ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ПРОГРАММНОЕ И АППАРАТНОЕ ОБЕСПЕЧЕНИЕ РАСПРЕДЕЛЕННЫХ И СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМ

УДК 519.687.1:004.75
10.25209/2079-3316-2026-17-3-275-314

Компактная MILP-модель для планирования многостадийных графов вычислений с однородными зависимостями в облаке

Юрий Дмитриевич **Семенов**[1✉], Олег Викторович **Сухорослов**[2]

[1] Высшая школа экономики, Москва, Россия
[2] Институт проблем передачи информации им. А. А. Харкевича РАН, Москва, Россия
[1✉] *ydsemenov@hse.ru*

Аннотация.

В статье рассматривается задача планирования ориентированных ациклических графов (DAG) вычислительных задач в облачной среде с целью минимизации длительности расписания или стоимости выполнения. Важным направлением исследований, связанных с решением сложных задач планирования, является разработка моделей смешанно-целочисленного линейного программирования (MILP), которые могут использоваться с точными решателями или служить основой для построения линейных релаксаций. Существующие MILP-модели для планирования DAG в облаке содержат большое число переменных и ограничений и известны как чрезвычайно трудные для решения. Хотя в общем случае улучить такие модели сложно, графы, встречающиеся на практике, часто обладают выраженными структурными свойствами, которые можно использовать для построения более эффективных моделей.

В работе такие свойства используются для разработки более компактной и лучше поддающейся решению MILP-модели для графов, состоящих из нескольких стадий (слоёв) с однородными зависимостями по данным. Преимущества предложенной модели демонстрируются с помощью вычислительных экспериментов.

Ключевые слова и фразы: планирование графов вычислений, облачные вычисления, оптимизационное моделирование, смешанно-целочисленное линейное программирование

Благодарности: Работа выполнена в рамках государственного задания ИППИ РАН, утвержденного Минобрнауки России.

© Семенов Ю. Д., Сухорослов О. В. 2026

Для цитирования: Семенов Ю. Д., Сухорослов О. В. *Компактная MILP-модель для планирования многостадийных графов вычислений с однородными зависимостями в облаке* // Программные системы: теория и приложения. 2026. **Т. 17**. № 3(72). С. 275–314. https://psta.psiras.ru/read/psta2026_3_275-314.pdf

1. Введение

Параллельные и распределённые приложения часто моделируются как направленные ациклические графы (DAG) вычислительных задач с зависимостями по данным между ними. Такие приложения возникают в научных вычислениях [1], распределённом машинном обучении [2] и больших данных [3]. Задача планирования графа заключается в определении для каждой задачи машины, на которой она будет выполняться, и времени начала выполнения. Традиционно алгоритмы планирования графа были сосредоточены на минимизации времени выполнения графа. Однако если граф выполняется в облаке, а не в локальном кластере, возникает другая важная метрика: стоимость выполнения. Хотя заманчиво оптимизировать оба критерия одновременно, такой подход обычно даёт несколько несравнимых Парето-эффективных расписаний, и выбор окончательного расписания требует вмешательства лица, принимающего решения, что делает его непригодным для автономных систем планирования. Поэтому мы рассматриваем однокритериальный случай, когда одна метрика оптимизируется при ограничении на другую метрику (бюджет или срок выполнения) или даже вообще без учёта другой метрики.

Основным вкладом этой статьи является улучшенная MILP-модель для частного случая многостадийных графов с однородными зависимостями по данным. Как будет показано далее, этот случай имеет определённую практическую значимость. Модель сравнивается с общей MILP-моделью для планирования графов, представленной в [4].

2. Постановка задачи

2.1. Модель графа задач

Направленный ациклический граф $G = (T, D)$ состоит из множества задач T (вершин графа) и множества зависимостей по данным D (рёбер графа). Для каждой задачи t известно время её выполнения на каждой машине v, обозначаемое как $p_{t,v}$. Обычно время выполнения получается делением размера задачи во флопсах на скорость машины. Однако мы используем более общую модель, поскольку хотим, чтобы наша модель была применима к более сложным сценариям, когда время выполнения не масштабируется линейно с производительностью машины (например, задачи, ограниченные вводом-выводом).

Рёбра графа соответствуют зависимостям по данным между задачами. Каждое ребро представляется тройкой (i, j, w), означающей, что задача i требует w байт данных, произведённых задачей i. Естественно, задача не может начать выполнение, пока не получит все необходимые данные. Если задача, произведшая данные, была выполнена на другой машине, данные должны быть переданы по сети, что вызывает коммуникационную задержку. Мы называем две задачи независимыми, если между соответствующими вершинами в графе нет направленного пути ни в одном направлении.

На практике некоторые задачи могут получать внешние входные данные и отправлять выходные данные во внешнее хранилище. В существующих исследованиях это обычно включается в модель путём добавления одной входной задачи, которая распределяет входные данные, и одной выходной задачи, которая собирает выходные данные. Мы следуем тому же подходу.

2.2. Многостадийные графы с однородными зависимостями

Ниже рассматриваются графы с определёнными структурными свойствами. Мы называем граф «многостадийным графом с однородными зависимостями», если выполняется следующее:

- Вершины графа можно разбить на k стадий $L_1, \ldots, L_k$ так, что все задачи внутри одной стадии независимы, для всех пар соседних стадий (L_i, L_{i+1}) любая задача из L_{i+1} непосредственно зависит от любой задачи из L_i, и все такие зависимости имеют одинаковый размер.
- Могут существовать зависимости между несоседней парой стадий, но в этом случае должны выполняться те же условия, что и выше (зависимости одинакового размера между всеми парами задач).

Как будет показано далее, некоторые графы, возникающие в практических приложениях, соответствуют этому определению с точностью до точной аппроксимации. Под точной аппроксимацией мы подразумеваем, что зависимости могут на самом деле не иметь одинаковый размер, но разница между ними пренебрежимо мала (например, десятки или сотни байт), что на практике не влияет на время передачи по сети.

Пример многостадийного графа с однородными зависимостями показан на рисунке 1.

2.3. Модель облачных ресурсов

Мы рассматриваем сценарий, в котором граф выполняется на наборе виртуальных машин (ВМ), арендованных у IaaS-облачного провайдера. Наиболее распространённая модель облачных ресурсов IaaS, используемая в существующих исследованиях, заключается в следующем. Пользователь

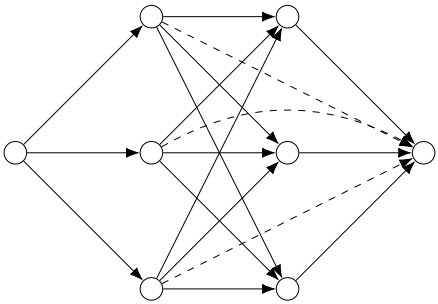


Рисунок 1. Пример многостадийного графа с однородными зависимостями. Сплошные стрелки обозначают зависимости между соседними стадиями, а пунктирные стрелки обозначают зависимости между несоседними стадиями.

имеет доступ к набору виртуальных машин V, где каждая машина принадлежит к одному из типов, предлагаемых провайдером (то есть набор машин обычно гетерогенен, но некоторые машины могут иметь эквивалентные характеристики). Каждый тип ВМ характеризуется ценой c_v и производительностью. Как мы установили ранее, мы не задаём производительность явно, а определяем её через то, насколько быстро каждая задача выполняется на данной ВМ. Мы предполагаем, что машина может выполнять одну задачу за раз, и задачи не могут быть прерваны во время выполнения.

Мы предполагаем, что передача зависимостей между задачами, выполняемыми на разных машинах, осуществляется через общее облачное хранилище (например, Amazon S3). Данные, произведённые задачей, загружаются в хранилище сразу после их вычисления и извлекаются из указанного хранилища по требованию зависимыми задачами. Передача данных размера w по сети между машиной и хранилищем требует времени w/B, где B – пропускная способность сети. Мы предполагаем, что одновременные передачи не конкурируют за пропускную способность. Наши результаты также могут быть непосредственно адаптированы к модели без общего хранилища.

Облачные провайдеры обычно предлагают ВМ на основе модели оплаты по факту использования с интервалом биллинга P в качестве параметра. Если ВМ v создана в момент времени svm_v и удалена в момент времени fvm_v, стоимость аренды машины вычисляется как

$$C_v = c_v \cdot \lceil \frac{fvm_v - svm_v}{P} \rceil. \tag{1}$$

Общая стоимость выполнения графа затем рассчитывается как сумма индивидуальных затрат на аренду ВМ. Мы не учитываем затраты на общее хранилище, поскольку для вычислительно-интенсивных графов они пренебрежимо малы по сравнению с затратами на аренду ВМ [4].

Существуют некоторые естественные ограничения, которые должны выполняться при вычислении времени создания и удаления ВМ. ВМ должна быть запущена во время всех выполнений задач, запланированных на данной ВМ, и во время всех передач данных на данную ВМ или с неё.

3. MILP модели

3.1. Базовая модель

Прежде чем представить MILP-модель для многостадийных графов с однородными зависимостями, мы опишем общую MILP-модель для планирования графов задач в облаке, представленную в [4]. Нам потребуются дополнительные обозначения.

Для простоты мы переопределяем множество зависимостей D так, чтобы оно содержало тройки $(i, j, \Delta t)$, где Δt обозначает время передачи данных в/из хранилища, вместо троек (i, j, w), где w обозначает размер данных. $\mathcal{N}$ обозначает множество всех пар (i, j) задач, между которыми нет пути в графе ни от i к j, ни наоборот.

U обозначает верхнюю границу времени выполнения графа, вычисляемую как сумма по всем задачам максимального времени выполнения задачи и максимального времени передачи данных по её исходящим зависимостям, включая как загрузку в общее хранилище, так и скачивание из него. Эта граница намеренно выбрана с большим запасом надёжности, так что её достаточно для всех ограничений big-M, используемых ниже. Наконец, ε обозначает очень малое положительное число, используемое для преобразования строгого неравенства в нестрогое.

Для простоты мы предполагаем, что граф имеет единственную выходную задачу, обозначаемую out. Если это не так, то можно ввести новую задачу размера 0 и добавить зависимости нулевого веса от всех выходных задач к этой новой задаче.

Модель содержит следующие переменные:

$x_{i,v} \in \{0, 1\}$ равно 1 тогда и только тогда, когда задача i размещена на ВМ v.

$st_i \in \mathbb{R}_+$ обозначает время начала выполнения задачи i.

$ft_i \in \mathbb{R}_+$ обозначает время завершения выполнения задачи i.

$\omega_{i,j} \in \{0, 1\}$ фиксирует перекрытие между независимыми задачами i и j. Если выполнения этих двух задач перекрываются, то и $\omega_{i,j}$, и $\omega_{j,i}$ устанавливаются в 1.

$svm_v \in \mathbb{R}_+$ обозначает время начала работы ВМ v.

$fvm_v \in \mathbb{R}_+$ обозначает время завершения работы ВМ v.

$I_v \in \mathbb{Z}_+$ обозначает число интервалов биллинга, использованных ВМ v.

$M \in \mathbb{R}_+$ обозначает время выполнения графа.

$C \in \mathbb{R}_+$ обозначает стоимость выполнения графа.

Модель содержит следующие ограничения:

$$\sum_{v \in V} x_{i,v} = 1 \qquad \forall i \in T \tag{2}$$

$$st_i + \sum_{v \in V} p_{i,v} \cdot x_{i,v} = ft_i \qquad \forall i \in T \tag{3}$$

$$ft_j - st_i \leqslant U \cdot \omega_{i,j} \qquad \forall (i,j) \in \mathcal{N} \tag{4}$$

$$\omega_{i,j} + \omega_{j,i} + x_{i,v} + x_{j,v} \leqslant 3 \qquad \forall (i,j) \in \mathcal{N}\ \forall v \in V \tag{5}$$

$$svm_v \leqslant fvm_v \qquad \forall v \in V \tag{6}$$

$$svm_v \leqslant st_i + U \cdot (1 - x_{i,v}) \qquad \forall v \in V\ \forall t \in T \tag{7}$$

$$fvm_v + U \cdot (1 - x_{i,v}) \geqslant ft_i \qquad \forall v \in V\ \forall i \in T \tag{8}$$

$$I_v \cdot P \geqslant fvm_v - svm_v \qquad \forall v \in V \tag{9}$$

$$(I_v - 1) \cdot P \leqslant fvm_v - svm_v - \varepsilon \qquad \forall v \in V \tag{10}$$

$$M = ft_{out} \tag{11}$$

$$C = \sum_{v \in V} c_v \cdot I_v \tag{12}$$

$$st_s \geqslant ft_p + 2\Delta t \cdot (x_{p,v} - x_{s,v}) \qquad \forall v \in V\ \forall (p, s, \Delta t) \in D \tag{13}$$

$$svm_v \leqslant st_s + U \cdot (1 - x_{s,v}) - \Delta t \cdot (x_{s,v} - x_{p,v}) \qquad \forall v \in V\ \forall (p, s, \Delta t) \in D \tag{14}$$

$$fvm_v + U \cdot (1 - x_{p,v}) \geqslant ft_p + \Delta t \cdot (x_{p,v} - x_{s,v}) \qquad \forall v \in V\ \forall (p, s, \Delta t) \in D \tag{15}$$

Ограничение (2) гарантирует, что каждая задача выполняется ровно на одной ВМ. Ограничение (3) связывает время завершения задач с временем их начала. Ограничения (4) и (5) обеспечивают корректность переменных перекрытия. Ограничения (6)–(8) устанавливают время запуска и завершения ВМ, не включая передачи данных. Ограничения

(9) и (10) обеспечивают корректный подсчёт количества биллинговых интервалов. Ограничения (11) и (12) определяют время и стоимость выполнения соответственно. Ограничения, учитывающие передачи данных, различаются для каждой стратегии передачи данных. Ограничение (13) гарантирует, что никакая задача не начинает выполнение до получения всех данных от своих родительских задач. Ограничение (14) гарантирует, что все передачи данных на ВМ лежат в пределах времени её жизни. Ограничение (15) делает то же самое, но для передач с ВМ. Наконец, целевые функции, а также ограничения на бюджет и срок выполнения тривиально выражаются с использованием переменных C и M.

Переменные и ограничения, представленные выше, могут также использоваться для построения классической модели планирования графа (только с минимизацией времени выполнения). Для этого следует удалить переменные svm_v, fvm_v, I_v, C. Что касается ограничений, следует сохранить 2-5, 11 и 13.

3.2. Модель для многостадийных графов

В дальнейшем мы предполагаем, что на множестве ВМ V задан произвольный линейный порядок. ВМ сравниваются согласно этому порядку и нумеруются в соответствии с ним числами $1, \ldots, |V|$. Допуская некоторую вольность в обозначениях, мы иногда будем отождествлять ВМ из множества V с индексами $1, \ldots, |V|$.

Ключевое наблюдение, позволяющее построить более компактные модели для многостадийных графов с однородными зависимостями, заключается в следующем.

Лемма 1. *Для любого корректного расписания выполняются следующие свойства:*

1. *Если $i < i'$, то каждая задача $j \in L_i$ завершается не позже, чем начинается каждая задача $j' \in L_{i'}$.*
2. *Если задачи j и j' стадии L_i выполняются на ВМ v без других задач между ними, всегда возможно и оптимально начать j' сразу после завершения j.*
3. *Предположим, что задача j стадии L_i начинается на ВМ v в момент времени t. Если задача $j' \in L_i$ начинается сразу после j, их можно поменять местами так, что j' начнётся в момент t, а j – сразу после j'. Эта операция сохраняет корректность расписания и не изменяет ни время выполнения, ни стоимость.*

Доказательство. *1*. По определению, между всеми парами задач для всех пар соседних стадий существуют зависимости. Это означает, что если $i < i'$, каждая задача из $L_{i'}$ прямо или косвенно зависит от каждой задачи из L_i, следовательно, каждая задача из $L_{i'}$ не может начаться, пока все задачи из L_i не завершены.

2. По определению, задачи j и j', принадлежащие одной стадии L_i, имеют одно и то же множество предшественников, и для каждого предшественника размер данных, отправляемых в j и j', предполагается одинаковым. Это означает, что если задача j получила все необходимые данные к моменту t, то задача j' также может получить свои данные. Следовательно, j' может начаться сразу после j. Устранение промежутка между задачами не может продлить время жизни какой-либо ВМ, поэтому ни время выполнения, ни стоимость не увеличиваются.

3. По той же логике, что и выше, обе задачи j и j' получили все необходимые данные к моменту t, поэтому обе могут начаться в момент t. Что касается задач, зависящих от j и j', каждая такая задача зависит от обеих j и j', и зависимости имеют одинаковый размер. Поскольку в обоих случаях последняя из j и j' завершается в момент $t + p_{j,v} + p_{j',v}$, перестановка не влияет на зависимые задачи. Время выполнения и стоимость остаются прежними, поскольку время жизни ни одной ВМ не затрагивается перестановкой. □

Лемма, по существу, утверждает, что почти для всех целей нам достаточно рассматривать лишь временные отрезки $[l_{i,v}, r_{i,v}]$, в течение которых задачи стадии i выполняются на ВМ v. Внутри отрезка задачи выполняются без простоев, и точный порядок задач не имеет значения, поскольку их можно свободно менять местами. Используя эту идею, мы строим нашу модель на основе следующих переменных:

$x_{j,v} \in \{0, 1\}$ равно 1 тогда и только тогда, когда задача j размещена на ВМ v.

$l_{i,v} \in \mathbb{R}_+$ обозначает время начала стадии i на ВМ v.

$r_{i,v} \in \mathbb{R}_+$ обозначает время завершения стадии i на ВМ v. Если никакая задача стадии i не выполняется на v, $r_{i,v} = l_{i,v}$.

$r^{<}_{i,v} \in \mathbb{R}$ обозначает максимальное время завершения стадии i на любой ВМ $v' < v$ или некоторое малое отрицательное число, если ни одна ВМ $v' < v$ не выполняет стадию i.

$r^{>}_{i,v} \in \mathbb{R}$ обозначает максимальное время завершения стадии i на любой ВМ $v' > v$ или некоторое малое отрицательное число, если ни одна ВМ $v' > v$ не выполняет стадию i.

$h_{i,v} \in \{0,1\}$ равно 1 тогда и только тогда, когда хотя бы одна задача из L_i размещена на ВМ v.

$\widehat{h}_{i,v} \in \{0,1\}$ равно 1 тогда и только тогда, когда хотя бы одна задача из L_i не размещена на ВМ v.

$svm_v \in \mathbb{R}_+$ обозначает время начала работы ВМ v.

$fvm_v \in \mathbb{R}_+$ обозначает время завершения работы ВМ v.

$I_v \in \mathbb{Z}_+$ обозначает число интервалов биллинга, использованных ВМ v.

$M \in \mathbb{R}_+$ обозначает время выполнения графа.

$C \in \mathbb{R}_+$ обозначает стоимость выполнения графа.

В дальнейшем для заданной стадии i через S_i мы обозначаем множество пар $(i', \Delta t)$ таких, что существуют рёбра от всех задач из L_i ко всем задачам из $L_{i'}$ с временем передачи Δt. Аналогично, множество пар $(i', \Delta t)$ таких, что существуют рёбра от всех задач из $L_{i'}$ ко всем задачам из L_i с временем передачи Δt, мы обозначаем через P_i. Этапы нумеруются от 1 до k, а $[i,k]$ обозначает числа $i, i+1, \ldots, k$. U обозначает ту же верхнюю границу времени выполнения, что и в базовой модели. Модель имеет следующие ограничения:

$$\sum_{v \in V} x_{j,v} = 1 \qquad \forall j \in T \tag{16}$$

$$l_{i,v} \geqslant r_{i-1,v} \qquad \forall i \in [2,k]\ \forall v \in V \tag{17}$$

$$l_{i,v} + \sum_{j \in L_i} p_{j,v} x_{j,v} = r_{i,v} \qquad \forall i \in [1,k]\ \forall v \in V \tag{18}$$

$$h_{i,v} \leqslant \sum_{j \in L_i} x_{j,v} \qquad \forall i \in [1,k]\ \forall v \in V \tag{19}$$

$$|L_i| \cdot h_{i,v} \geqslant \sum_{j \in L_i} x_{j,v} \qquad \forall i \in [1,k]\ \forall v \in V \tag{20}$$

$$r^{<}_{i,v} \geqslant r^{<}_{i,v-1} \qquad \forall i \in [1,k]\ \forall v \in V, v > 1 \tag{21}$$

$$r^{<}_{i,v} \geqslant r_{i,v-1} - U \cdot (1 - h_{i,v-1}) \qquad \forall i \in [1,k]\ \forall v \in V, v > 1 \tag{22}$$

$$r^{>}_{i,v} \geqslant r^{>}_{i,v+1} \qquad \forall i \in [1,k]\ \forall v \in V, v < |V| \tag{23}$$

$$r^{>}_{i,v} \geqslant r_{i,v+1} - U \cdot (1 - h_{i,v+1}) \qquad \forall i \in [1,k]\ \forall v \in V, v < |V| \tag{24}$$

$$l_{i,v} + U \cdot (1 - h_{i,v}) \geqslant r^{<}_{i',v} + 2\Delta t \qquad \forall i \in [1,k]\ \forall (i', \Delta t) \in P_i\ \forall v \in V \tag{25}$$

$$l_{i,v} + U \cdot (1 - h_{i,v}) \geqslant r^{>}_{i',v} + 2\Delta t \qquad \forall i \in [1,k]\ \forall (i', \Delta t) \in P_i\ \forall v \in V \tag{26}$$

$$\widehat{h}_{i,v} \leqslant |L_i| - \sum_{j \in L_i} x_{j,v} \qquad \forall i \in [1,k]\ \forall v \in V \tag{27}$$

$$|L_i| \cdot \widehat{h}_{i,v} \geqslant |L_i| - \sum_{j \in L_i} x_{j,v} \qquad \forall i \in [1,k]\ \forall v \in V \tag{28}$$

$$svm_v \leqslant l_{1,v} \qquad \forall v \in V \tag{29}$$

$$fvm_v \geqslant r_{k,v} \qquad \forall v \in V \tag{30}$$

$$svm_v \leqslant l_{i,v} - \Delta t \cdot (h_{i,v} + \widehat{h}_{i',v} - 1) \qquad \forall i \in [1,k]\ \forall (i', \Delta t) \in P_i\ \forall v \in V \tag{31}$$

$$fvm_v \geqslant r_{i,v} + \Delta t \cdot (h_{i,v} + \widehat{h}_{i',v} - 1) \qquad \forall i \in [1,k]\ \forall (i', \Delta t) \in S_i\ \forall v \in V \tag{32}$$

$$I_v \cdot P \geqslant fvm_v - svm_v \qquad \forall v \in V \tag{33}$$

$$(I_v - 1) \cdot P \leqslant fvm_v - svm_v - \varepsilon \qquad \forall v \in V \tag{34}$$

$$M \geqslant r_{k,v} \qquad \forall v \in V \tag{35}$$

$$C = \sum_{v \in V} c_v \cdot I_v \tag{36}$$

Ограничение (16) гарантирует, что каждая задача выполняется ровно на одной ВМ. Ограничение (17) гарантирует, что каждая последующая стадия выполняется после предыдущей. Ограничение (18) связывает время начала и завершения для каждой стадии. Если ни одна задача стадии i не выполняется на ВМ v, имеем $l_{i,v} = r_{i,v}$.

Ограничения (19) и (20) определяют переменные $h_{i,v}$. Ограничения (21)-(24) определяют переменные $r^{<}_{i,v}$ и $r^{>}_{i,v}$. Они работают путём вычисления текущих максимумов $r_{i,v}$ для v, выполняющих хотя бы одну задачу стадии i, по префиксам и суффиксам V.

Ограничения (25) и (26) гарантируют, что каждая задача получает все необходимые данные перед выполнением. Ограничения (27) и (28) определяют переменные $\widehat{h}_{i,v}$.

Ограничения (29) и (30) гарантируют, что время жизни ВМ включает все задачи, выполняемые на этой ВМ. В сочетании с (17) они дополнительно гарантируют, что $svm_v \leqslant fvm_v$.

Ограничение (31) гарантирует, что все передачи данных на ВМ лежат в пределах времени её жизни. Ограничение (32) делает то же самое, но для передач с ВМ.

Ограничения (33) и (34) обеспечивают корректный подсчёт интервалов биллинга. Наконец, ограничения (35) и (36) определяют время и стоимость выполнения графа соответственно.

3.3. Сравнение моделей

Рассмотрим граф с k стадиями и l парами стадий, между которыми есть рёбра. На практике $k \ll |T|$ и $l \ll |D|$. Базовая модель содержит $O(|T||V| + |T|^2)$ целочисленных переменных и $O(|V||T|^2 + |V||D|)$ ограничений. Заметим, что $|T|^2$ – это граница размера множества $\mathcal{N}$ пар независимых задач, и в общем случае $|T|^2$ может быть чрезмерно пессимистичной границей (в качестве экстремального примера рассмотрим граф-цепочку, у которого $\mathcal{N} = \varnothing$). Однако для графов, встречающихся на практике, размер $\mathcal{N}$ часто составляет $\Theta(|T|^2)$.

В отличие от базовой модели, наша модель имеет $O(|T||V|)$ целочисленных переменных и $O(|T| + (k+l)|V|)$ ограничений.

Ещё одной важной метрикой является количество ограничений big-M, поскольку известно, что они негативно влияют на производительность решателя. Базовая модель имеет $O(|T|^2 + |V|(|T| + |D|))$ ограничений big-M, в то время как многостадийная модель имеет лишь $O((k+l)|V|)$.

4. Экспериментальное сравнение моделей

4.1. Данные

Широко используемым набором данных в современных исследованиях по планированию графов задач является WfInstances [1]. WfInstances содержит графы, собранные из 9 реальных распределённых приложений для научных вычислений. Хотя графы в этом наборе данных не соответствуют в точности нашему определению многостадийного графа с однородными зависимостями, некоторые из них соответствуют ему приближённо. Под «приближённо» мы подразумеваем следующее:

(1) Не все рёбра между парой стадий в точности равны, но время передачи для разных рёбер отличается не более чем на 10^{-3} секунды (при типичной пропускной способности облачной сети 5 Гбит/с).

В этом случае мы можем положить все времена передачи равными их среднему значению ценой небольшой погрешности, которой на практике можно пренебречь (в худшем случае погрешность времени выполнения составит не более $10^{-3}n$ секунд). Следует отметить, что использование среднего времени передачи в некоторых редких случаях может сделать результирующее расписание недопустимым.

Эту проблему можно смягчить, применив более консервативный подход и установив все времена передачи равными их максимуму. Однако это оказывает пренебрежимо малое влияние на результирующую MILP-модель, и в экспериментах ниже мы используем среднее время передачи. Для приложений, требующих гарантированной допустимости относительно исходного графа, вместо этого можно использовать максимальное время передачи без изменения формулировки.

(2) Может существовать такая пара стадий, что некоторые задачи непосредственно зависят друг от друга, а некоторые – нет, но в этом случае одна из стадий является либо первой стадией, состоящей из единственной входной задачи, либо последней стадией, состоящей из единственной выходной задачи, и существующие зависимости имеют почти нулевое время передачи (не более 10^{-3} секунды).

В этом случае мы можем добавить недостающие рёбра с нулевым временем передачи. Идея, лежащая в основе этой процедуры, состоит в том, что задачи, которые не получают внешние входные данные или не отправляют выходные данные, всё равно сообщают свой статус планировщику, и это можно рассматривать как внешние входные/выходные данные пренебрежимо малого размера.

Таким образом, добавление дополнительных рёбер нулевого размера от входной задачи или к выходной задаче не делает модель нереалистичной. Пример показан на рисунке 2. Некоторая стадия может лишь частично

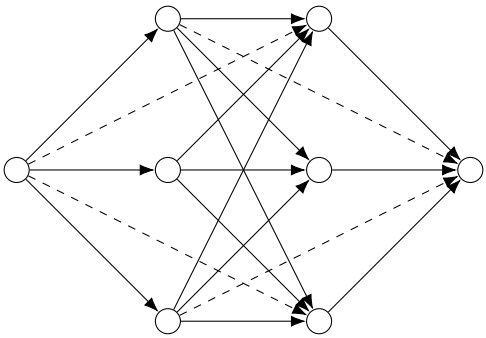


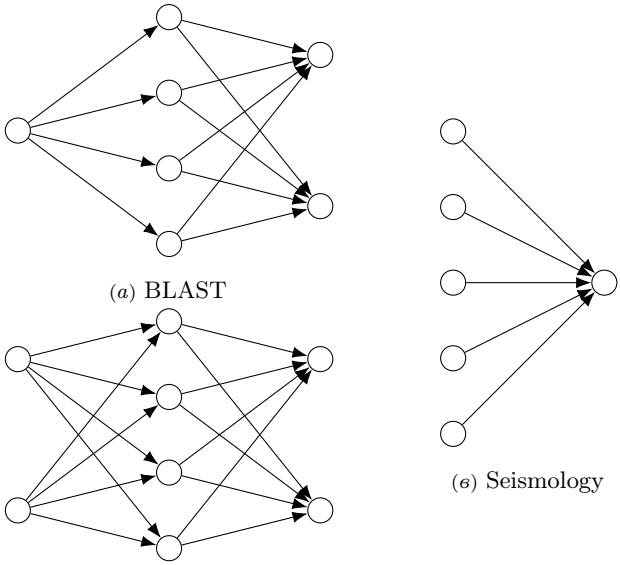
Рисунок 2. Граф, приближённо удовлетворяющий определению многостадийного графа с однородными зависимостями

зависеть от входной стадии, и выходная стадия может лишь частично зависеть от некоторой стадии, но это не страшно, если эти зависимости очень малого размера.

Анализируя набор данных WfInstances, мы обнаружили следующие приложения, которые приближённо соответствуют определению многостадийного графа с однородными зависимостями:

- **BLAST.** Графы этого биоинформатического приложения имеют простую трёхстадийную структуру fork-join.
- **BWA.** Аналогично BLAST, но первая стадия имеет две задачи вместо одной.
- **Seismology.** Эти графы имеют простую двухстадийную структуру: первая стадия состоит из параллельных задач, а вторая стадия представляет собой всего одну задачу, которая обрабатывает и объединяет результаты предыдущих задач.

Примеры, показывающие структуру графов BLAST, BWA и Seismology, приведены на рисунке 3. Для наглядности графы изображены без искусственных входной и выходной задач.



(*а*) BLAST

(*б*) BWA

(*в*) Seismology

Рисунок 3. Примеры структур графов

4.2. Описание эксперимента

Мы сравниваем производительность двух MILP-моделей на графах из WfInstances для трёх упомянутых выше приложений. Мы ограничиваемся графами, содержащими не более 110 задач, включая дополнительные входную и выходную задачи. Следуя [4], мы рассматриваем набор ВМ из 4 самых дешёвых типов ВМ в каждом из семейств Amazon EC2 m5 и c5, по 3 ВМ каждого типа (всего 24 ВМ). Интервал биллинга составляет 1 секунду.

Модели решались с помощью MILP-решателя CPLEX 20.1 на процессоре Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz со 128 ГБ оперативной памяти. Решателю было разрешено использовать до 32 параллельных потоков. Ограничение по времени на решение модели составляло 1 час.

Мы тестируем три постановки задачи: минимизация времени выполнения без дополнительных ограничений, минимизация стоимости при ограничении на срок выполнения и минимизация времени выполнения при ограничении на бюджет. Последние две постановки требуют задания срока выполнения или бюджета.

Стандартный подход [4–6] состоит в том, чтобы взять некоторую грубую оценку оптимального времени выполнения или стоимости и преобразовать её в ограничение путём умножения на некоторый заранее определённый параметр α. Оценка стоимости вычислялась путём последовательного запуска всех задач на самом дешёвом типе ВМ, как в [4, 5]. Оценка времени выполнения вычислялась путём запуска каждой задачи на отдельном экземпляре самого быстрого типа ВМ в предположении, что передачи данных происходят мгновенно, как в [4, 6].

Параметр α был установлен равным 10 для срока выполнения и 2 для бюджета. Причина использования разных α для разных ограничений заключается в том, что слишком малое значение α сделало бы задачу тривиально недопустимой, а слишком большое значение α сделало бы ограничение тривиально выполнимым и, следовательно, бесполезным. Что является слишком малым или слишком большим, зависит от качества оценки, которое различается для времени и стоимости, поэтому и параметры разные.

4.3. Результаты

В таблице 1 показано количество целочисленных переменных и ограничений в MILP-моделях для задач без ограничений и с ограничениями на срок выполнения/бюджет соответственно (заметим, что минимизация стоимости при ограничении на срок выполнения и минимизация времени выполнения при ограничении на бюджет порождают MILP-модели одинакового размера).

В таблицах 2, 3 и 4 представлены время решения модели, относительный зазор решателя и статус решателя для минимизации времени выполнения без ограничений, минимизации стоимости при ограничении на срок выполнения и минимизации времени выполнения при ограничении на бюджет соответственно.

Таблица 1. Размеры моделей в количествах целочисленных переменных и ограничений для минимизации времени выполнения

Графы	Размер модели			
	общей		многостадийной	
	$I^{(1)}$	ограничений	$I^{(1)}$	ограничений
без ограничений на срок выполнения или бюджет				
blast-chameleon-large-*	13545	257480	3231	1573
blast-chameleon-small-*	3105	43100	1791	1513
bwa-chameleon-small-*	13780	259956	3255	1574
seismology-chameleon-100p-*	13081	252554	3041	1239
при ограничеиях на срок выполнения или бюджет				
blast-chameleon-large-*	13570	277042	3256	1959
blast-chameleon-small-*	3130	51142	1816	1899
bwa-chameleon-small-*	13805	284366	3280	1960
seismology-chameleon-100p-*	13106	262372	3066	1529

(1) количество целочисленных переменных

Зазор решателя вычисляется как $\frac{best-bound}{best} \cdot 100\%$, где $best$ – лучшее расписание, найденное решателем, а $bound$ – лучшая нижняя граница оптимального значения целевой функции, найденная решателем. Если модель решена до оптимальности], то зазор равен 0. Если модель не нашла ни одного допустимого решения, то зазор равен ∞. Если зазор равен ∞ и решатель завершил работу до достижения лимита времени, то это означает, что задача доказуемо недопустима.

Статус решателя – одно из следующего:

«optimal», если модель была решена до оптимальности;

«feasible», если было найдено некоторое допустимое решение, но его оптимальность не доказана;

«infeasible», если решатель доказал, что допустимого решения не существует; и

«unknown», если допустимое решение не было найдено, но и недопустимость также не удалось доказать.

Решателю не удалось решить базовую MILP-модель до оптимальности ни для одной задачи ни на одном графе в течение 1 часа. Многостадийная модель была решена на графах *blast-chameleon-small-** (за несколькими исключениями), на графе Seismology (кроме минимизации времени исполнения) и на двух графах BWA в случае минимизации стоимости при ограничении на срок выполнения.

Несмотря на то, что в большинстве случаев многостадийную модель не удаётся решить до оптимальности в течение 1 часа, для графов BLAST и Seismology она даёт решения с зазором оптимальности ниже 4% (за единственным исключением в лице оптимизации при ограничении на бюджет для *blast-chameleon-small-001*), если задача допустима (и во многих случаях зазор составляет менее 1%). Её производительность на графах BWA несколько ухудшается, но она всё равно даёт меньший зазор, чем базовая MILP-модель.

Базовая MILP-модель, с другой стороны, часто даёт решения с зазором более 60%, а для оптимизации при ограничении на срок выполнения зазор никогда не бывает ниже 90%. Что касается размера модели, многостадийная модель достигает сокращения количества ограничений до 200 раз и сокращения количества целочисленных переменных до 4 раз.

Таблица 2. Минимизация времени выполнения без ограничений: время решения, зазор и статус решателя

граф	общая модель			многостадийная модель		
	время (с)	статус	зазор (%)	время (с)	статус	зазор (%)
blast-chameleon-large-001	3600.00	feasible	90.63	3600.00	feasible	0.25
blast-chameleon-large-002	3600.00	feasible	92.64	3600.00	feasible	0.12
blast-chameleon-large-003	3600.00	feasible	95.44	3600.00	feasible	0.24
blast-chameleon-large-004	3600.00	feasible	93.08	3600.00	feasible	0.17
blast-chameleon-large-005	3600.00	feasible	93.37	3600.00	feasible	0.17
blast-chameleon-small-001	3600.00	feasible	80.22	3600.00	feasible	3.09
blast-chameleon-small-002	3600.00	feasible	71.92	20.40	optimal	0.00
blast-chameleon-small-003	3600.00	feasible	69.96	1578.76	optimal	0.00
blast-chameleon-small-004	3600.00	feasible	71.43	638.41	optimal	0.00
blast-chameleon-small-005	3600.00	feasible	71.31	42.10	optimal	0.00
bwa-chameleon-small-001	3600.00	feasible	19.23	3600.00	feasible	4.81
bwa-chameleon-small-002	3600.00	feasible	21.95	3600.00	feasible	9.91
bwa-chameleon-small-003	3600.00	feasible	23.34	3600.00	feasible	15.50
bwa-chameleon-small-004	3600.00	feasible	21.29	3600.00	feasible	11.50
bwa-chameleon-small-005	3600.00	feasible	21.04	3600.00	feasible	7.49
seismology-chameleon-100p-001	3600.00	feasible	65.79	3600.00	feasible	0.14

Таблица 3. Минимизация стоимости с предельным сроком ($\alpha = 10$): время решения, зазор и статус решателя

граф	общая модель			многостадийная модель		
	время (с)	статус	зазор (%)	время (с)	статус	зазор (%)
blast-chameleon-large-001	3600.00	unknown	∞	3600.00	feasible	0.15
blast-chameleon-large-002	3600.00	unknown	∞	3600.00	feasible	0.07
blast-chameleon-large-003	3600.00	unknown	∞	3600.00	feasible	0.17
blast-chameleon-large-004	3600.00	unknown	∞	3600.00	feasible	0.18
blast-chameleon-large-005	3600.00	unknown	∞	3600.00	feasible	0.11
blast-chameleon-small-001	3600.00	unknown	∞	0.07	infeasible	∞
blast-chameleon-small-002	3600.00	feasible	93.22	59.95	optimal	0.00
blast-chameleon-small-003	3600.00	feasible	93.34	3222.04	optimal	0.00
blast-chameleon-small-004	3600.00	feasible	92.86	3600.00	feasible	2.34
blast-chameleon-small-005	3600.00	feasible	93.45	3600.00	feasible	0.13
bwa-chameleon-small-001	3600.00	feasible	100.00	42.41	optimal	0.00
bwa-chameleon-small-002	3600.00	feasible	100.00	3600.00	feasible	0.01
bwa-chameleon-small-003	3600.00	feasible	100.00	3600.00	feasible	1.27
bwa-chameleon-small-004	3600.00	feasible	100.00	16.40	optimal	0.00
bwa-chameleon-small-005	3600.00	feasible	100.00	3600.00	feasible	1.34
seismology-chameleon-100p-001	3600.00	feasible	100.00	75.13	optimal	0.00

Таблица 4. Минимизация времени выполнения с бюджетом ($\alpha = 2$): время решения, зазор и статус решателя

граф	общая модель			многостадийная модель		
	время (с)	статус	зазор (%)	время (с)	статус	зазор (%)
blast-chameleon-large-001	3600.00	unknown	∞	3600.00	feasible	0.52
blast-chameleon-large-002	3600.00	feasible	96.74	3600.00	feasible	0.20
blast-chameleon-large-003	3600.00	feasible	95.78	3600.00	feasible	0.24
blast-chameleon-large-004	3600.00	unknown	∞	3600.00	feasible	0.37
blast-chameleon-large-005	3600.00	unknown	∞	3600.00	feasible	0.19
blast-chameleon-small-001	3600.00	feasible	76.62	3600.00	feasible	8.76
blast-chameleon-small-002	3600.00	feasible	74.32	44.74	optimal	0.00
blast-chameleon-small-003	3600.00	feasible	72.85	2194.55	optimal	0.00
blast-chameleon-small-004	3600.00	feasible	71.63	109.33	optimal	0.00
blast-chameleon-small-005	3600.00	feasible	73.26	29.13	optimal	0.00
bwa-chameleon-small-001	3600.00	feasible	22.49	3600.00	feasible	2.54
bwa-chameleon-small-002	3600.00	feasible	72.44	3600.00	feasible	22.73
bwa-chameleon-small-003	3600.00	unknown	∞	3600.00	feasible	5.85
bwa-chameleon-small-004	3600.00	unknown	∞	3600.00	feasible	11.60
bwa-chameleon-small-005	3600.00	unknown	∞	3600.00	feasible	13.19
seismology-chameleon-100p-001	3600.00	feasible	99.58	821.60	optimal	0.00

5. Заключение

В данной работе предложена новая компактная MILP-модель для планирования многостадийных графов задач с однородными зависимостями в облаке. Эта модель значительно меньше стандартной модели благодаря сжатию стадий. Эксперименты демонстрируют преимущество предложенной модели: она существенно сокращает количество целочисленных переменных и ограничений и в большинстве случаев находит оптимальные или близкие к оптимальным допустимые решения в пределах 1-часового лимита времени. Перспективным направлением будущих исследований является усиление формулировки для нахождения оптимальных решений в большем числе случаев.

Список использованных источников

[1] Coleman T., Casanova H., Pottier L., Kaushik M., Deelman E., Ferreira da Silva R. *Wfcommons: A framework for enabling scientific workflow research and development* // Future Generation Computer Systems.– March 2022.– Vol. **128**.– Pp. 16–27. ↑296, 305

[2] Mayer R., Mayer Ch., Laich L. *The tensorflow partitioning and scheduling problem: it's the critical path!Proceedings of the 1st Workshop on Distributed Infrastructures for Deep Learning*, DIDL'17 (Las Vegas, Nevada, USA, December 11–15, 2017), New York: ACM.– 2017.– ISBN 978-1-4503-5169-0.– Pp. 1–6. ↑296

[3] Aseman-Manzar M.-M., Karimian-Aliabadi S., Entezari-Maleki R., Egger B., Movaghar A. *Cost-aware resource recommendation for DAG-based big data workflows: An apache spark case study* // IEEE Transactions on Services Computing.– 2023.– Vol. **16**.– No. 3.– Pp. 1726–1737. ↑296

[4] Semenov Yu., Sukhoroslov O. *MIP models and complexity results for DAG scheduling in the cloud* // *Mathematical Optimization Theory and Operations Research*, MOTOR 2025 (Novosibirsk, Russia, July 7–11, 2025), Lecture Notes in Computer Science.– vol. **15681**, eds. Kochetov Yu., Khachay M., Eremeev A., Pardalos P., Cham: Springer.– 2025.– ISBN 978-3-031-97076-4.– Pp. 317–331. ↑296, 299, 307, 308

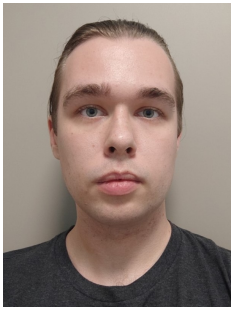
[5] Li H., Wang D., Cañizares Abreu J. R., Zhao Q., Bonilla Pineda O. *PSO+LOA: hybrid constrained optimization for scheduling scientific workflows in the cloud* // The Journal of Supercomputing.– 2021.– Vol. **77**.– No. 11.– Pp. 13139–13165. ↑308

[6] Shobeiri P., Rastaghi M. A., Abrishami S., Shobiri B. *PCP-ACO: a hybrid deadline-constrained workflow scheduling algorithm for cloud environment* // The Journal of Supercomputing.– 2024.– Vol. **80**.– No. 6.– Pp. 7750–7780. ↑308

Поступила в редакцию 29.05.2026;
одобрена после рецензирования 29.05.2026;
принята к публикации 28.07.2026;
опубликована онлайн 21.09.2026.

Рекомендовал к публикации *д.ф.-м.н. С. М. Абрамов*

Информация об авторах:



Юрий Дмитриевич Семенов

Аспирант НИУ ВШЭ. Научные интересы включают распределённые вычисления и комбинаторную оптимизацию.

0009-0004-8996-4857

e-mail: *ydsemenov@hse.ru*



Олег Викторович Сухорослов

Кандидат технических наук, старший научный сотрудник ИППИ РАН, доцент НИУ ВШЭ. Научные интересы: распределённые вычисления, облачные вычисления, управление ресурсами, планирование и имитационное моделирование.

0000-0003-2836-2524

e-mail: *sukhoroslov@iitp.ru*

Вклад авторов: *Ю. Д. Семенов* – 80% (методология, программное обеспечение, валидация, расследование, написание черновой версии); *О. В. Сухорослов* – 20% (сбор материала, доработка и редактирование, администрирование).

Декларация об отсутствии личной заинтересованности: *благополучие авторов не зависит от результатов исследования.*