Построение расписаний в гетерогенных системах с использованием адаптивной нейронной гиперэвристики

Александр Аллахвердян[1†], Анастасия Жадан[1†],
Иван Кондратов[1†], Ованес Петросян[1, 4],
Алексей Романовский[2], Виталий Харин[2], Инь Ли[3,4*]

[1]Санкт-Петербургский государственный университет,
Санкт-Петербург, 198504, Россия.
[2]Санкт-Петербургский исследовательский центр, Huawei Russian
Research Institute, Санкт-Петербург, 191119, Россия.
[3]Факультет математики, Харбинский политехнический университет,
Хэйлунцзян, Харбин, 150001, Китай.
[4]Факультет математики и компьютерных наук, Университет Яньань,
Яньань, 716000, Китай.

*Корреспондирующий автор(ы). E-mail(s): dr.liyin@hit.edu.cn;
Соавторы: aleka_alexander@mail.ru; anastasiya_markel@mail.ru;
kondratov.ivan.vladimirovich@gmail.com; petrosian.ovanes@yandex.ru;
aleksei.romanovskii@huawei.com; vitaliy.kharin@huawei.com;
†Данные авторы внесли равный вклад в данную работу.

Аннотация

В гетерогенных вычислительных средах эффективное планирование задач, особенно тех, которые формируют направленные ациклические графы, имеет решающее значение. Это особенно актуально для различных задач облачных и граничных вычислений, а также для обучения больших языковых моделей. В данной работе представлен новый подход к составлению расписания с помощью адаптивной нейронной гиперэвристики. Используя нейронную сеть, обученную с помощью генетического алгоритма, наш метод направлен на минимизацию времени выполнения расписания. Подход двухуровневый: на первом уровне приоритеты задач определяются с помощью адаптивной эвристики, а на втором уровне ресурсы распределяются на основе алгоритма Earliest Finish Time (EFT). Тесты показали, что этот метод

значительно превосходит традиционные эвристики планирования и другие подходы, основанные на машинном обучении. Он сокращает время выполнения на 6,7% для малых и на 28,49% для больших направленных ациклических графов по сравнению с лидирующим алгоритмом DONF. Кроме того, он достигает близости от 84,08% до 96,43% к оптимальным решениям, полученным с помощью методов решения задач смешанного целочисленного линейного программирования, демонстрируя свою эффективность в различных условиях.

Keywords: Нейронные сети, построение расписаний, направленные ациклические графы, генетические алгоритмы

1 Введение

В сложных современных системах, использующих гетерогенные вычислительные устройства, из-за взаимозависимости задач возникают существенные сложности в планировании. Эти взаимозависимости часто формализуются в виде направленного ациклического графа. Проблема назначения задач (узлов) на разнообразный набор исполнителей, различающихся по типу и вычислительной мощности, с целью минимизации времени выполнения является NP-полной [1]. Эта проблема распространена в нескольких областях, включая облачные вычисления (как видно на примере вычислительных мощностей Amazon [2] и приложений Apache [3]) и операционные системы [4]. В последнее время, с ростом популярности больших языковых моделей, эта проблема приобрела значимость в области машинного обучения. Обучение нейронных сетей обычно реализуется как обход статического вычислительного графа (например, в TensorFlow [5]) и влечет за собой значительную вычислительную нагрузку. Крупные модели обучаются в течение сотен или тысяч часов на гетерогенных системах [6]. Учитывая длительность и масштаб современных процессов обучения больших языковых моделей, даже небольшие улучшения в планировании могут привести к значительному сокращению количество затраченных вычислительных часов и, следовательно, к экономии средств.

В данной статье рассматривается проблема статического планирования с известными до выполнения параметрами вычислительного направленного ациклического графа, включая матрицу смежности и веса узлов, ребер. Предлагается новый подход к планированию направленного ациклического графа, использующий адаптивную метрику, реализованную через нейронную гиперэвристику в виде нейронной сети, обученной с помощью генетического программирования [7] — метода оптимизации, который эволюционирует программы для решения задач, имитируя биологическую эволюцию [8, 9]. Наш метод рассматривает направленный ациклический граф как неизменяемый и опирается на современные метрики построения расписаний (раздел 4.2) для построения набора признаков. Применение генетического программирования в процессе обучения устраняет

необходимость в построении дифференцируемой функции потерь для минимизации суммарного времени выполнения на наборе направленных ациклических графов.

Алгоритм состоит из двух уровней: нейронной гиперэвристики, которая выставляет приоритет выполнения узлов на основе их характеристик, и алгоритма Earliest Finish Time (EFT) для назначения задач вычислительным исполнителям. Генетический алгоритм [10] используется для обучения нейронной сети [11]. Алгоритм планирования на основе приоритетов и нейронной сети приспосабливается к различным топологиям и превосходит другие решения в минимизации суммарного времени выполнения.

Структура статьи следующая: в разделе 2 рассматриваются эвристические методы и подходы машинного обучения в статическом планировании направленных ациклических графов; в разделе 3 описывается модель гетерогенных систем и генерация случайных направленных ациклических графов для принцип работы симуляции; в разделе 4 подробно описывается предложенный алгоритм планирования; в разделе 5 проводится оценка сравнение предложенного решения с современными алгоритмами планирования и близкими к оптимальным решениями, найденным с помощью методов для решения задач смешанного целочисленного линейного программирования (посредством Gurobi [12]) для графов малой размерности.

2 Обзор литературы

Эвристические алгоритмы широко распространены и применяются в различных задачах [13], при этом полиномиальная сложность предпочтительнее экспоненциальной для точных решений. К ключевым эвристикам в планировании задач для гетерогенных сред относятся Levelized-Min Time (LMT) [14], которая минимизирует время выполнения, Dynamic Level Scheduling (DLS) [15], и Mapping Heuristic (MH) [16], оптимизирующая сопоставление задач процессорам путем минимизации времени коммуникации. Эмпирические исследования [14, 17] выделяют CPOP, HCPT, HPS, PETS и DONF как наиболее эффективные для минимизации суммарного времени выполнения в статическом планировании на направленных ациклических графах; они выбраны для сравнения с предлагаемым алгоритмом и описаны в разделе 4.3.

Более продвинутые подходы, включая методы машинного обучения и другие методы оптимизации, используются для решения задачи планирования на направленных ациклических графах. Обучение с подкреплением широко применяется в ситуациях, где трудно аналитически построить дифференцируемую функцию потерь. В работе [18] представлен метод обучения с подкреплением для динамического распределения задач в гетерогенных системах, но для него были ослаблены некоторые ограничения, которые присутствуют в данной работе. Метод поиска по дереву Монте-Карло (MCTS) [19] также использовался в аналогичной по постановке задачи, но с упрощением некоторых предположения о зависимости между задачами. В работе [9] была предложена основанная на

графах система линейного генетического программирования (LGP) с новыми операторами для направленных ациклических графах, улучшая традиционный LGP и генерируя эффективные эвристики для динамического распределения задач между исполнителями.

3 Описание задачи

В данной работе рассматривается планирование направленного ациклического графа в рамках распределённой гетерогенной системы. Модель планирования включает три элемента: сам направленный ациклического графа, распределённую вычислительную среду и критерии производительности. Описание нашей задачи соответствует стандартной структуре для построения расписаний на направленных ациклических графах, описанной в предыдущих исследованиях [14, 17, 20–22], с единственным отличием, что как исполнители, так и задачи являются гетерогенными по типам.

3.1 Ориентированный ациклический граф

Дан направленный ациклический граф $G = (J, E)$, где J — множество узлов, а E — множество рёбер:

- Ребро $(v_i, v_j) \in E$ обозначает отношение предшествования: узел v_j должен ждать завершения выполнения задачи v_i.
- Стоимость коммуникации $b_{j,m}$ между узлами v_j и v_m, где $(v_j, v_m) \in E$, учитывается, если узлы v_j и v_m назначены различным исполнителям; в противном случае стоимость коммуникации отсутствует. Если узел v_m имеет несколько родителей $(v_j, \ldots, v_k)$, которые были выполнены на исполнителях, отличных от того, которому назначен узел v_m, то стоимость коммуникации учитывается относительно каждого из родителей: $b_{j,m} + \ldots + b_{k,m}$.
- Каждый узел v_j имеет тип, который указывает, на каком типе исполнителя этот узел может быть выполнен.
- Множество непосредственных предшественников узла v_j в графе обозначается как $pred(v_j)$. Узел без предшественников называется входным узлом. В направленном ациклическом графе в общем случае может быть несколько входных узлов.
- Множество непосредственных наследников узла v_j обозначается как $succ(v_j)$. Узел без наследников называется выходным узлом. В направленном ациклическом графе в общем случае может быть несколько выходных узлов.

3.2 Распределённая вычислительная среда

Распределённая вычислительная среда состоит из множества исполнителей P, где $p_i \in P$ — гетерогенные исполнители с полносвязной топологией:

- D — матрица стоимости вычислений, где $D^{j,i} = w_{j,i}$ — время выполнения исполнителем $p_i \in P$ узла $v_j \in J$.

- Каждый исполнитель p_i имеет тип, определяющий, какие типы задач могут быть выполнены этим исполнителем. Чтобы назначить узел $v_j \in J$ исполнителю p_i, необходимо, чтобы тип узла v_j и тип исполнителя p_i совпадали.

3.3 Критерий производительности

Прежде чем представить окончательную целевую функцию планирования, определим следующие понятия: суммарное время выполнения (makespan), наиболее раннее время начала выполнения (EST) и наиболее раннее время окончания выполнения (EFT):

- Суммарное время выполнения — это время завершения последнего узла в заданном графе. Определяется как $makespan = \max AFT(v_{exit})$, где $AFT(v_{exit})$ — фактическое время окончания (AFT) выполнения выходного узла v_{exit}. Если имеется несколько выходных узлов, makespan является максимальным фактическим временем окончания выполнения по всем выходным узлам.
- $EST(v_j, p_i)$ обозначает наиболее раннее время начала выполнения узла v_j на исполнителе p_i и определяется как

$$EST(v_j, p_i) = \max(T_{Ava}(p_i), \max_{v_m \in pred(v_j)} AFT(v_m) + b_{m,j}),$$

  где $T_{Ava}(p_i)$ — наиболее раннее время, когда станет доступен исполнитель p_i.
- $EFT(v_j, p_i)$ обозначает наиболее раннее время окончания исполнения узла v_j на исполнителе p_i и определяется как

$$EFT(v_j, p_i) = EST(v_j, p_i) + w_{j,i}.$$

Целевая функция в задаче построения расписания на направленном ациклическом графе заключается в определении политики назначения узлов гетерогенным исполнителям таким образом, чтобы минимизировать суммарное время выполнения.

3.4 Генерация графов

Таблица 1 Параметры использованные для генерации графов

Название параметра	**Множество значений параметра**
Количество вершин (n)	{30, 60, 90, 36, 114, 576, 2400, 9600, 36864}
Ширина (f)	{0.2, 0.5}
Регулярность (r)	{0.2, 0.8}
Межуровневая связность (j)	{2, 4}
Отношение весов ребер и вершин (c)	{0.2, 0.8}
Плотность (d)	{0.1, 0.4, 0.8}

Для генерации случайных ориентированных ациклических графов было использовано программное обеспечение с открытым исходным кодом DAGGEN [23].

Следует отметить, что DAGGEN является популярным инструментом для создания синтетических наборов данных и использовался в работах [14, 17, 20–22], что позволяет напрямую сравнивать предложенное решение с этими исследованиями. DAGGEN применяет структурированный, послойный метод генерации графов, которые характеризуются набором из шести параметров. Количество вершин в каждом слое является случайной величиной и получается из равномерного распределения с заданными границами интервала. Среднее этого интервала центрировано вокруг параметра «ширина» (f), а отношение ширины каждого слоя к их количеству задается параметром «регулярность» (r). Связи между слоями устанавливаются на основе параметра «межуровневая связность» (j), который определяет максимальное количество слоев, которые может пересекать одно ребро — ребра соединяют только непосредственные следующие друг за другом слои, когда параметр «межуровневая связность» равен единице. Для каждой вершины число предшественников выбирается случайным образом из равномерного распределения, определенного в диапазоне от 1 до максимума, определяемого параметром «плотность» (d). Отношение суммарных весов ребер к суммарным весам узлов контролируется параметром «отношение весов ребер и вершин» (c).

4 Алгоритм

4.1 Архитектура алгоритма

Algorithm 1 Псевдокод алгоритма

Вход: Множество направленных ациклических графов D, множество исполнителей P с типами T, порогом улучшения ts и весами нейронной сети W
Выход: C_{max} для каждого графа из D их сумма C_{sum}
$C_{sum}^{prev} \leftarrow infinity$, $C_{sum}^{curr} \leftarrow 0$
while Истина **do**
    for каждого графа d в D **do**
        Построить расписание для d используя гиперэвристику и EFT
        Вычислить C_{max}^{d} для d
        C_{sum}^{curr} += C_{max}^{d}
    end for
    if $Diff(C_{sum}^{curr}, C_{sum}^{prev}) < ts$: **выйти из цикла**
    Обновить веса W используя генетический алгоритм
    $C_{sum}^{prev} \leftarrow C_{sum}^{curr}$, $C_{sum}^{curr} \leftarrow 0$
end while
return W

Алгоритм 4.1 нейронную сеть с генетическим алгоритмом для оптимизации, итеративно оптимизируя веса сети для минимизации суммарного времени выполнения направленного ациклического графа. В каждой итерации адаптивная нейронная гиперэвристика используется для вычисления *приоритета* и EFT

для выбора исполнителя в рамках каждого из графов. После построение расписания для заданного графа его суммарное время выполнения суммируется с остальными.

После планирования всех графов и агрегирования их суммарной длительности выполнения, если разница между текущей и предыдущей суммарными значениями меньше заданного порога, цикл завершается. Этот порог, установленный на уровне 0,1% из-за вычислительных ограничений, обозначает минимальное приемлемое улучшение.

Если обнаружено достаточное улучшение, веса нейронной сети обновляются с помощью генетического алгоритма для повышения эффективности планирования в следующей итерации. Предыдущее суммарное значение обновляется текущей суммой, которая затем сбрасывается до нуля для следующего цикла. Этот процесс повторяется, постоянно стремясь уменьшить суммарное время выполнения для всех графов, пока улучшения не станут меньше порога. По завершении алгоритм возвращает оптимизированные веса для алгоритма планирования на основе нейронной сети, представляя усовершенствованную политику планирования.

Предлагаемый алгоритм имеет вычислительную сложность $O(k \times o \times p)$ по сравнению с DONF, у которого сложность составляет $O(v \times o \times p)$, где k — количество узлов в списке готовых к выполнению, o — максимальная исходящая степень любого узла в графе, p обозначает число исполнителей, а v — число задач (узлов). В сценариях, где o столь же велико, как $v - 1$, оба алгоритма — предлагаемый и DONF — имеют верхнюю границу сложности $O(v^2 p)$. Однако в большинстве случаев число исполнителей не так велико, и предложенное решение показывает лучшие результаты при сопоставимой или меньшей вычислительной сложности.

4.2 Метрики для построения расписаний на графах

Одним из основных этапов в разработке алгоритма является подбор метрик, которыми можно на локальном уровне описать каждый из узлов в графе. Обширный обзор метрик для направленных ациклических графов представлен в работе [24]. На её основе были выбраны приведенные ниже метрики как входные параметры нейронной сети, описанные в таблице 2.

4.3 Современные эвристики построения расписания на графах

В данном разделе приведено подробное описание современных алгоритмов планирования на направленных ациклических графах, которые служат эталонами для сравнения с предложенным подходом. Выбор этих алгоритмов основан на обзоре [14].

Алгоритм Degree of Node First (DONF) [17] нацелен на максимальное использование параллелизма для полного задействования ресурсов гетерогенной системы. Он приоритизирует узлы с большей взвешенной выходящей степенью на ранних этапах. Взвешенная выходящая степень (WOD) узла v_i формулируется

Таблица 2 Метрики для построения расписаний на графах

Обозначение	Описание
$WOD(v_i)$	взвешенная выходящая степень (WOD) узла v_i: $WOD(v_i) = \sum_{v_j \in succ(v_i)} \frac{1}{ID(v_j)}$, где $ID(v_j)$ обозначает входящую степень узла v_j
$WOD_2(v_i)$	WOD второй степени: $WOD_2(v_i) = \sum_{v_j \in succ(v_i)} \left(\frac{1}{ID(v_j)} + \alpha \cdot \sum_{v_k \in succ(v_j)} \frac{1}{ID(v_k)} \right)$, где α параметр влияния WOD второй степени
$rank(v_i)$	приоритет узла: $rank(v_i) = \overline{w_i} + \max_{v_j \in succ(v_i)}(\overline{b_{i,j}} + rank(v_j))$
$C(v_i)$	Вычислительная сложность узла v_i
$\|pred(v_i)\|$	количество непосредственных предшественников v_i
$\|succ(v_i)\|$	количество непосредственных наследников v_i
TW_{in}	Суммарный вес входящих ребер узла v_i
TW_{ou}	Суммарный вес исходящих ребер узла v_i

как:

$$WOD(v_i) = \sum_{v_j \in succ(v_i)} \frac{1}{ID(v_j)}, \tag{1}$$

где $ID(v_j)$ — входящая степень узла v_j. После приоретизации узла для него выбирается исполнитель, минимизирующий EFT.

Алгоритм High-Performance Task Scheduling (HPS) [21] работает в три фазы. Сначала он обходит направленный ациклический граф сверху вниз для упорядочения узлов. Вычисление приоритета узлов использует вес входящий и исходящих ребер, а также издержки на передачу данных. Для выбора исполнителя используется EFT.

Алгоритм Performance Effective Task Scheduling (PETS) [22], аналогично HPS, состоит из трёх этапов. Сортировка узлов происходит на каждой итерации, после чего следует стадия приоритизация с использованием издержек на передачу данных (DTC), средней стоимости исполнения (ACC) и ранга предшествующей задачи (RPT). Формула приоритизации задаётся как:

$$rank(v_i) = rankACC(v_i) + DTC(v_i) + RPT(v_i), \tag{2}$$

где наивысший ранг обозначает наивысший приоритет. Выбор исполнителя выполняется с помощью EFT.

Алгоритм Heterogeneous Critical Parent Trees (HCPT) [14] выполняет планирование с ограниченным числом гетерогенных исполнителей (BNP) через

разделение графа на две группы неупорядоченных деревьев предшественников. Критический узел (CN) характеризуется нулевой дисперсией между средними наиболее поздним и наиболее ранним временем начала. Эмпирические исследования показали, что HCPT обеспечивает относительно лучшие результаты при меньшей сложности.

Алгоритм Critical Path on a Processor (CPOP) [20] определяет свою стратегию планирования в две фазы. Сначала он присваивает приоритеты узлам на основе их восходящего ранга $rank_u(v_i)$ и нисходящего ранга $rank_d(v_i)$, определённых следующим образом:

$$rank_u(v_i) = \overline{w_i} + \max_{v_j \in succ(v_i)} (\overline{b_{i,j}} + rank_u(v_j)), \tag{3}$$

$$rank_d(v_i) = \max_{v_j \in pred(v_i)} (\overline{b_{i,j}} + \overline{w_j} + rank_d(v_j)), \tag{4}$$

где финальный приоритет узла v_i является суммой его восходящего и нисходящего рангов:

$$priority(v_i) = rank_d(v_i) + rank_u(v_i). \tag{5}$$

Критический исполнитель, который является самым быстрым, определяется как тот, который минимизирует суммарные вычислительные затраты вдоль критического пути. Затем узлы на критическом пути назначаются этому исполнителю, а остальные следуют критерию EFT.

5 Численный эксперимент

В этом разделе описывается дизайн эксперимента и оценивается производительность предлагаемого алгоритма. Эксперимент произведен на Macbook M1 Pro с процессором ARM.

5.1 Среда моделирования

Предлагаемый алгоритм тестируется на направленных ациклических графах малого и большого размеров, с целью оценки его обобщающей способности и устойчивости относительно различных конфигураций системы. Кроме того, для графов малой размерности анализируется близость суммарного времени выполнения современных алгоритмов построения расписания на направленных ациклических графах к глобально оптимальному суммарному времени выполнения. Во всех тестах в симуляции передача данных начинается только после завершения всех предшествующих задач. Скорость передачи данных между вычислительными узлами (используется для расчёта штрафа за передачу данных, см. раздел 3.2) составляет 1085 МБ/с.

Конфигурации для направленных ациклических графов малой размерности используются гетерогенные конфигурации количеством исполнителей от трех до девяти, подробно описанные в таблице 3, выбранные из литературы, использующей DAGGEN. Сгенерировано три графа для обучения на каждую топологию (всего 144) и десять графов для оценки на каждую топологию (всего 480).

Таблица 3 Описание конфигураций экспериментов для графов малой размерности (30, 60, 90 узлов)

Исполнители	Типы	Вычислительная мощность (Гфлоп/с)
	Конфигурация 1: 3 исполнителя	
[1, 2, 3]	[1, 2 ,3]	[26, 134, 34]
	Конфигурация 2: 6 исполнителей	
[1, 2, 3, 4, 5, 6]	[1, 2, 3, 1, 2, 3]	[26, 134, 50, 70, 20]
	Конфигурация 3: 9 исполнителей	
[1, 2, 3, 4, 5, 6, 7, 8, 9]	[1, 2, 3, 1, 2, 3, 1, 2, 3]	[26, 134, 50, 70, 20, 125, 40, 60]

Таблица 4 Описание конфигураций экспериментов для графов большой размерности (36, 114, 576, 2400, 9600, 36864 узлов)

Конфигурация	Количество исполнителей каждого типа	Размер графа
Конфигурация 4: 3 исполнителя	1	36
Конфигурация 5: 6 исполнителей	2	114
Конфигурация 6: 12 исполнителей	4	576
Конфигурация 7: 24 исполнителя	8	2400
Конфигурация 8: 48 исполнителей	16	9600
Конфигурация 9: 96 исполнителей	32	36864

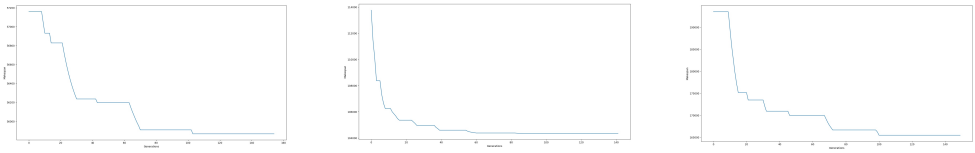
Для направленных ациклических графов большой размерности составлены конфигурации количеством исполнителей от 3 до 96, с вычислительной мощностью, случайно сгенерированной в диапазоне от 10 до 300 Гфлоп/с (см. таблицу 4). Каждая конфигурация составлена для фиксированного размера направленного ациклического графа. Для обучения сгенерированно 3000 графов на топологию, а для оценки — 10000 графов на топологию. В общей сложности было протестировано 78000 направленных ациклических графов большой размерности. Сравнительный анализ представлен в разделе 5.3.

5.2 Обучение нейронной сети

Для оптимизации весов нейронной сети был использован генетический алгоритм со следующими компонентами:

1. Кодирование хромосомы: веса и смещения нейронной сети представлены в виде списка действительных чисел.
2. Целевая функция: оценка основана на сумме суммарных длительностей исполнения для всех графов в обучающей выборке.
3. Инициализация: начальные веса популяции случайно выбираются из равномерного распределения в диапазоне от –1 до 1.
4. Операторы: в обучении используются кроссоверы TPI и SBX, а также оператор мутации, который изменяет элементы хромосомы одного родителя для получения потомка.
5. Настройки параметров: ключевые параметры включают вероятность мутации 80%, размер популяции 150, 1000 поколений и 95 весов.

Нейронная сеть содержит пять полносвязных слоёв, с количеством нейронов в каждом слое уменьшающимся от 8 до 1. Функции активации включают ReLU, которая вводит нелинейность, возвращая ноль для отрицательных входов и само значение входа для положительных, и сигмоиду, которая нормализует входы в диапазон от 0 до 1. Структура сети содержит 95 параметров, охватывающим как веса, так и смещения.



Граф с 30 узлами Граф с 60 узлами Граф с 90 узлами

Рис. 1 Кривая обучения нейронной сети с применением генетического алгоритма

Процесс обучения алгоритма на основе нейронной сети представлен на трёх рисунках, демонстрирующих процесс оптимизации весов посредством генетического алгоритма и его влияние на среднее суммарное время выполнения для графов малой размерности.

Рисунок 1 иллюстрирует производительность алгоритма на малых (30 узлов), средних (60 узлов) и больших (90 узлов) направленных ациклических графах малой размерности. Каждый график отображает поколения генетического алгоритма по оси X и среднее суммарное время выполнения по оси Y. Все три графика показывают резкое снижение суммарного времени выполнения на ранних поколениях, указывая на быструю оптимизацию, после чего следует плато по мере приближения алгоритма к оптимальному решению. Это поведение демонстрирует устойчивость и масштабируемость алгоритма, эффективно оптимизирующего построение расписаний для графов разных размеров в гетерогенных вычислительных средах.

5.3 Сравнительный анализ

В этом разделе представлен сравнительный анализ предложенного алгоритма с передовыми методами, такими как DONF, CPOP, HCPT, HPS, PETS, а также сравнение с глобально оптимальным решением, полученным с помощью методов решения задач смешанного целочисленного линейного программирования для направленных ациклических графов малой размерности. Также сравнивается производительность предложенного алгоритма на графах большой размерности с наиболее эффективными из этих алгоритмов.

Для каждого графа сравнение проводится с использованием следующей метрики производительности:

$$p = \frac{(makespan_{\text{sota}} - makespan_{\text{NN}}) \times 100}{makespan_{\text{sota}}}, \tag{6}$$

где $makespan_{\text{sota}}$ — суммарное время выполнения, достигнутое передовыми алгоритмами, а $makespan_{\text{NN}}$ — суммарное время выполнения, достигнутое предложенным алгоритмом. Полученные значения агрегируются и усредняются по всем конфигурациями и размерам графов для оценки.

5.3.1 Результаты для графов малой размерности

Таблица 5 Средний процент улучшения предложенного алгоритма по сравнению с передовыми методами.

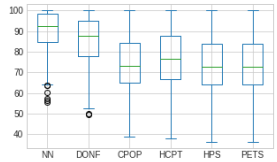
Количество узлов	DONF	CPOP	HCPT	HPS	PETS
Конфигурация 1: 3 исполнителя					
30	**4.81**	17.75	15.04	18.31	18.41
60	**9.71**	24.14	22.01	25.11	25.03
90	**12.14**	26.07	24.86	27.59	27.6
Конфигурация 2: 6 исполнителей					
30	**2.20**	14.17	10.43	13.37	13.47
60	**6.36**	22.81	18.59	22.51	22.41
90	**9.44**	25.87	21.81	26.59	26.40
Конфигурация 3: 9 исполнителей					
30	**1.57**	10.72	6.33	8.06	8.19
60	**5.10**	20.0	14.46	17.67	17.83
90	**5.73**	21.51	16.55	20.14	20.31

Предлагаемый алгоритм на основе нейронных сетей был оценен на графах малой размерности, проверяя его способность к обобщению и устойчивость в различных конфигурациях системы и разных размерах графов, представленных в таблицах 3 и 4. Цель анализа заключалась в вычислении близости суммарного времени исполнения полученного с помощью алгоритма к глобально оптимальному, описанного в разделе 5.4.

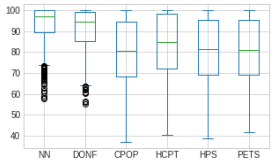
Результаты сравнительного анализа, представленные в таблице 5, показывают, что алгоритм DONF был лучшим среди современных алгоритмов построения расписаний на направленных ациклических графах. В частности, в конфигурации с 30 узлами предлагаемый алгоритм показал улучшение на 4.81% по сравнению с DONF и ещё более существенное улучшение на 17.5% по сравнению с другими передовыми алгоритмами. Хотя в конфигурациях 2 и 3 предлагаемый алгоритм оставался конкурентоспособным с DONF, он не продемонстрировал значительных улучшений по сравнению с ним. Тем не менее он постоянно превосходил другие эталонные алгоритмы на всех протестированных размерностях.

Хотя DONF показывает конкурентоспособные результаты по сравнению с предлагаемым алгоритмом на размерностях графов в 60 и 90 узлов, предлагаемый алгоритм превосходит DONF на всех остальных размерах. Наиболее значительное улучшение наблюдается в конфигурации 1, со средним повышением на 7.33% по всем размерам, слегка снижаясь до 6.01% в конфигурации 2 и далее до 4.13% в конфигурации 3. Кроме того, результаты моделирования демонстрируют тенденцию к увеличению процентного прироста в целевой метрике по сравнению

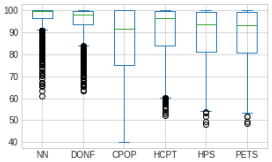
с другими современными алгоритмами по мере увеличения размера графов, со средним улучшением 2.85% при 30 узлах, 7.10% при 60 узлах и 9.23% при 90 узлах во всех конфигурациях.



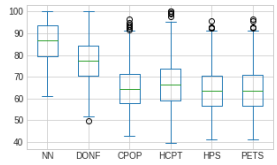
Конфигурация 1, 30 узлов



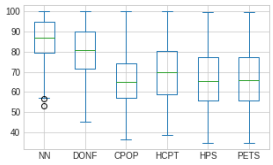
Конфигурация 2, 30 узлов



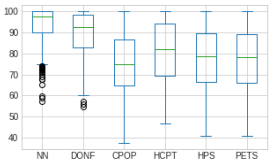
Конфигурация 3, 30 узлов



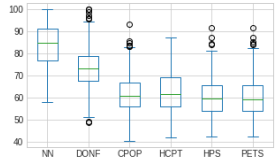
Конфигурация 1, 60 узлов



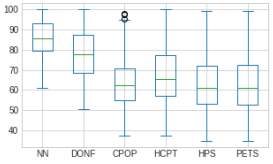
Конфигурация 2, 60 узлов



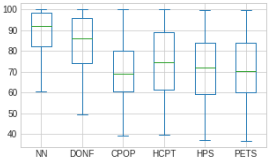
Конфигурация 3, 60 узлов



Конфигурация 1, 90 узлов



Конфигурация 2, 90 узлов



Конфигурация 3, 90 узлов

Рис. 2 Распределение близости к глобально оптимальному решению современных алгоритмов в каждом рабочем пространстве и при каждом размере

На рисунке 2 сравнивается эффективность различных алгоритмов планирования с оптимальным решением, где близость определяется как степень приближения суммарного времени выполнения к оптимуму (подробности в разделе 5.4). Диаграммы размаха показывают, что медианная производительность предложенного алгоритма превосходит показатели других алгоритмов во всех рабочих областях, последовательно достигая суммарную длительность выполнения которая ближе к оптимальной. Примечательно, что минимальный уровень производительности алгоритма на основе нейронной сети выше, демонстрируя его устойчивость даже в неблагоприятных условиях. В конфигурации 1 алгоритм на основе нейронной сети особенно хорошо проявляет себя на меньших размерностях графа, тогда как в более масштабных конфигурациях, таких как 2 и 3, он сохраняет узкий межквартильный размах при 60 и 90 узлах, что свидетельствует о стабильной производительности. Предложенный алгоритм демонстрирует более последовательные и надёжные результаты, что отражается в более узком межквартильном размахе и меньшем числе выбросов, делая его надёжным выбором для построения расписаний на графах в различных масштабах и средах.

5.3.2 Результаты для графов большой размерности

Предлагаемый алгоритм на основе нейронной сети был протестирован на графах большой размерности и было произведено сравнение с DONF, который превзошёл другие современные алгоритмы при сравнении на графах малой размерности. Чтобы определить его обобщающую способность и устойчивость в зависимости от размеров графов и различных конфигураций системы, использован следующий набор тестов:

Таблица 6 Тесты для оценки алгоритма на основе нейронной сети для графов большой размерности

Тест	Конфигурация	Количество исполнителей	Количество узлов
1	4	3	36
2	5	6	144
3	6	12	576
4	7	24	2400
5	8	48	9600
6	9	96	36864

График 3 демонстрирует улучшение производительности, предлагаемого алгоритма по сравнению с DONF на различных графах большой размерности. Ось X обозначает различные тестовые случаи, в то время как ось Y количественно выражает процентное улучшение предложенного алгоритма относительно DONF.

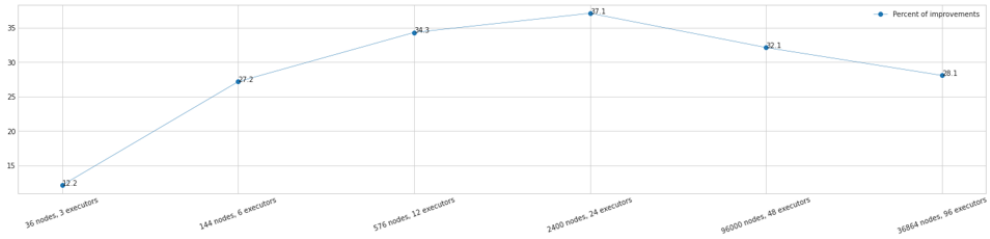


Рис. 3 Процентное улучшение makespan предложенного алгоритма по сравнению с DONF

График демонстрирует, что алгоритм на основе нейронных сетей постоянно превосходит DONF, со средним улучшением на 28.49% во всех случаях. Улучшение составляет 12.2% для самых маленьких DAG с 36 узлами и 3 исполнителями, что указывает на ограниченные возможности для оптимизации в меньших графах. Однако преимущество использования алгоритма на основе нейронных сетей становится более заметным с увеличением размера DAG, достигая улучшения в 37.1% в сценариях с 2400 узлами и 24 исполнителями, что свидетельствует о масштабируемости алгоритма.

Однако тенденция показывает небольшое снижение производительности на самых крупных протестированных DAG с 9600 и 36864 узлами, где улучшения

составляют 32.1% и 28.1% соответственно по сравнению с DONF. Это может указывать на то, что, хотя алгоритм на основе нейронных сетей хорошо масштабируется, ему может потребоваться дополнительная доработка или обучение для поддержания или улучшения своего преимущества в производительности в чрезвычайно больших и сложных конфигурациях DAG.

5.4 Сравнение с глобально оптимальным решением

В этом разделе производится сравнение предложенного алгоритма с DONF, оценивая их близость к глобально оптимальным решениям, полученным с помощью методов решения задач смешанного целочисленного линейного программирования (MILP) [25].

Используя Gurobi 9.1.2 [12] и Python 3.6, каждое решение MILP вычислялось с 15-минутным ограничением по времени, с относительным зазором оптимальности 0,03 и применением детерминированного параллельного метода.

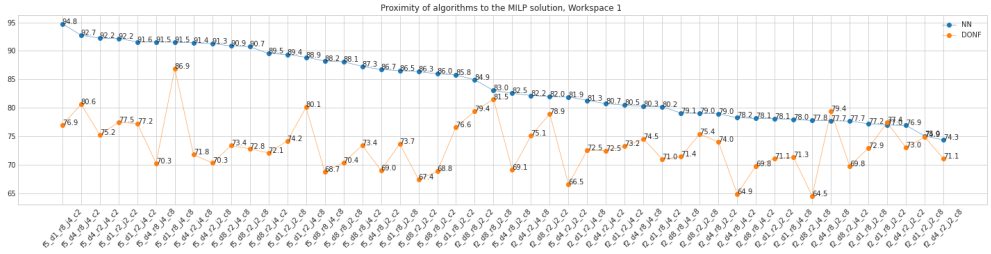
Как показано в таблице 7, предложенный алгоритм последовательно превосходит DONF, достигая более близкого приближения к решению MILP. Алгоритм на основе нейронных сетей сократил разрыв в производительности с MILP-оптимумом на 39.2%, достигая средней близости 84.08% по сравнению с 73.37% у DONF.

Таблица 7 Близость DONF предложенного алгоритма к решению MILP

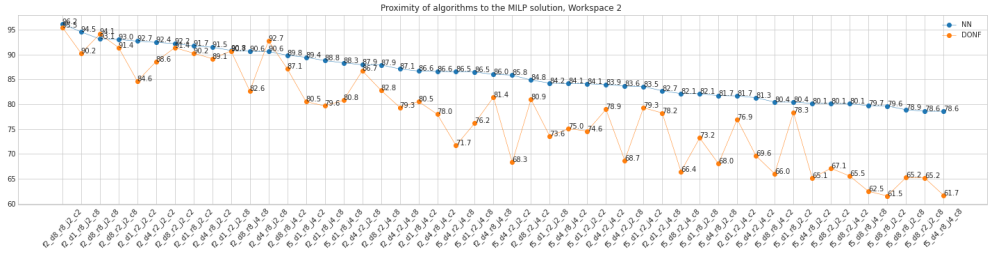
	Конфигурация 3		Конфигурация 2		Конфигурация 1	
Количество узлов	DONF	NN	DONF	NN	DONF	NN
30	94.89	**96.43**	91.13	**93.23**	85.85	**90.33**
60	89.27	**94.06**	80.66	**86.22**	77.48	**86.20**
90	84.36	**89.38**	77.82	**85.89**	73.37	**84.08**

Рисунок 4 демонстрирует производительность предложенного алгоритма и DONF к MILP-оптимуму на различных топологиях графов. По мере увеличения сложности графов — от конфигурации 1 к 3 — производительность алгоритма на основе нейронных сетей заметно улучшается.

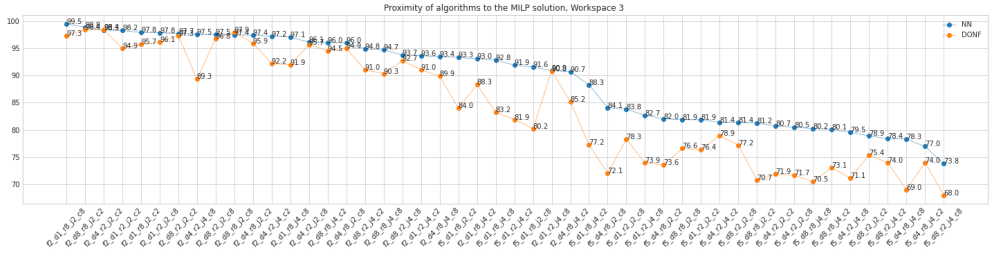
Ось X упорядочивает графы по их параметрам генерации — f, r, d, j, c — подробно описанным в разделе 3.4. Ось Y измеряет процент близости, показывая, что предложенный алгоритм превосходит DONF на всех топологиях, особенно в более «простых» конфигурациях. Хотя его превосходство уменьшается на самых крупных графах для конфигураций 2 и 3, что предполагает потенциальные области для улучшения процесса обучения, алгоритм на основе нейронных сетей последовательно демонстрирует высокую производительность, подчёркивая его эффективность как инструмента планирования в различных вычислительных средах.



Конфигурация 1: 3 исполнителя



Конфигурация 2: 6 исполнителей



Конфигурация 3: 9 исполнителей

Рис. 4 Близость решений на основе нейронных сетей и DONF к решению MILP для графа с 90 узлов

6 Заключение

В данном исследовании мы представили основанный на нейронных сетях подход для оптимизации статического планирования направленных ациклических графов в гетерогенной вычислительной среде с целью минимизации суммарного времени выполнения. Подход характеризуется уникальной архитектурой алгоритма, которая интегрирует современные метрики планирования на графах в нейронную сеть, совмещённую с генетическим программированием. Нейронная сеть, посредством обучения и оптимизации весов, эволюционирует в продвинутую метрику, точно настроенную на структуру графа, что значительно сокращает makespan.

Изначально, нейронная сеть, обученная с помощью генетического алгоритма, сортирует узлы путем анализа их характеристик с использованием комплексных метрик. Затем задачи назначаются исполнителям на основе алгоритма Earliest

Finish Time (EFT). Симуляция показывает, что алгоритм на основе нейронных сетей последовательно превосходит как традиционные, так и передовые эвристики, улучшая суммарную длительность выполнения в среднем на 6.7% для графов малой размерности и на 28.49% для графов большой размерности по сравнению с ведущим алгоритмом DONF. Он также демонстрирует среднего улучшения на 22.1% по сравнению с другими популярными алгоритмами, такими как CPOP, HCPT, HPS и PETS для графов малой размерности. При сравнении с глобальным оптимальным решением MILP для графов малой размерности алгоритм достигает близости в диапазоне к нему от 84.08% до 96.43%, а для крупномасштабных графов он покрывает 39.2% разрыва близости между DONF и решением MILP.

Будущие исследования направлены на усовершенствование алгоритма для многомерных требований к исполнителям и изучение планирования динамически поступающих узлов графа в условиях потоковой обработки в режиме онлайн.

Благодарности. Работа выполнена в рамках исследовательского проекта 95438429 в Санкт-Петербургском государственном университете.

Список литературы

[1] Ullman, J.D.: Np-complete scheduling problems. Journal of Computer and System Sciences **10**, 384 (1975)

[2] Durillo, J.J., Prodan, R.: Multi-objective workflow scheduling in amazon ec2. Cluster Computing: The Journal of Networks, Software Tools and Applications **17**(2), 169–189 (2014)

[3] Duan, Y., Wang, N., W, J.: Reducing makespans of dag scheduling through interleaving overlapping resource utilization. In: 2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), pp. 392–400 (2020)

[4] Huang, B.: Security modeling and efficient computation offloading for service workflow in mobile edge computing. Future Generation Computer System **97**, 755–774 (2019)

[5] Abadi, M., et al.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems (2016)

[6] Naveed, H., Khan, A.U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., Mian, A.: A Comprehensive Overview of Large Language Models (2024). https://arxiv.org/abs/2307.06435

[7] Banzhaf, W.: Evolutionary Computation and Genetic Programming, pp. 429–447. Elsevier, (2013). https://doi.org/10.1016/B978-0-12-415995-2.00017-9

[8] Fang, H., Ross, P.M., Corne, D.: A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling programs. In: Proc. of the 5th International Conference on Genetic Algorithms (1993)

[9] Huang, Z., Mei, Y., Zhang, F., Zhang, M.: Graph-based linear genetic programming: a case study of dynamic scheduling. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22), pp. 955–963. Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3512290.3528730

[10] Carson, J.: Genetic Algorithms: Advances in Research and Applications. Computer Science, Technology and Applications. Nova Science Publishers, Inc, New York (2017)

[11] Esfahanian, P., Akhavan, M.: GACNN: Training Deep Convolutional Neural Networks with Genetic Algorithm (2019)

[12] Gurobi Optimization. https://www.gurobi.com. Accessed: 1 March 2022 (2022)

[13] Hwang, J.J., Chow, Y.C., Anger, F.D., Lee, C.Y.: Scheduling precedence graphs in systems with interprocessor communication times. SIAM Journal on Computing **18**(2), 244–257 (1989)

[14] Suman, C., Kumar, G.: Analysis of process scheduling algorithm for multiprocessor system. In: 2018 7th International Conference on Reliability, pp. 564–569 (2018)

[15] Sih, G.C., Lee, E.A.: A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. IEEE Transactions on Parallel and Distributed Systems **4**(2), 175–187 (1993)

[16] El-Rewini, H., Lewis, T.G.: Scheduling parallel program tasks onto arbitrary target machines. Journal of Parallel and Distributed Computing **9**(2), 138–153 (1990)

[17] Lin, H., Li, M., Jia, C., Liu, J., An, H.: Degree-of-node task scheduling of fine-grained parallel programs on heterogeneous systems. Journal of Computer Science and Technology **34**(5), 1096–1108 (2019)

[18] Orhean, A.I., Pop, F., Raicu, I.: New scheduling approach using reinforcement learning for heterogeneous distributed systems. Journal of Parallel and Distributed Computing **117**, 292–302 (2018)

[19] Loth, M., et al.: Hybridizing constraint programming and Monte-Carlo tree search: application to the job shop problem (2013)

[20] Arabnejad, H.: List Based Task Scheduling Algorithms on Heterogeneous Systems-an Overview. In: Doctoral Symposium in Informatics Engineering, p. 93 (2013)

[21] Ilavarasan, E., Thambidurai, P., Mahilmannan, R.: Performance effective task

scheduling algorithm for heterogeneous computing system. In: Parallel and Distributed Computing, The 4th International Symposium, pp. 28–38 (2005)

[22] Ilavarasan, E., Thambidurai, P.: Low complexity performance effective task scheduling algorithm for heterogeneous computing environments. Journal of Computer Sciences **3**(2), 94–103 (2007)

[23] Daggen. https://github.com/frs69wq/daggen. Last accessed: 1 March 2024 (2024)

[24] Flint, C., Bramas, B.: Finding new heuristics for automated task prioritizing in heterogeneous computing (2020)

[25] Appendix C: Integer Programming Models for Sequencing, pp. 471–477. John Wiley Sons, Ltd (2009). https://doi.org/10.1002/9780470451793.app3