ТРУДЫ ИНСТИТУТА МАТЕМАТИКИ И МЕХАНИКИ УрО РАН

Том 30 № 1 2024

УДК 517.977

ВЕРХНИЕ И НИЖНИЕ ОЦЕНКИ ОПТИМУМА ДЛЯ ЗАДАЧИ ДИНАМИЧЕСКОЙ УПАКОВКИ В КОНТЕЙНЕРЫ[1]

Ю. А. Кочетов, А. В. Ратушный

Рассматривается новая задача динамической упаковки в контейнеры, возникающая в облачных вычислениях. Имеется конечное множество виртуальных машин. Для каждой машины задано временное окно для обслуживания и два размера: число ядер и объем оперативной памяти. Все машины делятся на два типа: большие и малые. Каждый сервер состоит их двух одинаковых узлов. Малые машины помещаются полностью на один из них. Большие машины делятся пополам и помещаются на оба узла. Требуется разместить все машины в минимальное число серверов. Для решения задачи разработана эвристика, основанная на методе генерации столбцов. Для получения нижних оценок оптимума выбираются моменты времени с большой суммарной нагрузкой и для них решается статическая задача. Для построения верхних оценок решение статической задачи расширяется на все моменты времени известным алгоритмом "Первый подходящий" (First Fit). Приводятся теоретические утверждения о точности оценок в худшем случае. Вычислительные эксперименты для реальных тестовых примеров указывают на небольшой разрыв между границами, не более 0.9% для недельного интервала, 50000 машин и среднем времени расчетов 26 с на персональном компьютере. В работе удалось улучшить некоторые известные результаты на открытых примерах.

Ключевые слова: задача о рюкзаке, метод генерации столбцов, NUMA-архитектура, виртуальная машина.

Yu. A. Kochetov, A. V. Ratushnyi. Upper and lower bounds for the optimum in a temporal bin packing problem.

A new temporal bin packing problem originating from cloud computing is introduced. There is a finite set of virtual machines. For each machine, the time window for processing, the number of cores, and the RAM size are known. Each machine can be of one of two types: large or small. Each server consists of two identical nodes. Each small machine is placed completely on one of the nodes, and each large machine is divided into two identical parts placed on both nodes of a server. The goal is to pack all the machines into the minimum number of servers. For this problem, we develop a heuristic based on the column generation method. To get lower bounds of the optimum, we choose the times with the greatest total load and solve the static problem for them. To derive upper bounds, we extend the solution of the static problem to all times using the well-known First Fit algorithm. Computational experiments for real test instances indicate a small gap between the bounds, which is at most 0.9% for a week horizon, 50 000 machines, and 26 s average running time on a personal computer. We manage to improve some well-known results for open instances.

Keywords: knapsack problem, column generation method, NUMA architecture, virtual machine.

MSC: 80M50, 90C27

DOI: 10.21538/0134-4889-2024-30-1-109-127

Введение

Задача об упаковке в контейнеры [1] является одной из классических задач комбинаторной оптимизации и была исследована в разных постановках [2]. Ее приложения возникают в таких областях, как логистика, роботизация, облачные вычисления и др. [3]. Популярными вариациями являются онлайн-задача об упаковке в контейнеры и векторная упаковка.

Актуальность задачи динамической упаковки в контейнеры породила много различных формулировок и исследований. Такая задача, в сравнения с классической задачей упаковки в контейнеры, дополнительно включает в себя временной горизонт и учитывает интервалы времени существования предметов. В статье [4] исследуется подход, основанный на заранее

[1]Работа выполнена в рамках государственного задания ИМ СО РАН (FWNF 2022-0019).

вычисленных шаблонах упаковки, которые применяются для построения быстрого онлайн-алгоритма. В [5] исследуется сразу несколько эвристик, а также описывается способ уменьшить количество моментов времени. Авторы работы [6] рассматривают задачу с несколькими целевыми функциями, мотивируя данный подход тем, что для уменьшения затрат важно не только количество серверов, но и другие факторы. (Например, частое включение/выключение серверов (контейнеров) может потребовать большое количество дополнительной энергии). В [7] приводится математическая модель для динамической постановки, а также несколько способов избавиться от симметрии. В работах [8; 9] описываются динамическая задача о рюкзаке, некоторые модели и различные подходы для ее решения. Достаточно подробная классификация задач распределения виртуальных машин (динамической упаковки предметов в контейнеры) и различных техник решения описана в [10].

Другим интересным направлением исследований в этой области нам видится развитие идей из работы [11] по представлению задач упаковки в виде задач о потоке минимальной стоимости. К настоящему времени эти идеи получили свое развитие в целом ряде исследований [12–16]. Основным недостатком новых формулировок является огромная размерность получаемых задач. Поэтому усилия были направлены преимущественно на сокращение размерности, теоретические исследования и алгоритмические улучшения [17–19]. Именно на этом пути получены наиболее мощные точные методы решения задач упаковки, в том числе и в динамической постановке с одним параметром, задающим вместимость контейнера [20–24].

В нашем исследовании рассматривается новая задача в динамической постановке для размещения виртуальных машин (ВМ) различной конфигурации и временными окнами на одинаковых серверах со специальной NUMA-архитектурой [25]. Информация о каждой машине предполагается известной заранее: время создания и удаления, необходимое количество RAM и CPU. Однако объем требуемых ресурсов определяется типом машины; предполагается, что число различных типов значительно меньше числа машин, что позволяет отнести эту задачу к более узкому классу задач раскроя [13]. Для каждой машины необходимо так указать сервер, на котором она будет обслуживаться, чтобы минимизировать общее число используемых серверов за рассматриваемый промежуток времени. Задача NP-трудна в сильном смысле, так как является обобщением широко известной задачи об упаковке в контейнеры. Представлены нижние и верхние оценки оптимума, основанные на методе генерации столбцов [26, с. 455] (см. также [27]). Для получения нижних оценок выбираются моменты времени с большой суммарной нагрузкой, и для них решается статическая задача. Лучшая из таких нижних оценок предъявляется как нижняя оценка для всей задачи. Для построения верхних оценок решение статической задачи расширяется на все моменты времени известным алгоритмом “Первый подходящий” (First Fit) [28] несколькими способами. Однако, в отличие от работы [4], шаблоны, полученные для некоторого момента времени, не сохраняются ни одним из предложенных способов расширения. Вычислительные эксперименты на реальных тестовых примерах крупного интернет-провайдера указывают на небольшой разрыв между границами, не более 0,9% для недельного интервала, 50000 машин и при среднем времени расчетов 26 с на персональном компьютере. Данная работа является расширенной версией статьи [29], представленной в трудах международной конференции “MOTOR 2021” [30]. В частности, в нее включена расширенная серия вычислительных экспериментов, а также неопубликованные утверждения 1, 2 и теоремы 1, 2. Помимо этого, схожее исследование имелось в трудах “MOTOR 2023” [31], но здесь задача отличалась наличием конфликтов и подходом к вычислению верхней оценки.

Статья организована следующим образом. В разд. 1 приводятся детальное описание задачи, архитектура и компоненты сервера, типы виртуальных машин, математическая модель в терминах целочисленного линейного программирования и подчеркивается вычислительная сложность задачи. В разд. 2 рассматриваются метод генерации столбцов для получения нижних оценок оптимума, модель двумерной упаковки в контейнеры для одного из наиболее нагруженных моментов времени и специализированная задача о рюкзаке для порождения новых шаблонов в методе генерации столбцов. Алгоритм получения верхних оценок оптимума (допу-

стимых решений задачи) описан в разд. 3. Раздел 4 содержит описание используемых данных для численных экспериментов и результаты сравнения получаемых верхних и нижних оценок, а также время работы алгоритмов. В заключении кратко подводятся итоги исследования.

1. Постановка задачи

Обозначим через S конечное множество одинаковых серверов, каждый из которых имеет объем C_1 оперативной памяти (RAM) и C_2 ядер (CPU). Серверы имеют NUMA-архитектуру и разделены на два идентичных NUMA-узла (рис. 1). Каждый узел обладает половиной памяти и ядер. Каждая виртуальная машина должна располагаться на сервере в течение своего временного окна и характеризоваться своим типом. Тип машины определяет требуемые память и число ядер. Обозначим через L множество всех типов машин и выделим в нем подмножество L^l так называемых больших машин. При размещении на сервере такие машины располагаются на двух узлах и занимают на каждом из них половину требуемых им ресурсов (ядер и памяти). Дополнение этого множества обозначим через L^s, $L^s = L \setminus L^l$. Данные машины будем называть малыми, они целиком размещаются на одном из узлов сервера. Будем предполагать, что миграции машин невозможны как между серверами, так и внутри его между узлами. Как только машина разместилась на сервере, она не может быть перемещена на другой сервер или на другой узел.

Пусть множество M задает набор всех машин, которые надо разместить на серверах. Так как каждая машина имеет свой тип, то множество M разбивается на два непересекающихся подмножества: малые виртуальные M^s и большие M^l машины. Для каждой машины $m \in M$ заданы временное окно $[\alpha_m, \omega_m)$ для размещения на сервере и ресурсы d_{1m}, d_{2m}, требуемые в каждый момент времени из этого окна. Время создания α_m и время удаления ω_m машины являются уникальными для каждой машины. Предполагается, что $|M| \gg |L|$.

Введем следующие обозначения:

N — множество NUMA-узлов сервера ($N = \{1, 2\}$);

R — множество типов ресурсов ($R = \{1, 2\}$);

$\mathcal{T}$ — множество моментов времени, когда создавалась хотя бы одна машина;

M_t^l и M_t^s — множества больших и малых машин, которые существуют в момент $t \in \mathcal{T}$;

$M_t = M_t^l \cup M_t^s$ — множество всех машин в момент времени $t \in \mathcal{T}$.

Введем следующие 0-1 переменные:

$x_{msn} = 1$, если и только если малая машина m расположена на NUMA-узле n сервера s;

$y_{ms} = 1$, если и только если большая машина m расположена на сервере s;

$z_s = 1$, если и только если сервер s активен хотя бы в один момент времени.

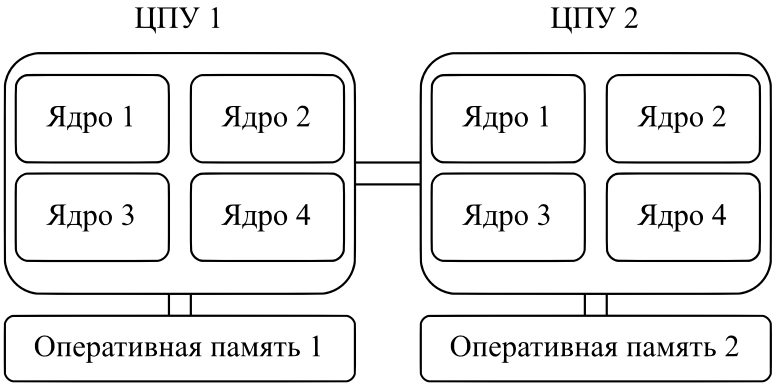

Рис. 1. NUMA-архитектура сервера.

Математическая модель имеет следующий вид:

$$\min_{x_{msn},y_{ms},z_s} \sum_{s\in S} z_s \tag{1.1}$$

$$\sum_{s\in S}\sum_{n\in N} x_{msn} = 1, \quad m \in M^s, \tag{1.2}$$

$$\sum_{s\in S} y_{ms} = 1, \quad m \in M^l, \tag{1.3}$$

$$\sum_{m\in M_t^s} d_{rm}x_{msn} + \frac{1}{2}\sum_{m\in M_t^l} d_{rm}y_{ms} \le \frac{C_r}{2}z_s, \quad t \in \mathcal{T}, \quad s \in S, \quad n \in N, \quad r \in R, \tag{1.4}$$

$$x_{msn}, y_{ms}, z_s \in \{0,1\}. \tag{1.5}$$

Целевая функция (1.1) определяет число используемых серверов. Ограничения (1.2), (1.3) гарантируют, что каждая машина будет размещена на одном сервере. Неравенства (1.4) ограничивают количество ресурсов на каждом сервере и делают сервер активным, если на нем располагалась хотя бы одна машина хотя бы в один момент времени.

Несмотря на небольшую мощность множеств N, R и L, решение задачи с помощью коммерческого решателя Gurobi занимает значительное время. Например, решение задачи с 2000 виртуальных машин требует около 5 мин. Для поиска оптимального решения задачи с 50000 виртуальных машин (см. разд. 4) необходимо более 5 ч. Следующее утверждение отчасти объясняет столь быстрый рост вычислительных затрат при точном решении задачи.

Теорема 1. *Классическая задача об упаковке в контейнеры является частным случаем задачи* (1.1)–(1.5).

Доказательство. Пусть все виртуальные машины будут большими и занимают на сервере весь временной интервал. Тогда NUMA-архитектура сервера теряет актуальность и решение о распределении машин в первый момент времени задает решение всей задачи. Нетрудно заметить, что это и есть классическая задача об упаковке в контейнеры со всеми ее свойствами, касающимися вычислительной сложности и неаппроксимируемости [26]. □

2. Нижние оценки

Поиск нижних оценок оптимума будем проводить методом генерации столбцов для статической задачи. Выберем некоторый момент времени T, например момент максимального спроса на один из ресурсов. Нижняя оценка для числа серверов в этот момент времени будет, очевидно, нижней оценкой и для всей задачи. Соберем все машины, которые должны быть на серверах в этот момент. Заметим, что для статической задачи все машины одного типа можно считать идентичными; это значительно сокращает число переменных. Приведем точную постановку статической задачи упаковки виртуальных машин на серверы (задача упаковки в контейнеры с NUMA-архитектурой).

Рассмотрим множество J всех возможных вариантов (шаблонов) упаковки одного сервера. Пусть подмножество $J' \subset J$ задает часть этого множества, достаточное для получения допустимого решения задачи. Это подмножество можно получить любой эвристикой для упаковки виртуальных машин. Обозначим через a_{ij} количество виртуальных машин типа i в шаблоне $j \in J$. Таким образом, шаблон j можно ассоциировать со следующим вектором: $(a_{1j}, \ldots, a_{|L|j})$ малой размерности. Пусть величина n_i — это общее число виртуальных машин типа i, которые должны быть размещены на серверах в выбранный момент времени. Целочисленная неотрицательная переменная x_j будет задавать число серверов, упакованных по шаблону j.

С использованием этих обозначений статическая задача принимает следующий вид:

$$\min \sum_{j \in J} x_j \tag{2.1}$$

$$\sum_{j \in J} a_{ij} x_j \geq n_i, \quad i \in L, \tag{2.2}$$

$$x_j \geq 0, \text{ целые}, j \in J. \tag{2.3}$$

Целевая функция (2.1) задает количество серверов. Неравенства (2.2) гарантируют, что все виртуальные машины будут упакованы. Заметим, что ограничения исходной задачи, связанные с архитектурой серверов, разделением машин на большие и малые, а также ресурсные ограничения на ядра и память учитываются при формировании шаблонов. Неравенства (2.3) ограничивают область изменения переменных.

Отбросим условие целочисленности переменных и решим точно полученную задачу линейного программирования с экспоненциальным числом переменных. Оптимальное значение в этой задаче будет требуемой нижней оценкой. Следуя методу генерации столбцов, рассмотрим задачу для подмножества J' и выпишем для нее двойственную модель с неотрицательными переменными $\lambda_i, i \in L$:

$$\max \sum_{i \in L} n_i \lambda_i \tag{2.4}$$

$$\sum_{i \in L} a_{ij} \lambda_i \leq 1, \quad j \in J', \tag{2.5}$$

$$\lambda_i \geq 0, \quad i \in L. \tag{2.6}$$

Пусть λ_i^* — оптимальные значения двойственных переменных и x_j^* — оптимальные значения переменных соответствующей прямой задачи. Легко проверить, что при выполнении неравенства

$$\sum_{i \in L} a_{ij} \lambda_i^* \leq 1 \tag{2.7}$$

для всех шаблонов $j \in J$ получаем оптимальное решение

$$x_j = \begin{cases} x_j^*, & j \in J', \\ 0, & j \in J \backslash J' \end{cases}$$

всей задачи (2.1)–(2.3) для исходного множества шаблонов J. В этом случае величина $H = \sum_{j \in J'} x_j^* = \sum_{i \in L} n_i \lambda_i^*$ и ее округление вверх являются нижней оценкой.

Проверять неравенство (2.7) для всех столбцов $j \in J$ достаточно сложно в силу экспоненциальной мощности этого множества. Поэтому будем искать такой шаблон, столбец матрицы (a_{ij}), который доставляет максимальное значение левой части неравенства при известных оптимальных значениях двойственных переменных. Рассмотрим задачу упаковки одного сервера с NUMA-узлами. Пусть новые переменные y_i определяют количество виртуальных машин типа $i \in L$ на сервере. Переменные z_i^n задают количество малых машин типа i на NUMA-узле $n \in N$. Тогда математическая модель может быть записана следующим образом:

$$\max \alpha = \sum_{i \in L} \lambda_i^* y_i \tag{2.8}$$

$$y_i \leq n_i, \quad i \in L, \tag{2.9}$$

$$\sum_{n \in N} z_i^n = y_i, \quad i \in L^s, \tag{2.10}$$

$$\sum_{i \in L^s} d_{ri} z_i^n + \frac{1}{2} \sum_{i \in L^l} d_{ri} y_i \le \frac{C_r}{2}, \quad r \in R, \quad n \in N, \tag{2.11}$$

$$y_i \ge 0, \text{ целые, } i \in L, \tag{2.12}$$

$$z_i^n \ge 0, \text{ целые, } i \in L^s, \ n \in N. \tag{2.13}$$

Целевая функция (2.8) задает ценность упаковки. В связи с этим данную задачу часто называют в англоязычной литературе *Pricing problem*. Неравенства (2.9) задают верхнюю оценку на число виртуальных машин каждого типа. Равенства (2.10) гарантируют, что каждая малая машина расположена на одном узле сервера. Неравенства (2.11) ограничивают доступность ресурсов на каждом узле, учитывая различие малых и больших машин. Условия (2.12), (2.13) определяют типы переменных задачи.

Пусть α^* — значение целевой функции в оптимальном решении задачи. Если $\alpha^* \le 1$, то неравенство (2.7) выполняется для всех шаблонов. Если $\alpha^* > 1$, то получаем новый шаблон $(y_1^*, \ldots, y_{|L|}^*)$ и включаем его в множество J'. Общая схема поиска нижней оценки представлена ниже.

Процедура вычисления нижней оценки.

1: Выбираем начальное множество J'.

2: Решаем задачу (2.4)–(2.6) и находим вектор λ^*.

3: Решаем задачу (2.8)–(2.13) с вектором λ^*, находим α^* и шаблон y^*.

4: Если $\alpha^* \le 1$, то полагаем $H(T) = \lceil \sum_{i \in L} n_i \lambda_i^* \rceil$, Stop.

5: Добавляем шаблон y^* в множество J' и возвращаемся на шаг 2.

Для инициализации множества J' применима любая эвристика, строящая допустимое решение задачи упаковки в контейнеры [28]. Неудачный выбор может привести к большому числу возвращений на шаг 2 и неоправданному росту времени счета. Ниже предлагается общая схема алгоритма инициализации этого множества, основанного на модели (2.8)–(2.13).

Построение начального множества J'.

1: Полагаем $J' = \varnothing \ \forall i \in L$.

2: Решаем задачу (2.8)–(2.13) с набором виртуальных машин, активных в момент времени T.

3: Добавляем новый шаблон y^* в множество J'.

4: Сокращаем число неупакованных машин $n_i := n_i - y_i \ \forall i \in L$.

5: Если $\exists n_i \ne 0$, то возвращаемся на шаг 2, иначе Stop.

Нижняя оценка вычисляется для статической задачи, значение которой зависит от выбранного момента времени. Найти наилучший момент — непростая задача (рис. 2). Момент времени с наибольшей нагрузкой по одному из ресурсов не обязан давать наибольшее значение нижней оценки. Можно рассмотреть все моменты старта машин и выбрать из полученных нижних оценок наибольшую. Однако такая процедура представляется слишком трудоемкой. Более того, полученная таким способом нижняя оценка может как минимум в полтора раза отличаться от оптимума [5, с. 336]. Разумным выходом в такой ситуации представляется вычисление нижней оценки для нескольких достаточно загруженных моментов времени и предъявление наибольшей из них.

Отметим, что в работе [5] предложен другой способ получения нижних оценок, основанный на методе генерации столбцов для исходной, а не статической задачи. Каждый контейнер имеет один ресурс и не имеет NUMA-архитектуры. Этот подход дает возможность получать точное решение линейной релаксации с числом предметов до 1000 и приближенные решения с числом

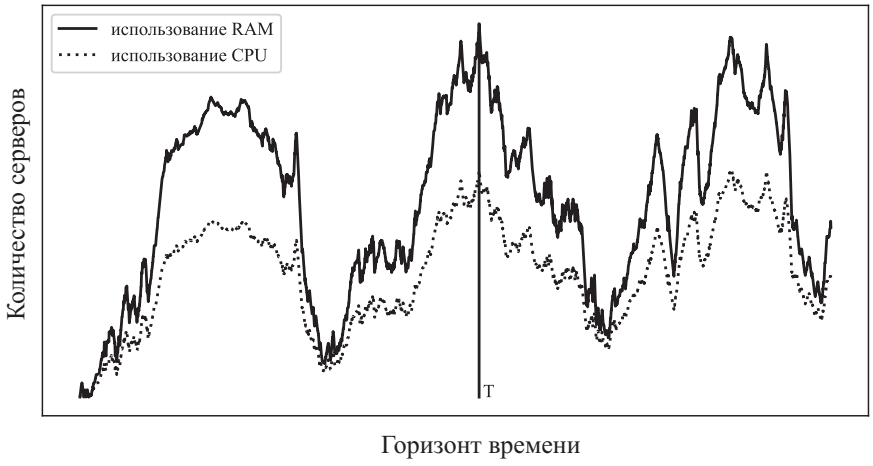

Рис. 2. Общая нагрузка на серверы

предметов до 7000. Адаптация такого подхода к данной задаче представляет несомненный интерес. Тем не менее в настоящей работе делается акцент в первую очередь на примеры еще большей размерности, что значительно усложняет вычисления. Кроме того, новый способ, и это кажется более важным, лишает нас возможности сократить размерность задачи, объявляя предметы одного типа идентичными, несмотря на разные временные окна для их выполнения.

Помимо упомянутых нижних оценок, основанных на решении задач линейного программирования экспоненциальной размерности, можно рассмотреть следующую полиномиально вычислимую нижнюю оценку:

$$Slb = \max_{t \in \mathcal{T}} \max \left\{ \left\lceil \frac{\sum_{m \in M(t)} d_{m1}}{C1}, \frac{\sum_{m \in M(t)} d_{m2}}{C2} \right\rceil \right\}.$$

Она получается по аналогии с оценкой из работы [5] игнорированием NUMA-архитектуры. К сожалению, простота получения такой примитивной оценки сказывается на ее точности. Примитивная нижняя оценка может почти в два раза отличаться от оптимума. Пусть F^* — количество серверов в оптимальном решении задачи (1.1)–(1.5). Тогда справедливо следующее утверждение.

Утверждение 1. *Для любого числа $M > 1$ больших машин и любого количества моментов времени в множестве $\mathcal{T}$ существует индивидуальная задача, на которой выполняется неравенство $2Slb \le F^* + 1$.*

Доказательство. Рассмотрим M больших машин, существующих весь рассматриваемый период и требующих один ресурс, но чуть больше половины: $d_{1m} = \dfrac{C_1}{2}(1 + \dfrac{1}{M})$, $d_{2m} = 0$. Тогда $Slb = \dfrac{M}{2} + 0.5$, но $F^* = M$. □

Таким образом, примитивная нижняя оценка Slb может давать достаточно плохие результаты, что демонстрируется в разд. 4. В то же время Slb совпадает с релаксацией линейного программирования для (1.1)–(1.5) (см. теорему 2), тогда как решение целочисленной задачи занимает несколько часов даже для небольшого числа виртуальных машин. Это отчасти объясняет невысокую эффективность коммерческих решателей типа GUROBI для задач упаковки и стимулирует исследование неполиномиальных формулировок и численных методов на их основе. Как мы увидим ниже, для некоторых примеров некомпактные формулировки позволяют не только найти точное решение задачи, но и доказать его оптимальность без ветвления и дополнительных отсечений. Если же нижние оценки не совпадают с верхними, то зазор как относительный, так и абсолютный, в основном оказывается ничтожно малым.

Наряду с оценкой Slb можно рассмотреть нижнюю оценку С. Мартелло и П. Тот [32]. Для ее подсчета мы провели адаптацию, связанную со взятием максимума по времени и по ресурсам. Вычислительные эксперименты показывают, что такая оценка дает более точные результаты в некоторых случаях, но все еще является недостаточно точной.

Теорема 2. *Оценка Slb совпадает с релаксацией линейного программирования для задачи* (1.1)–(1.5) *при* $N = \{1, 2\}$ *и любых множеств* R *и* $\mathcal{T}$.

Доказательство. Так как переменные x_{msn} являются действительными, то малые машины могут быть упакованы на оба NUMA-узла сервера аналогично большим. Тем самым мы можем исключить NUMA-архитектуру и ограничения (1.2) из рассмотрения, а переменные x_{msn} заменить на переменные y_{ms}. Тогда следующее решение является одним из оптимальных: $z_1 = \ldots = z_{\lfloor Slb \rfloor} = 1$, $z_{\lfloor Slb \rfloor + 1} = \{Slb\}$, $y_{ms} = z_s / Slb$. □

3. Верхние оценки

Для построения верхних оценок сначала получим допустимое решение статической задачи для выбранного момента времени, а затем достроим это решение до допустимого решения всей исходной задачи. Вернемся к процедуре вычисления нижней оценки и пусть на шаге 4 сработал критерий остановки с некоторым множеством шаблонов J'. Подставим данное множество в задачу (2.1)–(2.3) вместо исходного множества J и найдем точное решение этой упрощенной задачи. Такой эвристический прием часто дает минимальный зазор между границами (см. например, [33]).

Следующим шагом необходимо достроить указанное решение статической задачи до решения исходной задачи. Разделим множество M на три непересекающихся подмножества M_1, M_2 и M_3: $M_1 = \{m \in M | \alpha_m \leq T < \omega_m\}$, $M_2 = \{m \in M | \alpha_m > T\}$, $M_3 = \{m \in M | T \geq \omega_m\}$. Для множества M_1 решаем статическую задачу, как это было отмечено выше, и распределяем машины из этого множества по выбранным шаблонам. В статической задаче все машины одного типа идентичны, а при возвращении в исходную задачу они должны стать уникальными, со своим временем старта и окончания. Для каждого типа эта процедура проводится независимо от других типов и в произвольном порядке для машин одного типа. Свобода в выборе этого порядка дает простой способ рандомизации алгоритма, что открывает дополнительные возможности для повышения точности. В частности, используя случайный порядок машин, можно породить популяцию допустимых решений задачи и применить идеи генетических алгоритмов для получения более точных верхних оценок оптимума.

Для множеств M_2 и M_3 применяется широко известная эвристика "Первый подходящий" (First Fit) [26]. В результате получается допустимое решение исходной задачи. Для множества M_2 она выглядит следующим образом.

Процедура "Первый подходящий".

1: Сортируем множество M_2 по неубыванию времени старта машин.

2: Берем первую еще нераспределенную машину m.

3: На каждом сервере удаляем все уже завершенные машины $\tilde{m}$: $\omega_{\tilde{m}} \leq \alpha_m$.

4: Ищем сервер, имеющий достаточно ресурсов для машины m.

Если такого нет, то создаем новый сервер. Кладем машину m на этот сервер.

5: Если остались нераспределенные машины, то возвращаемся на шаг 2, иначе Stop.

Для множества M_3 процедура имеет некоторые отличия. Во-первых, множество M_3 сортируется в порядке невозрастания времени завершения работ, чтобы последовательно встраивать в решение ранее стартовавшие машины. Во-вторых, на шаге 3 удаляются еще не стартовавшие машины, $\alpha_{\tilde{m}} \geq \omega_m$.

При размещении малых машин на сервере оба NUMA-узла могут иметь достаточное количество ресурсов. Чтобы выбрать узел, для каждого из них подсчитывается суммарная нагрузка и выбирается тот, на котором больше свободных ресурсов. Такая стратегия создает лучшие условия для дальнейшего размещения больших машин, так как им требуются ресурсы на каждом узле.

Мы уже отмечали в предыдущем разделе, что выбор момента времени для статической задачи может сильно влиять на нижние и, соответственно, на верхние оценки. Момент с максимальным спросом, как показывает следующее утверждение, может быть не лучшим выбором.

Утверждение 2. *В момент с наибольшей нагрузкой разница между оптимальным решением статической задачи и решением исходной задачи может быть сколь угодно большой.*

Доказательство. Рассмотрим два момента времени с непересекающимися множествами больших машин M_1^l и M_2^l. Каждая машина из первого множества требует $0.5C_1 + \varepsilon$ ресурса. Каждая машина из второго множества требует весь ресурс C_1. Мощность первого множества равна $2k$, мощность второго — $k + 1$. Второй ресурс, малые машины и другие моменты времени можно игнорировать. Легко заметить, что для первого множества требуется $2k$ серверов, а для второго — $k + 1$, но нагрузка в первый момент времени равна $kC_1 + \varepsilon 2k$ и $kC_1 + C_1$ — во второй момент времени. Таким образом, абсолютный разрыв $(k - 1)$ может быть сколь угодно большим, а относительное уклонение — не меньше двух. □

Заметим, что в приведенном примере нижняя оценка статической задачи совпадает с оптимальным решением, если считать ее для обоих моментов времени и выбирать наибольшее значение. В связи с этим ниже предлагается эвристика, позволяющая подбирать несколько моментов времени, не просматривая все из них. Идея этого подхода состоит в том, чтобы начинать с момента наибольшей нагрузки, строить допустимое решение задачи, а затем искать в этом решении моменты времени, требующие дополнительных серверов, и брать эти моменты времени для вычисления новых значений нижней оценки.

Процедура генерации моментов времени.

1: Выбираем момент T наибольшей нагрузки по одному из ресурсов, решаем статическую задачу и строим допустимое решение алгоритмом “Первый подходящий”.

2: Находим момент времени T' с наибольшим количеством активных серверов в полученном решении.

3: Если $T \neq T'$, то полагаем $T := T'$, решаем статическую задачу для момента T и строим новое допустимое решение.

4: Если сработал критерий остановки, то Stop, иначе возвращаемся на шаг 2.

В качестве критерия остановки могут выступать время счета, число возвращений на шаг 2 или факт зацикливания алгоритма при попадании в уже просмотренный момент времени. Если моментов времени с максимальным числом серверов несколько, то на шаге 3 можно проверять все из них в надежде получить лучшую верхнюю или нижнюю оценки.

Алгоритм “Первый подходящий” является быстрой и экономичной по памяти эвристикой, так как необходимо хранить только один момент времени, но он ориентируется исключительно на временные окна машин и игнорирует затраты ресурсов. Этот недостаток можно исправить, сортируя машины по другим критериям, таким как:

1: Невозрастание $\sum_{r \in R} d_{mr}$.

2: Невозрастание $\sum_{r \in R} d_{mr}(\omega_m - \alpha_m)$.

3: Неубывание $\sum_{r \in R} d_{mr}\alpha_m$.

4: Невозрастание $(\omega_m - \alpha_m) \prod_{r \in R} d_{mr}$.

Указать наилучший порядок не удается. На каких-то примерах лучше работает один, на иных — другой. Поэтому логично использовать каждый из них и выбирать лучшее из построенных решений. Заметим, что при таком подходе пропадает необходимость выделения множеств M_2 и M_3. Достаточно построить решение статической задачи в некоторый момент времени, а затем достроить это решение алгоритмом "Первый подходящий", используя каждый из критериев. Такой алгоритм, использующий несколько моментов времени для статической задачи и несколько критериев сортировки машин, будем называть итерационным алгоритмом.

4. Численные эксперименты

Для проведения экспериментов все алгоритмы были реализованы на языке Java. Вычисления осуществлялись на персональном компьютере с процессором i7-9700 и 8Gb RAM. Для расчетов по математическим моделям использовалось коммерческое программное обеспечение, Gurobi 9.1 со стандартными настройками [34]. Исследовались тестовые примеры с реальными типами машин (https://aws.amazon.com/ec2/instance-types/); каждый сервер имеет 276 ядер CPU и 324 Gb RAM. Рассматривалось 14 типов машин. Машина считается большой, если у нее не меньше 24 ядер. Общее число машин достигало $5e + 4$.

Кроме того, исследовались общедоступные примеры из работы [5] без NUMA-архитектуры и с одним ресурсом. Эти примеры были слегка модифицированы. Все машины считались большими, а второй ресурс копировал первый, чтобы можно было проводить сравнение с результатами этой работы и другим подходом к построению нижних и верхних оценок.

До проведения численных экспериментов на каждом примере применялась процедура редукции данных для сокращения рассматриваемых моментов времени [5]. Среди всех моментов времени выбираются только недоминируемые. Обозначим через M_T множество всех виртуальных машин, существующих в момент T. Тогда момент времени T называется недоминируемым, если $\nexists\, T' : M_T \subseteq M'_T$. При таком подходе оптимальные решения не меняются, но число моментов времени в среднем сокращается в 20 раз, а время работы алгоритмов — в 3.5 раза. Результаты расчетов приводятся в табл. 1–7, каждая из которых включает следующие параметры:

- Slb — примитивная нижняя оценка;
- CGlb — нижняя оценка статической задачи;
  полученная в моменты максимальной нагрузки;
- MTlb — нижняя оценка С. Мартелло и П. Тот [32];
- CGub — верхняя оценка алгоритма "Первый подходящий";
- ICGub — верхняя оценка итерационного алгоритма;
- CGub time — время получения оценки CGub;
- ICGub time — время работы итерационного алгоритма;
- CGub gap = 100%(CGub-CGlb)/CGlb;
- ICGub gap = 100%(ICGub-CGlb)/CGlb.

Согласно утверждению 1 нижняя оценка в момент наибольшей нагрузки может сильно отличаться от оптимума и оценок в другие моменты времени. Однако в ходе экспериментов такое отличие наблюдалось крайне редко. В связи с этим в таблицах приводится только значение CGlb. Влияние на верхнюю оценку выбора нескольких моментов времени можно считать существенным, о чем можно судить при сравнении величин CGub и ICGub.

В табл. 1 приведены результаты экспериментов для примеров, в которых около 70% виртуальных машин необходимо обслуживать весь рассматриваемый период. Это достаточно простые примеры. Однако время работы итерационного алгоритма ICGub заметно превышает время CGub, и это сказывается на точности получаемых решений. Среднее отклонение ICGub gap

Таблица 1

Примеры с машинами большой длительности

	Slb	MTlb	CGlb	CGub	ICGub	CGub time	ICGub time	CGub gap	ICGub gap
1	820	820	820	825	820	6.0	29.2	0.61%	0.00%
2	847	847	847	849	847	45.8	197.4	0.24%	0.00%
3	871	871	871	874	872	4.5	37.1	0.34%	0.11%
4	898	898	898	900	899	6.1	62.2	0.22%	0.11%
5	919	919	919	926	920	3.1	33.0	0.76%	0.11%
6	984	984	984	986	985	2.7	18.6	0.20%	0.10%
7	1047	1047	1047	1050	1047	4.6	20.5	0.29%	0.00%
8	1105	1105	1105	1112	1105	4.5	27.0	0.81%	0.00%
9	1101	1101	1101	1103	1101	3.5	28.8	0.18%	0.00%
10	1227	1227	1227	1227	1227	3.1	16.8	0.00%	0.00%
11	1249	1249	1249	1249	1249	2.0	6.8	0.00%	0.00%
Avg.						7.8	43.4	0.33%	0.04%

Таблица 2

Примеры со сбалансированной нагрузкой

	Slb	MTlb	CGlb	CGub	ICGub	CGub time	ICGub time	CGub gap	ICGub gap
1	747	747	747	749	748	2.6	25.0	0.27%	0.13%
2	660	664	664	668	665	5.0	36.4	0.60%	0.15%
3	727	727	727	727	727	5.4	22.9	0.00%	0.00%
4	691	691	691	692	691	3.4	58.9	0.14%	0.00%
5	878	878	878	878	878	4.8	15.7	0.00%	0.00%
6	828	828	828	828	828	3.2	12.0	0.00%	0.00%
7	823	823	823	833	823	2.6	39.8	1.22%	0.00%
8	751	752	752	752	752	4.7	20.8	0.00%	0.00%
9	713	718	718	732	719	4.3	49.1	1.95%	0.14%
10	884	884	884	885	885	2.7	12.3	0.11%	0.11%
11	1178	1178	1178	1178	1178	1.6	5.6	0.00%	0.00%
Avg.						3.7	27.2	0.39%	0.05%

близко к нулю. Отклонение CGub gap не превышает 0.81%, что также можно считать хорошим результатом, учитывая время получения таких решений.

Заметим, что нижние оценки Slb, MTlb и CGlb совпадают на всех примерах этого класса, хотя второй пример явно отличается от остальных по времени счета. Этот пример оказался трудным для метода генерации столбцов. Алгоритм "Первый подходящий" не сумел найти оптимальное решение, используя только моменты времени с максимальной нагрузкой. Однако итерационным методом удалось найти оптимальное решение. В четырех примерах итерационным методом получены решения, не совпадающие по целевой функции с нижней оценкой, но разница между ними составляет только один сервер. Возможно, это также оптимальные решения, но для проверки этой гипотезы потребуются более трудоемкие процедуры.

В табл. 2 приводятся результаты расчетов для случая сбалансированной нагрузки по ресурсам, $\sum_{m\in M} d_{1m} / \sum_{m\in M} d_{2m} \approx 1$. Нижние оценки уже слегка отличаются (см. примеры 2, 8 и 9). Итерационный алгоритм снова дает наилучшие результаты, а там где не удается получить оптимальное решение, разница между верхней и нижней оценками оказывается минимальной и составляет единицу. Нижняя оценка MTlb показывает себя лучше Slb на примерах 8 и 9, но совпадает с CGlb.

Таблица 3

Примеры с 20% больших машин

	Slb	MTlb	CGlb	CGub	ICGub	CGub time	ICGub time	CGub gap	ICGub gap
1	676	676	676	677	677	2.3	7.1	0.15%	0.15%
2	689	689	689	689	689	2.1	8.1	0.00%	0.00%
3	720	720	720	723	720	2.1	27.2	0.42%	0.00%
4	695	695	695	696	695	3.4	31.3	0.14%	0.00%
5	638	638	638	638	638	4.1	16.9	0.00%	0.00%
6	648	648	648	660	648	7.3	50.3	1.85%	0.00%
7	621	621	621	627	622	3.4	34.2	0.97%	0.16%
8	643	643	643	646	644	2.0	16.0	0.47%	0.16%
9	747	747	747	747	747	2.2	8.5	0.00%	0.00%
10	745	745	745	746	746	2.3	8.5	0.13%	0.13%
Avg.						3.1	20.8	0.41%	0.06%

Таблица 4

Примеры с 1% больших машин

	Slb	MTlb	CGlb	CGub	ICGub	CGub time	ICGub time	CGub gap	ICGub gap
1	770	770	770	770	770	4.3	14.3	0.00%	0.00%
2	777	777	777	783	777	2.0	19.3	0.77%	0.00%
3	833	833	833	833	833	3.2	19.9	0.00%	0.00%
4	756	756	756	758	756	2.0	11.6	0.26%	0.00%
5	758	758	758	758	758	1.9	7.2	0.00%	0.00%
6	826	826	826	826	826	1.9	5.8	0.00%	0.00%
7	731	731	731	732	732	2.5	11.7	0.14%	0.14%
8	848	848	848	849	848	2.8	22.7	0.12%	0.00%
9	848	848	848	848	848	1.8	6.5	0.00%	0.00%
10	782	782	782	785	782	1.8	39.9	0.38%	0.00%
Avg.						2.45	15.9	0.17%	0.01%

Для табл. 3 выборка из реальных данных проводилась таким образом, чтобы количество больших машин было высоким — около 20% от общего числа. Результаты алгоритмов близки к результатам из предыдущих таблиц. Оценки Slb, MTlb и CGlb совпадают на всех примерах.

Для табл. 4 рассматривалась ситуация, противоположная предыдущей. Только около 1% всех виртуальных машин являются большими. Среднее отклонение и время счета алгоритмов слегка уменьшились. Нижние оценки Slb, MTlb и CGlb также совпадают.

Для табл. 5 из реальных данных выбирались только большие машины. Спрос на RAM и CPU также получился достаточно сбалансированным. Число виртуальных машин варьируется между 7 и 22 тысячами для разных примеров. Количество типов виртуальных машин равняется 5. Среднее время счета очень мало. Оба алгоритма смогли найти оптимальное решение для всех примеров. Нижние оценки Slb и MTlb заметно уступают оценке CGlb. Такой эффект обусловлен сложностью плотной упаковки сервера при наличии только больших виртуальных машин. Здесь NUMA-архитектура не оказывает влияния на решение.

Для примеров в табл. 6 виртуальные машины выбирались случайным образом из реальных данных. Число виртуальных машин и типов такое же, как для табл. 1–4, и равняется соответственно $5e + 4$ и 14. Время выполнения алгоритма не отличается от времени в табл. 1–4. Алгоритм CGub показывает значительно худшие результаты. Метод генерации столбцов де-

Таблица 5

Примеры только с большими машинами

	Slb	MTlb	CGlb	CGub	ICGub	CGub time	ICGub time	CGub gap	ICGub gap
1	533	533	562	562	562	0.6	2.6	0.00%	0.00%
2	541	541	571	571	571	0.7	4.8	0.00%	0.00%
3	549	549	579	579	579	0.7	2.6	0.00%	0.00%
4	550	550	578	578	578	0.4	1.4	0.00%	0.00%
5	762	762	773	773	773	0.8	2.6	0.00%	0.00%
6	734	734	750	750	750	0.8	2.3	0.00%	0.00%
7	784	784	796	796	796	0.8	2.4	0.00%	0.00%
8	796	796	808	808	808	0.9	2.8	0.00%	0.00%
9	775	775	797	797	797	0.8	2.7	0.00%	0.00%
10	819	819	836	836	836	0.9	2.6	0.00%	0.00%
11	659	659	705	705	705	0.3	0.8	0.00%	0.00%
12	616	616	646	646	646	0.2	0.8	0.00%	0.00%
Avg.						0.7	2.4	0.00%	0.00%

Таблица 6

Примеры, сгенерированные случайной выборкой

N	Slb	MTlb	CGlb	CGub	ICGub	CGub time	ICGub time	CGub gap	ICGub gap
1	796	796	1103	1104	1104	2.1	21.3	0.09%	0.09%
2	1253	1253	1739	1739	1739	2.6	24.2	0.00%	0.00%
3	1016	1016	1218	1219	1219	2.1	21.1	0.08%	0.08%
4	873	873	1076	1094	1084	3.5	90.2	1.67%	0.74%
5	818	818	948	967	948	2.2	46.8	2.00%	0.00%
6	313	313	314	315	315	5.5	22.3	0.32%	0.32%
7	1102	1102	1507	1542	1507	2.5	48.2	2.32%	0.00%
8	594	594	797	880	798	1.9	34.1	10.41%	0.13%
9	167	167	167	173	168	6.0	77.1	3.59%	0.60%
10	335	335	335	347	338	3.6	59.8	3.58%	0.90%
Avg.						3.2	44.5	2.41%	0.29%

монстрирует заметно лучшие результаты (CGlb), чем оценки Slb и MTlb.

На рис. 3 представлена диаграмма для итерационного алгоритма. Она показывает число примеров, в которых итерационным алгоритмом T раз вычислялась нижняя оценка и строились допустимые решения задачи. В 23 случаях дополнительные моменты времени не генерируются вообще и используется только один из моментов с максимальной нагрузкой. В 95% используется не более четырех моментов времени. Также одним из наблюдений, которые можно сделать, является отклонение верхней оценкий от нижней не более чем на один контейнер на большом числе примеров. Возникает вопрос о выполнимости MIRUP-гипотезы [35] для данной задачи, в частности для статической векторной задачи и динамической задачи упаковки в контейнеры. Ее проверка, вероятно, будет весьма трудоемкой, однако это может стать интересной темой для будущих исследований.

Вычислительные эксперименты показывают, что итерационный алгоритм работает достаточно стабильно на примерах с разными свойствами. Максимальное время выполнения не превышает 200 с, но в среднем этот алгоритм работает значительно быстрее. Процедура генерации столбцов также способна показывать хорошие результаты, однако в некоторых случаях

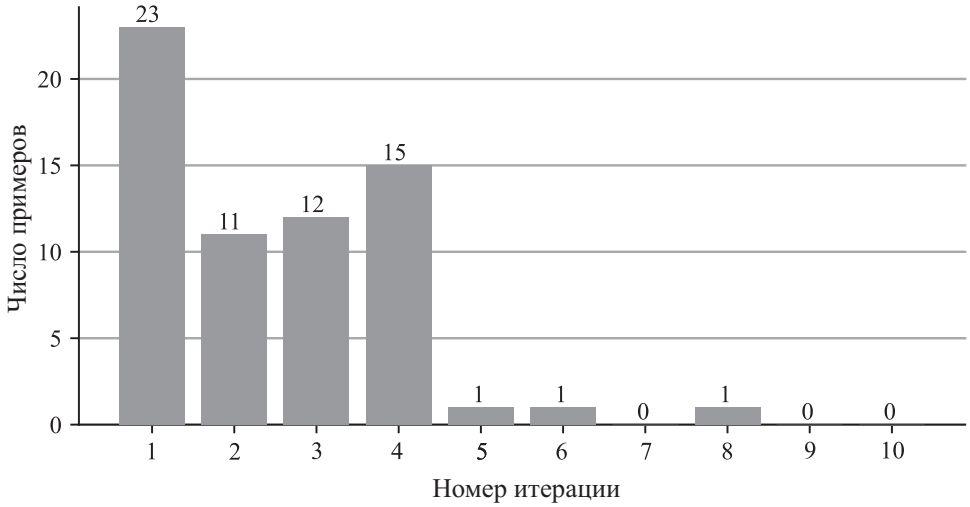

Рис. 3. Требуемое число итераций для получения величины ICGub.

можно получать решения, сильно отклоняющиеся от нижней оценки.

Помимо сгенерированных данных также интересно провести эксперименты на примерах, предложенных в работе [5] для задачи Temporal Bin Packing Problem без NUMA-архитектуры и с одним ресурсом. Для проведения расчетов потребовалась процедура преобразования примеров, как уже упоминалось в начале этого раздела. Во всех примерах используется около 90 типов виртуальных машин, а время существование каждой машины не превышает 10% от общего

Таблица 7

Результаты расчетов на примерах из [5]

N	Lb	Ub	Cols	Time	Slb	MTlb	CGlb	ICGub	CGcols	ICGub time
I40	23	23	789	1034.28	19	23	23	23	53	2.29
I44	26	27	1631	2639.48	22	26	26	**26**	53	1.95
I45	27	27	81	0.53	23	27	27	27	49	1.58
I61	26	40	1924	3011.65	21	25	26	**26**	60	2.53
I64	27	28	1091	2551.92	22	27	27	**27**	54	1.64
I71	28	29	514	494.61	24	28	28	**28**	53	1.84
I72	27	28	985	1823.78	24	27	27	**27**	96	3.46
I73	28	28	593	831.12	23	28	28	28	68	1.86
I75	27	27	284	13.33	24	27	27	27	67	3.78
I76	29	30	736	948.59	24	29	29	**29**	58	2.25
I77	29	29	530	38.47	24	29	29	29	68	4.53
I78	27	27	566	1295.63	23	27	27	27	65	3.48
I82	26	26	904	2757.68	23	26	26	26	48	3.75
I83	28	29	896	564.07	25	27	28	**28**	57	4.12
I86	28	28	742	161.43	24	28	28	28	55	2.80
I88	27	27	135	12.7	22	27	27	27	59	4.78
I89	29	29	717	23.91	24	29	29	29	68	3.08
I90	29	29	1512	229.19	25	29	29	29	55	4.33
I93	31	32	752	1235.46	27	31	31	32	74	3.88
I95	31	31	417	81.47	25	31	31	31	46	3.17
I96	32	32	477	88.91	26	32	32	32	81	2.51
I98	33	33	378	26.27	28	33	33	33	83	2.02
I99	31	32	664	1438.21	28	31	31	**31**	66	3.59

количества моментов времени. Среднее время работы итерационного алгоритма по 100 примерам равняется 3.5 с, а число сгенерированных столбцов (колонка CGcol) не превышает 96. Оптимальное решение найдено в 45 случаях, при этом наибольший разрыв между границами равняется двум контейнерам.

В табл. 7 представлено сравнение предложенного подхода с динамической генерацией столбцов [5]. Приводятся только те примеры, на которых авторам [5] удалось получить решение с помощью генерации столбцов менее чем за один час (табл. 4 в работе [5]). В первом столбце в табл. 7 дается название примера. Следующие четыре столбца — это нижняя оценка, верхняя оценка, число сгенерированных столбцов и время работы метода генерации столбцов из работы [5]. Последние четыре столбца соответствуют результатам итерационного алгоритма. Нижняя оценка MTlb дает более точные результаты, чем Slb, на всех примерах, однако уступает CGlb на примере I61. В восьми примерах удалось улучшить результаты предыдущего алгоритма и получить точное решение задачи. В примере I61 это улучшение составило 14 контейнеров. Оценки для многих примеров совпадают, но время работы и число сгенерированных столбцов значительно отличаются, что связано со статической природой моделей (2.1)–(2.3) и (2.8)–(2.13).

Заключение

В данной статье рассмотрена новая задача упаковки для виртуальных машин и серверов с NUMA-архитектурой. Для ее решения разработана и протестирована эвристика, основанная на методе генерации столбцов и алгоритме “Первый подходящий”, а также ее модификация в виде итерационного алгоритма, подбирающего моменты времени для решения статической задачи. Применение генерации столбцов только для статической задачи положительно сказывается на эффективности и отличает предложенные алгоритмы от других, основанных на данной технике (например, [5]). Проведены вычислительные эксперименты на двух классах исходных данных: полусинтетических данных для реальных типов виртуальных машин и серверов и общедоступных данных с одним ресурсом и без NUMA-архитектуры. Разработанные алгоритмы продемонстрировали малое отклонение от оптимума и высокую скорость работы. Для многих примеров удалось либо найти оптимальное решение и доказать оптимальность, либо получить небольшой зазор между верхней и нижней оценками оптимума.

СПИСОК ЛИТЕРАТУРЫ

1. **Johnson D.S.** Near-optimal bin packing algorithms: Ph.D. dissertation / Department of Mathematics, Massachusetts Institute of Technology. Cambridge: MA, 1973.
2. **Christensen H.I., Khan A., Pokutta S., Tetali P.** Multidimensional bin packing and other related problems: A survey // Computer Science Review. 2017. Vol. 24. P. 63–79. doi: 10.1016/j.cosrev.2016.12.001
3. **Aggoun A., Rhiat A., Fages A.** Panorama of real-life applications in logistics embedding bin packing optimization algorithms, robotics and cloud computing technologies // Proc. of 3rd Internat. Conf. on Logistics Operations Management (GOL). 2016. P. 1–4. doi: 10.1109/GOL.2016.7731693
4. **Shi J., Luo J., Dong F., Jin J., Shen J.** Fast multi-resource allocation with patterns in large scale cloud data center // J. Comput. Sci. 2018. Vol. 26. P. 389–401. doi: 10.1016/j.jocs.2017.05.005
5. **Furini F., Shen X.** Matheuristics for the temporal bin packing problem // Oper. Res./Comput. Sci. Interfaces Series (book series). 2018. Vol. 62. P. 333–345. doi: 10.1007/978-3-319-58253-5_19
6. **Aydin N., Muter I., Birbil I.S.** Multi-objective temporal bin packing proble: An application in cloud computing // Comput. Oper. Res. 2021. Vol. 121. Article no. 104959. doi: 10.1016/j.cor.2020.104959
7. **de Cauwer M., Mehta D., O’Sullivan B.** The temporal bin packing problem: An application to workload management in data centres. // IEEE 28th Internat. Conf. on Tools with Artificial Intelligence (ICTAI 2016). P. 157–164. doi: 10.1109/ICTAI.2016.0033
8. **Bartlett M., Frisch A.M., Hamadi Y., Miguel I., Tarim S.A., Unsworth C.** The Temporal Knapsack Problem and its solution // Proc. of the Integration of AI and OR Techniques in Constraint

Programming for Combinatorial Optimization Problems: Second Internat. Conf. (CPAIOR 2005) / eds. R. Barták, M. Milano. Berlin; Heidelberg: Springer, 2005. P. 34–48. (Lecture Notes in Comp. Sci. 2005; vol. 3524). doi:10.1007/11293853_5

9. **Caprara A., Furini F., Malaguti E.** Uncommon Dantzig–Wolfe reformulation for the Temporal Knapsack Problem // INFORMS J. Comput. 2013. Vol. 25. P. 560–571. doi: 10.1287/ijoc.1120.0521
10. **Pires F.L., Baran B.** A virtual machine placement taxonomy // Proc. of the 15th IEEE/ACM Internat. Symposium on Cluster, Cloud, and Grid Computing. 2015. P. 159–168. doi: 10.1109/CCGrid.2015.15
11. **Wolsey L.A.** Valid inequalities, covering problems and discrete dynamic programs // Annal. Discrete Math. 1977. Vol. 1. P. 527–538. doi: 10.1016/S0167-5060(08)70758-1
12. **Clautiaux F., Sadykov R., Vanderbeck F., Viaud Q.** Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem // Discr. Optim. 2018. Vol. 29. P. 18–44. doi: 10.1016/j.disopt.2018.02.003
13. **Delorme M., Iori M., Martello S.** Bin packing and cutting stock problems: mathematical models and exact algorithms // Eur. J. Oper. Res. 2016. Vol. 255. P. 1–20. doi:10.1016/j.ejor.2016.04.030
14. **Macedo R., Alves C., Valério de Carvalho J.M.** Arc-flow model for the two-dimensional guillotine cutting stock problem // Comput. Oper. Res. 2010. Vol. 37. P. 991–1001.
15. **Martinovic J., Scheithauer G.** Integer linear programming models for the skiving stock problem // Eur. J. Oper. Res. 2016. Vol. 251, no. 2. P. 356–368. doi: 10.1016/j.ejor.2015.11.005
16. **Valério de Carvalho J.M.** Exact solution of cutting stock problems using column generation and branch-and-bound // Int. Trans. Oper. Res. 1998. Vol. 5. P. 35–44.
17. **Brandao F., Pedroso J.P.** Bin packing and related problems: general arc-flow formulation with graph compression // Comput. Oper. Res. 2016. Vol. 69. P. 56–67. doi: 10.1016/j.cor.2015.11.009
18. **Delorme M., Iori M.** Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems // INFORMS J. Comput. 2020. Vol. 32, no. 1. P. 101–119. doi: 10.1287/ijoc.2018.0880
19. **Cote J.-F., Iori M.** The meet-in-the-middle principle for cutting and packing problems // INFORMS J. Comput. 2018. Vol. 30, no. 4. P. 625–786 . doi: 10.1287/ijoc.2018.0806
20. **Dell'Amico M., Furini F., Iori M.** A branch-and-price algorithm for the temporal bin packing problem // Comput. Oper. Res. 2020. Vol. 112. Article no. 104825. doi: 10.1016/j.cor.2019.104825
21. **Martinovic J., Strasdat N., Selch M.** Compact Integer Linear Programming Formulations for the temporal bin packing problem with fire-ups // Comput. Oper. Res. 2021. Vol. 132. Article no. 105288. doi: 10.1016/j.cor.2021.105288
22. **Martinovic J., Strasdat N., Valerio de Carvalho J.M., Furini F.** Variable and constraint reduction techniques for the temporal bin packing problem with fire-ups // Optim. Letters. 2021. Vol. 16. P. 2333–2358. doi: 10.1007/s11590-021-01825-x
23. **Martinovic J., Strasdat N.** Theoretical insights and a new class of valid inequalities for the temporal bin packing problem with fire-ups. Preprint MATH-NM-01-2022 / Technische Universitat Dresden. 2022. Available at: http://www.optimization-online.org/DB_HTML/2022/02/8791.html .
24. **Valerio de Carvalho J.M.** LP Models for bin packing and cutting stock problems // Eur. J. Oper. Res. 2002. Vol. 141, no. 2. P. 253–273. doi: 10.1016/S0377-2217(02)00124-8
25. **Manchanda N., Anand K.** Non-uniform memory access (NUMA). NY: New York University, 2014.
26. **Kellerer H., Pferschy U., Pisinger D.** Knapsack problems. Berlin, Heidelberg: Springer, 2004. 548 p. doi: 10.1007/978-3-540-24777-7
27. **Канторович Л.В., Залгаллер В.А.** Расчет рационального раскроя промышленных материалов. Л.: Лениздат, 1951. 198 с.
28. **Johnson D.S.** Fast algorithms for bin packing // J. Comp. System Sci. 1974. Vol. 8, no. 3. P. 272–314. doi: 10.1016/S0022-0000(74)80026-7
29. **Ratushnyi A., Kochetov Y.** A column generation based heuristic for a temporal bin packing problem // Mathematical Optimization Theory and Operations Research: Proc. of 20th Internat. Conf. (MOTOR 2021) / eds. P. Pardalos, M. Khachay, A. Kazakov. 2021. P. 96–110. (Lecture Notes Comp. Sci.; vol. 12755). doi: 10.1007/978-3-030-77876-7_7
30. Mathematical Optimization Theory and Operations Research: Proc. of 20th Internat. Conf. (MOTOR 2021) / eds. P. Pardalos, M. Khachay, A. Kazakov. Cham: Springer, 2021. 493 p. doi: 10.1007/978-3-030-77876-7
31. **Ratushnyi A.** A pattern-based heuristic for a temporal bin packing problem with conflicts // Mathematical Optimization Theory and Operations Research: Recent Trends: Proc. of 22nd Internat. Conf. (MOTOR 2023) / eds. M. Khachay et al. 2023. P. 152–166. (Communications in Computer and Information Sci.; vol. 1881). doi: 10.1007/978-3-031-43257-6_13

32. **Martello S., Toth P.** Lower bounds and reduction procedures for the bin packing problem // Discrete Appl. Math. 1990. Vol. 28. P. 59–70.
33. **Kochetov Yu., Kondakov A.** A hybrid VNS matheuristic for a bin packing problem with a color constraint // Yugoslav J. Oper. Res. 2021. Vol. 31, no. 3. P. 285–298. doi:10.2298/YJOR200117009K
34. Gurobi optimization: Gurobi optimizer reference manual. 2021. Available at: http://www.gurobi.com .
35. **Scheithauer G., Terno J.** The modified integer round-up property of the one- dimensional cutting stock problem // Eur. J. Oper. Res. 1995. Vol. 84. P. 562–571.

Поступила 30.05.2023
После доработки 18.09.2023
Принята к публикации 2.10.2023

Кочетов Юрий Андреевич
д-р физ.-мат. наук, профессор
главный науч. сотрудник
Институт математики имени С. Л. Соболева СО РАН
г. Новосибирск
e-mail: jkochet@math.nsc.ru

Ратушный Алексей Владленович
аспирант
Институт математики имени С. Л. Соболева СО РАН
г. Новосибирск
e-mail: alexeyratushny@gmail.com

REFERENCES

1. Johnson D.S. *Near-optimal bin packing algorithms*: Ph.D. dissertation, Department of Mathematics, Massachusetts Institute of Technology. Cambridge: MA, 1973.
2. Christensen H.I., Khan A., Pokutta S., Tetali P. Multidimensional bin packing and other related problems: A survey. In: *Computer Science Review*, 2017, vol. 24, pp. 63–79. doi: 10.1016/j.cosrev.2016.12.001
3. Aggoun A., Rhiat A., Fages A. Panorama of real-life applications in logistics embedding bin packing optimization algorithms, robotics and cloud computing technologies. In: *Proc. of 3rd Internat. Conf. on Logistics Operations Management* (GOL), 2016, pp. 1–4. doi: 10.1109/GOL.2016.7731693
4. Shi J., Luo J., Dong F., Jin J., Shen J. Fast multi-resource allocation with patterns in large scale cloud data center. *J. Comput. Sci.*, 2018, vol. 26, pp. 389–401. doi: 10.1016/j.jocs.2017.05.005
5. Furini F., Shen X. Matheuristics for the temporal bin packing problem. *Oper. Res./Comput. Sci. Interfaces Series* (book series), 2018, vol. 62, pp. 333–345. doi: 10.1007/978-3-319-58253-5_19
6. Aydin N., Muter I., Birbil I.S. Multi-objective temporal bin packing proble: An application in cloud computing. *Comput. Oper. Res.*, 2021, vol. 121, article no. 104959. doi: 10.1016/j.cor.2020.104959
7. de Cauwer M., Mehta D., O'Sullivan B. The temporal bin packing problem: An application to workload management in data centres. In: IEEE 28th Internat. Conf. on Tools with Artificial Intelligence (ICTAI 2016). P. 157–164. doi: 10.1109/ICTAI.2016.0033
8. Bartlett M., Frisch A.M., Hamadi Y., Miguel I., Tarim S.A., Unsworth C. The temporal knapsack problem and its solution. In: (eds.) R. Barták, M. Milano, Proc. of the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: Second Internat. Conf. (CPAIOR 2005), Ser. Lecture Notes in Comp. Sci., vol. 3524, Berlin, Heidelberg: Springer, 2005, pp. 34–48. doi:10.1007/11293853_5
9. Caprara A., Furini F., Malaguti E. Uncommon Dantzig–Wolfe reformulation for the temporal knapsack problem. *INFORMS J. Comput.*, 2013, vol. 25, pp. 560–571. doi: 10.1287/ijoc.1120.0521
10. Pires F.L., Baran B. A virtual machine placement taxonomy. In: *Proc. of the 15th IEEE/ACM Internat. Symposium on Cluster, Cloud, and Grid Computing*, 2015, pp. 159–168. doi: 10.1109/CCGrid.2015.15
11. Wolsey L.A. Valid inequalities, covering problems and discrete dynamic programs. *Annal. Discrete Math.*, 1977, vol. 1, pp. 527–538. doi: 10.1016/S0167-5060(08)70758-1

12. Clautiaux F., Sadykov R., Vanderbeck F., Viaud Q. Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem. *Discr. Optim.*, 2018, vol. 29, pp. 18–44. doi: 10.1016/j.disopt.2018.02.003

13. Delorme M., Iori M., Martello S. Bin packing and cutting stock problems: mathematical models and exact algorithms. *Eur. J. Oper. Res.*, 2016, vol. 255, pp. 1–20. doi:10.1016/j.ejor.2016.04.030

14. Macedo R., Alves C., Valério de Carvalho J.M. Arc-flow model for the two-dimensional guillotine cutting stock problem. *Comput. Oper. Res.*, 2010, vol. 37, pp. 991–1001.

15. Martinovic J., Scheithauer G. Integer linear programming models for the skiving stock problem. *Eur. J. Oper. Res.*, 2016, vol. 251, no. 2, pp. 356–368. doi: 10.1016/j.ejor.2015.11.005

16. Valério de Carvalho J.M. Exact solution of cutting stock problems using column generation and branch-and-bound. *Int. Trans. Oper. Res.*, 1998, vol. 5, pp. 35–44.

17. Brandao F., Pedroso J.P. Bin packing and related problems: general arc-flow formulation with graph compression. *Comput. Oper. Res.*, 2016, vol. 69, pp. 56–67. doi: 10.1016/j.cor.2015.11.009

18. Delorme M., Iori M. Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. *INFORMS J. Comput.*, 2020, vol. 32, no. 1, pp. 101–119. doi: 10.1287/ijoc.2018.0880

19. Cote J.-F., Iori M. The meet-in-the-middle principle for cutting and packing problems. *INFORMS J. Comput.*, 2018, vol. 30, no. 4, pp. 625–786 . doi: 10.1287/ijoc.2018.0806

20. Dell'Amico M., Furini F., Iori M. A branch-and-price algorithm for the temporal bin packing problem. *Comput. Oper. Res. 2020.*, vol. 112, article no. 104825. doi: 10.1016/j.cor.2019.104825

21. Martinovic J., Strasdat N., Selch M. Compact integer linear programming formulations for the temporal bin packing problem with fire-ups. *Comput. Oper. Res.*, 2021, vol. 132, article no. 105288. doi: 10.1016/j.cor.2021.105288

22. Martinovic J., Strasdat N., Valerio de Carvalho J.M., Furini F. Variable and constraint reduction techniques for the temporal bin packing problem with fire-ups. *Optim. Letters*, 2021, vol. 16, pp. 2333–2358. doi: 10.1007/s11590-021-01825-x

23. Martinovic J., Strasdat N. *Theoretical insights and a new class of valid inequalities for the temporal bin packing problem with fire-ups*, Preprint MATH-NM-01-2022. Dresden: Technische Universität Dresden, 2022. Available at: http://www.optimization-online.org/DB_HTML/2022/02/8791.html .

24. Valerio de Carvalho J.M. LP Models for bin packing and cutting stock problems. *Eur. J. Oper. Res.*, 2002, vol. 141, no. 2, pp. 253–273. doi: 10.1016/S0377-2217(02)00124-8

25. Manchanda N., Anand K. *Non-uniform memory access* (NUMA), NY: New York University, 2014.

26. Kellerer H., Pferschy U., Pisinger D. *Knapsack problems*. Berlin, Heidelberg: Springer, 2004, 548 p. doi: 10.1007/978-3-540-24777-7

27. Kantorovich, L.V., Zalgaller, V.A. *Raschet ratsional'nogo raskroya promyshlennykh materialov* [Computation of rational cutting of industrial materials]. Leningrad, Lenizdat, 1951, 198 p.

28. Johnson D.S. Fast algorithms for bin packing. *J. Comp. System Sci.*, 1974, vol. 8, no. 3, pp. 272–314. doi: 10.1016/S0022-0000(74)80026-7

29. Ratushnyi A., Kochetov Y. A column generation based heuristic for a temporal bin packing problem. In: Mathematical Optimization Theory and Operations Research: Proc. of 20th Internat. Conf. (MOTOR 2021), Ser. Lecture Notes Comp. Sci., vol. 12755, 2021, pp. 96–110. doi: 10.1007/978-3-030-77876-7_7

30. Mathematical Optimization Theory and Operations Research: Proc. of 20th Internat. Conf. (MOTOR 2021), (eds.) P. Pardalos, M. Khachay, A. Kazakov, 493 p, Cham: Springer, 2021.
doi: 10.1007/978-3-030-77876-7

31. Ratushnyi A. A pattern-based heuristic for a temporal bin packing problem with conflicts. In: (eds.) M. Khachay et al., Mathematical Optimization Theory and Operations Research: Recent Trends: Proc. of 22nd Internat. Conf. (MOTOR 2023), Communications in Computer and Information Sci., vol. 1881, 2023, pp. 152–166. doi: 10.1007/978-3-031-43257-6_13

32. Martello S., Toth P. Lower bounds and reduction procedures for the bin packing problem. *Discrete Appl. Math.*, 1990, vol. 28, pp. 59–70.

33. Kochetov Yu., Kondakov A. A hybrid VNS matheuristic for a bin packing problem with a color constraint. *Yugoslav J. Oper. Res.*, 2021, vol. 31, no. 3, pp. 285–298. doi:10.2298/YJOR200117009K

34. Gurobi optimization: Gurobi optimizer reference manual, 2021. Available at: http://www.gurobi.com .
35. Scheithauer G., Terno J. The modified integer round-up property of the one- dimensional cutting stock problem. *Eur. J. Oper. Res.*, 1995, vol. 84, pp. 562–571.

Received May 30, 2023
Revised September 18, 2023
Accepted October 2, 2023

Funding Agency: The research was carried out within the State Contract of the Sobolev Institute of Mathematics (FWNF-2022-0019).

Yury Andreevich Kochetov, Dr. Phys.-Math. Sci., Prof., Sobolev Institute of Mathematics of the Siberia Branch of the Russian Academy of Sciences, Novosibirsk, 630090 Russia,
e-mail: jkochet@math.nsc.ru .

Alexey Vladlenovich Ratushnyi, doctoral student, Sobolev Institute of Mathematics of the Siberian Branch of the Russian Academy of Sciences, Novosibirsk, Russia,
e-mail: alexeyratushny@gmail.com .

Cite this article as: Yu. A. Kochetov, A. V. Ratushnyi. Upper and lower bounds for the optimum in a temporal bin packing problem. *Trudy Instituta Matematiki i Mekhaniki UrO RAN*, 2023, vol. 30, no. 1, pp. 109–127 .