Экспериментальное сравнение моделей планирования биохимического производства.

В. В. Пырэу, С. А. Тренин
kondratiukvitalik@gmail.com; s.trenin@gmail.com

Целью данной работы является исследование задачи оперативного планирования производства для биохимической промышленности. Анализируются различные постановки задачи составления расписания, учитывающие ограничения, приходящие из практики хранение промежуточных веществ, требования к работе производственных узлов и подготовка станков, например наладка и очистка между запусками. Рассматриваются модели смешанного целочисленного линейного программирования. Это означает, что задачи являются NP-трудными. Основная трудность заключается в том, что при моделировании задач с схожим описанием модели могут сильно отличаться по сложности (числу переменных и ограничений), что негативно сказывается на процессе внедрения моделей в практику. Для решения этого вопроса проводится экспериментальный запуск моделей, разработанных для одной предметной области, в задачах из другой и интерпретация полученных результатов.

Ключевые слова: *planning; scheduling; MILP*

DOI:

1 Введение

Основная решаемая задача — планирование производства и создание расписаний в ограничениях, диктуемых особенностями ресурсов завода. Эта задача имеет практическое значение, так как производственные процессы усложняются, количество машин растёт, а, как следствие, растут размеры ограничений и имеющегося сырья. Это влечёт за собой то, что ручное создание расписаний и распределения сырья по машинам становится невозможным, а получаемые расписания — слишком неэкономными. В работе [3] приводится детальное описание и классификация постановок задач создания расписаний, а так же богатая классификация разработанных на момент методов моделирования. Основное поле для исследований — это различные способы описать практические требования к расписанию так, чтобы математическая постановка задачи, как задачи оптимизации, было (1) простым и коротким и (2) позволяло найти оптимальное решение быстрее. Так как время работы является зависит не только от сложности постановки задачи, то основным критерием качества модели будет её сложность.

Определение 1. **Модель** — задача целочисленного линейного программирования в общей форме, описывающая практические требования. Значения переменных из допустимых решений задачи будут интерпретироваться как план производства. **Сложность модели** — число её переменных и ограничений типа равенства и неравенства.

В работах [1], [2], [5], [4] описаны различные задачи из практики и предложены методы по их решению. Утверждается, что методы можно использовать перекрёстно, то есть не только на тех данных, для которых они предложены. Некоторые авторы, как например [1] предлагают модели, которые не гарантируют оптимальность полученных расписаний, однако гарантируют простоту самой модели. В работе описано, в при каких предположениях

об описании процессов какие методы моделирования работают лучше, то есть создают модели с более простым описанием.

2 Описание имеющихся данных

Так как авторы статьи поставили одной из своих целью собрать различные варианты постановок задач планирования из разных предметных областей, то в первую очередь необходимо эти задачи описать. Задача планирования состоит в том, что по описанию процесса необходимо указать, какие операции на каких машинах и с какими входными веществами нужно произвести, чтобы по имеющимся на складе прекурсорам получить заявленное количество продукта.

Определение 2. Прекурсор — вещество, имеющееся изначально на складе, из которого будут произведены все требуемые продукты. Также **промежуточным прекурсором** будем называть вещество, получаемое после некоторых этапов производства, но не требуемое для получения в финале.

В описание задачи входят описания всех прекурсоров, описание производственных узлов — сущностей, способных проводить реакции и описания рецептов приготовления промежуточных прекурсоров и финальных продуктов. Рецепт представляет собой описание одной операции: сколько нужно обрабатывать, на каком оборудовании и какие прекурсоры в какой пропорции, чтобы получить продукт.

Вещества разрешается хранить на складе, который описывается максимальной вместимостью. Некоторые вещества, возможно, хранить нельзя вовсе.

В данной работе рассматривается только **пакетное производство**. В этом случае на узел подаётся пакет некоторого размера и обрабатывается. Продуктом является пакет вещества-продукта. Для каждого узла известен максимальный и минимальный размер пакета, который можно подать. Этот подход является альтернативным к **непрерывному производству**, где узлы оперируют с непрерывными потоками данных, но это требует другого подхода к созданию моделей, и поэтому не рассматривается для перекрёстного запуска моделей.

В зависимости от задачи, время обработки пакета может как зависеть, так и не зависеть от его размера. В работе описывается, какие модели способны работать с неравными временами обработки пакета, а какие нет и насколько изменится сложность модели.

Процессы, в зависимости от их структуры делятся на две большие группы: последовательные и сетевые.

Определение 3. Последовательный процесс — процесс, который может быть разделён на несколько стадий, упорядоченных во времени, и пакеты передаются на производство только от предыдущей стадии к следующей. В зависимости от числа стадий эти процессы делятся на **одноступенчатые** и **многоступенчатые**.

Собственно, **сетевой** процесс — это тот процесс, что не является последовательным. Как будет показано далее, сетевые процессы являются более общими, но и более сложными для моделирования.

Процессы (в особенности сетевые) удобно представлять себе в виде графов состояний (STN), впервые предложенных в работе [6]. Пример такого графа можно видеть на рисунке 1. Круглыми вершинами обозначаются склады веществ, прямоугольными — задачи, а рёбрами — поток материала (число над ребром — процент вещества в пакете).

Рис. 1 Пример STN-диаграммы

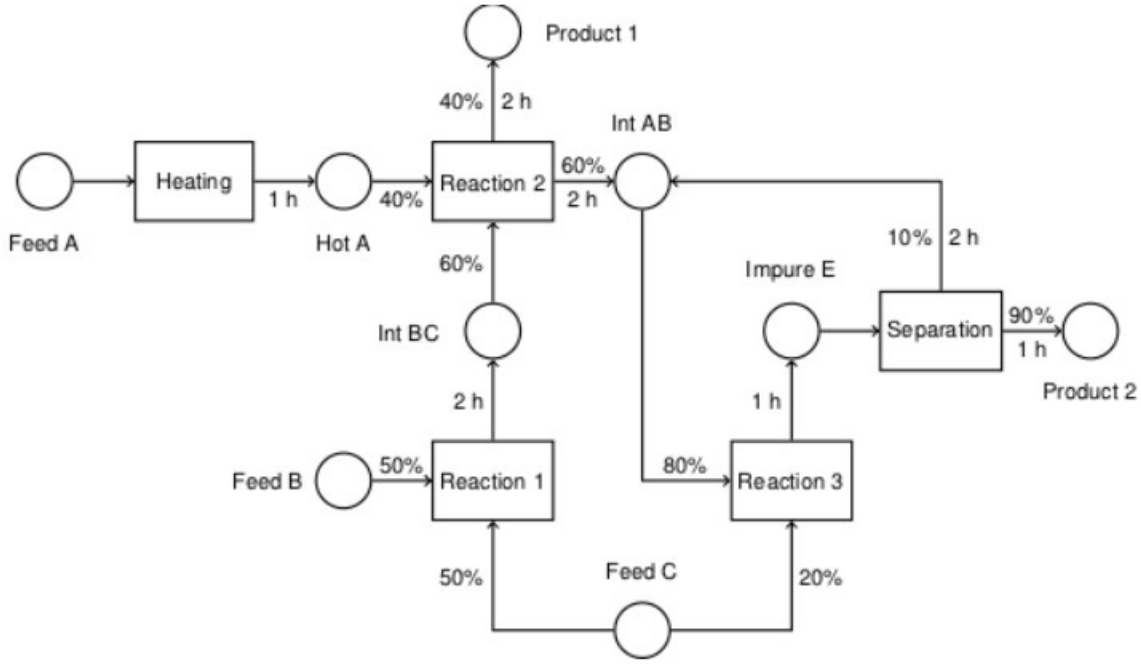
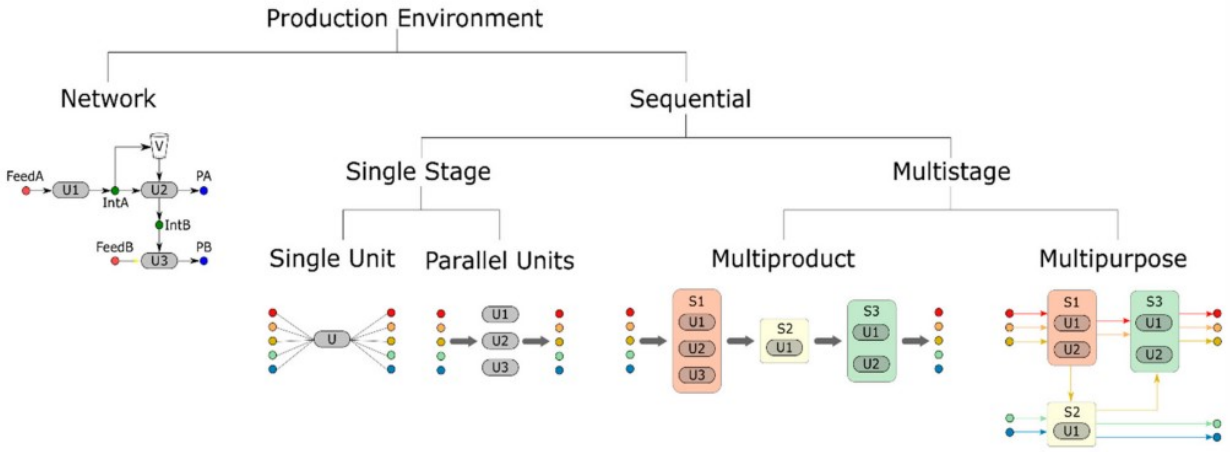


Рис. 2 Классификация процессов



Классификация 2 взята из [3] где она является более подробной, но в этой статье параллелизм и наличие многих продуктов/прекурсоров являются параметрами модели. Основной вопрос состоит в том, как переход от последовательных процессов к сетевым изменяет модель и какие методы, изобретённые для последовательных моделей применимы к сетевым и наоборот.

3 Описание моделей

Моделью в рамках данной работы будем называть задачу смешанного целочисленного линейного программирования в общей форме. На данном этапе необходимо закодировать решения, принимаемые системой составления расписаний в переменные, которые будут

называться **решающими переменными**. Далее T будет обозначать множество промежутков времени и переменные, с ним связанные. Другими большими латинскими буквами (с индексами) будут обозначаться непрерывные решающие переменные, малыми — бинарные решающие переменные и индексы (в некоторых моделях бинарные переменные и являются своеобразными «индексами» того, что какое-то утверждение верно или нет). Также большими буквами будут обозначаться параметры модели и их множества. Строчные греческие буквы по умолчанию означают разные затраты, сопровождающие процесс (в моделях, где они присутствуют).

Опишем все обозначения, которые будут встречаться в эксперименте.

1. Множества:
   - U — множество производственных узлов. Индексы у этого множества означают:
     - U_j — узлы, способные производить задачу $j \in J$.
   - P — множество продуктов (как прекурсоры, так и те, что получаются в процессе реакций). Индексы у этого множества означают:
     - P_j^{in} — продукты, потребляемые задачей $j \in J$.
     - P_j^{out} — продукты, производимые задачей $j \in J$.
     - $P^{`}$ — продукты, которые надо произвести к концу (заказ).
   - T — множество временных промежутков
   - J — множество задач (задача есть производство некоторого вещества по его рецепту). Индексы у этого множества означают:
     - J_p^{in} — задачи, потребляющие продукт $p \in P$.
     - J_p^{out} — задачи, производящие продукт $p \in P$.
     - J^u — задачи, которые могут быть произведены на узле $u \in U$.
2. Параметры:
   - B_u^{min}, B_u^{max} — минимальный и максимальный размеры пакета для запуска узла $u \in U$.
   - S_p^{max} — максимальное количество продукта $p \in P$, которое может находиться на складе в любой момент времени.
   - D_p — внешние требования на производство продукта $p \in P^{`}$.
   - I_p — изначальное количество продукта $p \in P$ на складе.
   - $\mathcal{T}_{u,j}$ — время работы задачи $j \in J$ на узле $u \in U$. По умолчанию считается, что время исполнения задачи не зависит от размера пакета. Будут рассмотренны случаи, в которых время может зависеть от размера пакета, в таком случае этот параметр значит «время работы за килограмм входных веществ».
   - $\mathcal{Q}_{p,j}^{in}, \mathcal{Q}_{p,j}^{out}$ — пропорции входного/выходного продукта $p \in P$ в пакете в рецепте задачи $j \in J$ в случае мультипотребления/мультипроизводства.

В моделях будет рассматриваться целевая функция «Makespan» — общее время производства:

$$\begin{gathered} min\, MS \\ s.t. \\ MS \leqslant T_{u,i}\ \forall u \in U, i \in \mathbb{N} \end{gathered} \tag{1}$$

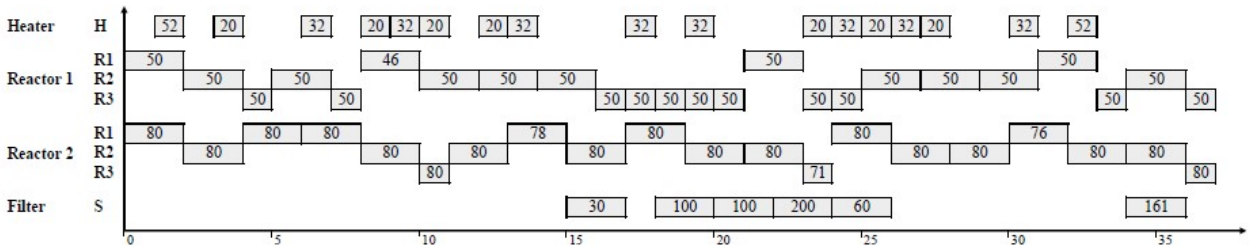
Минимизация общего времени выполнения (makespan) при условиях, что оно больше, чем время окончания каждого запуска каждого узла — $T_{u,i}$. В зависимости от представления времени в модели величина справа будет по-разному выражаться из переменных.

4 Вычислительный эксперимент

Эксперимент состоит в том, что практическая задача будет закодирована разным способом в набор решающих переменных и ограничений и подана на вход COIN-OR Branch-and-Cut (CBC) алгоритму, а далее полученные значения будут интерпретированы и визуализированы в виде диаграмм Ганта. Цель эксперимента состоит в том, чтобы сравнить разные алгоритмы на разных данных и проинтерпретировать полученные результаты. Способ кодирования плана для солвера и интерпретации зависит от подхода к моделированию времени и является основным объектом исследований. Пример диаграммы Ганта можно найти в [5] и на рисунке 3. Эта диаграмма соответствует некоторому плану для процесса 1. По оси Ox отложено время, а по оси oY — производственные узлы. Прямоугольники — это запуски задачи, а числа в них обозначают размер пакета для обработки. Такие диаграммы помогают визуально понять качество расписания и работы решающих алгоритмов.

Основные метрики сравнения подходов к моделированию: количество переменных, количество ограничений, качество расписания, получаемого за ограниченное время, время, необходимое для получения оптимального расписания.

Рис. 3 Пример диаграммы Ганта



5 Модели

5.1 Базовая модель с дискретным временем

Для начала опишем ещё одну классификацию рассматриваемых моделей. Как было описано ранее, от алгоритма ожидаются ответы на два основных вопроса: Когда запускать задачи на узлах и какое количество вещества подавать на вход. Размер пакета кодируется достаточно просто — это переменная непрерывного типа (нет требования на целочисленность). О времени есть два основных подхода, хорошо описанные в [reallife]: моделирование **порядка** запусков на узлах и моделирование фиксированного **времени** начала запуска. Говоря формально, в первом случае вводятся переменные-индикаторы того, что один процесс начинается раньше другого, а потом выбираются времена как непрерывные переменные. Во втором случае временная шкала делится на периоды и вводятся переменные-индикаторы того, что процесс начался в заданный момент. После этого выбираются времена, как непрерывные переменные. Помимо этого, можно классифицировать второй тип дальше: считаем ли мы ширину промежутка фиксированной (между точками на шкале проходит одинаковый период времени) или плавающей (длина промежутка или точное значение момента времени — переменная).

Базовая модель, описываемая в этой части будет принадлежать второму типу: мы поделим заранее шкалу на моменты времени и введём переменные для каждого запуска в

каждый момент времени. Эта модель взята из [1] и критикуется за большое число переменных в случае плотной временной сетки и низкое качество в случае разряженной.

Опишем переменные детальнее:

1. MS — общее время работы системы.
2. $b_{u,j,t}$ — общее количество вещества, потребряемого процессом $j \in J$ на узле $u \in U$ в момент времени $t \in T$, где T — натуральные числа некоторого отрезка (времена на шкале). Заранее подбирается константа, являющаяся некоторой оценкой на общее время работы.
3. $s_{p,t}$ — количество вещества $p \in P$ на складе в момент времени $t \in T$. Считается, что $s_{p,0}$ даны (начальное состояние складов).
4. $x_{u,j,t}$ — индикатор того, что задача $j \in J$ началась на узле $u \in U$ в момент времени $t \in T$.

Опишем ограничения:

1. $MS \geqslant x_{u,j,t}t + \mathcal{T}_{u,j}\ \forall u \in U, j \in J, t \in T$ — общее время работы не меньше, чем время окончания каждого процесса.
2. $x_{u,j,t}B_u^{min} \leqslant b_{u,j,t} \leqslant x_{u,j,t}B_u^{max}\ \forall u \in U, j \in J, t \in T$ — пакет, потребляемый процессом $j \in J$ на узле $u \in U$ лежит между максимальным и минимальным размерами, которые узел может принять. Если процесс не запускается, то вещества он не потребляет.
3. $s_{p,t} = s_{p,t-1} + \sum\limits_{j \in J_p^{out}, u \in U_j, t - \mathcal{T}_{u,j} \geqslant 1} \mathcal{Q}_{p,j}^{out} b_{u,j,t} - \sum\limits_{j \in J_p^{in}, u \in U_j, t \geqslant 1} \mathcal{Q}_{p,j}^{in} b_{u,j,t}\ \forall t \geqslant 1, p \in P$ — уравнения баланса склада. Количество вещества в момент времени t есть количество вещества в момент $t-1$ плюс то, что успели к этому моменту произвести, минус количество, которое потребляют начатые процессы.
4. $0 \leqslant s_{p,t} \leqslant S_p^{max}$ — ограничения на объем хранящихся на складе веществ.
5. $\sum\limits_{j \in U_j, t' \in [t - \mathcal{T}_{u,j}, t]} x_{u,j,t'} \leqslant 1\ \forall t \in T, u \in U$ — ограничения одновременности. В момент времени t узел u выполняет не более одной задачи.
6. $s_{p,0} = I_p\ \forall p \in P$ — начальное состояние складов.
7. $s_{p,max(T)} \geqslant D_p\ \forall p \in P$ — требование на заказ.

При использовании этого подхода к процессу, изображенному на диаграмме 1 и данных, описанных в [5] было получено 20860 решающих переменных. За 10 минут работы было найдено расписание, затрачивающее 60 рабочих часов, что уступает почти в 2 раза результатам, полученным в оригинальной работе [5].

5.2 Двухступенчатая схема.

Основная проблема базового подхода к моделированию состоит в том, что из-за плотной шкалы времени число переменных становится очень большим. Также сложность получаемых моделей растет не от количества запусков задач на производственных узлах, а преимущественно от их длительности (так как увеличивается количество точек на шкале). Это делает модель абсолютно неприменимой в случае, если makespan растёт.

Однако для того, чтобы записать уравнения баланса склада, которые представляют основное ограничения на запуски процессов, достаточно знать не точное время запуска, а порядок запусков. Это наводит на мысль о том, что можно разрядить временную сетку, позволив системе очень грубо расставить задачи на временной шкале. Очевидно, этот подход даст расписание малого качества, так как задачам станет запрещено запускаться в произвольное время. Однако, получив приближенное расписание, можно решить задачу

оптимизации на второй фазе и, зная размеры пакетов всех задач постараться разместить задачи максимально плотно, чтобы минимизировать общее время производства. Более детально этот метод моделирования описан в оригинальной статье [1], в которой он был впервые предложен.

Первая фаза совпадает с базовой моделью с дискретным временем из 1. Единственное отличие, что теперь множество временных меток T содержит не всевозможные времена начала задачи, а только некоторые (например, кратные некоторому значению). **Шириной** временной сетки будем называть расстояние между соседними точками (подразумевается, что оно одинаковое). Заметим, что если взять ширину сетки равной максимальному времени, то задача составления расписания упрощается, так как на одной машине при любом выборе времени начала задачи сами задачи не пересекутся во времени. Это позволяет не писать отдельно ограничения. Выбор ширины сетки представляет собой отдельную исследовательскую задачу, так как ведёт к упрощению модели, но ухудшению качества расписания. Это явление в английской литературе называется «tradeoff».

На второй фазе ставится задача оптимизации, однако размеры пакетов для задач считаются известными, что упрощает модель. Также грубое приближение даёт некоторые ограничения на максимальные и минимальные значения времени старта и окончания задачи.

Переменные:

1. MS — общее время работы системы.
2. $S_{n,u}, F_{n,u}$ — минимальные времена старта и окончания n-той задачи, работающей на узле $u \in U$. В отличие от первой фазы, тут задачи не делятся по типу, так как тип задачи уже не важен. В первой фазе алгоритм определил задачи и их последовательность, а так же потребляемые материалы, а значит теперь необходимо их просто плотнее поставить на шкале. Значит, решающих переменных стало меньше
3. $s_{n,u,t}, f_{n,u,t}$ — «переменные Хевисайда», индикаторы того, что $S_{n,u} \leqslant t$ и $F_{n,u} \leqslant t$. На сей раз t берется не из разряженной шкалы, а из полной. Эти переменные требуются для описания баланса склада — для момента времени t можно выразить всё, что на склад было отправлено как произведение размера пакета операции на индикатор того, что операция успела начаться, а так же всё, что с склада было извлечено.
4. $stock_{p,t}$ — объем вещества на складе к моменту времени t.

Будем обозначать $\overline{S}_{n,u}, \underline{S}_{n,u}$ — верхняя и нижняя границы на время начала. Аналогичные ограничения можно ввести на время окончания. Верхней границей на время начала окончания будет полученное время на первой фазе: цель второй фазы состоит в том, что алгоритм располагает плотнее задачи на шкале. Очевидно, ожидается, что новое время начала будет меньше предыдущего. Нижняя граница считается тривиально: $\underline{S}_{n,u} = \sum_{m<n} \mathcal{T}_{u,j_m}$ — сумма времён выполнения всех предыдущих задач. $\underline{T}_{n,u} = \underline{S}_{n,u} + \mathcal{T}_{u,j_n}$ — нижняя граница на время окончания задачи получается из границы на время старта прибавлением длительности задачи.

Ограничения:

1. $MS \geqslant F_{u,n}$ — общее время работы не меньше, чем время окончания каждого процесса.
2. $S_{n,u} \geqslant F_{n-1,u}$ — очередная задача начинается не раньше, чем закончится предыдущая.
3. $F_{n,u} = S_{n,u} + \mathcal{T}_{u,j_n}$ — время окончания это время начала плюс длительность задачи.
4. $s_{n,u,t} \leqslant \frac{t-S_{n,u}}{max(T)} + 1$, $s_{n,u,t} \geqslant \frac{t-S_{n,u}+1}{max(T)}$, $f_{n,u,t} \leqslant \frac{t-F_{n,u}}{max(T)} + 1$, $f_{n,u,t} \geqslant \frac{t-F_{n,u}+1}{max(T)}$ — «переменные Хевисайда» действительно являются индикаторами того, событие начала и конца вы-

полнения задачи произошло не раньше некоторого времени. $max(T)$ — максимальное рассматриваемое время.

5. $s_{n,u,t} = 0$ при $t < \underline{S}_{n,u}$ и $s_{n,u,t} = 1$ при $t \geqslant \overline{S}_{n,u}$. $f_{n,u,t} = 0$ при $t < \underline{F}_{n,u}$ и $f_{n,u,t} = 1$ при $t \geqslant \overline{F}_{n,u}$ — ограничения на минимальное и максимальное время старта и окончания, описанное выше.
6. $stock_{p,t} = s_{p,0} + \sum\limits_{u \in U, n \in N_u} q^{out}_{u,n,p} f_{n,u,t} - q^{in}_{u,n,p} s_{n,u,t}$, где $q^{in}_{u,n,p}, q^{out}_{u,n,p}$ — потребляемое и производимое количество материала n-той операцией, производимой на узле u. Эти величины можно посчитать, используя размеры пакетов $b_{u,j,t}$, посчитанные в первой фазе. Для этого следует выбрать те моменты времени t, когда задача действительно была запущена (пакет имеет ненулевой размер), отсортировать пакеты по времени и назначить номера задачам, а также умножить размер пакета на $\mathcal{Q}^{in}_{p,j}$ или $\mathcal{Q}^{out}_{p,j}$ — долю содержания в нём нужного вещества.
7. $0 \leqslant stock_{p,t} \leqslant S^{max}_p$ — ограничение на корректность склада в любой момент времени.
8. $stock_{p,max(T)} \geqslant D_p$ — требование на заказ.

Основной выигрыш данного подхода состоит в разбиении задачи составления на две подзадачи: определение примерного порядка и размера пакета для каждой задачи и уточнение времён запуска для достижения оптимальности. Именно из-за использования на втором этапе грубого приближения из первого теряется оптимальность расписания.

5.3 Модель порядка.

В прошлом разделе была предложена эвристика построения моделей, основанная на прореживании временной сетки. Как было замечено ранее, расписание, полученное на первом этапе, является некоторым приближением, которое можно получить при сравнительно малом размере модели. Детальное рассмотрение модели с прореженной сеткой показывает, что для записи ограничений требуется знать не точное время начала и конца задачи, запускаемой на узле, а порядок задач на разных узлах. Действительно, ограничения 3 и 5 из 1, представляющие основные практические требования: отсутствие одновременности (не более одной задачи исполняется на одном узле за раз) и корректность склада (он не перегружен и всякая задача может потребить заявленное в плане число вещества), оперируют переменными, относящимися к задачам, что исполнились раньше. Фиксируя времена начала и конца, модель явно пользуется наличием линейного порядка на временной шкале, но платит за это наличием бинарных переменных, относящихся к каждой точке на шкале.

Эти соображения наводят на мысль о том, что можно кодировать не время, а порядок запуска задач на узлах. Тогда вместо бинарных индикаторов начала работы задачи в определенный момент времени можно будет записать индикаторы того, что задачи запускаются в некотором порядке. После этого можно будет так выбрать времена запуска, чтобы все ограничения порядка были удовлетворены. Заметим, что в таком случае времена начала и конца работы задачи выбираются уже из непрерывного множества (а не дискретного), то есть сами переменные перестают быть целочисленными в общем случае. Будем называть начало и конец запуска процесса **событиями** После того, как порядок событий определён, выписать ограничения на параллельность и корректность склада становится просто: в первом случае само наличие порядка избавляет от параллельности, а во втором надо просто просуммировать производство и потребление прекурсора по всем событиям, произошедшим раньше данного.

Замечание Если склад корректен во времена всех событий, то он корректен всегда. Действительно, между событиями с ним ничего не происходит, значит утверждение верно.

Это замечание позволяет гарантировать корректность построенного расписания.

Эта модель была вдохновлена работой [7], где описана модель с схожей структурой и ограничениями.

Перед описанием модели необходимо описать специальный класс «индикаторных» ограничений. Пусть дано два выражения a и b, представляющие собой линейные комбинации переменных и бинарная переменная p, равная истинности выражения $a \leqslant b$. То есть, $p = 1 \Leftrightarrow a \leqslant b$. Это можно записать при помощи двух неравенств: $p \leqslant \frac{b-a}{M} + 1$, где M — число, заведомо большее, чем $|a| + |b|$. Это неравенство заведомо ложно при $p = 1$ и $a > b$. Осталось написать обратную импликацию. Заметим, что этот случай сводится к прямой импликации рассмотрением выражения $1 - p$: $1 - p \leqslant \frac{a-b-1}{M} + 1$. По аналогии с первым неравенством, оно заведомо ложно при $p = 0$ и $b + 1 > a$. В силу того, что все длительности задач целочисленные, $b + 1 > a \Leftrightarrow b \geqslant a$.

Таким образом, при помощи линейных ограничений можно выразить то, что одно выражение не больше другого. Так как в отличии от первых двух моделей, где системе было разрешено самой заполнять булеву маску (в какое время начинать задачи), в данном случае система для каждого запуска выбирает время его начала и конца. Следовательно, требуется для каждого запуска выделить некоторый набор переменных. Однако заранее неизвестно, сколько раз нужно запустить каждую задачу (как минимум по причине того, что размер пакета — переменная). Значит, нужно будет разрешить модели некоторые запуски делать фиктивными. Фиктивные задачи нужны для того, чтобы модель могла сама решить, сколько запусков ей нужно — остальные используют нулевой пакет вещества, производят нулевой и не ограничивают общее время работы системы.

Эта модель подразумевает, что продукты не потребляются, а только производятся.

Теперь можно выписать решающие переменные:

1. MS — общее время работы системы.
2. $begin_{u,j,n}, end_{u,j,n}$ — время начала и конца n-того запуска задачи j на узле u
3. $active_{n,u,j}$ — является ли n-тый запуск задачи фиктивным или нет.
4. $batch_{n,u,j}$ — размер пакета для запуска.
5. $begin\_begin_{n_1,u_1,j_1,n_2,u_2,j_2}$ — верно ли, что событие «начало одного запуска одной задачи» произошло не позже события «начало другого запуска другой задачи». Эти переменные являются индикаторами того, что $begin_{u_1,j_1,n_1} \leqslant begin_{u_2,j_2,n_2}$. Аналогично можно ввести индикаторы других пар событий.
6. $begin\_end_{n_1,u_1,j_1,n_2,u_2,j_2}$
7. $end\_begin_{n_1,u_1,j_1,n_2,u_2,j_2}$
8. $end\_end_{n_1,u_1,j_1,n_2,u_2,j_2}$
9. $s_{p,begin,begin,n_1,u_1,j_1,n_2,u_2,j_2}$ — количество вещества p, которое будет взято со склада n_1-ым запуском задачи j_1 на узле u_1 до начала n_2-го запуска задачи j_2 на узле u_2. Это значение может быть либо равно нулю, если индикатор соответствующего события нулевой (то есть, первый запуск происходит позже второго), либо равно $\mathcal{Q}^{in}_{p,j} batch_{n_1,u_1,j_1}$ — количеству вещества, отгружаемого со склада. Аналогично можно ввести переменные, отражающие количество вещества, взятого/добавленного на склад одним запуском до начала/конца другого. Считается, что все задачи в начале только забирают вещества со склада, а в момент окончания только отгружают продукт.

10. $stock_{p,begin,n,j,u}, stock_{p,end,n,j,u}$ — количество вещества на складе в момент начала/конца запуска задачи на узле.

Опишем ограничения:

1. Если $active_{n,u,j}$, то $MS \geqslant end_{n,u,j}$ — общее время работы не меньше, чем время окончания каждого нефиктивного процесса.
2. $begin_{n,u,j} \geqslant end_{n-1,u,j}$ — очередная задача начинается не раньше, чем закончится предыдущая.
3. $end_{n,u,j} = begin_{n,u,j} + \mathcal{T}_{u,j}$ — время окончания это время начала плюс длительность задачи.
4. $active_{n,u,j} B_u^{min} \leqslant batch_{n,u,j} \leqslant active_{n,u,j} B_u^{max}$ — Фиктивные задачи не потребляют вещества, а нефиктивные потребляют не больше и не меньше задаваемых ограничений.
5. $s_{p,begin,begin,n_1,u_1,j_1,n_2,u_2,j_2} = batch_{n_1,u_1,j_1} \mathcal{Q}^{in}_{p,j_1} \Leftrightarrow begin\_begin_{n_1,u_1,j_1,n_2,u_2,j_2} = 1$ и $s_{p,begin,begin,n_1,u_1,j_1,n_2,u_2,j_2} = 0 \Leftrightarrow begin\_begin_{n_1,u_1,j_1,n_2,u_2,j_2} = 0$ — такого рода ограничения создаются по описанному в замечании принципу.
6. Бинарные ограничения порядка для упорядочивания событий начала и конца всех запусков всех задач.
7. $stock_{p,begin,n,j,u} = I_p + \sum\limits_{j_1 \in J_p^{out},n_1,u_1} s_{p,begin,end,n,u,j,n_1,u_1,j_1} - \sum\limits_{j_1 \in J_p^{in},n_1,u_1} s_{p,begin,begin,n,u,j,n_1,u_1,j_1}$ — когда зафиксирован приток и отток веществ между задачами, легко записать состояние склада — сумма всех произведенных веществ минус сумма всех потреблённых.
8. $0 \leqslant stock_{p,begin,n,j,u} \leqslant S_p^{max}$
9. $\sum\limits_{j \in J_p^{out},n,u} batch_{n,u,j} \mathcal{Q}^{out}_{p,j} \geqslant D_p$ — для всех продуктов p.

В этой модели уже нет зависимости от временной шкалы и длительности каждой задачи, однако теперь появилось много бинарных индикаторов порядка.

6 Результаты

В предыдущем разделе было описано 3 модели для получения расписаний производства. Эти алгоритмы были реализованы в `https://gitlab.com/Vitalgor/semanticmes`. В этом же репозитории находится скрипт, который строит STN-сеть для процесса по его json-описанию, а так же удобный инструментарий для генерации json-описания. Там же находятся данные всех использованных процессов. Основными критериями оценки качества модели были сложность (число переменных и ограничений) и makespan расписания, полученное за заранее отведённое время.

В таблице 1 показаны числовые характеристики моделей, полученных в результате запуска описанных алгоритмов на процессе 1 с различными требованиями на производство. Эксперимент показал, что двухступенчатая схема работает значительно лучше базовой модели с дискретным временем при всех росте требований на производство, позволяя получить грубое приближения расписания моделью с малым числом параметров. На втором этапе переменных больше, однако получаемые на первом этапе ограничения (см 1) позволяют зафиксировать большую их часть, что делает процесс уточнения (левого сдвига) расписания проще и быстрее. Этот эффект также описан в оригинальной статье [?].

Схема, основанная на порядке, проигрывает двухступенчатой, что видно из таблицы, так как STN-граф процесса плотный. Это позволяет сформулировать некоторый эмпирический критерий, полезный для внедрения описанных моделей в реальный производственный процесс, а именно:

Модель	Demand 500, время 50	Demand 200, время 30	Demand 50, время 10	Общий случай
Дискретная	1276/1859	776/1119	276/379	$O(Tn)$
Двухступенчатая	276+1471/379+1589	176+767/231+841	151+829/194+854	$O(\frac{Tn}{L_{max}}) + O(n + Tn)$
Порядковая	41431/121494	11905/34890	1735/5070	$O(n^2)$

Таблица 1 Количество переменных и ограничений для разных заказов на производство процесса 1

1. Следует использовать двухступенчатую схему, если процессы являются сетевыми, задачи делят общие склады, а само производство нельзя подразделить на этапы, выполняющиеся строго друг за другом и использующие продукты предыдущего этапа. Помимо этого, данный подход предпочтителен в случае, если задач много и они работают быстро. Для настройки сложности и получения качественных расписаний следует менять плотность моделирования: чем плотнее шкала, тем проще модель и менее оптимально расписание.
2. Следует использовать модель, основанную на порядке, если процесс подразделяется на этапы по очередности и с складом взаимодействует только один процесс-производитель и один процесс-потребитель. Этот подход предпочтительнее в случае малого числа задач, сравнимых по времени производства с всем процессом.

7 Заключение

В данной работе была произведена реализация некоторых подходов к моделированию производства и получению расписаний, описанных в разных работах для разных задач ([1], [2], [3], [7]). Полученные модели были испытаны на синтетических и реальных данных, для которых изначально не были разработаны.

Работа может быть полезна как исследователям в области автоматического планирования и массового обслуживания, так и разработчикам, желающим применить эти методы на практике реального производства. Также модели реализованы в достаточной общности, чтобы работать не только с описанными в оригинальных статьях процессами, но с произвольными. В прилагаемом репозитории присутствует генератор STN диаграммы по файлам, которые соответствуют сторогой системе типов, основанной на B2MML (описания системы типов прилагаются в репозитории). Язык B2MML, на котором основана типизация, разработанная для исследования описан в статье [8] Это позволяет продолжить экспериментальный запуск на новых данных и проанализировать результаты.

В продолжении этого исследования предлагается провести тонкую настройку параметров имеющихся методов с целью повышения их эффективности. Помимо этого, строгая типизация позволяет реализовать и другие методы, известные современной теории планирования, для того, чтобы они могли работать с произвольными задачами.

Литература

[1] *F. Blomer, H.-O. Gunther* LP-based heuristics for scheduling chemical batch processes, 2010 International Journal of Production Research, 38:5, 1029-1051 doi: `http://dx.doi.org/10.1080/002075400189004`.

[2] *Georgios P. Georgiadis, Georgios M. Kopanos, Antonis Karkaris, Harris Ksafopoulos and Michael C. Georgiadis* Optimal Production Scheduling in the Dairy Industries, 2019 Industrial & Engineering Chemistry Research 58 (16), 6537-6550 doi: http://dx.doi.org/10.1021/acs.iecr.8b05710.

[3] *Georgiadis, Georgios P. and Elekidis, Apostolos P. and Georgiadis, Michael C.* Optimization-Based Scheduling for the Process Industries: From Theory to Real-Life Industrial Applications, 2019 Industrial & Engineering Chemistry Research 58 (16), 6537-6550 doi: http://dx.doi.org/10.3390/pr7070438.

[4] *Siqun Wang, Monique Guignard* Hybridizing Discrete- and Continuous-Time Models For Batch Sizing and Scheduling Problems, 2006 Computers & Operations Research Volume 33, Issue 4 doi: http://dx.doi.org/10.1016/j.cor.2004.11.013.

[5] *Christos T. Maravelias and Ignacio E. Grossmann* Minimization of the Makespan with a Discrete-Time State-Task Network Formulation, 2003 Industrial & Engineering Chemistry Research doi: http://dx.doi.org/10.1021/ie034053b.

[6] *E.Kondili, C.C.Pantelides, R.W.H.Sargent* A general algorithm for short-term scheduling of batch operations—I. MILP formulation, 1993 Computers & Chemical Engineering doi: http://dx.doi.org/10.1021/ie034053b10.1016/0098-1354(93)80015-F.

[7] *C.A. Méndez, G.P. Henning, J. Cerdá* An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilitie, 2001 Computers & Chemical Engineering, Volume 25, Issues 4–6 doi: http://dx.doi.org/10.1016/S0098-1354(01)00671-8.

[8] *Harjunkoski Iiro, Bauer Reinhard* Sharing Data for Production Scheduling Using the ISA-95 Standard, 2014 Frontiers in Energy Research, Volume 2, Pages 44 doi: http://dx.doi.org/10.3389/fenrg.2014.00044.

Поступила в редакцию