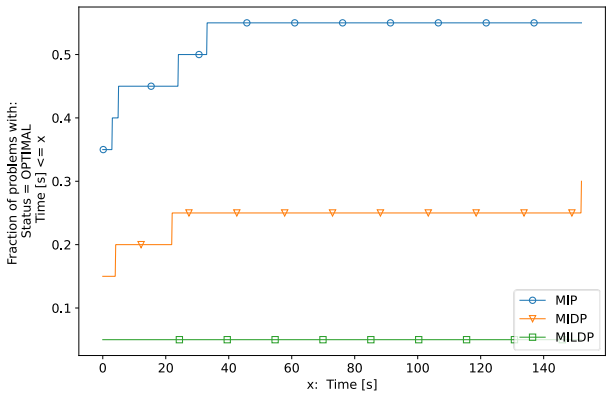

Figure 5. Solution time performance profile for the modified version of case 2.

CONCLUSIONS

The optimal STN scheduling problem has traditionally been addressed through standard MIP formulations and classical solution algorithms. In contrast, this work introduced the first set-based STN models, opening a new avenue for research in optimal network scheduling. Documentation suggests Hexaly performs well for specific classes of industrial-scale problems, such as scheduling and vehicle routing, precisely because it exploits this set-based modeling paradigm effectively. We have shown promising results for determining a strong set-based model for representing and solving STN formulations using Hexaly, namely a mixed-interval disjunctive programming (MIDP) formulation.

As future work, we aim to continue refining and studying the proposed set-based formulations. Also, a systematic comparison of set-based, constraint programming, and MIP optimization beyond Hexaly and across multiple modeling and solution platforms was outside the scope of this work. The solver embedded in Hexaly is known to be competitive with commercial and open-source solvers such as Gurobi, CPLEX, HiGHS, and OR Tools, e.g., in job shop scheduling [18]. Consequently, a broader benchmarking study with a larger set of test instances is necessary to further clarify how STN scheduling benefits from the proposed set-based formulations.

DIGITAL SUPPLEMENTARY MATERIAL

A file containing the parameters needed to define the case studies and to replicate the sensitivity analyses shown in the result section are available at https://psecommunity.org/LAPSE:2026.0003. All results can be replicated by running the code available at https://psecommunity.org/LAPSE:2026.0002.

ACKNOWLEDGEMENTS

The authors would like to thank the Hexaly team for

their valuable feedback on the manuscript.

David A. Liñán and Jan Kronqvist acknowledge the financial support provided by LKAB and Digital Futures.

AUTHOR IDENTIFIERS

Author ORCIDs:
Liñán DA: 0000-0002-3190-7612
Stinchfield G: 0009-0004-5130-5661
Kronqvist J: 0000-0003-0299-5745

REFERENCES

1. Agnetis A, Billaut JC, Pinedo M, Shabtay D. Fifty years of research in scheduling — theory and applications. European Journal of Operational Research 327:367-393 (2025). https://doi.org/10.1016/j.ejor.2025.01.034
2. Kondili E, Pantelides CC, Sargent RWH. A general algorithm for short-term scheduling of batch operations—i. MILP formulation. Computers & Chemical Engineering 17:211-227 (1993). https://doi.org/10.1016/0098-1354(93)80015-f
3. C.C. Pantelides. Unified frameworks for optimal process planning and scheduling. In: Proceedings of the Second Conference on Foundations of Computer Aided Operations. CACHE (1994)
4. Maravelias CT. Chemical production scheduling. Cambridge University Press (2021). https://doi.org/10.1017/9781316650998
5. Méndez CA, Cerdá J, Grossmann IE, Harjunkoski I, Fahl M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers & Chemical Engineering 30:913-946 (2006). https://doi.org/10.1016/j.compchemeng.2006.02.008
6. Samadi A, Maravelias CT. A comprehensive chemical production scheduling representation. Computers & Chemical Engineering 181:108552 (2024). https://doi.org/10.1016/j.compchemeng.2023.108552
7. Lekkerkerk K. Scheduling optimization in a refinery for vegetable oils. TU Delft (2020)
8. Moniz S, Barbosa-Póvoa AP, de Sousa JP. Simultaneous regular and non-regular production scheduling of multipurpose batch plants: a real chemical–pharmaceutical case study. Computers & Chemical Engineering 67:83-102 (2014). https://doi.org/10.1016/j.compchemeng.2014.03.017
9. Huang Y, Su R, Yang J, Li J, Du Z, Huang Z. An intelligent electricity scheduling model for cement plants based on STN. 2024 IEEE 2nd International Conference on Power Science and Technology (ICPST) :618-624 (2024). https://doi.org/10.1109/icpst61417.2024.10602163
10. Georgiadis GP, Elekidis AP, Georgiadis MC. Optimization-based scheduling for the process industries: from theory to real-life industrial applications. Processes 7:438 (2019). https://doi.org/10.3390/pr7070438
11. Perez HD, Amaran S, Iyer SS, Wassick JM, Grossmann IE. Applications of the RTN scheduling model in the chemical industry. Simulation and Optimization in Process Engineering :365-400 (2022). https://doi.org/10.1016/b978-0-323-85043-8.00006-4
12. Mostafaei H, Harjunkoski I. Continuous?time scheduling formulation for multipurpose batch plants. AIChE Journal 66: (2019). https://doi.org/10.1002/aic.16804
13. Castro PM, Grossmann IE. Generalized disjunctive programming as a systematic modeling framework to derive scheduling formulations. Ind. Eng. Chem. Res. 51:5781-5792 (2012). https://doi.org/10.1021/ie2030486
14. Blaise L, Artigues C, Benoist T. Solution repair by inequality network propagation in localsolver. Lecture Notes in Computer Science :332-345 (2020). https://doi.org/10.1007/978-3-030-58112-1_23
15. Grapendal L, Hirn C. Mathematical Optimization of Scheduling Problems: A State-Task Network and Mixed Integer Linear Programming Approach. KTH (2025)
16. Velez S, Maravelias CT. Reformulations and branching methods for mixed-integer programming chemical production scheduling models. Ind. Eng. Chem. Res. 52:3832-3841 (2013). https://doi.org/10.1021/ie303421h
17. Laborie P. Bornes rapides pour l'ordonnancement dans LocalSolver. ROADEF (2023)
18. Abreu LR. Mixed-integer versus constraint programming solvers. Proceeding Series of the Brazilian Society of Computational and Applied Mathematics : (2026). https://doi.org/10.5540/03.2026.012.01.0308