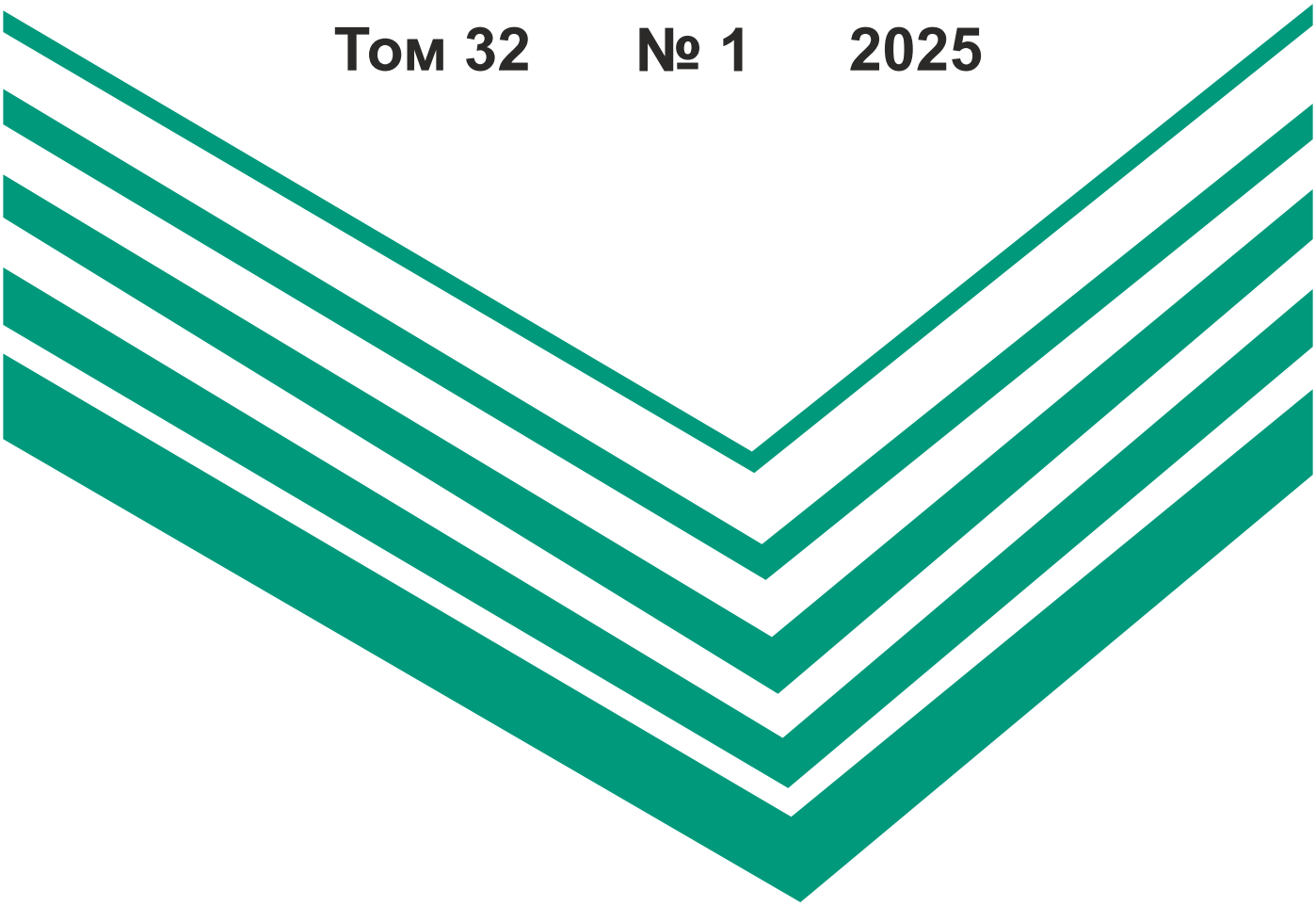
ISSN 2949-5598

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

Том 32 № 1 2025



Новосибирск
Издательство Института математики

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Январь–март 2025. Т. 32, № 1. С. 99–121

УДК 519.8 DOI: 10.33048/daio.2025.32.813

ДВУХСТАДИЙНЫЙ АЛГОРИТМ ДЛЯ ДИНАМИЧЕСКОЙ ЗАДАЧИ УПАКОВКИ В КОНТЕЙНЕРЫ С ГРУППАМИ РАЗМЕЩЕНИЯ

А. В. Ратушный [a], *Ю. А. Кочетов* [b]

Институт математики им. С. Л. Соболева,
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия
E-mail: [a] alexeyratushny@gmail.com, [b] jkochet@math.nsc.ru

Аннотация. Рассматривается новая задача динамической упаковки в контейнеры, актуальная для облачных вычислений. Для каждого предмета (виртуальной машины) известны время создания, удаления и требуемые ресурсы. Контейнеры (серверы) имеют NUMA-архитектуру и определённые правила при размещении машин. Серверы собраны в стойки, а некоторые машины образуют группы. Каждая группа разделена на партиции. Машины из разных партиций нельзя размещать на одной стойке для обеспечения отказоустойчивости системы. Требуется упаковать все машины в минимальное число стоек на заданном горизонте планирования. Для решения задачи разработан двухстадийный алгоритм: построение начального решения, в котором нарушается часть ограничений, и итеративное улучшение с помощью локального поиска, направленное на устранение нарушений. Применяя предложенный подход на открытых тестовых примерах, удалось достичь среднего отклонения от нижней границы в 3,8%. Табл. 3, ил. 4, библиогр. 27.

Ключевые слова: задача упаковки в контейнеры, виртуальная машина, конфликт, группа размещения.

Введение

Развитие облачных вычислений в последние десятилетия породило множество актуальных задач в области информационных технологий. Одним из ключевых направлений является оптимизация использования серверных ресурсов с целью повышения энергоэффективности, что включает в себя решение задачи динамической упаковки в контейнеры. В этой задаче упаковки фиксируется некоторый горизонт планирования, в течение которого необходимо оптимально распределить виртуальные машины (ВМ) по серверам. Все серверы идентичны и обладают NUMA-

архитектурой [1, 2], т. е. состоят из нескольких NUMA-узлов, каждый из которых характеризуется определённым объёмом оперативной памяти (RAM) и числом процессорных ядер (CPU). Каждой виртуальной машине требуются серверные ресурсы, количество которых определяется её типом. Важно уточнить, что число типов ВМ обычно значительно меньше общего числа виртуальных машин. Тип ВМ также определяет её размер — малая или большая. Малая виртуальная машина занимает ресурсы только одного NUMA-узла, тогда как большая ВМ равномерно распределяется между двумя различными NUMA-узлами. Временной интервал использования ресурсов каждой ВМ задаётся во входных данных задачи. В дополнение в задаче имеются стойки и группы размещения виртуальных машин с партициями [3], что является основным отличием от задач, рассмотренных в [4, 5]. Стойка представляет собой ограниченный набор серверов, причём каждый сервер принадлежит только одной стойке. Часть виртуальных машин объединена в группы, разделенные на непересекающиеся партиции (подмножества ВМ), находящиеся в конфликте друг с другом. Виртуальные машины из разных партиций одной группы не могут располагаться на одной стойке одновременно, а конфликт между двумя такими ВМ возникает только при пересечении их интервалов существования.

Описанная задача NP-трудна в сильном смысле, поскольку является обобщением задачи упаковки в контейнеры. Однако если рассматривать задачу разрешения конфликтов в группах ВМ с партициями изолированно, то задача раскраски графа конфликтов в этом случае становится полиномиально разрешимой. Это обусловлено возможностью раскрасить каждую партицию в отдельный цвет, что приводит к оптимальному решению. Данная особенность существенно отличает рассматриваемую задачу от сценария, где любые две ВМ могут конфликтовать, а также от задачи с конфликтами между типами ВМ, исследованной в [5]. Таким образом, наличие партиций вводит специфические структурные особенности, которые могут быть учтены при разработке алгоритмических подходов к решению данной задачи. Похожая задача рассмотрена в статье [6]. В отличие от постановки, представленной ниже, в которой рассматриваются конфликты, в [6] акцент делается на ограничениях, касающихся совместного размещения виртуальных машин из одной группы. Авторы приводят MIP модель и формулируют несколько теорем, касающихся сложности данной задачи. Оказывается, что даже при сильных предположениях, таких как наличие только одного момента времени или одной группы, задача остаётся NP-трудной.

Задача оптимизации ресурсов остаётся актуальной, причём продолжают возникать её новые вариации. Так, например, в [7] представлен широкий спектр сценариев и задач, возникающих в силу необходимости

разработки оптимальных расписаний для виртуальных машин с учётом технических особенностей современных вычислительных кластеров. Авторы работы [14] уделяют внимание важности обеспечения отказоустойчивости серверов в дата-центрах, чтобы уменьшить потери производительности и обеспечить непрерывность работы системы при возникновении сбоев. Параллельно с развитием постановок задач происходит эволюция методов их решения. В работах [9, 13] проводится анализ классических алгоритмов, таких как First Fit, Best Fit и Next Fit. В дополнение к этому авторы предлагают новые алгоритмы Hybrid First Fit и онлайн Move-to-front, построенные на основе вышеупомянутых классических подходов. Исследование [8] посвящено изучению многокритериальной постановки задачи оптимизации ресурсов. Авторы предлагают метаэвристический подход, интегрирующий жадные эвристики в адаптивную эволюционную схему. Предложенный метод демонстрирует высокую эффективность при тестировании на различных бенчмарках, показывая результаты, сопоставимые с другими современными метаэвристическими алгоритмами.

Другим важным направлением исследований является задача оптимизации энергетических, а не вычислительных ресурсов. Например, в [10] авторы детально анализируют проблемы эффективного использования энергетических ресурсов в дата-центрах и предлагают новые алгоритмы динамической упаковки: Dynamic Energy Efficient Best-Fit Decreasing и Energy Mitigation Switcher. Первый из них нацелен на оптимизацию утилизации ресурсов облачной системы, а второй выступает в качестве переключателя между различными подходами. Проведённое моделирование показывает, что эти алгоритмы могут значительно снизить общее энергопотребление и уменьшить время выполнения задач. Кроме того, в [10] применение технологий виртуализации предлагается как эффективный инструмент оптимизации распределения ресурсов. В [11] также изучается задача понижения энергопотребления, но основное внимание акцентируется на возможности динамического перераспределения виртуальных машин, что может привести к значительной экономии. Аналогично в [12] авторы рассматривают вариацию этой задачи с возможностью переупаковки виртуальных машин и предлагают алгоритм, отличительная особенность которого состоит в независимости числа переупаковок от общего числа виртуальных машин. Потенциально такой подход может повысить масштабируемость и эффективность управления ресурсами в крупных дата-центрах.

Кроме того, в последние годы приобретают популярность подходы, основанные на методах машинного обучения, в том числе это отражается на развитии программных библиотек (см., например, [15, 16]). В исследованиях [17, 18] изучаются подходы, основанные на методах обучения

с подкреплением. Авторы проводят сравнительный анализ эффективности алгоритмов машинного обучения и традиционных методов, демонстрируя возрастающую конкурентоспособность первых, что подчёркивает их потенциал в решении практических задач оптимизации. В [19] представлен обзор ограничений, присущих традиционным методам машинного обучения при решении задачи упаковки в контейнеры. Авторы показывают, как их подход может преодолеть эти ограничения благодаря обширному набору обучающих данных и точной разметке. В [20] предлагается метод решения задачи о рюкзаке, основанный на применении нейронных сетей. Приводится подробное описание архитектуры сети, принимающей на вход все доступные параметры и выдающей оптимальный набор предметов. Результаты демонстрируют потенциал данного подхода в контексте решения реальных задач. В [21] поднимается вопрос планирования в сфере распределённых вычислений. Авторы представляют прототип онлайн-планировщика, основанного на гибридном подходе, включающем в себя методы машинного обучения и динамического программирования.

Для решения рассматриваемой задачи предлагается двухстадийный подход, основанный на итеративном исправлении решения. На первом шаге строится некоторое исходное решение с помощью простого алгоритма, а затем ломается таким образом, чтобы нарушалась часть ограничений, связанных с конфликтами. На втором шаге для каждой группы определяются все конфликты, которые были нарушены, и запускается процедура их исправления. Каждая виртуальная машина с недопустимым расположением меняется местами с некоторой другой виртуальной машиной таким образом, чтобы уменьшить общее число нарушенных ограничений в текущей рассматриваемой группе. Если избавиться от всех конфликтов в группе не получается, то часть этой группы подвергается повторному размещению. Основной научный вклад данной работы заключается в исследовании новой NP-трудной задачи. Разработана математическая модель, которая учитывает различные аспекты и позволяет формально описать данную задачу. Кроме того, представлены алгоритмы получения верхних оценок. Проведённые вычислительные эксперименты на открытых данных продемонстрировали высокую эффективность предложенных методов, что подтверждает их практическую применимость для решения рассматриваемой задачи. Аналогичная задача рассмотрена в [22], где авторы предложили стохастический жадный алгоритм. Сравнение с данным подходом производится в разделе с вычислительными экспериментами.

Далее в разд. 1 приводятся необходимые обозначения, детальная постановка задачи и математическая модель. Разд. 2 содержит алгоритмы построения верхних оценок. Разд. 3 описывает используемые данные,

а также результаты численных экспериментов. В заключении кратко подводятся итоги исследования.

1. Математическая модель

Для постановки задачи и построения её математической модели введём следующие обозначения:

- T — множество моментов времени или горизонт планирования;
- $R = \{\text{RAM}, \text{CPU}\}$ — множество типов ресурсов;
- F — множество стоек;
- S_f — множество серверов, принадлежащих стойке $f \in F$;
- $S = \bigcup\limits_{f \in F} S_f$ — множество всех серверов;
- N — множество NUMA-узлов.

Стойки представляют собой наборы серверов, причём вместимость каждой стойки ограничена и, как правило, не превышает 10–20 единиц. Каждый сервер должен быть ассоциирован ровно с одной стойкой, иначе говоря, принадлежать ей. Хотя число серверов в стойках может варьироваться, основная цель задачи заключается в минимизации общего числа используемых стоек. Следовательно, в рамках данной постановки задачи можно всегда использовать максимальное заполнение каждой стойки серверами. Все серверы идентичны, но важно отметить, что различные NUMA-узлы одного и того же сервера могут отличаться по количеству доступных ресурсов.

Для работы с группами и виртуальными машинами введём следующие обозначения:

- M^S — множество малых виртуальных машин;
- M^L — множество больших виртуальных машин;
- $M = M^S \cup M^L$ — множество всех виртуальных машин;
- G — множество групп виртуальных машин;
- $M_g \subset M$ — множество виртуальных машин из группы $g \in G$;
- P_g — множество партиций в группе $g \in G$;
- $M_{gp} \subset M$ — множество ВМ из партиции $p \in P_g$ группы $g \in G$;
- K — множество типов виртуальных машин.

Группа представляет собой множество виртуальных машин: например, это могут быть виртуальные машины одного клиента. Каждая группа разделена дополнительно на подмножества, называемые партициями. Число партиций в группе обычно невелико и не превышает 10. Ключевое ограничение заключается в том, что любые две виртуальные машины из различных партиций одной группы не могут быть размещены на одной стойке (рис. 1). Таким образом, партиции можно мыслить как частный случай конфликтов между ВМ. Однако структура графа таких конфликтов (где вершины — виртуальные машины, а рёбра — конфликты между

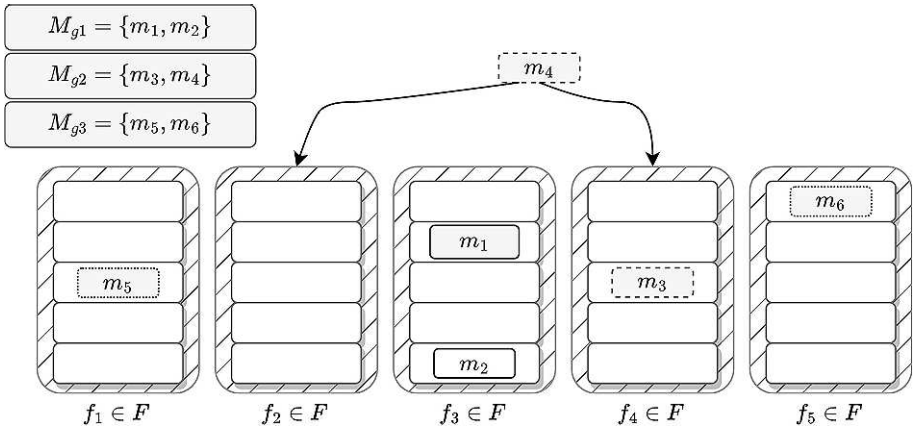


Рис. 1. Пример упаковки группы с тремя партициями

ними) достаточно специфична. Граф разбивается на несколько компонент связности, соответствующих группам. Каждую компоненту связности можно эффективно раскрасить в минимальное число цветов, просто присваивая отдельный цвет каждой партиции. Тип определяет количество памяти RAM и число ядер CPU, которые ВМ занимает на сервере. Множества типов виртуальных машин у большинства компаний, предоставляющих услуги облачных вычислений, имеют значительное пересечение. Пример такого множества типов можно найти в [24]. Также следует отметить, что мощность множества типов ВМ, как правило, намного меньше мощности множества виртуальных машин.

Следующие обозначения вводятся для описания характеристик виртуальных машин и серверов:

- C_{nr} — количество ресурса $r \in R$ на NUMA-узле $n \in N$;
- d_{mr} — количество ресурса $r \in R$, необходимого виртуальной машине для одного NUMA-узла;
- d_{kr} — количество ресурса $r \in R$, необходимого типу $k \in K$ для одного NUMA-узла;
- α_m — время создания машины $m \in M$;
- ω_m — время удаления машины $m \in M$.

Каждая виртуальная машина $m \in M^S$ ($m \in M^L$) занимает d_{mr} ($2d_{mr}$) ресурса $r \in R$ на сервере во все моменты времени $\alpha_m \leqslant t < \omega_m$. Для любого ресурса $r \in R$ справедливо равенство $d_{mr} = d_{kr}$, если виртуальной машине $m \in M$ приписан тип $k \in K$. Во входных данных гарантируется, что $d_{mr} \leqslant C_{nr}$ для любых $n \in N$, $r \in R$, $m \in M$. Важно уточнить, что если между двумя ВМ m_1 и m_2 есть конфликт, но они не пересекаются по времени, то данный конфликт можно не рассматривать.

С учётом всего вышесказанного множество конфликтов D можно задать следующим образом. Пара ВМ (m_1, m_2), обозначающая конфликт, принадлежит D, если $m_1 \in M_{gp_1}$ и $m_2 \in M_{gp_2}$, где p_1 и p_2 — различные партиции ($p_1 \neq p_2$) некоторой группы g, и существует момент времени $t \in T$ такой, что $\alpha_{m_1} \leqslant t < \omega_{m_1}$ и $\alpha_{m_2} \leqslant t < \omega_{m_2}$.

Определим также следующие множества ВМ, функционирующих в заданный момент времени $t \in T$:

- $M_t^S = \{m \in M^S \mid \alpha_m \leqslant t < \omega_m\}$ — множество малых ВМ;
- $M_t^L = \{m \in M^L \mid \alpha_m \leqslant t < \omega_m\}$ — множество больших ВМ;
- $M_t = M_t^S \cup M_t^L$ — множество всех ВМ.

Для описания математической модели понадобятся следующие булевы переменные. Для произвольных $f \in F$, $s \in S_f$, $n \in N$ определим

$$x_{fsnm} = \begin{cases} 1, & \text{если малая или первая часть большой ВМ } m \in M \\ & \text{расположена на узле } n \text{ сервера } s \text{ в стойке } f, \\ 0 & \text{иначе;} \end{cases}$$

$$y_{fsnm} = \begin{cases} 1, & \text{если вторая часть большой ВМ } m \in M^L \text{ распо-} \\ & \text{ложена на узле } n \text{ сервера } s \text{ в стойке } f, \\ 0 & \text{иначе;} \end{cases}$$

$$z_f = \begin{cases} 1, & \text{если стойка } f \text{ была активна хотя бы в один} \\ & \text{момент времени } t \in T, \\ 0 & \text{иначе.} \end{cases}$$

В указанных обозначениях математическую модель рассматриваемой задачи можно записать следующим образом:

$$\min \sum_{f \in F} z_f, \tag{1}$$

$$\sum_{f \in F} \sum_{s \in S_f} \sum_{n \in N} x_{fsnm} = 1, \quad m \in M, \tag{2}$$

$$x_{fsnm} + y_{fsnm} \leqslant 1, \quad f \in F,\ s \in S_f,\ n \in N,\ m \in M^L, \tag{3}$$

$$\sum_{n \in N} x_{fsnm} = \sum_{n \in N} y_{fsnm}, \quad f \in F,\ s \in S_f,\ m \in M^L, \tag{4}$$

$$\sum_{m \in M_t} d_{mr} x_{fsnm} + \sum_{m \in M_t^L} d_{mr} y_{fsnm} \leqslant C_{nr},$$
$$f \in F,\ s \in S_f,\ n \in N,\ t \in T,\ r \in R, \tag{5}$$

$$\sum_{s \in S_f} \sum_{n \in N} x_{fsnm} \leqslant z_f, \quad f \in F,\ m \in M, \tag{6}$$

$$\sum_{s \in S_f} x_{fsnm_1} + \sum_{s \in S_f} x_{fsnm_2} \leqslant 1, \quad (m_1, m_2) \in D,\ f \in F,\ n \in N, \qquad (7)$$

$$x_{fsnm}, y_{fsnm}, z_f \in \{0, 1\}. \qquad (8)$$

Целевая функция (1) выражает минимизацию числа задействованных стоек. Ограничения (2) гарантируют, что все виртуальные машины упакованы, а (3)–(4) отвечают за правильную упаковку больших ВМ. Неравенства (5) ограничивают количество ресурсов на каждом сервере, а (6) необходимы для подсчёта числа стоек. Считаем, что стойка была активна, если на ней обслуживалась хотя бы одна виртуальная машина. Неравенства (7) задают конфликты между виртуальными машинами.

Потребность в решении задачи для достаточно больших примеров (более 10^4 виртуальных машин) не позволяет использовать математическую модель (1)–(8), так как она требует значительных вычислительных ресурсов и времени. В качестве альтернативы предлагается использовать хорошо зарекомендовавшие себя эвристические подходы [5, 25]. Описание рассматриваемого алгоритма приводится в разд. 2.

2. Верхние оценки

При решении задачи в онлайн-постановке существует необходимость упаковывать виртуальные машины в порядке их создания, учитывая конфликты непосредственно на этапе размещения. Динамическая постановка обладает большей свободой действий, что позволяет разрабатывать более гибкие и эффективные эвристические методы. Предлагаемый подход основан на двухэтапном процессе оптимизации. На первом этапе формируется начальное решение задачи, игнорирующее часть ограничений, связанных с конфликтами. На втором этапе применяется процедура коррекции, направленная на устранение конфликтов. Разделение поиска решения на два этапа позволило достичь большей гибкости и качества итогового размещения.

На первом этапе применяется рандомизированная версия алгоритма First Fit [23] для размещения групп виртуальных машин (RFFG). Алгоритм последовательно упаковывает группы ВМ в случайном порядке одну за другой и не приступает к упаковке следующей группы, пока не закончит с текущей. При этом упаковка каждой группы производится таким образом, чтобы не возникло конфликтов между партициями одной группы. Если в примере имеются ВМ, не принадлежащие какой-либо группе, то они упаковываются последними. Случайный порядок групп позволяет достичь заметно лучшего результата, сохраняя при этом производительность алгоритма. С помощью такого подхода можно генерировать несколько потенциальных решений, из которых затем выбирается наилучшее. Важно отметить, что порядок ВМ внутри каждой группы

также случаен, что дополнительно увеличивает вариативность решений. Далее для создания возможности дальнейшего улучшения распаковывается некоторый процент $\widehat{U}$ стоек, после чего неразмещённые ВМ перепаковываются рандомизированным алгоритмом First Fit (RFF), при этом игнорируются ограничения на конфликты.

Определим схожесть виртуальных машин с помощью коэффициентов Жаккара $J(m_1, m_2)$ [26], в частности, его адаптации для прямоугольников IoU (intersection over union). В нашем случае ширина прямоугольника (виртуальной машины m) равняется $\omega_m - \alpha_m$, а высота определяется относительным значением требуемого ресурса $d_{mr} / \sum\limits_{n \in N} C_{nr}$ ($2d_{mr} / \sum\limits_{n \in N} C_{nr}$ в случае большой ВМ). Тем самым коэффициент $J(m_1, m_2, r)$ зависит не только от выбранных виртуальных машин m_1 и m_2, но и от рассматриваемого ресурса $r \in R$. Для того чтобы избавиться от данной неоднозначности, будем рассматривать адаптацию этого коэффициента:

$$J(m_1, m_2) = \min_{r \in R} J(m_1, m_2, r).$$

На втором шаге имеющееся решение итеративно перестраивается так, чтобы последовательно избавляться от нарушений ограничений. Группы виртуальных машин просматриваются одна за другой в случайном порядке. Для каждой ВМ m анализируется возможность перестановки с другими ВМ $\widehat{m}$, исключая из рассмотрения те, которые удовлетворяют некоторым правилам: находятся на той же стойке f, что и m; принадлежат уже просмотренным группам; входят в ту же группу и партицию, что и m; или имеют низкую степень схожести с m ($J(m, \widehat{m}) < \widehat{J}$). Эти критерии отсева позволяют сфокусироваться на наиболее перспективных перестановках, которые могут уменьшить число нарушений ограничений без риска добавления новых конфликтов. Перестановка двух виртуальных машин не увеличивает целевую функцию, однако такой подход не всегда может свести к нулю число нарушенных ограничений. Это приводит к необходимости дополнительного шага. Если на некоторой итерации (полном просмотре всех виртуальных машин в группе) не удалось полностью избавиться от нарушений ограничений, то часть данной группы полностью перепаковывается. В таком случае на каждой стойке остаётся только та партиция из группы, которая имеет наибольшее число виртуальных машин, а все остальные ВМ удаляются и упаковываются заново с помощью RFF. Однако в этот раз алгоритм упаковывает так, чтобы не нарушить никаких ограничений, т. е. для каждой ВМ определяется некоторый набор запрещённых стоек, куда упаковка не производится. При необходимости создаются новые стойки и серверы, так что алгоритм всегда возвращает допустимое решение. Схему алгоритма можно представить следующим образом.

Алгоритм RALR

Шаг 1. С помощью алгоритма RFFG строится начальное допустимое решение, не содержащее нарушений по ограничениям (2)–(8).

Шаг 2. Случайный набор из $\widehat{U}\%$ стоек распаковывается, а освободившиеся ВМ упаковываются заново с помощью RFF с возможным нарушением ограничений на конфликты между партициями.

Шаг 3. Для очередной группы и для каждой ВМ в ней запускается локальный поиск в пространстве перестановок местами с другими ВМ таким образом, чтобы уменьшить число нарушенных ограничений. Каждая группа просматривается полностью несколько раз до тех пор, пока удаётся устранять нарушения. Если удалось устранить все нарушения, то переходим к следующей группе, иначе переходим на шаг 4.

Шаг 4. Если не удалось устранить все нарушения для рассматриваемой группы, то она частично перепаковывается алгоритмом RFF, который исправляет все ограничения, при необходимости создавая новые сервера и стойки.

Пример поведения целевой функции и числа нарушенных ограничений во время шагов 3–4 можно найти на рис. 2. Здесь обе кривые нормализованы к интервалу [0, 1]. Кривая отклонения целевой функции показывает, как менялось её значение относительно решения, полученного на втором шаге. Резкие скачки на обеих кривых соответствуют переупаковкам групп, что, как можно видеть, не всегда приводит к ухудшению решения.

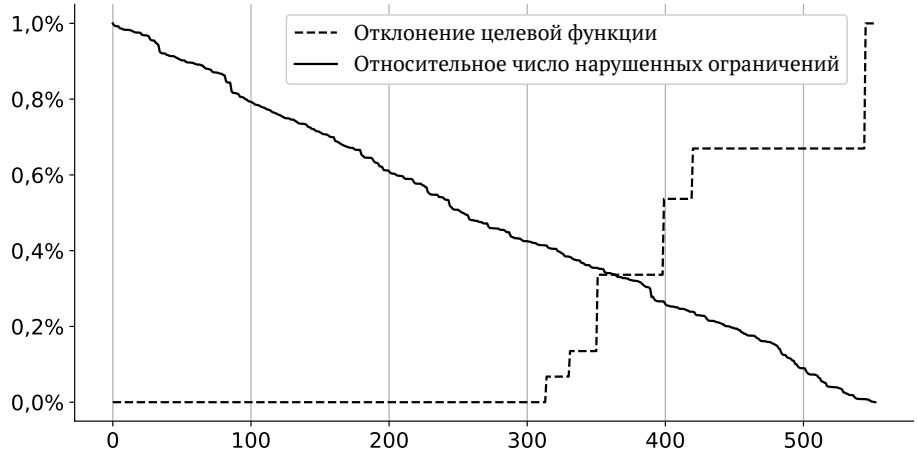


Рис. 2. Зависимость целевой функции
и числа нарушенных ограничений от номера итерации

Следует отметить, что шаги 2–4 иногда способны ухудшить изначальное решение. В случае, если такие ухудшения действительно наблюдаются, то решение, полученное методом RFFG на первом шаге возвращается в качестве окончательного. Однако результаты численных экспериментов показывают, что такая ситуация возникает достаточно редко.

Сложность шагов 1, 2 и 4 имеет порядок $O(|M||N||T|\overline{S})$, что сопоставимо со сложностью алгоритма RFF, который учитывает динамику и NUMA-архитектуру. Параметр $\overline{S}$ ограничивает число серверов сверху: например, можно использовать $\overline{S} = |M|$. Шаг 3 наиболее трудоёмкий в алгоритме RALR. Для каждой пары виртуальных машин необходимо проверить допустимость обмена их местами, что требует $O(|M|^2|T||N|)$ операций. Количество полных просмотров одной группы ограничено числом возможных конфликтов порядка $O(|M|^2)$. На практике число итераций ограничивается сверху. В проведённых вычислительных экспериментах граница равна 20, хотя алгоритму обычно требуется не более 10 итераций для достижения локального минимума. Несмотря на сложность, благодаря различным оптимизациям и ограничениям размера окрестностей локального поиска шаг 3 выполняется достаточно быстро для получения допустимого решения в обозримое время. Например, поиск решения заметно ускоряется за счёт значения порога схожести двух виртуальных машин $\widehat{J}$. Многие пары виртуальных машин (m_1, m_2) имеют достаточно низкий коэффициент $J(m_1, m_2)$, например из-за сильного разброса по времени. Попытка переставить их местами скорее всего закончится неудачей. Даже небольшое значение $\widehat{J}$ позволяет эффективно исключить

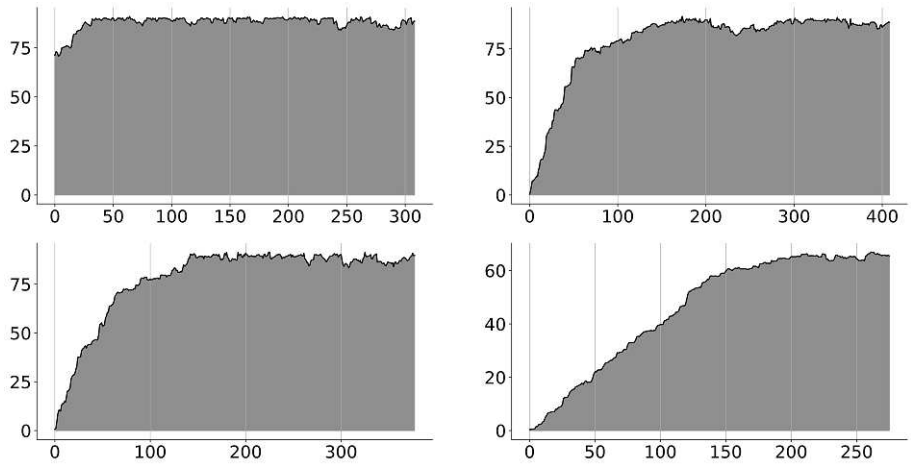


Рис. 3. Нижняя оценка по моментам времени для различных примеров

из рассмотрения большую часть неудачных пар и значительно уменьшить размер окрестности.

3. Численные эксперименты

Для проведения численных экспериментов применён случайный генератор, предоставленный одной из компаний, занимающихся облачными вычислениями. Генератор имитирует поведение реальной нагрузки на серверы. На каждом шаге он создаёт запросы на размещение виртуальных машин последовательно, стараясь, чтобы загрузка кластера (ограниченного набора стоек) находилась в определённом интервале. Новые группы размещения и типы виртуальных машин генерируются в соответствии с частотами, полученными в результате анализа истории работы одного из кластеров той же компании. Примеры суммарной нагрузки по числу стоек можно увидеть на рис. 3. По оси абцисс отложено время, а по оси ординат — минимально необходимое число стоек, вычисленное по формуле

$$\max_{r\in R}\frac{\sum\limits_{m\in M_t^S} d_{mr} + 2\sum\limits_{m\in M_t^L} d_{mr}}{\sum\limits_{n\in N} C_{nr}}.$$

Все рассмотренные примеры находятся в открытом доступе [27].

Анализ примеров и различные статистики приведены в табл. 1. Все примеры поделены на несколько семейств. Обозначения LPO, DMP и MP в названии семейства показывают тип примеров, где LPO — примеры только с большими группами партиций, MP — примеры с различными группами партиций (с большими и малыми), а DMP — обобщение примеров MP с добавлением виртуальных машин, которые не принадлежат никакой группе, т. е. не имеют конфликтов ни с какими другими ВМ. Буквенные обозначения «l», «m» и «s» соответствуют размеру примера, т. е. числу виртуальных машин и количеству моментов времени, а обозначения «2» и «4» показывают число NUMA-узлов на серверах. Максимальный размер стойки во всех примерах равен 10 серверам. В среднем во всех примерах по 5 партиций в группе. Каждое семейство состоит из 25 примеров, а суммарно рассмотрено 450 примеров. В табл. 1 приводятся средние значения по всем примерам каждого семейства.

Для сравнения выбраны следующие алгоритмы:

• RALR — алгоритм, описанный в разд. 2, с параметрами $\widehat{J} = 0{,}25$ и $\widehat{U} = 66\%$;

• INIT + NMSLI — стохастический алгоритм, приведённый в [22]; этот подход основан на алгоритме First Fit и методе бисекции;

Таблица 1

Анализ примеров

Семейство примеров	Среднее число ВМ	Среднее число моментов времени	Среднее число групп с партициями	Среднее число ВМ в группе	Среднее число ВМ без конфликтов
LPO_l_2	64 703,12	338,28	605,92	106,85	0,00
LPO_l_4	68 349,76	280,40	671,84	101,80	0,00
LPO_m_2	28 825,24	153,72	272,20	105,89	0,00
LPO_m_4	27 328,32	115,04	271,76	100,59	0,00
LPO_s_2	13 090,92	70,28	124,56	105,20	0,00
LPO_s_4	12 781,92	55,24	124,92	102,20	0,00
DMP_l_2	38 074,36	379,68	354,36	69,60	13 452,12
DMP_l_4	42 577,24	311,80	425,72	63,27	15 668,76
DMP_m_2	32 460,80	343,88	273,60	69,51	13 452,12
DMP_m_4	32 767,40	276,00	271,36	62,99	15 668,76
DMP_s_2	22 066,76	274,48	124,88	68,95	13 452,12
DMP_s_4	23 220,84	235,72	121,88	62,06	15 668,76
MP_l_2	44 820,88	308,88	641,40	69,96	0,00
MP_l_4	49 286,20	217,60	737,04	66,91	0,00
MP_m_2	19 213,04	136,16	273,84	70,08	0,00
MP_m_4	18 679,64	84,28	278,64	66,98	0,00
MP_s_2	8 838,44	63,32	126,16	69,98	0,00
MP_s_4	8 213,20	37,24	121,92	67,46	0,00

• RFFG — первый шаг алгоритма RALR; метод является вариацией алгоритма First Fit, в которой группы упаковываются одна за другой в случайном порядке таким образом, чтобы не нарушались ограничения на конфликты, а виртуальные машины вне групп упаковываются так же в случайном порядке в конце алгоритма;

• RAR — упрощённый вариант алгоритма RALR, в котором отсутствует самый трудоёмкий шаг 3;

• RFF — рандомизированный First Fit алгоритм, который упаковывает виртуальные машины в случайном порядке, не учитывает конфликты и необходим, чтобы оценить дополнительную сложность примеров, вызванную наличием групп.

Все алгоритмы реализованы на языке программирования Python, вычисления проводились на компьютере с процессором AMD EPYC 7502 32-core. В каждом примере вычислено относительное отклонение по формуле $\text{gap} = (\text{UB} - \text{LB})/\text{UB}$, где UB — это результат некоторого алгоритма,

Таблица 2

Сравнение алгоритмов

Семейство примеров	RALR		INIT + MSLI		RFFG		RAR	
	сред.	стд.	сред.	стд.	сред.	стд.	сред.	стд.
LPO_l_2	5,14	0,87	**4,36**	0,89	12,39	0,68	13,64	1,09
LPO_l_4	**6,12**	1,33	7,57	0,56	9,93	0,49	13,49	0,80
LPO_m_2	**1,49**	2,21	1,55	1,51	6,85	1,91	9,98	1,85
LPO_m_4	**4,52**	2,37	8,25	1,89	8,04	1,36	12,68	1,72
LPO_s_2	**3,27**	3,09	3,90	2,85	6,24	3,22	9,63	2,69
LPO_s_4	**6,27**	2,72	11,95	3,34	9,32	2,75	14,52	2,73
DMP_l_2	3,02	1,10	**2,11**	0,87	7,34	1,48	8,26	1,41
DMP_l_4	**3,06**	0,46	3,31	0,61	4,36	1,00	4,56	1,94
DMP_m_2	1,94	1,21	**1,42**	1,03	6,42	1,82	7,33	1,93
DMP_m_4	**2,85**	0,87	3,01	0,91	4,35	1,11	4,40	3,33
DMP_s_2	1,43	1,13	**1,19**	1,16	5,21	2,08	6,03	1,93
DMP_s_4	**3,10**	1,19	3,11	1,24	4,40	1,54	4,95	1,34
MP_l_2	3,96	0,78	**3,13**	0,63	9,79	0,73	11,16	0,80
MP_l_4	**5,81**	1,00	6,58	0,64	7,64	0,57	10,64	0,70
MP_m_2	2,20	1,21	**2,02**	1,49	7,00	2,22	9,93	2,67
MP_m_4	**5,45**	1,74	7,91	1,26	7,10	1,58	12,22	3,93
MP_s_2	**3,05**	2,48	3,85	3,60	5,94	3,79	11,05	3,88
MP_s_4	**6,41**	3,65	11,26	3,87	7,58	2,69	12,23	2,64
Среднее	3,84	1,63	4,80	1,57	7,22	1,72	9,81	2,08

a LB — нижняя оценка, полученная с помощью генерации столбцов для задачи упаковки в контейнеры с одним моментом времени без конфликтов [5]. В табл. 2 для каждого алгоритма и семейства примеров представлено по два значения в процентах: среднее значение gap (сред.) по семейству примеров и стандартное отклонение gap (стд.), которое иллюстрирует стабильность работы алгоритма на данном подмножестве примеров. Так как во всех алгоритмах присутствует фактор случайности, они запускались до достижения установленного ограничения по времени. Вне зависимости от семейства на каждый пример размера l выделялось 3 часа, на размер m — 2 часа, а на размер s — всего 1 час. Исключение составил алгоритм INIT + MSLI, параметры запуска которого описаны более подробно в [22].

Анализ результатов показывает, что алгоритм RALR демонстрирует значительно лучшие результаты по сравнению с RFFG и RAR на всех представленных наборах данных, даже несмотря на то, что RFFG и RAR более эффективны и успевали обработать больше запусков за отведённое

время. Наблюдаемая разница в отклонениях подчёркивает значимость наличия всех этапов, реализованных в RALR. Тем не менее, несмотря на свою простоту, алгоритм RFF смог продемонстрировать среднее отклонение, равное 2,51%, что заметно ниже, чем у других подходов. Нужно отметить, что RFF не учитывает наличие конфликтов, и выявленная разница, вероятно, обусловлена дополнительной сложностью примеров, связанной с наличием групп и партиций. Кроме того, при анализе результатов работы RFF замечено, что на примерах с четырьмя NUMA-узлами он имеет более высокие значения gap, чем на примерах с двумя NUMA-узлами. По всей видимости, примеры с четырьмя NUMA-узлами оказываются несколько сложнее, и для них обеспечение высокой утилизации ресурсов серверов вызывает больше сложностей. Важно, что такое наблюдение сделано именно для алгоритма RFF, поскольку этот подход не учитывает некоторые дополнительные ограничения и имеет более простую структуру. Дополнительно алгоритму RALR удалось улучшить результаты алгоритма INIT + MSLI на части примеров. В среднем получилось уменьшить gap более чем на 1%. Однако большая часть данной разницы связана с семействами примеров LPO_s_4, LPO_m_4 и MP_s_4, которые оказались для INIT + MSLI достаточно сложными, даже в сравнении с RFFG.

Таблица 3

Влияние параметра $\widehat{U}$ на отклонение gap

Семейство примеров	Значение $\widehat{U}$ (%)						
	10	25	33	50	66	75	90
	Значение gap (%)						
DMP_l_2	6,62	5,30	4,16	3,46	3,02	4,19	4,62
DMP_l_4	4,29	3,65	3,62	3,32	3,06	3,69	3,73
DMP_m_2	4,34	3,47	2,82	2,39	1,94	2,52	2,80
DMP_m_4	4,33	3,47	3,29	3,12	2,85	3,59	3,68
DMP_s_2	3,71	2,97	2,41	2,23	1,43	1,80	1,89
DMP_s_4	4,25	3,82	3,57	3,40	3,10	3,42	3,48
Среднее	4,59	3,78	3,31	3,02	2,57	3,20	3,37

В ходе исследования вызвал интерес также анализ влияния параметра $\widehat{U}$ на эффективность работы алгоритма RALR. Результаты данного анализа представлены в табл. 3 и на рис. 4. При относительно малых значениях параметра $\widehat{U}$ результаты RALR схожи с результатами RFFG. Это означает, что при такой малой переупаковке алгоритму RALR не удаётся эффективно исправлять нарушения, и он часто возвращает решение, полученное на первом шаге. При увеличении значения параметра $\widehat{U}$

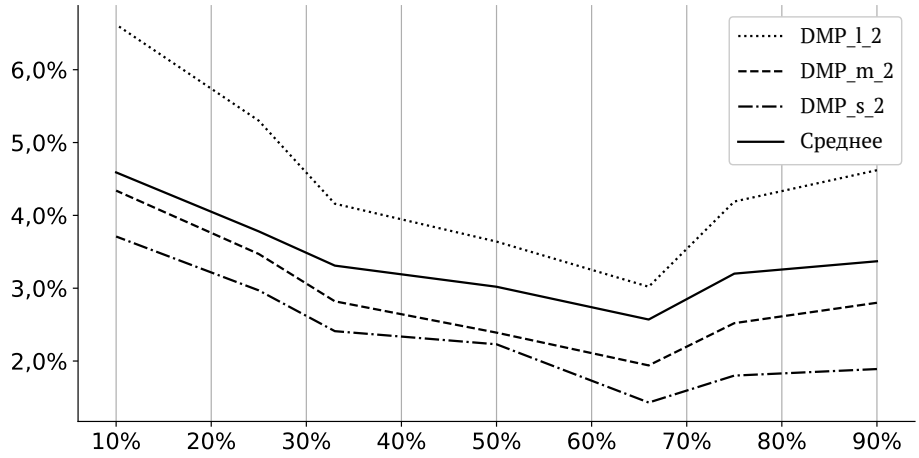


Рис. 4. Зависимость отклонения gap от значения параметра $\widehat{U}$

до слишком больших величин качество алгоритма RALR также остаётся низким. В данном случае разрушается слишком большая часть решения. Упаковка с помощью RFF значительной части BM, по всей видимости, оказывается недостаточно хорошей для того, чтобы её можно было эффективно исправить. Это приводит к слишком частому использованию процедуры переупаковки групп с шага 4, что дополнительно снижает качество.

Заключение

В представленной работе рассмотрена динамическая задача упаковки в контейнеры с учётом групп размещения, которые накладывают ограничения на расположение виртуальных машин. Представлена математическая модель задачи, продемонстрирована работа нескольких эвристических методов, а также проведено их сравнение на большом объёме открытых примеров. Кроме того, выполнен анализ одного из алгоритмов для более глубокого понимания его поведения.

Предложенный метод в дальнейшем может быть обобщён на другие модификации задачи с учётом развития облачных сервисов и появления новых возможностей для клиентов. Например, можно исследовать дополнительные ограничения на группы размещения, такие как конфликты на уровне серверов вместо стоек, или требования по близости размещения виртуальных машин внутри одной группы [6]. Также перспективным направлением является улучшение нижних оценок, так как текущий подход не учитывает влияние групп размещений.

Финансирование работы

Исследование выполнено в рамках гос. задания Института математики им. С. Л. Соболева (проект № FWNF–2022–0019). Дополнительных грантов на проведение или руководство этим исследованием получено не было.

Конфликт интересов

Авторы заявляют, что у них нет конфликта интересов.

Литература

1. **Lameter C.** NUMA (Non-uniform memory access): An overview // Queue. 2013. V. 11. P. 40–51. DOI: 10.1145/2508834.2513149.
2. **Li P., Wu X., Luo Y., Wang L.-M., Yadav N., Pepin M., Morgan J.** NUMAware: Accelerate VM-to-VM I/O performance in NUMA servers for NFV applications // IEEE Conf. Network Function Virtualization and Software Defined Networks (Palo Alto, CA, USA, Nov. 7–9, 2016). Piscataway: IEEE, 2016. URL: nfvsdn2016.ieee-nfvsdn.org/program/demonstrations/index.html (accessed: 20.08.2024).
3. Placement groups for your Amazon EC2 instances // Amazon elastic compute cloud: User guide. Seattle: Amazon Web Serv., 2024. URL: docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html (accessed: 20.08.2024).
4. **Ratushnyi A. V., Kochetov Yu. A.** A column generation based heuristic for a temporal bin packing problem // Mathematical optimization theory and operations research. Proc. 20th Int. Conf. MOTOR 2021 (Irkutsk, Russia, July 5–10, 2021). Cham: Springer, 2021. P. 96–110. (Lect. Notes Comput. Sci.; V. 12755). DOI: 10.1007/978-3-030-77876-7_7.
5. **Ratushnyi A. V.** A pattern-based heuristic for a temporal bin packing problem with conflicts // Mathematical optimization theory and operations research. Proc. 22nd Int. Conf. MOTOR 2023 (Yekaterinburg, Russia, July 2–8, 2023). Cham: Springer, 2023. P. 161–175. (Commun. Comput. Inf. Sci.; V. 1881). DOI: 10.1007/978-3-031-43257-6_13.
6. **Borisovsky P. A., Eremeev A. V., Panin A. A., Sakhno M. A.** Temporal bin packing problems with placement constraints: MIP-models and complexity // Mathematical optimization theory and operations research. Proc. 23rd Int. Conf. MOTOR 2024 (Omsk, Russia, June 30 – July 6, 2024). Cham: Springer, 2024. P. 157–169. (Lect. Notes Comput. Sci.; V. 14766). DOI: 10.1007/978-3-031-62792-7_11.
7. **Grushin D. A., Kuzyurin N. N.** On effective scheduling in computing clusters // Program. Comput. Softw. 2019. V. 45. P. 398–404. DOI: doi.org/10.1134/S0361768819070077.
8. **Spencer K. Y., Tsvetkov P. V., Jarrell J. J.** A greedy memetic algorithm for a multiobjective dynamic bin packing problem for storing cooling objects // J. Heuristics. 2019. V. 25. P. 1–45. DOI: 10.1007/s10732-018-9382-0.

9. **Li Y., Tang X., Cai W.** Dynamic bin packing for on-demand cloud resource allocation // IEEE Trans. Parallel Distrib. Syst. 2016. V. 27, No. 1. P. 157–170. DOI: 10.1109/TPDS.2015.2393868.
10. **Gupta N., Gupta K., Gupta D.** [et al.]. Enhanced virtualization-based dynamic bin-packing optimized energy management solution for heterogeneous clouds // Math. Probl. Eng. 2022. V. 2022, No. 1. Article ID 8734198. 11 p. DOI: 10.1155/2022/8734198.
11. **Beloglazov A., Buyya R.** Energy efficient allocation of virtual machines in cloud data centers // 10th IEEE/ACM Int. Conf. Cluster, Cloud and Grid Computing (Melbourne, Australia, May 17–20, 2010). Piscataway: IEEE, 2014. P. 577–578. DOI: 10.1109/CCGRID.2010.45.
12. **Berndt S., Jansen K., Klein K.-M.** Fully dynamic bin packing revisited // Math. Program. 2020. V. 179. P. 109–155. DOI: 10.1007/s10107-018-1325-x.
13. **Murhekar A., Arbour D., Mai T., Rao A.** Brief announcement: Dynamic vector bin packing for online resource allocation in the cloud // Proc. 35th ACM Symp. Parallelism in Algorithms and Architectures (Orlando, FL, USA, June 17–19, 2023). New York: ACM, 2023. P. 307–310. DOI: 10.1145/3558481.3591314.
14. **Daudjee K., Kamali S., López-Ortiz A.** On the online fault-tolerant server consolidation problem // Proc. 26th ACM Symp. Parallelism in Algorithms and Architectures (Prague, Czech Rep., June 23–25, 2014). New York: ACM, 2014. P. 12–21. DOI: 10.1145/2612669.2612686.
15. **Hubbs C., Perez H., Sarwar O., Sahinidis N., Grossmann I., Wassick J.** OR-Gym: A reinforcement learning library for operations research problem. Ithaca, NY: Cornell Univ., 2020. 29 p. (Cornell Univ. Libr. e-Print Archive; arXiv:2008.06319). DOI: 10.48550/arXiv.2008.06319.
16. **Wan C., Li T., Wang J.** RLOR: A flexible framework of deep reinforcement learning for operation research. Ithaca, NY: Cornell Univ., 2023. 21 p. (Cornell Univ. Libr. e-Print Archive; arXiv:2303.13117). DOI: 10.48550/arXiv.2303.13117.
17. **Balaji B., Bell-Masterson J., Bilgin E.** [et al.]. ORL: Reinforcement learning benchmarks for online stochastic optimization problems. Ithaca, NY: Cornell Univ., 2019. 12 p. (Cornell Univ. Libr. e-Print Archive; arXiv:1911.10641). DOI: 10.48550/arXiv.1911.10641.
18. **Yang T., Luo F., Fuentes J., Ding W., Gu C.** A flexible reinforced bin packing framework with automatic slack selection // Math. Probl. Eng. 2021. V. 2021. Article ID 6653586. 15 p. DOI: 10.1155/2021/6653586.
19. **Mao F., Blanco E., Fu M.** [et al]. Small boxes big data: A deep learning approach to optimize variable sized bin packing // Proc. IEEE 3rd Int. Conf. Big Data Computing Service and Applications (San Francisco, CA, USA, Apr. 6–9, 2017). Piscataway: IEEE, 2017. P. 80–89. DOI: 10.1109/BigDataService.2017.18.
20. **Nomer H. A., Alnowibet K. A., Elsayed A., Mohamed A. W.** Neural knapsack: A neural network based solver for the knapsack problem // IEEE Access. 2020. V. 8. P. 224200–224210. DOI: 10.1109/ACCESS.2020.3044005.

21. Toporkov V. V., Yemelyanov D. M., Bulkhak A. N. Machine learning-based online scheduling in distributed computing // Parallel processing and applied mathematics. Rev. Sel. Pap. 14th Int. Conf. PPAM 2022 (Gdansk, Poland, Sept. 11–14, 2022). Pt. II. Cham: Springer, 2023. P. 248–259. (Lect. Notes Comput. Sci.; V. 13827). DOI: 10.1007/978-3-031-30445-3_21.

22. Turnaev A. M., Panin A. A. Stochastic greedy algorithms for a temporal bin packing problem with placement groups // Mathematical optimization theory and operations research. Proc. 23rd Int. Conf. MOTOR 2024 (Omsk, Russia, June 30 – July 6, 2024). Cham: Springer, 2024. P. 199–211. (Lect. Notes Comput. Sci.; V. 14766). DOI: 10.1007/978-3-031-62792-7_14.

23. Johnson D. S. Fast algorithms for bin packing // J. Comput. Syst. Sci. 1974. V. 8. P. 272–314. DOI: 10.1016/S0022-0000(74)80026-7.

24. Amazon EC2 instance types // Amazon elastic compute cloud documentation. Seattle: Amazon Web Serv., 2024. URL: docs.aws.amazon.com/ec2/latest/instancetypes/instance-types.html (accessed: 20.08.2024).

25. Munien C., Ezugwu A. Metaheuristic algorithms for one-dimensional bin-packing problems: A survey of recent advances and applications // J. Intell. Syst. 2021. V. 30. P. 636–663. DOI: 10.1515/jisys-2020-0117.

26. Kosub S. A note on the triangle inequality for the Jaccard distance // Pattern Recognit. Lett. 2019. V. 120. P. 36–38. DOI: 10.1016/j.patrec.2018.12.007.

27. Temporal bin packing problem with placement groups // Discrete location problems. Benchmark library. Novosibirsk: Sobolev Inst. Math., 2024. URL: old.math.nsc.ru/AP/benchmarks/Temporal%20Bin%20Packing/binpack.html (accessed: 20.08.2024).

Ратушный Алексей Владленович
Кочетов Юрий Андреевич

Статья поступила
11 сентября 2024 г.

После доработки —
17 сентября 2024 г.

Принята к публикации
22 сентября 2024 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
January–March 2025. Vol. 32, No. 1. P. 99–121

UDC 519.8 DOI: 10.33048/daio.2025.32.813

A TWO-STAGE ALGORITHM FOR THE DYNAMIC BIN PACKING PROBLEM WITH PLACEMENT GROUPS

A. V. Ratushnyi[a] and *Yu. A. Kochetov*[b]

Sobolev Institute of Mathematics,
4 Acad. Koptyug Avenue, 630090 Novosibirsk, Russia
E-mail: [a] alexeyratushny@gmail.com, [b] jkochet@math.nsc.ru

Abstract. A new dynamic bin packing problem relevant to cloud computing is considered. The creation time, deletion time, and required resources are known for each item (virtual machine). The containers (servers) have a NUMA architecture and specific rules for placing the machines. The servers are grouped into racks, and some machines form groups. Each group is divided into partitions. Machines from different partitions cannot be placed on the same rack to ensure system fault tolerance. The objective is to pack all the machines into the minimum number of racks over a given planning horizon. A two-stage algorithm is developed to solve the problem: an initial solution is constructed, where some constraints may be violated, followed by iterative improvement using local search aimed at eliminating the violations. Using the proposed approach, an average deviation of 3.8% from the lower bound was achieved on open test cases. Tab. 3, illustr. 4, bibliogr. 27.

Keywords: bin packing problem, virtual machine, conflict, placement group.

References

1. **C. Lameter,** NUMA (Non-uniform memory access): An overview, *Queue* **11**, 40–51 (2013), DOI: 10.1145/2508834.2513149.
2. **P. Li, X. Wu, Y. Luo, L.-M. Wang, N. Yadav, M. Pepin,** and **J. Morgan,** NUMAware: Accelerate VM-to-VM I/O performance in NUMA servers for NFV applications, in *IEEE Conf. Network Function Virtualization and Software Defined Networks, Palo Alto, CA, USA, Nov. 7–9, 2016* (IEEE, Piscataway, 2016), URL: nfvsdn2016.ieee-nfvsdn.org/program/demonstrations/index.html (accessed: 20.08.2024).

English transl.: *Journal of Applied and Industrial Mathematics* **19** (1), 92–103 (2025), DOI: 10.1134/S1990478925010090.

3. Placement groups for your Amazon EC2 instances, in *Amazon Elastic Compute Cloud: User Guide* (Amazon Web Serv., Seattle, 2024), URL: `docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html` (accessed: 20.08.2024).

4. **A. V. Ratushnyi** and **Yu. A. Kochetov,** A column generation based heuristic for a temporal bin packing problem, in *Mathematical Optimization Theory and Operations Research* (Proc. 20th Int. Conf. MOTOR 2021, Irkutsk, Russia, July 5–10, 2021) (Springer, Cham, 2021), pp. 96–110 (Lect. Notes Comput. Sci., Vol. 12755), DOI: `10.1007/978-3-030-77876-7_7`.

5. **A. V. Ratushnyi,** A pattern-based heuristic for a temporal bin packing problem with conflicts, in *Mathematical Optimization Theory and Operations Research* (Proc. 22nd Int. Conf. MOTOR 2023, Yekaterinburg, Russia, July 2–8, 2023) (Springer, Cham, 2023), pp. 161–175 (Commun. Comput. Inf. Sci., Vol. 1881), DOI: `10.1007/978-3-031-43257-6_13`.

6. **P. A. Borisovsky, A. V. Eremeev, A. A. Panin,** and **M. A. Sakhno,** Temporal bin packing problems with placement constraints: MIP-models and complexity, in *Mathematical Optimization Theory and Operations Research* (Proc. 23rd Int. Conf. MOTOR 2024, Omsk, Russia, June 30 – July 6, 2024) (Springer, Cham, 2024), pp. 157–169 (Lect. Notes Comput. Sci., Vol. 14766), DOI: `10.1007/978-3-031-62792-7_11`.

7. **D. A. Grushin** and **N. N. Kuzyurin,** On effective scheduling in computing clusters, *Program. Comput. Softw.* **45**, 398–404 (2019), DOI: `doi.org/10.1134/S0361768819070077`.

8. **K. Y. Spencer, P. V. Tsvetkov,** and **J. J. Jarrell,** A greedy memetic algorithm for a multiobjective dynamic bin packing problem for storing cooling objects, *J. Heuristics* **25**, 1–45 (2019), DOI: `10.1007/s10732-018-9382-0`.

9. **Y. Li, X. Tang,** and **W. Cai,** Dynamic bin packing for on-demand cloud resource allocation, *IEEE Trans. Parallel Distrib. Syst.* **27** (1), 157–170 (2016), DOI: `10.1109/TPDS.2015.2393868`.

10. **N. Gupta, K. Gupta, D. Gupta,** [et al.], Enhanced virtualization-based dynamic bin-packing optimized energy management solution for heterogeneous clouds, *Math. Probl. Eng.* **2022** (1), ID 8734198 (2022), DOI: `10.1155/2022/8734198`.

11. **A. Beloglazov** and **R. Buyya,** Energy efficient allocation of virtual machines in cloud data centers, in *10th IEEE/ACM Int. Conf. Cluster, Cloud and Grid Computing, Melbourne, Australia, May 17–20, 2010* (IEEE, Piscataway, 2014), pp. 577–578, DOI: `10.1109/CCGRID.2010.45`.

12. **S. Berndt, K. Jansen,** and **K.-M. Klein,** Fully dynamic bin packing revisited, *Math. Program.* **179**, 109–155 (2020), DOI: `10.1007/s10107-018-1325-x`.

13. **A. Murhekar, D. Arbour, T. Mai,** and **A. Rao,** Brief announcement: Dynamic vector bin packing for online resource allocation in the cloud, in *Proc. 35th ACM Symp. Parallelism in Algorithms and Architectures, Orlando, FL, USA, June 17–19, 2023* (ACM, New York, 2023), pp. 307–310, DOI: `10.1145/3558481.3591314`.

14. **K. Daudjee, S. Kamali,** and **A. López-Ortiz,** On the online fault-tolerant server consolidation problem, in *Proc. 26th ACM Symp. Parallelism in Algorithms and Architectures, Prague, Czech Rep., June 23–25, 2014* (ACM, New York, 2014), pp. 12–21, DOI: 10.1145/2612669.2612686.
15. **C. Hubbs, H. Perez, O. Sarwar, N. Sahinidis, I. Grossmann,** and **J. Wassick,** OR-Gym: A reinforcement learning library for operations research problem (Cornell Univ., Ithaca, NY, 2020) (Cornell Univ. Libr. e-Print Archive; arXiv:2008.06319), DOI: 10.48550/arXiv.2008.06319.
16. **C. Wan, T. Li,** and **J. Wang,** RLOR: A flexible framework of deep reinforcement learning for operation research (Cornell Univ., Ithaca, NY, 2023) (Cornell Univ. Libr. e-Print Archive; arXiv:2303.13117), DOI: 10.48550/arXiv.2303.13117.
17. **B. Balaji, J. Bell-Masterson, E. Bilgin,** [et al.], ORL: Reinforcement learning benchmarks for online stochastic optimization problems (Cornell Univ., Ithaca, NY, 2019) (Cornell Univ. Libr. e-Print Archive; arXiv:1911.10641), DOI: 10.48550/arXiv.1911.10641.
18. **T. Yang, F. Luo, J. Fuentes, W. Ding,** and **C. Gu,** A flexible reinforced bin packing framework with automatic slack selection, *Math. Probl. Eng.* **2021**, ID 6653586 (2021), DOI: 10.1155/2021/6653586.
19. **F. Mao, E. Blanco, M. Fu,** [et al], Small boxes big data: A deep learning approach to optimize variable sized bin packing, in *Proc. IEEE 3rd Int. Conf. Big Data Computing Service and Applications, San Francisco, CA, USA, Apr. 6–9, 2017* (IEEE, Piscataway, 2017), pp. 80–89, DOI: 10.1109/BigDataService.2017.18.
20. **H. A. Nomer, K. A. Alnowibet, A. Elsayed,** and **A. W. Mohamed,** Neural knapsack: A neural network based solver for the knapsack problem, *IEEE Access* **8**, 224200–224210 (2020), DOI: 10.1109/ACCESS.2020.3044005.
21. **V. V. Toporkov, D. M. Yemelyanov,** and **A. N. Bulkhak,** Machine learning-based online scheduling in distributed computing, in *Parallel Processing and Applied Mathematics* (Rev. Sel. Pap. 14th Int. Conf. PPAM 2022, Gdansk, Poland, Sept. 11–14, 2022) (Springer, Pt. II. Cham, 2023), pp. 248–259 (Lect. Notes Comput. Sci., Vol. 13827), DOI: 10.1007/978-3-031-30445-3_21.
22. **A. M. Turnaev** and **A. A. Panin,** Stochastic greedy algorithms for a temporal bin packing problem with placement groups, in *Mathematical Optimization Theory and Operations Research* (Proc. 23rd Int. Conf. MOTOR 2024, Omsk, Russia, June 30 – July 6, 2024) (Springer, Cham, 2024), pp. 199–211 (Lect. Notes Comput. Sci., Vol. 14766), DOI: 10.1007/978-3-031-62792-7_14.
23. **D. S. Johnson,** Fast algorithms for bin packing, *J. Comput. Syst. Sci.* **8**, 272–314 (1974), DOI: 10.1016/S0022-0000(74)80026-7.
24. Amazon EC2 instance types, in *Amazon Elastic Compute Cloud Documentation* (Amazon Web Serv., Seattle, 2024), URL: docs.aws.amazon.com/ec2/latest/instancetypes/instance-types.html (accessed: 20.08.2024).
25. **C. Munien** and **A. Ezugwu,** Metaheuristic algorithms for one-dimensional bin-packing problems: A survey of recent advances and applications, *J. Intell. Syst.* **30**, 636–663 (2021), DOI: 10.1515/jisys-2020-0117.

26. S. Kosub, A note on the triangle inequality for the Jaccard distance, *Pattern Recognit. Lett.* **120**, 36–38 (2019), DOI: 10.1016/j.patrec.2018.12.007.

27. Temporal bin packing problem with placement groups, in *Discrete Location Problems. Benchmark Library* (Sobolev Inst. Math., Novosibirsk, 2024), URL: old.math.nsc.ru/AP/benchmarks/Temporal%20Bin%20Packing/binpack.html (accessed: 20.08.2024).

Aleksey V. Ratushnyi
Yury A. Kochetov

Received September 11, 2024
Revised September 17, 2024
Accepted September 22, 2024