Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal

Check for updates

Cite this: DOI: 10.1039/d4dd00327f

Schedule optimization for chemical library synthesis†

Qianxiang Ai, ‡[a] Fanwang Meng, ‡[a] Runzhong Wang,[a] J. Cullen Klein,[b] Alexander G. Godfrey [b] and Connor W. Coley *[ac]

Automated chemistry platforms hold the potential to enable large-scale organic synthesis campaigns, such as producing a library of compounds for biological evaluation. The efficiency of such platforms will depend on the schedule according to which the synthesis operations are executed. In this work, we study the scheduling problem for chemical library synthesis, where operations from interdependent synthetic routes are scheduled to minimize the makespan—the total duration of the synthesis campaign. We formalize this problem as a flexible job-shop scheduling problem with chemistry-relevant constraints in the form of a mixed integer linear program (MILP), which we then solve in order to design an optimized schedule. The scheduler's ability to produce valid, optimal schedules is demonstrated by 720 simulated scheduling instances for realistically accessible chemical libraries. Reductions in makespan up to 58%, with an average reduction of 20%, are observed compared to the baseline scheduling approach.

Received 15th October 2024
Accepted 15th December 2024

DOI: 10.1039/d4dd00327f

rsc.li/digitaldiscovery

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM. This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

1 Introduction

Automated synthetic chemistry has evolved from one-step synthesis of similar targets[1,2] to multi-step synthesis with chemically distinct reactions.[3–6] Recent studies have demonstrated the potential of automated synthesis modules in synthesis campaigns for a large set of targets, *i.e.*, syntheses of chemical libraries.[7–12] Computer-aided synthetic planning with the goal of automated execution must include not only the task of retrosynthetic analysis, but also the subsequent elaboration of abstract routes into specific, actionable hardware operations.[13] This elaboration requires high-level considerations such as spatial arrangement/connectivity of devices (hardware orchestration),[14] as well as methods to specify operations. The precise timing of these hardware operations directly impacts the total duration of a synthesis campaign. As a simple example, parallel synthesis in reaction vials using a multi-position dry block heater is likely to be more efficient than sequential synthesis if multiple reaction steps can be run with the same temperature setpoint. The impact of optimizing the schedule of operations can be significant as synthetic chemistry laboratories tend to be highly modular and comprise independent hardware modules that allow highly parallel execution. As the size of chemical libraries for synthesis grows, so does the number of requisite hardware operations and interdependencies, motivating the use of formal optimization techniques for time-efficient multi-target synthesis.

In examples of automated chemistry platforms, an operation scheduler is typically reported as a component of the overall platform manager rather than as a standalone, reusable module. Most of these schedulers use rule-based, heuristic approximation methods.[11,15–19] For example, Canty uses a greedy algorithm to arrange concurrent workflows on a molecular synthesis platform.[20] Simple prioritization policies are also popular choices for dynamic scheduling. The platform by Burger *et al.* uses an oldest-job-first policy,[21] and the laboratory automation manager by Delaney *et al.* adopts a shortest-job-first policy.[22] These simple scheduling policies, however, may not provide feasible solutions to comply with constraints that are common to chemical synthesis. A key constraint type is a time lag constraint that restricts the temporal gap between the end of an operation and the start of another. For example, if a stock solution of a homogeneous catalyst is prone to deactivation by oxidation, there is a maximum time lag between the preparation of this solution and its addition to the reactor. A shortest-job-first or an oldest-job-first policy does not guarantee that the shortest/oldest next operation would obey such a maximum time lag constraint. Aarts *et al.* propose a modified greedy algorithm to address time lag constraints among operations by rejecting infeasible schedules in local searches.[18] While these heuristic methods provide solutions with low computational complexity, the obtained solutions can be far from optimal. Few studies describe formal mathematical formulations of laboratory operation schedulers,[23–25] and, to the best of our knowledge,

[a]*Department of Chemical Engineering, MIT, Cambridge, MA, 02139, USA. E-mail: ccoley@mit.edu*

[b]*National Center for Advancing Translational Sciences, Rockville, MD, 20850, USA*

[c]*Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 02139, USA*

† Electronic supplementary information (ESI) available. See DOI: https://doi.org/10.1039/d4dd00327f

‡ Authors contributed equally.

© 2025 The Author(s). Published by the Royal Society of Chemistry

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM. This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

only the study by Itoh *et al.* achieves mathematically optimal solutions.[25] However, while Itoh *et al.* consider constraints on temporal boundaries of operations (time lags) for life science laboratory applications, the formulation still lacks other constraints that are important to generic library synthesis scenarios. For example, module capacity, as illustrated in the case of multi-position heaters, and work shifts, where some operations can only be performed in predefined time slots. It should also be noted that not all of these previous schedulers were designed for the synthesis of chemical libraries.

Chemical production scheduling has become an extensive research area with applications in a variety of industries, such as oil refining,[26] polymer production,[27] and ceramic production.[28] These scheduling problems are posed to optimize a given objective, most often the total execution time (makespan). As a special case of chemical production scheduling, library synthesis deals with small-scale batch reactors due to their flexibility in accommodating different synthetic routes. We identify library synthesis as exhibiting the following characteristics:

(1) Pre-planned batches: while materials merging and splitting are allowed in a generic campaign, the quantities of materials involved in all operations are planned ahead of operation scheduling. This means batch quantities are pre-calculated, and the number of batches/operations is fixed.

(2) Flexible assignment: an operation can be processed by one or more hardware modules in the laboratory, and the assignment of operations to hardware modules is a sub-problem of the scheduling problem.

(3) Interdependent synthetic routes: the reactions to afford a general set of target chemicals can have arbitrary interdependencies. This often originates from shared intermediate products for multiple downstream reactions. Alternative scenarios include precedence relations imposed by functional modules, such as flow systems with fixed operation orders. This thus requires the scheduling algorithm to model arbitrary precedence relations among operations;

(4) Laboratory constraints: the operations are commonly subject to temporal constraints imposed by materials, hardware, and operators. Constraints of this type include time lags, hardware capacity, and operator shifts as mentioned earlier.

Following the classification by Maravelias,[29] the library synthesis scheduling problem falls into the category of flexible job-shop scheduling problem (FJSP) due to characteristics 1 and 2. The term "flexible" indicates that a specific operation may be processed by one of many machines. FJSP was first introduced by Brucker and Schlie in 1990 to model linear operation sequences in multipurpose facility environments,[30] and is at least as hard as the NP-complete job-shop scheduling problem.[31,32] FJSP has since been extended to include arbitrary inter-operation precedence relations,[33,34] sometimes referred to as the extended FJSP,[35,36] which maps to the dependent synthetic routes issue in library synthesis (characteristics 3). Previous studies have further generalized extended FJSP to include additional hard constraints. For example, Pawar and Bhosale investigated a FJSP in the press working industry with machine capacity constraints.[37] Braune and Doerner focus on scheduling workflows in chemical research laboratories with maximum time lag constraints.[38] Boyer *et al.* introduced "generalized FJSP" to include machine setup time and holding time in addition to aforementioned hard constraints.[39] While these previous studies include constraints that could partially map onto the constraints in library synthesis, none of these previous studies fully covers the types of constraints that are commonly seen in organic synthesis (Table 1). We found the study by Boyer *et al.*, despite being designed for ring rolling processes, most comprehensive in terms of constraint types, and their formulation is modified to accommodate library synthesis scheduling in this study (Section 2.2).

In this study, we report an open-source library synthesis scheduler along with visualization modules (Fig. 1). Our formulation optimizes the makespan of arbitrary batch chemistry synthetic routes while respecting common laboratory-relevant constraints, including minimum/maximum lag time, hardware capacity, and work shifts. Our workflow produces optimal schedules for library synthesis by representing synthetic routes (proposed by predictive chemistry models) as reaction and operation networks. We demonstrate the scheduler on exemplary real-world chemical libraries, including a library for cannabinoid receptors[40] and a library of FDA-approved drugs.[41]

2 Methods

2.1 Problem definition

We focus on minimizing the time (makespan) required to synthesize a target chemical library, given the specifications of available hardware units. If the temporal cost of each operation can be reliably estimated, the library synthesis scheduling problem can then be treated as a flexible job-shop scheduling problem (FJSP), where the makespan is optimized as a function of both the operation-to-module assignment and the start times of operations.

Table 1 Components and constraints covered by previous scheduling studies. The last column shows if an optimal schedule is guaranteed

Reference	Application	Arbitrary precedence	Module assignment	Module capacity	Time lag	Work shift	Optimal schedule
Aarts *et al.*[18]	Chemical synthesis				✓		
Braune and Doerner[38]	Molding process	✓	✓	✓	✓		✓
Itoh *et al.*[25]	Biochemical analysis	✓	✓				✓
Boyer *et al.*[39]	Ring rolling	✓	✓	✓	✓		✓
Pawar and Bhosale[37]	Press working		✓	✓			✓
This work	Chemical synthesis	✓	✓	✓	✓	✓	✓

© 2025 The Author(s). Published by the Royal Society of Chemistry

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM.
This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

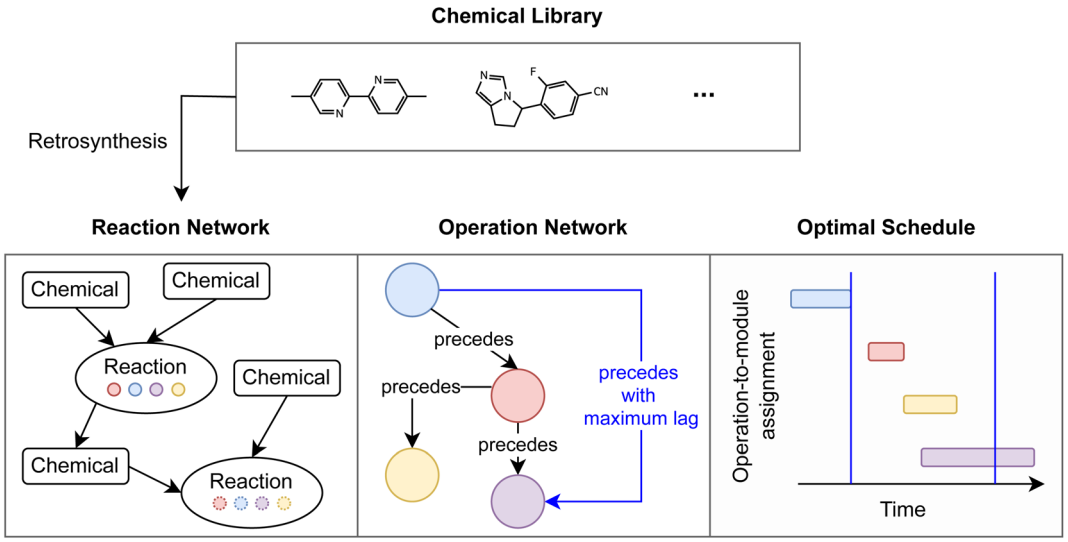

Fig. 1 Schematic representation of the workflow from real-world chemical libraries to the optimal schedule. Retrosynthesis is used to generate synthetic routes that are represented by a reaction network. This reaction network is then used to create an operation network including specific operations, constraints, and the precedence relations among them. The operation network constitutes the input to the scheduler to produce an optimal schedule.

2.1.1 Reaction network definition. We assume the synthesis routes to all target chemicals have already been determined prior to scheduling. These routes can be determined by expert chemists or predictive chemistry tools such as ASKCOS.[42] Algorithms such as SPARROW can be used in preprocessing to downselect routes, *e.g.*, to maximize common intermediates.[43] These routes are represented as a reaction network – a bipartite acyclic directed graph in which nodes are either chemicals or chemical reactions. The production or consumption of a chemical by a reaction is depicted by directed edges in the reaction network. We further assume the yields of all reactions in the reaction network can be conservatively estimated or are known. Given the desired quantities of target chemicals, the quantities of other chemicals are determined by traversing the reaction network in reverse. The quantities are often required to estimate operation processing time; for example, the duration of a solvent removal step depends on the amount of solvent.

2.1.2 Operation network definition. An operation is defined as a planned physical or chemical process that can be executed by one or more functional modules which are sets of hardware units. For example, the operation of solvent evaporation can be carried out using a rotary evaporator or a blow-down evaporator, assuming they are present in the laboratory. The flexibility in choosing which functional module to use for a given operation introduces an operation-to-module assignment problem, which is a part of library synthesis scheduling.

Each reaction in a reaction network consists of a predetermined set of operations which can be represented by a directed acyclic graph, the operation graph, where the nodes represent the operations of this reaction and edges represent the required precedence relations among the operations. We use the term "required" here to distinguish it from the implied precedence relations in an existing schedule. The required precedence relations of a chemical reaction sometimes lead to a linear sequence of operations (a path graph), such as the additions of reagents with a fixed addition order followed by a heating operation. In other cases, however, an operation can be required to precede, or to be preceded by, multiple other operations. For example, when the addition order of reagents is flexible, there is no required precedence relations among addition operations, and the heating operation becomes a node that has multiple incoming edges from the addition operation nodes in the operation graph of this reaction. Thus, in contrast to previous studies,[18] we refrain from making assumptions about the topology of operation graphs and consider them generic directed acyclic graphs.

We further assume that throughout the synthesis campaign, the set of functional modules is fixed, so no modules are added or removed from the laboratory. Given a reaction network and the operation graphs of its reactions, we can generate the operation network of this reaction network by connecting nodes from different operation graphs with directed (precedence) edges. Note these edges still represent required precedence relations among operations, thus the precedence of reaction A over reaction B in the reaction network does not imply all operations of reaction A must precede all operations of reaction B in the operation network. This network can be viewed as a State-Task Network[44] where chemicals and operations are states and tasks, respectively.

2.1.3 Mathematical formulation of the operation network. The input parameters of our scheduling problem are defined as follows.

(1) The operation network $\mathcal{O}$ defined by the set of operations $\mathfrak{O} = \{O_1, O_2, \ldots\}$ and the required precedence relations among them;

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM. This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

(2) The set of functional modules $\mathfrak{M} = \{M_1, M_2, \ldots\}$;
(3) The processing times

$$p_{\mathrm{im}} = \begin{cases} \text{the processing time of } O_i \text{ on } M_m,\ p_{\mathrm{im}} \in \mathbb{R}^+ \\ +\infty, \text{ if } O_i \text{ cannot be processed by } M_m. \end{cases} \quad (1)$$

Note that $\mathcal{O}$ includes the ordinary precedence constraints between two operations, *i.e.*, there is a path from O_i to O_j in $\mathcal{O} \Leftrightarrow$ the end time of O_i must precede the start time of O_j. The term "ordinary" distinguishes these constraints from other precedence constraints (*vide infra*).

2.1.4 Mathematical formulation of laboratory constraints. An important aspect of library synthesis scheduling, in addition to the ordinary precedence constraints among operations, is the set of constraints imposed by both reactions and the laboratory. In this study we introduce the following additional parameters to model common scenarios in library syntheses:

(1) $l\mathrm{min}_{ij} \in (-\infty, +\infty)$: the minimum lag time between the end of O_i and the start of O_j. This is used when the undetermined delay between two operations has a lower bound. For example, the solution prepared by an exothermic mixing process has to wait for more than a certain time before being used by a subsequent operation;

(2) $l\mathrm{max}_{ij} \in (-\infty, +\infty)$: the maximum lag time between the end of O_i and the start of O_j. This is used when the undetermined delay between two operations has an upper bound, frequently present in reaction input additions. For example, a freshly prepared Grignard reagent must be used within a short period of time before expiring if not stored under fully inert conditions;

(3) $K_m \in \mathbb{N}^+$: the module capacity of M_m. This allows efficient usage of functional modules. For example, a heater block with three slots can be used for three independent heating operations with similar set temperatures. For functional modules that can process only one operation at a time, their capacities are set to one;

(4) $C_{ij} \in \{0, 1\}$: the compatibility between O_i and O_j. This determines whether two operations can be processed simultaneously by the same functional module. For example, two heating operations cannot be processed concurrently on the same heater block if they require very different set temperatures.

(5) $S_n \in \mathbb{R}^+ \cap \{0\}$ and $E_n \in \mathbb{R}^+$: the start time and the end time of the nth work shift $W_n \in \mathfrak{W}$ during which the operations in $\mathfrak{P} \subseteq \mathfrak{O}$ are permitted to be processed. This is useful for scheduling operations with safety concerns, *e.g.*, operations involving elevated temperature/pressure would often require human supervision in the laboratory.

By combining $l\mathrm{min}_{ij}$ and $l\mathrm{max}_{ij}$, the precedence relations can be described in a more constrained manner than the ordinary precedence constraints in $\mathcal{O}$. For ordinary precedence constraints, we let $l\mathrm{min}_{ij} = 0$, $l\mathrm{max}_{ij} = +\infty$ if O_i is required to precede O_j, and $l\mathrm{min}_{ij} = -\infty$, $l\mathrm{max}_{ij} = +\infty$ otherwise. All input parameters are summarized in Table 2.

2.2 Mixed-integer linear programming (MILP) formulation

We solve the library synthesis scheduling problem as a mixed-integer linear program (MILP), which includes operation-to-module assignment, arbitrary precedence relations, and common laboratory constraints, as schematically shown in Fig. 2. The variables used in this formulation are defined in Table 3. x_{ijm} and y_{ijm} are indicators of whether one of $\{O_i, O_j\}$ precedes another and both are assigned to the M_m. z_{ijm} is the indicator of whether there is overlap between O_i and O_j on the same M_m. Y_{in} and Z_{in} are indicators describing temporal

Table 2 Input parameters of the scheduling problem

Parameter(s)	Definition
Operations	
$\mathcal{O}$	The operation network
$\mathfrak{O}$	The set of operations
$\mathfrak{P}$	The set of operations that require human supervision
i, j	Operation indices
O_i	The ith operation
$l\mathrm{min}_{ij}$	The minimum lag time between the end of O_i and the start of O_j
$l\mathrm{max}_{ij}$	The maximum lag time between the end of O_i and the start of O_j
C_{ij}	If O_i can be processed with O_j on the same functional module
Functional modules	
$\mathfrak{M}$	The set of functional modules
m	Function module index
M_m	The mth functional module
K_m	The capacity of M_m
p_{im}	The processing time of O_i on M_m
Work shifts	
$\mathfrak{W}$	The set of work shifts
n	Work shift index
W_n	The nth work shift
S_n	The start time of W_n
E_n	The end time of W_n

© 2025 The Author(s). Published by the Royal Society of Chemistry

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM.
This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

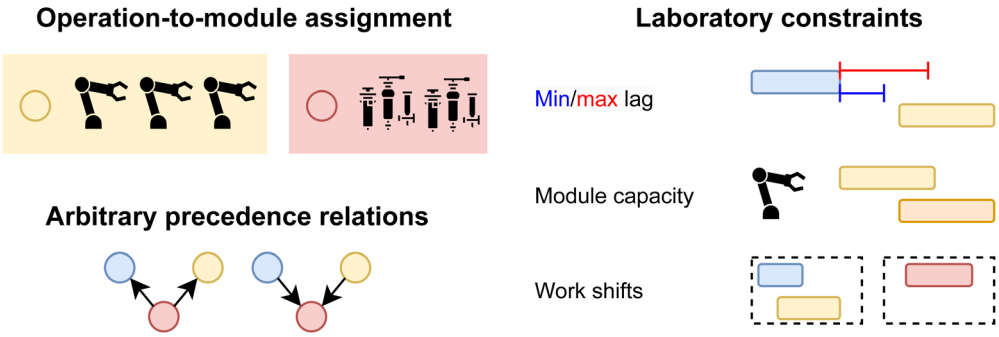

Fig. 2 Components of the library synthesis scheduler.

positions between the boundary of W_n and the start/end time of O_i. A_{in}, defined by Y_{in} and Z_{in}, is the indicator of whether O_i is assigned to the work shift W_n.

With the variables and the parameters defined in Table 2, we now introduce the MILP formulation of the library synthesis scheduling problem. Eqn (2) and (3) define the optimization objective. Eqn (4) defines e_i based on s_i and the assignment of O_i (a_{im}). Eqn (5) represents the unique assignment constraint for every operation. Eqn (6) and (7) represent the time lag constraints. Eqn (8)–(12) establish z_{ijm} as the indicator for overlap between O_i and O_j on the same M_m. Note for the same (i, j, m), $x_{ijm} + y_{ijm} \leq 1$ holds automatically. This mathematical formulation is based on the work by Boyer *et al.* originally developed for ring rolling processes.[39] The choice of indicator z_{ijm} is different from the design in Boyer *et al.* since they assume first-in-first-out (FIFO) policies for all functional modules. This assumption forbids the situation where the duration of O_i on M_m includes the duration of O_j, provided both O_i and O_j are assigned to the same M_m. For example, when two compatible heating operations are assigned to the same heater with time estimates of 1 hour and 3 hours, respectively, the FIFO policy would prevent the 1 hour operation to start until 2 hours after the start of the 3 hours operation. z_{ijm} is then used in (13) and (14) to enforce module capacity constraints. Compared to Boyer *et al.* which does not address work shift constraints, eqn (15)–(21) are included to guarantee work shift-constrained operations ($\mathfrak{P}$) are not scheduled between work shifts. Specifically, eqn (15) ensures that O_i is assigned to exactly one work shift, where A_{in}, the indicator for assigning O_i to W_n, is established by eqn (16)–(21) through the auxiliary indicators Y_{in} and Z_{in}.

Table 3 Variables in the MILP formulation. Parameters include O_i the ith operation, M_m the mth functional module, e_i the end time of the ith operation, s_j the start time of the jth operation, a_{im} if the ith operation is assigned to the mth functional module, S_n the start time of the nth work shift, and E_n the end time of the nth work shift

Variable(s)	Definition	Domain
s_i	The start time of O_i	$[0, +\infty)$
e_i	The end time of O_i	$[0, +\infty)$
$e_{\max}$	The makespan	$[0, +\infty)$
a_{im}	$\begin{cases} 1 & \text{if } O_i \text{ is processed by } M_m \\ 0 & \text{otherwise} \end{cases}$	$\{0, 1\}$
x_{ijm}	$\begin{cases} 1 & \text{if } e_i \leq s_j \text{ and } a_{im} = a_{jm} = 1 \\ 0 & \text{otherwise} \end{cases}$	$\{0, 1\}$
y_{ijm}	$\begin{cases} 1 & \text{if } e_j \leq s_i \text{ and } a_{im} = a_{jm} = 1 \\ 0 & \text{otherwise} \end{cases}$	$\{0, 1\}$
z_{ijm}	$\begin{cases} 0 & \text{if } x_{ijm} = 1 \text{ or } y_{ijm} = 1 \\ 1 & \text{otherwise} \end{cases}$	$\{0, 1\}$
Y_{in}	$\begin{cases} 1 & \text{if } S_n \leq s_i \\ 0 & \text{otherwise} \end{cases}$	$\{0, 1\}$
Z_{in}	$\begin{cases} 1 & \text{if } e_i \leq E_n \\ 0 & \text{otherwise} \end{cases}$	$\{0, 1\}$
A_{in}	$\begin{cases} 1 & \text{if } Y_{in} \text{ and } Z_{in} \\ 0 & \text{otherwise} \end{cases}$	$\{0, 1\}$

$$\min e_{\max} \tag{2}$$

$$\text{subject to : } e_{\max} \geq e_i, \quad \forall i \in \mathfrak{O} \tag{3}$$

$$e_i = s_i + \sum_{m \in \mathfrak{M}} p_{im} a_{im}, \quad \forall i \in \mathfrak{O} \tag{4}$$

$$\sum_{m \in \mathfrak{M}} a_{im} = 1, \quad \forall i \in \mathfrak{O} \tag{5}$$

$$s_j \geq e_i + l\min_{ij}, \quad \forall i \neq j \in \mathfrak{O} \tag{6}$$

$$s_j \leq e_i + l\max_{ij}, \quad \forall i \neq j \in \mathfrak{O} \tag{7}$$

$$e_i \leq s_j + \mathscr{M}(3 - x_{ijm} - a_{im} - a_{jm}), \quad \forall i < j \in \mathfrak{O}, \ \forall m \in \mathfrak{M} \tag{8}$$

$$s_j \leq e_i + \mathscr{M}(2 + x_{ijm} - a_{im} - a_{jm}), \quad \forall i < j \in \mathfrak{O}, \ \forall m \in \mathfrak{M} \tag{9}$$

$$e_j \leq s_i + \mathscr{M}(3 - y_{ijm} - a_{im} - a_{jm}), \quad \forall i < j \in \mathfrak{O}, \ \forall m \in \mathfrak{M} \tag{10}$$

$$s_i \leq e_j + \mathscr{M}(2 + y_{ijm} - a_{im} - a_{jm}), \quad \forall i < j \in \mathfrak{O}, \ \forall m \in \mathfrak{M} \tag{11}$$

$$x_{ijm} + y_{ijm} + z_{ijm} = 1 \tag{12}$$

$$\sum_{j \in \mathfrak{O} \setminus \{i\}} z_{ijm} \leq (K_m - 1) a_{im}, \quad \forall i \in \mathfrak{O}, \ \forall m \in \mathfrak{M} \tag{13}$$

$$z_{ijm} \leq C_{ij}, \quad \forall i, j \in \mathfrak{O}, \ \forall m \in \mathfrak{M} \tag{14}$$

© 2025 The Author(s). Published by the Royal Society of Chemistry

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM. This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

$$\sum_{n\in\mathfrak{W}} A_{in} = 1, \quad \forall i\in\mathfrak{P} \tag{15}$$

$$Y_{in} + Z_{in} - 2A_{in} \geq 0, \quad \forall i\in\mathfrak{P},\ \forall n\in\mathfrak{W} \tag{16}$$

$$A_{in} - Y_{in} - Z_{in} \geq -1, \quad \forall i\in\mathfrak{P},\ \forall n\in\mathfrak{W} \tag{17}$$

$$S_n \leq s_i + \mathscr{M}(1 - Y_{in}), \quad \forall i\in\mathfrak{P},\ \forall n\in\mathfrak{W} \tag{18}$$

$$s_i \leq S_n + \mathscr{M} Y_{in}, \quad \forall i\in\mathfrak{P},\ \forall n\in\mathfrak{W} \tag{19}$$

$$e_i \leq E_n + \mathscr{M}(1 - Z_{in}), \quad \forall i\in\mathfrak{P},\ \forall n\in\mathfrak{W} \tag{20}$$

$$E_n \leq e_i + \mathscr{M} Z_{in}, \quad \forall i\in\mathfrak{P},\ \forall n\in\mathfrak{W} \tag{21}$$

The big number $\mathscr{M}$ used in constraints[45] can be estimated as

$$p_i^{\max} = \max_{m\in\mathfrak{M}}\{p_{im} | p_{im} < +\infty\} \tag{22}$$

$$l_i^{\max} = \max_{j\in\mathfrak{O}}\{l\min_{ij} | l\min_{ij} \geq 0\} \tag{23}$$

$$E^{\max} = \max_{n\in\mathfrak{W}}\{E_n\} \tag{24}$$

$$\mathscr{M} = \max\left\{\sum_{i\in\mathfrak{O}}(p_i^{\max} + l_i^{\max}), E^{\max}\right\}. \tag{25}$$

It is worthwhile to discuss alternatives of treating laboratory constraints. A computationally equivalent treatment of work shifts is to introduce pre-scheduled "placeholder" operations that occupy gaps between work shifts. While the capacity constraint can be modeled by splitting multi-capacity modules into multiple single-capacity modules, this increases the size of $\mathfrak{M}$, thus increases the number of all constraints indexed by m. This is expected to be less computationally efficient than introducing capacity constraints explicitly as in (13). Feasible solutions that comply with lag time constraints can be selected from a set of potential solutions (*e.g.*, the heuristic method by Aarts *et al.*[18]), however, it is more computationally efficient to encode these as linear constraints than to select solutions from a less constrained problem in linear programming.

2.3 Case study construction

2.3.1 Network construction for case studies. Given a chemical library, hypothetical synthetic routes are proposed using ASKCOS[42] and SPARROW.[46] Synthetic routes are generated with the ASKCOS MCTS tree builder with the maximum price per gram for starting materials is set to be 120 USD by querying the ChemSpace (**https://chem-space.com/**) API. SPARROW is used to downselect routes using weighting factors of 50, 1, and 1 for `reward-weight`, `start-cost-weight`, and `reaction-weight`, respectively.

Without loss of generality, we define a fixed set of operation types. For each reaction, an operation graph is constructed by the following steps:

(1) For each solid reactant or reagent, create a `MakeSolution` operation for solution preparation;

(2) For each liquid reactant or reagent, create a `TransferLiquid` operation for its addition to reaction vessel;

(3) Create a `Heating` operation based on the recommended temperature condition of this reaction;

(4) Create a `ConcentrationAndPurification` operation for the product mixture.

Each operation is assigned a unique operation type that can be processed by one or more functional modules. The current list of operation types is shown in Table S1† along with the rules to estimate their processing times. Ordinary precedence relations are added according to the order in the above steps. Specifically, the `Heating` operation precedes all `TransferLiquid` and `MakeSolution` operations, and `TransferLiquid` operations for liquid reactant/reagent are preceded by `MakeSolution` operations for solution additions. If a reaction does not include solid reactant/reagent, the `TransferLiquid` operation of solvent addition precedes all other `TransferLiquid` operations. Additionally, a maximum lag time constraint is added between each `MakeSolution` and the `Heating` operation to model expiration of stock solutions. A minimum lag time constraint is also added between the `Heating` operation and the `ConcentrationAndPurification` operation to model the cooling process. Work shift constraints, if included in the formulation, are applied to all operations. The duration of a shift is set as 1.5 $\times$ the longest processing time with a gap between shifts set to half the duration of a shift.

2.3.2 Functional modules for case studies. Two distinct sets of functional modules (LAB-1 and LAB-2) are used for all case studies. Their components and capacities are shown in Table 4. Temperature bins (Table S2†) are defined and it is stipulated that `Heating` operations in the same bin are compatible and can be processed on the same heater concurrently if the heater has a capacity higher than one.

Table 4 Components and capacities of the functional module sets. The elements of a 2-tuple represent the module capacity and the number of modules, respectively. For example, "(1, 2)" represents there are two modules, each with a capacity of one

Module name	Compatible processes	LAB-1	LAB-2
Heater	`Heating`	(1, 2)	(2, 2)
Liquid handler	`TransferLiquid`	(1, 1)	(1, 1)
Stirrer	`MakeSolution`	(1, 2)	(1, 2)
Workup station	`ConcentrationAndPurification`	(1, 2)	(1, 2)

© 2025 The Author(s). Published by the Royal Society of Chemistry

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM.
This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

2.4 Baseline scheduler

A heuristic algorithm, as described in Algorithm 1, is implemented to provide baseline schedules for comparison.[47] This algorithm approximates optimal schedules by scheduling operations one-by-one in a greedy fashion that considers arbitrarily given precedence relations, operation-to-module assignments, module capacity, and minimum lag time by identifying the operation with the earliest possible end time in each iteration. Schedulers using only greedy iterations, however, may produce solutions that violate maximum lag time constraints. Inspired by Aarts *et al.*,[18] our baseline scheduler constructs feasible solutions by first identifying a reaction sequence that complies with the precedence relations defined in synthetic routes. This sequence is identified by, starting from a zero-precedent reaction, selecting the next reaction which has the fewest unscheduled precedents and does not depend on the last visited reaction. The algorithm further assumes an operation sequence for each reaction by adding artificial precedence relations among operations from the same reaction. This assumption allows adjusting the start times of all operations of a reaction by adding a delay to the first operation of this reaction, and, with a sufficiently large delay, the operations of this reaction can be made temporally close. Since maximum lag time constraints are only defined within a reaction in our case studies, this assumption aids in generating schedules complying with such constraints. We note while the baseline scheduler empirically produces schedules that comply with all constraints for our case studies, it does not guarantee compliance with maximum lag time constraints in general.

Input: *reactionSequence*: The reaction sequence as a list of operation lists.

Output: S: The set of scheduled operations.

$S \leftarrow \emptyset$

for *operationList* ***in*** *reactionSequence* **do**

  $delay \leftarrow 0$ `// A positive delay applied to operations of this reaction.`

  $valid \leftarrow False$ `// If S is valid. Set to` $False$ `in initialization.`

  S.reset() `// Reset` S `to its last commit.`

  while *not* *valid* **do**

    for O ***in*** *operationList* **do**

      FindEarliestStartTime($O, S,$ lower_bound $= delay$)

      `// Greedily find the start time of` O `based on its latest scheduled precedents and functional module availability.`

      S.temporarily_add(O) `// Temporarily include the start time of` O `into` S.

    end

    $valid \leftarrow$ ValidateSchedule(S)

    `// Check if the current schedule complies with all constraints.`

    $delay \mathrel{+}= delayIncrement$

    `// Increase` $delay$ `by a small amount and reschedule if` S `is invalid.`

  end

  S.commit() `// Commit adding start times of` $operationList$ `to` S.

end

Algorithm 1: Greedy scheduling algorithm

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM.
This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

2.5 Implementation

The scheduling problem is represented by a `pydantic` (version 2.7.4) schema,[48] for which an interactive visualization module is developed in `dash-cytoscape` (version 1.0.1).[49] Sample screenshots of the interface are shown in Fig. S2 and S3.† The MILP solver is implemented using `Gurobi` (version 11.0.2) through its `Python` API.[50] A thread limit of 12 is imposed for the MILP solver. A time limit of three wall hours is imposed for all scheduling instances. Empirically, we find the primal simplex method often outperforms others for MILP root relaxations in our formulation.

2.6 Chemical libraries

Target compounds from the following compound sets are used to build chemical libraries in case studies:

(1) VS-35: a set of 35 previously synthesized compounds from a screening campaign for cannabinoid receptors.[40]

(2) FDA-20: a set of 20 compounds from FDA-approved drug lists from 2019 to 2022.[41]

Subsets of VS-35 or FDA-20 are used as test libraries. These test libraries are grouped by the number of target chemicals, indicated by the suffix of the group name. Each group contains ten test libraries constructed by random sampling, *e.g.*, group VS.05 denotes the group of ten randomly selected 5-target libraries that are subsets of VS-35. The parameters of these libraries are summarized in Table S3.†

3 Results and discussion

3.1 Overall results of case studies

3.1.1 Solutions of the MILP scheduler. The MILP scheduler may terminate before finding an optimal solution due to the applied time limit (Section 2.5), especially for the larger libraries that tend to be more complex MILP problems. Premature termination can result in either a set of feasible solutions (suboptimal schedules) or no feasible solution. The runtime trend is shown in Fig. 3: While optimal schedules can be reliably found for small libraries (number of target chemicals ≤ 4), mid-size libraries ($5 \leq$ number of target chemicals ≤ 7) may result in only suboptimal schedules, and for large libraries ($8 \leq$ number of target chemicals), finding even feasible solutions becomes difficult. This is particularly true for scheduling instances including work shift constraints where more variables are needed. This also explains the fewer optimal schedules found for VS libraries compared to FDA libraries, as the former often includes more reactions/operations than the latter (Table S3†). Out of the 720 scheduling instances, optimal/suboptimal schedules are found for 363/323 of them, and for 34 of them no feasible solutions are found before reaching the time limit of 3 hours. As shown in Fig. S1,† temporal cost to find an optimal schedule increases exponentially with respect to the number of target chemicals. This suggest that time complexity is a limitation of the current MILP formulation, thus the motivation to explore better formulations or heuristic methods in future studies.

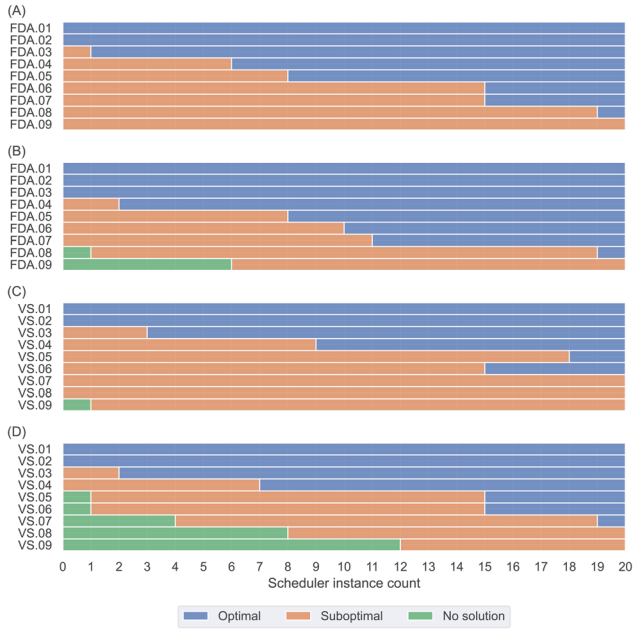

Fig. 3 Results of all 720 MILP scheduling instances. Each bar corresponds to the results of a library group scheduled on LAB-1 and LAB-2, totaling 20 scheduling instances. (A) FDA libraries scheduled on without work shift constraints; (B) FDA libraries scheduled on with work shift constraints; (C) VS libraries scheduled on without work shift constraints; (D) VS libraries scheduled on with work shift constraints. Results of these instances fall in three disjoint categories. "Optimal": the optimal solution is found, "Suboptimal": the scheduler terminates prematurely, and at least one feasible solution is found, and "No solution": no feasible solution is found before reaching time limit. Since all scheduling instances are feasible MILP problems, these three categories are jointly sufficient to cover all instances. As the number of target chemicals increases, it becomes increasingly difficult to find an optimal or even feasible solution within the maximum allowed 3 hours wall time.

3.1.2 Comparison between MILP and baseline schedules. The percentage makespan gap between the baseline and the MILP schedule, defined as

$$\text{percentage makespan gap} = \frac{e_{\max}^{\text{baseline}} - e_{\max}^{\text{MILP}}}{e_{\max}^{\text{baseline}}}$$

is used to evaluate the MILP scheduler. Fig. 4 shows the percentage makespan gap distributions between the baseline schedule and the optimal MILP schedule for all libraries grouped by the number of target chemicals (the groups are listed in Section S3†). Note only optimal schedules are included in Fig. 4 and not all scheduling instances converged to an optimal solution due to the 3 hours scheduler time limit (Section 2.5), as a result, several library groups have fewer than ten percentage gap values shown in Fig. 4; the full set of results including suboptimal MILP schedules can be found in the project repository. The makespan of the optimal MILP schedule tends to be shorter than that of baseline schedule, with a few exceptions for very small libraries (number of targets ≤ 2) where the gap is negligible. This is not surprising as the solution space of small libraries is also small.

© 2025 The Author(s). Published by the Royal Society of Chemistry

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM.
This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

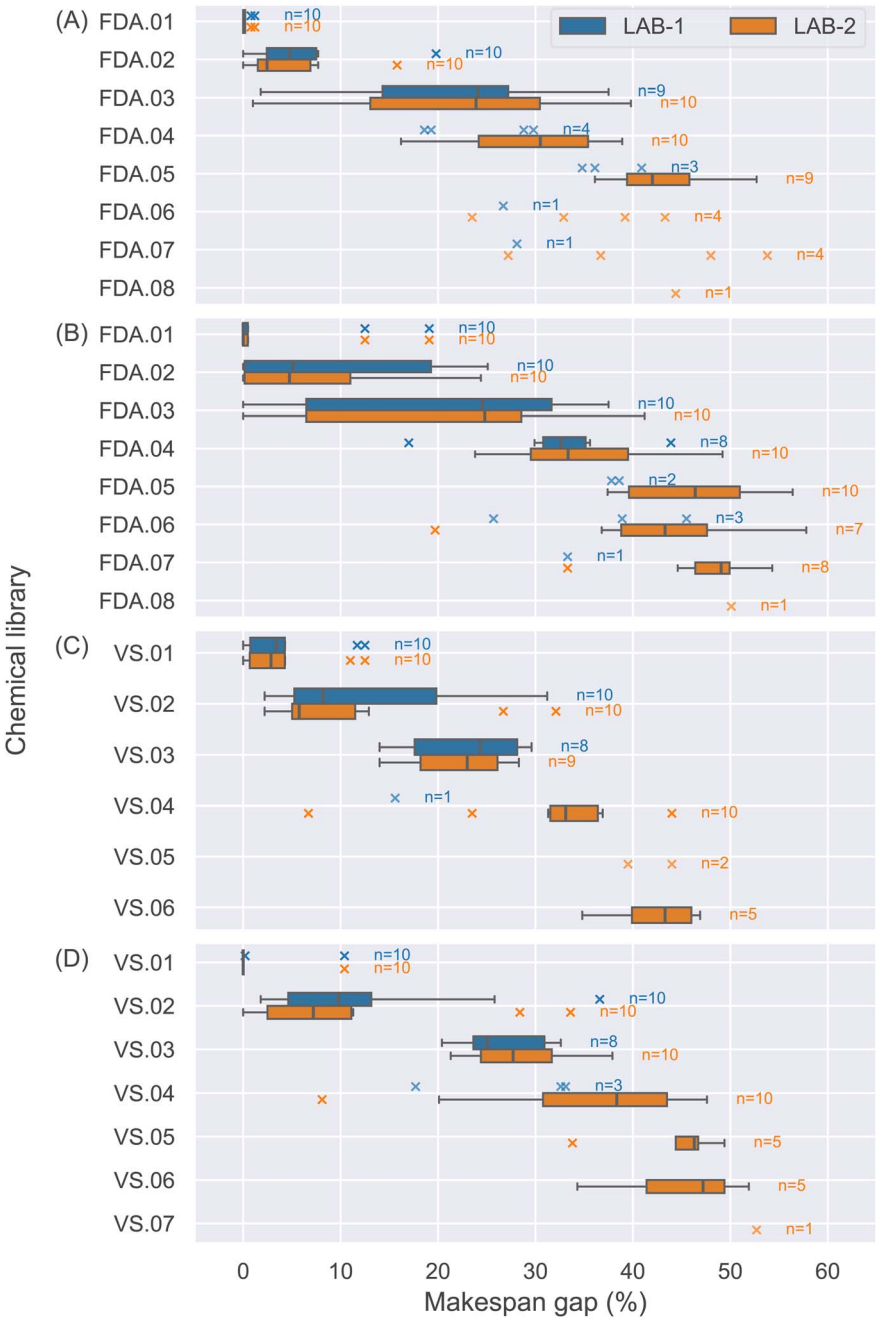

Fig. 4 Percentage makespan gaps between baseline and optimal MILP schedules. Two types of markers denote scheduling instances on LAB-1 and LAB-2, respectively. (A) FDA libraries without work shift constraints; (B) FDA libraries with work shift constraints; (C) VS libraries without work shift constraints; (D) VS libraries with work shift constraints. The percentage gap ranges from 0% to 57.8%, and tends to be higher for larger libraries. Percentage gaps were calculated based on the makespan values of baseline schedules.

For scheduling instances where the MILP scheduler found an optimal schedule (363 instances shown in Fig. 4), the percentage gap ranges from 0% to 57.8% with an average of 20.5%. For instances where at least one feasible solution is found (686 instances including 363 optimal schedules and 323 suboptimal schedules), the average percentage gap becomes 9.8%. The reduction in the average percentage gap is mainly due to 14 of 323 of suboptimal schedules having makespan values higher than baseline schedules, *i.e.*, the percentage gaps are negative for these instances. In general, the percentage gap tends to be larger for larger libraries, which is expected as larger operation graphs tend to have more degrees of freedom and provide greater opportunity for efficient scheduling. Another observation from Fig. 4 is that optimal schedules are hard to find for schedules on LAB-1 compared to LAB-2, *e.g.*, the imbalance of different markers in Fig. 4C and D. Note the only difference between LAB-1 and LAB-2 is the heater modules in the latter have a higher capacity than those in the former.

© 2025 The Author(s). Published by the Royal Society of Chemistry

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM. This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

To seek insights on how to improve heuristic algorithms, we investigate the differences between baseline and optimal MILP schedules are investigated through two metrics. The assignment difference, referring to the operation-to-module assignment, is calculated as the percentage of operations assigned to different modules between two schedules. The sequence difference is the normalized Levenshtein distances between two operation sequences of the same module from two schedules and averaged over all modules. We note that sequence difference is more sensitive than assignment difference, as operation sequences can change on modules without altering the operation-to-module assignment, but the reverse is not true. Fig. S4† shows the makespan gap between baseline and optimal MILP schedules along with corresponding assignment/sequence difference values. The values of assignment difference range from 0 to 25.5%, with 95% of them being lower than 21.5%, indicating the baseline schedule is not very far away from the optimal schedule in terms of operation-to-module assignment even for large libraries. In contrast, the operation sequences on modules from baseline and optimal schedules can differ significantly, as indicated by the high sequence difference values (averaged at 0.43, or 0.53 when libraries of size 1 and 2 are excluded) especially for large libraries. For very small libraries, a strong correlation between assignment and sequence differences is observed, which is weakened and eventually vanishes as the number of targets (library size) increases. These results suggest that the current baseline scheduler can approximate optimal operation-to-module assignments but struggles with approximating optimal operation sequences.

3.2 Illustration of representative schedules

3.2.1 Independent linear routes: FDA.03.09. We start by examining the scheduling instances of a small-size FDA library, FDA.03.09, where "03" denotes its size (number of target chemicals) and "09" the sampling index for targets. The optimal MILP solution achieves a 37.5% makespan reduction when scheduled on LAB-1 without work shift constraints. Given the three target chemicals, ASKCOS and SPARROW are used to generate synthetic routes that form a reaction network of 18 reactions. As shown in Fig. 5, the routes to the three targets do not share any common intermediates. The corresponding operation network of 110 operations (Fig. S2†) is then constructed using the rules defined in Section 2.3. This operation network, along with the functional module definition and processing time estimates, constitute the input to the scheduler for this library.

The optimal schedule of FDA.03.09 on LAB-1 produced by the MILP scheduler is shown in Fig. 6A. Compared to its heuristic counterpart (Fig. 6B), the optimal schedule differs in both operation-to-module assignment and processing order (operation sequence) on individual modules. Both schedules follow the precedence relations defined by the three independent routes in Fig. 5. They also share the same initial operations (both start with operations of reaction #1 and #17). However, they differ in terms of the absolute order of reactions (global precedence relations). For example, in the optimal schedule, reaction #9 is scheduled after #7, whereas in the baseline schedule, it precedes #7. As expected, the module utilization is higher for all modules using the optimal schedule. The percentage gap between the two solutions depends on the functional modules. For instance, when this library is scheduled on LAB-2 with other parameters unchanged, the makespan gap increases to 41.2%.

3.2.2 Scheduling with work shift constraints: VS.04.07. We now turn to a larger library, VS.04.07, corresponding to a reaction network of 29 reactions and 175 operations. In contrast to FDA.03.09, the synthetic routes of VS.04.07 are not linear graphs but trees (Fig. S5†). The optimal and baseline schedules on LAB-2 including work shift constraints are shown in Fig. 7. The baseline schedule requires two additional work shifts to complete the library synthesis campaign compared to the optimal schedule, contributing to a makespan gap of 36.0%.

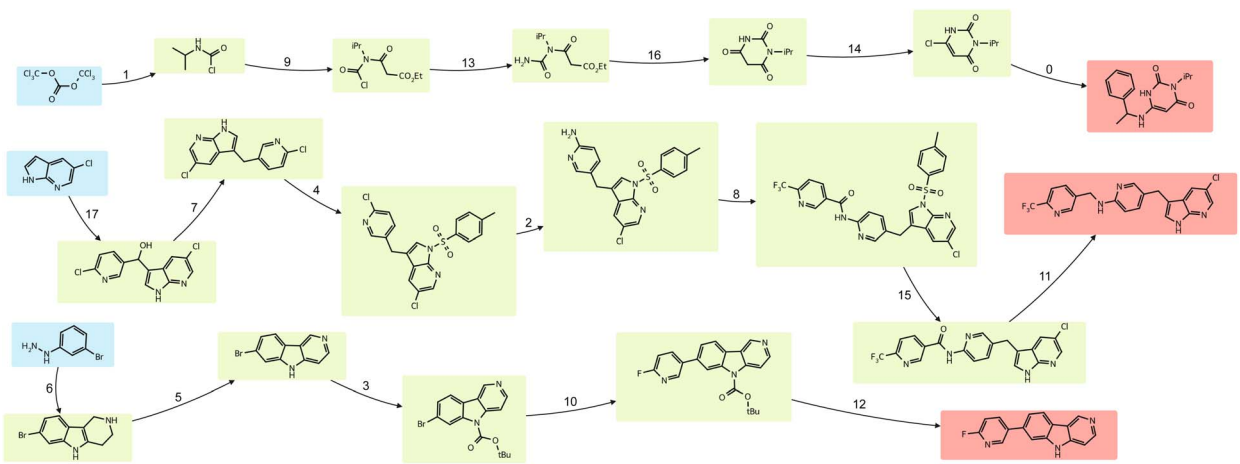

Fig. 5 Reaction network for FDA.03.09 generated by ASKCOS and SPARROW. Molecules with blue, green, and red backgrounds represent starting, intermediate, and target molecules, respectively. Reagents and solvents of these reactions, though part of the reaction network, are excluded from this figure for clarity purposes.

© 2025 The Author(s). Published by the Royal Society of Chemistry

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM.
This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

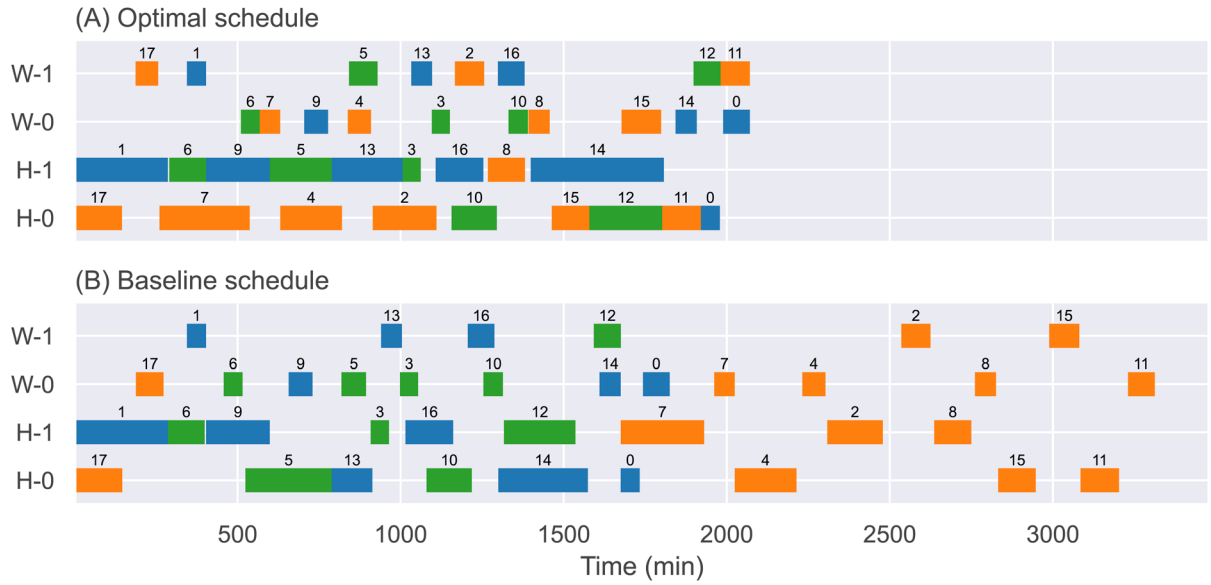

Fig. 6 Comparison of (A) the optimal and (B) the baseline schedules for FDA.03.09 scheduled on LAB-1 with a percentage gap of 51.3%. "H" and "W" represent heaters and workup stations, respectively. Only heaters and workup stations are shown, as the operation duration on these modules is significantly longer than others. Black numerical indices above colored bars are reaction indices indicating to which reaction the process belong. Bars are colored based on the corresponding operations' final target chemicals. The same operation can have different processing time on different functional modules, see Section S2.1† for more details about processing time estimation.

Since heaters in LAB-2 have a capacity of 2, compatible operations can be processed on the same module, as shown by the temporally overlapped operations on heaters in Fig. 7. The optimal schedule exhibits more same-module overlap than the baseline schedule, though this is not as significant as their difference in makespan. As in FDA.03.09, differences in

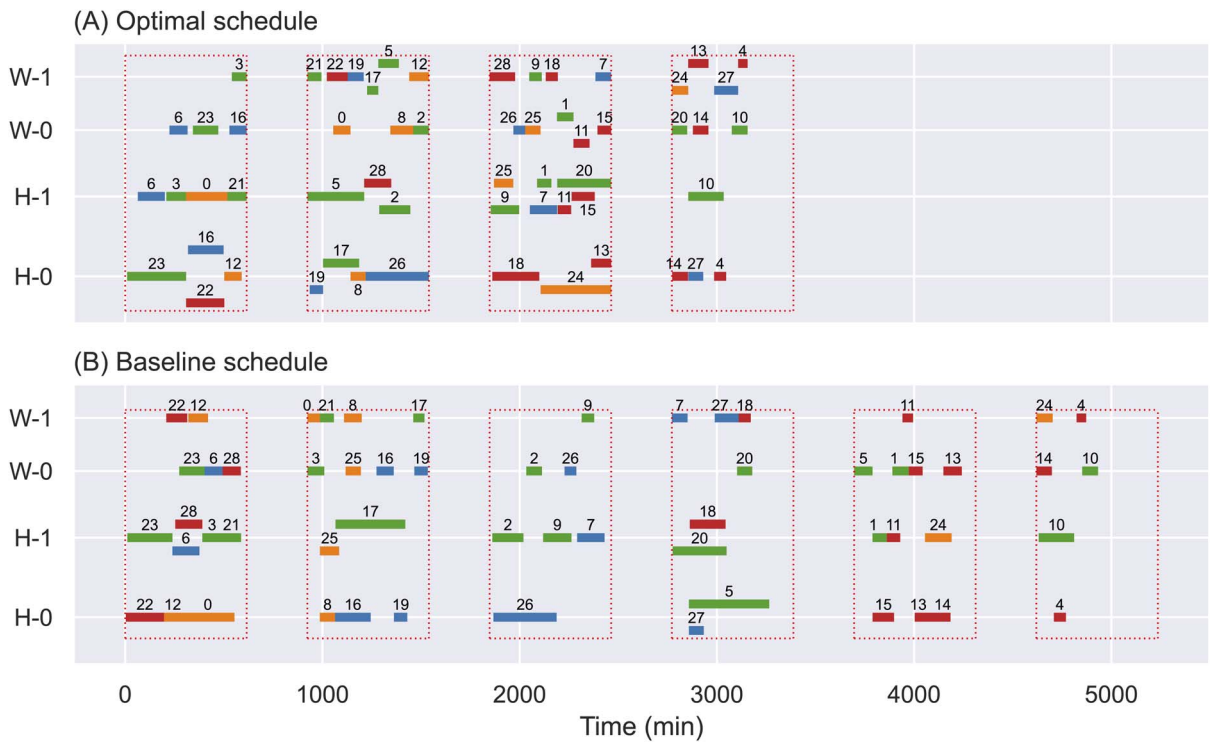

Fig. 7 Comparison of (A) the optimal and (B) the baseline schedules for VS.04.07 scheduled on LAB-2 with work shift constraints denoted by red dashed boxes. Some operations overlap with others temporally on the same heater, as the heaters have a capacity of 2 in LAB-2. Compared to the optimal schedule, two additional work shifts are needed for the baseline schedule.

© 2025 The Author(s). Published by the Royal Society of Chemistry

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM. This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

operation-to-module assignment and module processing order are observed.

4 Conclusion

In this study, the problem of scheduling chemical library synthesis is formalized as a generalized FJSP. A MILP formulation is proposed to consider common laboratory constraints including lag time, module capacity, and work shifts. Two scheduling algorithms, a greedy heuristic (baseline scheduler) and a MILP formulation, are evaluated for their ability to minimize the total makespan (duration) of a library synthesis campaign. To test the proposed schedulers, 720 scheduling instances are built from hypothetical synthetic routes to realistically accessible chemical libraries involving 3 to 63 reaction steps, or 20 to 402 operations. Within the 3 hours wall time limit, the MILP scheduler is able to produce optimal and suboptimal schedules for 363 and 323 scheduling instances, respectively, leaving 34 instances for which no feasible solution can be found. Compared to the baseline schedules, the optimal MILP schedules show makespan reductions ranging from 0% (very small libraries) to 58% (large libraries) with an average of 20%. We find that the baseline schedule can approximate optimal operation-to-module assignments but fails to approximate optimal operation sequences. Two illustrative case studies show that optimal schedules greatly reduce makespan and improve module utilization for synthesis campaigns of both linear routes and tree routes compared to baseline schedules.

There are still opportunities to improve chemical library synthesis scheduling. The current implementation of the MILP formulation suffers from its high time complexity and is thus not ideal for large libraries. An approximation of this formulation with lower time complexity could also enable dynamic scheduling to adapt real-time changes during a synthesis campaign. Additionally, both schedulers require reliable time estimate and pre-planned batches, which could be difficult in real library synthesis scenarios, such as those with large uncertainties in yield. This limitation means that operations with a dynamic end point, such as reaction whose end points are determined by color change, cannot be scheduled. These results provide open-source tools for improving synthesis efficiency and, as a standalone module, could be integrated into automated chemistry platforms in the future.

Data availability

All code and data included in this study are available in the project GitHub repository (**https://github.com/qai222/libsyn_tools**).

Author contributions

Qianxiang Ai: methodology, software, formal analysis, writing – original draft. Fanwang Meng: methodology, software, writing – review & editing. Runzhong Wang: methodology, software, writing – review & editing. J. Cullen Klein: methodology, writing – review & editing. Alexander G. Godfrey: methodology, writing – review & editing, project administration. Connor W. Coley: conceptualization, writing – review & editing, supervision, funding acquisition, project administration.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

Research reported in this publication was supported by the National Institutes of Health under award number U18TR004149. This research was supported in part by the Intramural/Extramural research program of the National Center for Advancing Translational Sciences, the National Institutes of Health. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

1 R. W. Wagner, F. Li, H. Du and J. S. Lindsey, Investigation of Cocatalysis Conditions Using an Automated Microscale Multireactor Workstation: Synthesis of meso-Tetramesitylporphyrin, *Org. Process Res. Dev.*, 1999, **3**, 28–37.
2 R. B. Merrifield and J. M. Stewart, Automated Peptide Synthesis, *Nature*, 1965, **207**, 522–523.
3 K. Machida, Y. Hirose, S. Fuse, T. Sugawara and T. Takahashi, Development and application of a solution-phase automated synthesizer, 'ChemKonzert', *Chem. Pharm. Bull.*, 2010, **58**, 87–93.
4 A. Weber, E. Von Roedern and H. U. Stilz, SynCar: an approach to automated synthesis, *J. Comb. Chem.*, 2005, **7**, 178–184.
5 T. Jiang, S. Bordi, A. E. McMillan, K.-Y. Chen, F. Saito, P. L. Nichols, B. M. Wanner and J. W. Bode, An integrated console for capsule-based, automated organic synthesis, *Chem. Sci.*, 2021, **12**, 6977–6982.
6 S. Steiner, J. Wolf, S. Glatzel, A. Andreou, J. M. Granda, G. Keenan, T. Hinkley, G. Aragon-Camarasa, P. J. Kitson, D. Angelone and L. Cronin, Organic synthesis in a modular robotic system driven by a chemical programming language, *Science*, 2019, **363**, eaav2211.
7 N. Collins, D. Stout, J.-P. Lim, J. P. Malerich, J. D. White, P. B. Madrid, M. Latendresse, D. Krieger, J. Szeto and V.-A. Vu, others Fully automated chemical synthesis: toward the universal synthesizer, *Org. Process Res. Dev.*, 2020, **24**, 2064–2077.
8 S. Chatterjee, M. Guidi, P. H. Seeberger and K. Gilmore, Automated radial synthesis of organic molecules, *Nature*, 2020, **579**, 379–384.
9 V. Fasano, R. C. Mykura, J. M. Fordham, J. J. Rogers, B. Banecki, A. Noble and V. K. Aggarwal, Automated stereocontrolled assembly-line synthesis of organic molecules, *Nat. Synth.*, 2022, **1**, 902–907.

 © 2025 The Author(s). Published by the Royal Society of Chemistry

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM. This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

10 G. Wang, H. T. Ang, S. R. Dubbaka, P. O'Neill and J. Wu, Multistep automated synthesis of pharmaceuticals, *Trends Chem.*, 2023, **5**, 432–445.
11 B. A. Koscher, *et al.*, Autonomous, multiproperty-driven molecular discovery: From predictions to measurements and back, *Science*, 2023, **382**, eadi1407.
12 F. Strieth-Kalthoff, *et al.*, Delocalized, asynchronous, closed-loop discovery of organic laser emitters, *Science*, 2024, **384**, eadk9227.
13 W. Gao, P. Raghavan and C. W. Coley, Autonomous platforms for data-driven organic synthesis, *Nat. Commun.*, 2022, **13**, 1075.
14 R. Vescovi, T. Ginsburg, K. Hippe, D. Ozgulbas, C. Stone, A. Stroka, R. Butler, B. Blaiszik, T. Brettin and K. Chard, others Towards a modular architecture for science factories, *Digital Discovery*, 2023, **2**, 1980–1998.
15 J. S. Lindsey, L. A. Corkan, D. Erb and G. J. Powers, Robotic work station for microscale synthetic chemistry: On-line absorption spectroscopy, quantitative automated thin-layer chromatography, and multiple reactions in parallel, *Rev. Sci. Instrum.*, 1988, **59**, 940–950.
16 L. Andrew Corkan and J. S. Lindsey, Experiment manager software for an automated chemistry workstation, including a scheduler for parallel experimentation, *Chemom. Intell. Lab. Syst.*, 1992, **17**, 47–74.
17 H. Du, L. A. Corkan, K. Yang, P. Y. Kuo and J. S. Lindsey, An automated microscale chemistry workstation capable of parallel, adaptive experimentation, *Chemom. Intell. Lab. Syst.*, 1999, **48**, 181–203.
18 R. J. Aarts, J. S. Lindsey, L. A. Corkan and S. F. Smith, Flexible protocols improve parallel experimentation throughput, *Clin. Chem.*, 1995, **41**, 1004–1010.
19 W. S. You, B. J. Choi, H. Moon, J. C. Koo and H. R. Choi, Robotic laboratory automation platform based on mobile agents for clinical chemistry, *Intell. Serv. Robot.*, 2017, **10**, 347–362.
20 R. B. Canty, Autonomous experimentation for molecular discovery applications, thesis, Massachusetts Institute of Technology, 2024.
21 B. Burger, P. M. Maffettone, V. V. Gusev, C. M. Aitchison, Y. Bai, X. Wang, X. Li, B. M. Alston, B. Li, R. Clowes, N. Rankin, B. Harris, R. S. Sprick and A. I. Cooper, A mobile robotic chemist, *Nature*, 2020, **583**, 237–241.
22 N. F. Delaney, J. I. Rojas Echenique and C. J. C. Marx, An Open-Source Manager for Laboratory Automation, *SLAS Technol.*, 2013, **18**, 171–177.
23 X. Gu, S. Neubert, N. Stoll and K. Thurow, Intelligent scheduling method for life science automation systems, *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2016, pp 156–161.
24 C. Cabrera, M. Fine-Morris, M. Pokross, K. Kish, S. Michalczyk, M. Cahn, H. Klei and M. F. Russo, Dynamically Optimizing Experiment Schedules of a Laboratory Robot System with Simulated Annealing, *SLAS Technol.*, 2014, **19**, 517–527.
25 T. D. Itoh, T. Horinouchi, H. Uchida, K. Takahashi and H. Ozaki, Optimal Scheduling for Laboratory Automation of Life Science Experiments with Time Constraints, *SLAS Technol.*, 2021, **26**, 650–659.
26 J. M. Pinto, M. Joly and L. F. L. Moro, Planning and scheduling models for refinery operations, *Comput. Chem. Eng.*, 2000, **24**, 2259–2276.
27 C. Chatzidoukas, S. Pistikopoulos and C. Kiparissides, A hierarchical optimization approach to optimal production scheduling in an industrial continuous olefin polymerization reactor, *Macromol. React. Eng.*, 2009, **3**, 36–46.
28 S. Wang, X. Wang, F. Chu and J. Yu, An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production, *Int. J. Prod. Res.*, 2020, **58**, 2283–2314.
29 C. T. Maravelias, General framework and modeling approach classification for chemical production scheduling, *AIChE J.*, 2012, **58**, 1812–1828, _eprint: **https://onlinelibrary.wiley.com/doi/pdf/10.1002/aic.13801**.
30 P. Brucker and R. Schlie, Job-shop scheduling with multi-purpose machines, *Computing*, 1990, **45**, 369–375.
31 M. R. Garey, D. S. Johnson and R. Sethi, The Complexity of Flowshop and Jobshop Scheduling, *Math. Oper. Res.*, 1976, **1**, 117–129.
32 Y. N. Sotskov, The complexity of shop-scheduling problems with two or three jobs, *Eur. J. Oper. Res.*, 1991, **53**, 326–336.
33 R. Alvarez-Valdés, A. Fuertes, J. M. Tamarit, G. Giménez and R. Ramos, A heuristic to schedule flexible job-shop in a glass factory, *Eur. J. Oper. Res.*, 2005, **165**, 525–534.
34 G. Vilcot and J.-C. Billaut, A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem, *Eur. J. Oper. Res.*, 2008, **190**, 398–411.
35 E. G. Birgin, P. Feofiloff, C. G. Fernandes, E. L. De Melo, M. T. Oshiro and D. P. Ronconi, A MILP model for an extended version of the flexible job shop problem, *Optim. Lett.*, 2014, **8**, 1417–1431.
36 G. A. Kasapidis, D. C. Paraskevopoulos, P. P. Repoussis and C. D. Tarantilis, Flexible Job Shop Scheduling Problems with Arbitrary Precedence Graphs, *Prod. Oper. Manag.*, 2021, **30**, 4044–4068.
37 P. J. Pawar and K. C. Bhosale, Flexible Job Shop Scheduling for Press Working Industries with Operation Precedence Constraint, *Process Integr. Optim. Sustain.*, 2022, **6**, 409–430.
38 R. Braune and K. F. Doerner, Real-world flexible resource profile scheduling with multiple criteria: learning scalarization functions for MIP and heuristic approaches, *J. Oper. Res. Soc.*, 2017, **68**, 952–972, DOI: **10.1057/s41274-017-0239-y**.
39 V. Boyer, J. Vallikavungal, X. C. Rodríguez and M. A. Salazar-Aguilar, The generalized flexible job shop scheduling problem, *Comput. Ind. Eng.*, 2021, **160**, 107542.
40 A. A. Sadybekov, A. V. Sadybekov, Y. Liu, C. Iliopoulos-Tsoutsouvas, X.-P. Huang, J. Pickett, B. Houser, N. Patel, N. K. Tran and F. Tong, others Synthon-based ligand discovery in virtual libraries of over 11 billion compounds, *Nature*, 2022, **601**, 452–459.

Open Access Article. Published on 17 December 2024. Downloaded on 2/3/2025 10:03:02 PM.
This article is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported Licence.

41 Food and Drug Administration, *Drug Approvals and Databases*, 2024, **https://www.fda.gov/drugs/development-approval-process-drugs/drug-approvals-and-databases**.
42 *ASKCOS*, 2024, original-date: 2024-30-12, **https://gitlab.com/mlpds_mit/askcosv2**.
43 J. C. Fromer and C. W. Coley, An algorithmic framework for synthetic cost-aware decision making in molecular design, *Nat. Comput. Sci.*, 2024, **4**, 440–450.
44 E. Kondili, C. C. Pantelides and R. W. Sargent, A general algorithm for short-term scheduling of batch operations—I. MILP formulation, *Comput. Chem. Eng.*, 1993, **17**, 211–227.
45 F. S. Hillier and G. J. Lieberman, *Introduction to operations research*, McGraw-Hill, 2015.
46 J. C. Fromer and C. W. Coley, An algorithmic framework for synthetic cost-aware decision making in molecular design, *Nat. Comput. Sci.*, 2024, 1–11.
47 R. Wang, Z. Hua, G. Liu, J. Zhang, J. Yan, F. Qi, S. Yang, J. Zhou and X. Yang, A bi-level framework for learning to solve combinatorial optimization on graphs, *Adv. Neural Inf. Process. Syst.*, 2021, **34**, 21453–21466.
48 S. Colvin pydantic: Data validation using Python type hints. **https://github.com/pydantic/pydantic**.
49 Team <cytoscape@plotly.com>, T. P. dash-cytoscape, A Component Library for Dash aimed at facilitating network visualization in Python, wrapped around Cytoscape.js. **https://dash.plotly.com/cytoscape**.
50 Gurobi Optimization, LLC, Gurobi Optimizer Reference Manual, 2024, **https://www.gurobi.com**.

© 2025 The Author(s). Published by the Royal Society of Chemistry