ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Июль–сентябрь 2022. Т. 29, № 3. С. 102–115

УДК 519.8+518.25 DOI 10.33048/daio.2022.29.741

ЗАДАЧА О РЮКЗАКЕ ДЛЯ ПРЯМОУГОЛЬНЫХ ПРЕДМЕТОВ С ОГРАНИЧЕНИЕМ НА РАСПОЛОЖЕНИЕ ЦЕНТРА ТЯЖЕСТИ

С. М. Шперлинг[1, a], *Ю. А. Кочетов*[2, b]

[1] Новосибирский гос. университет,
ул. Пирогова, 2, 630090 Новосибирск, Россия

[2] Институт математики им. С. Л. Соболева,
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия

E-mail: [a] s.shperling@g.nsu.ru, [b] jkochet@math.nsc.ru

Аннотация. Имеется множество предметов прямоугольной формы с заданными шириной, длиной и массой и большой прямоугольник (рюкзак) с известными шириной и длиной. Требуется выбрать такое подмножество предметов и найти их расположение в рюкзаке без взаимных пересечений, чтобы минимизировать свободное место в рюкзаке. Центр тяжести упакованных предметов не должен уклоняться от геометрического центра рюкзака по обеим координатам больше заданного порога. Мы представляем решение задачи в виде перестановки предметов и используем известный skyline алгоритм в роли декодирующей процедуры. Ограничение на расположение центра тяжести включается в целевую функцию в виде штрафа. Чтобы найти наилучшую перестановку, используется алгоритм имитации отжига. Для генерации соседних решений два предмета меняются местами в перестановке, а при нарушении ограничений используется специальное правило для сдвига центра тяжести в нужную сторону. Обсуждаются результаты численных экспериментов для примеров с известными оптимальными решениями. Табл. 2, ил. 3, библиогр. 14.

Ключевые слова: двумерная упаковка, алгоритм skyline, ограничение на расположение центра тяжести, локальный поиск.

Исследование выполнено в рамках государственного задания ИМ СО РАН (проект FWNF–2022–0019).

Введение

Даны набор прямоугольных предметов с известными длиной, шириной и массой и большой прямоугольник с заданными длиной и шириной. Этот большой прямоугольник будем называть рюкзаком. Суммарная площадь всех предметов превосходит площадь рюкзака, но каждый отдельный предмет помещается в рюкзак. Нужно найти такой поднабор предметов и их расположение в рюкзаке без взаимных пересечений, чтобы площадь общего незанятого места в рюкзаке была минимальной. Допускаются повороты предметов на 90°. Центр тяжести упакованных предметов должен быть в разрешённой зоне около геометрического центра рюкзака.

У этой оптимизационной задачи есть множество применений в логистике, например, упаковка предметов в грузовик. При упаковке грузовика нужно сложить предметы как можно компактнее, чтобы было меньше свободного места. Центр тяжести упакованных предметов не должен сильно отклоняться от геометрического центра кузова, чтобы грузовик не перевернулся.

Задача о рюкзаке достаточно широко изучена. Разработано и исследовано много точных алгоритмов и эвристик для её разных постановок. Впервые задача о рюкзаке была предложена в [1]. Несложно заметить, что новая постановка задачи NP-трудна, так как является обобщением задачи subset-sum [2, гл. 4]. Возьмём такой набор предметов, что все их длины будут совпадать с длиной рюкзака, а ограничение на расположение центра тяжести отсутствует. Вместимость рюкзака равна его площади. Таким образом, известная NP-трудная задача subset-sum оказывается частным случаем рассматриваемой задачи.

Первая постановка задачи о рюкзаке с условиями на веса предметов рассматривалась в работах [3, 4]. Авторы сформулировали несколько разных видов ограничений на расположение центра тяжести, а также предложили несколько адаптированных под эти ограничения алгоритмов. В [5] изучалась трёхмерная задача о рюкзаке с ограничениями на устойчивость. Для её решения построена модель целочисленного линейного программирования и применялся эвристический алгоритм. В [6] рассматривалась задача загрузки воздушного судна, которая является одним из вариантов задачи 3D упаковки с ограничениями на устойчивость. В работе [7] изучалась близкая к задаче о рюкзаке формулировка задачи упаковки в контейнеры с ограничениями устойчивости. Рассматривался случай, в котором все предметы имеют подобные размеры, но разные массы. Более того, предметы могут быть расположены только в предопределённых позициях.

Задача упаковки в контейнеры также хорошо изучена. В [8] описана многоразмерная задача упаковки в контейнеры с ограничениями

на устойчивость с помощью модели частично целочисленного линейного программирования. Двумерную задачу упаковки в контейнеры исследовали в [9], используя одну из разновидностей генетического алгоритма. В [10] для решения такой задачи предложен эволюционный подход, использующий роевые методы оптимизации.

В данной работе рассматривается новый вариант задачи о рюкзаке при ограничениях на расположение общего центра тяжести. Построена математическая модель и получены оценки работоспособности коммерческого решателя Gurobi: при каких размерностях всё ещё удаётся находить оптимальное решение задачи. Для больших размерностей разработан вариант алгоритма имитации отжига (SA). Он протестирован на примерах с известными оптимальными решениями.

Алгоритм имитации отжига предложен в [11]. Его можно рассматривать как способ генерации цепи Маркова на конечном множестве состояний, каждое из которых соответствует допустимому решению задачи. В [12] представлены результаты вычислительных экспериментов с алгоритмом SA для задачи упаковки. Показана эффективность данного метода для такого типа задач и наиболее перспективные способы генерации соседних решений. В данной работе решения представляются в виде перестановки предметов, а в роли декодирующей процедуры используется известный skyline алгоритм. Ограничения на расположение центра тяжести включаются в целевую функцию в виде штрафа. Другими словами, мы расширяем допустимую область и рассматриваем недопустимые решения по условиям на центр тяжести. Для нахождения наилучшей перестановки применяется алгоритм имитации отжига, где соседние решения порождаются с использованием модифицированной процедуры замены одного предмета другим. Представлены результаты вычислительных экспериментов для двух типов входных данных.

Работа организована следующим образом. В разд. 1 сформулирована задача и представлена модель частично целочисленного линейного программирования. В разд. 2 описан известный skyline алгоритм, который используется в алгоритме SA в роли декодирующей процедуры. В разд. 3 представлен модифицированный вариант классического алгоритма имитации отжига. Результаты вычислительных экспериментов приведены в разд. 4. В заключении подводятся итоги и описаны перспективы дальнейших исследований.

1. Постановка задачи

Пусть размеры рюкзака задаются шириной W и длиной H, а набор предметов — множеством I. Каждый предмет $i \in I$ имеет ширину w_i, длину h_i и массу m_i. Нужно найти такой поднабор предметов и такое их расположение без пересечений в рюкзаке, чтобы площадь свободного

места в рюкзаке была минимальной. Допускаются повороты предметов на 90°. Центр тяжести упакованных предметов должен находиться в разрешённой зоне около геометрического центра рюкзака. Обозначим через δ_x и δ_y максимально возможное отклонение общего центра тяжести упакованных предметов от геометрического центра по осям рюкзака.

Чтобы сформулировать задачу в виде частично целочисленной линейной программы, введём следующие переменные:

(x_i, y_i) — неотрицательные координаты левого нижнего угла предмета i,

$\left(c_i^x, c_i^y\right)$ — неотрицательные координаты геометрического центра предмета i,

$$l_{ij} = \begin{cases} 1, & \text{если предмет } i \text{ левее предмета } j, \\ 0 & \text{иначе,} \end{cases}$$

$$b_{ij} = \begin{cases} 1, & \text{если предмет } i \text{ ниже предмета } j, \\ 0 & \text{иначе,} \end{cases}$$

$$r_i = \begin{cases} 1, & \text{если предмет } i \text{ повёрнут на } 90°, \\ 0 & \text{иначе,} \end{cases}$$

$$z_i = \begin{cases} 1, & \text{если предмет } i \text{ упакован в рюкзак,} \\ 0 & \text{иначе.} \end{cases}$$

Введём большое число $B = \sum\limits_{i \in I} (w_i + h_i)$ и представим задачу в следующем виде:

$$HW - \sum_{i \in I} h_i w_i z_i \to \min, \tag{1}$$

$$x_i + w_i(1 - r_i) + h_i r_i \leqslant H + (1 - z_i)B, \quad i \in I, \tag{2}$$

$$y_i + h_i(1 - r_i) + w_i r_i \leqslant W + (1 - z_i)B, \quad i \in I, \tag{3}$$

$$x_i + w_i(1 - r_i) + h_i r_i \leqslant x_j + (1 - l_{ij})B + (1 - z_j)B, \quad i, j \in I, \tag{4}$$

$$y_i + h_i(1 - r_i) + w_i r_i \leqslant y_j + (1 - b_{ij})B + (1 - z_j)B, \quad i, j \in I, \tag{5}$$

$$l_{ij} + l_{ji} + b_{ij} + b_{ji} \geqslant 1, \quad i, j \in I,\ i \neq j, \tag{6}$$

$$c_i^x \leqslant z_i B, \tag{7}$$

$$c_i^y \leqslant z_i B, \tag{8}$$

$$c_i^x = x_i + \frac{w_i}{2}(1 - r_i) + \frac{h_i}{2} r_i - (1 - z_i)B, \tag{9}$$

$$c_i^y = y_i + \frac{h_i}{2}(1 - r_i) + \frac{w_i}{2} r_i - (1 - z_i)B, \tag{10}$$

$$\frac{W}{2} - \delta_x \leqslant \frac{\sum\limits_{i \in I} m_i c_i^x}{\sum\limits_{i \in I} z_i m_i} \leqslant \frac{W}{2} + \delta_x, \tag{11}$$

$$\frac{H}{2} - \delta_y \leqslant \frac{\sum\limits_{i \in I} m_i c_i^y}{\sum\limits_{i \in I} z_i m_i} \leqslant \frac{H}{2} + \delta_y. \tag{12}$$

Целевая функция (1) определяет площадь пустого места в рюкзаке. Неравенства (2), (3) гарантируют, что выбранные предметы удастся разместить в рюкзаке. Ограничения (4)–(6) не дают упакованным предметам пересекаться. Условия (7), (8) позволяют игнорировать неупакованные предметы. Равенства (9), (10) определяют координаты геометрического центра упакованных предметов. Двусторонние ограничения (11), (12) требуют, чтобы общий центр тяжести упакованных предметов был в разрешённой области около геометрического центра рюкзака.

2. Skyline алгоритм

При решении задачи будем применять skyline алгоритм [13] в роли декодирующей процедуры для перестановки предметов. Этот алгоритм был предложен для решения задачи упаковки в полосу. Он начинает свою работу с пустой полосы с известной шириной и набора предметов с заданными шириной и длиной. На каждом шаге алгоритма имеется упаковка некоторых предметов и их верхняя огибающая в виде последовательности горизонтальных и вертикальных линий. Шаг состоит в выборе самой нижней горизонтальной линии (сегмента огибающей) и нахождении наиболее подходящего предмета для этого сегмента. Все

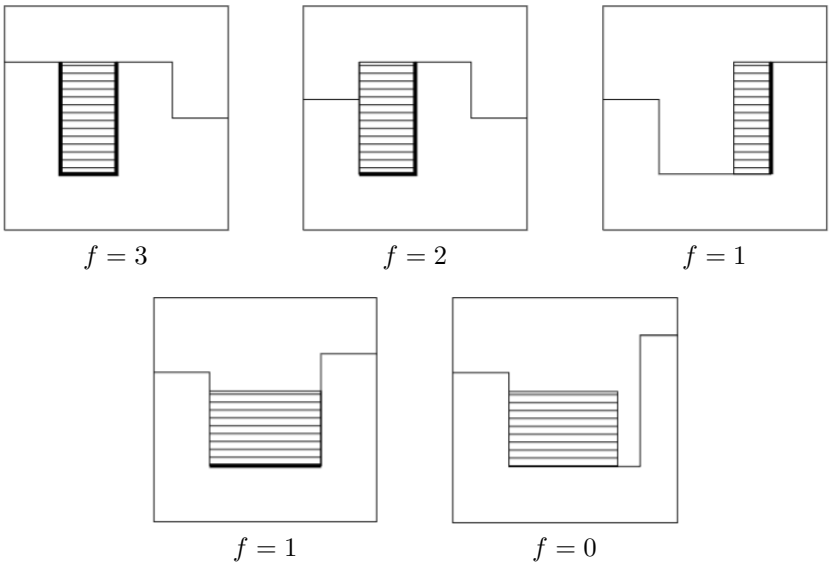

Рис. 1. Различные значения функции соответствия f

неупакованные предметы анализируются, чтобы понять, какой из них подходит для выбранного сегмента лучше всего.

Для оценки предметов вычисляется функция соответствия, которая определяется как число сторон прямоугольника, совпадающих с высотами или шириной выбранного сегмента. Это значит, что функция соответствия принимает значения от 0 (если ширина и высота предмета не совпадают с размерами сегмента) до 3 (когда три стороны предмета совпадают со сторонами сегмента). Примеры расположения предметов с разными значениями функций соответствия представлены на рис. 1. Если ни один предмет не входит в данный сегмент, то сегмент удаляется поднятием горизонтальной линии до ближайшего соседнего сегмента.

Мы добавили возможность вращать предметы на 90° на каждом шаге. Исходный алгоритм упаковывает все предметы в полосу, минимизируя её высоту. В данной задаче предметы упаковываются в рюкзак, поэтому алгоритм останавливается, как только не удаётся упаковать ни один предмет внутрь рюкзака.

3. Алгоритм имитации отжига

Будем представлять решения задачи в виде перестановок предметов. Для поиска поднабора предметов, которые будут упакованы, используем skyline алгоритм. Он оценивает предметы в перестановке и выбирает первый из предметов с наибольшим значением функции соответствия. Таким образом, разные перестановки могут давать разные решения. Требуется найти перестановку, которая даёт наилучшее решение при выполнении ограничений на расположение общего центра тяжести. Для этого применяется алгоритм имитации отжига [11].

Заметим, что skyline алгоритм игнорирует ограничения на расположение центра тяжести, поэтому включим их в целевую функцию в виде штрафа. Величина штрафа будет зависеть от суммарного нарушения

Алгоритм 1. Штрафная функция

Вход: *Упаковка* и текущее значение параметра r.

1: Вычисляем общий центр тяжести *упаковки*.
2: **if** ограничения (11) или (12) нарушаются **then**
3: $\quad r := r + 1$
4: $\quad \Delta =$ общее нарушение в ограничениях (11), (12)
5: $\quad$ *штраф* $= c\Delta r$
6: **else**
7: $\quad r = 0$
8: $\quad$ *штраф* $= 0$
9: **return** r, *штраф*

Алгоритм 2. Алгоритм имитации отжига

Вход: Начальная температура T_0 и минимальная температура $T_{\min}$.

Генерируем начальную перестановку π случайным образом, задаём $T = T_0$, $t = 0$.
while $T > T_{\min}$ **do**
    for i **in** range(K) **do**
        упаковка $\leftarrow$ skyline(π)
        if $F(\pi) + \textit{штраф}(\pi) = 0$ **then**
            break
        if *упаковка* удовлетворяет условиям (11), (12) **then**
            $\pi' \leftarrow$ случайная перестановка из swap(π)
        else
            $\pi' \leftarrow$ случайная перестановка из swap$'(\pi)$
        $\Delta \leftarrow F(\pi) + \textit{штраф}(\pi) - F(\pi') - \textit{штраф}(\pi')$
        if $\Delta \geqslant 0$ **then**
            $\pi \leftarrow \pi'$
        else
            $\pi \leftarrow \pi'$ с вероятностью $\exp(-\Delta/T)$
    $t := t + 1$
    $T = T_0/(1 + t)$
return лучшая найденная допустимая *упаковка*

условий (11), (12) (параметр Δ) и количества итераций подряд (параметр r), в которых было принято недопустимое решение. Штраф обнуляется при возвращении в допустимую область. Псевдокод вычисления функции штрафа представлен алгоритмом 1, где c является некоторой константой.

Будем использовать известную процедуру swap для генерации соседних решений в алгоритме SA. Выбираются два предмета из перестановки и меняются местами. Если же исходное решение было недопустимым, то применяется другое правило. Выбираем первый предмет по одну сторону с общим центром тяжести по одной из координат от разрешённой зоны. Второй предмет выбираем из тех, что находятся по другую сторону от разрешённой зоны по той же оси, если при замене общий центр тяжести становится ближе к разрешённой зоне. Множество таких соседних перестановок обозначим через swap$'$.

Псевдокод SA алгоритма представлен в виде алгоритма 2, где K — число итераций при данной температуре и $F(\pi)$ — значение целевой функции (1) для перестановки π.

4. Результаты вычислительных экспериментов

Чтобы протестировать эффективность коммерческого решателя Gurobi [14] на построенной математической модели и возможности алгоритма имитации отжига, были проведены вычислительные эксперименты. Математическая модель протестирована с использованием языка Python и Gurobi под академической лицензией. Все расчёты проходили на персональном компьютере с параметрами AMD Ryzen 5 3500U с Radeon Vega Mobile Gfx, 2,10 ГГц, RAM 16 ГБ, работающим на операционной системе Windows 10.

Таблица 1

Вычислительные результаты для модели (1)–(12)

		Gurobi		Алгоритм SA		
$\|I\|$	$\sigma_x = \sigma_y$	F	time	F	E(time)	D(time)
10	25	6840,0	7,166	6840,0	0,288	0,009
10	15	7151,0	4,133	7151,0	0,293	0,010
10	5	6927,0	13,081	6927,0	0,329	0,007
20	25	3962,0	173,4	3962,0	1,298	0,018
20	15	4157,0	212,884	4157,0	1,379	0,020
20	5	3877,0	166,55	3877,0	0,669	0,00042
50	25	1495,0	428,354	361,0	10,880	7,162
50	15	1160,0	328,271	766,0	11,669	0,115
50	5	1464,0	390,213	519,0	12,648	0,068

В алгоритме вычисления штрафа параметр c выбирался равным 10, как хорошо подходящий для рассматриваемых входных данных. Параметр K в алгоритме имитации отжига равен количеству предметов в исходных данных, $T_0 = 50$ и $T_{\min} = 1$. В численных экспериментах генерировались тестовые примеры с $H = W = 100$ и $\sigma_x = \sigma_y$. Все прямоугольные предметы имеют случайно сгенерированные длину и ширину из отрезка $[5, 30]$ и массу из отрезка $[1, 100]$. Эти величины генерировались независимо для каждого предмета с равномерным распределением. В табл. 1 представлены результаты работы решателя Gurobi. В этих экспериментах время работы ограничено 330 секундами. Получено оптимальное решение для входных данных с 10 и 20 прямоугольниками. Для этих же входных данных протестирован алгоритм имитации отжига. Мы тестировали его 30 раз и считали среднее время работы E(time) и дисперсию D(time). Так как алгоритм SA вероятностный, такие данные дадут больше представления о его эффективности. Алгоритм получил оптимальные решения или улучшил решения Gurobi за меньшее время.

Для дальнейших экспериментов созданы дополнительные тестовые примеры с известным оптимальным решением без пустого пространства

в рюкзаке. Примеры формировались по следующему правилу. Строилось гильотинное разбиение рюкзака. Массы задавались обратно пропорциональными площади. Если условие на расположение центра тяжести не было выполнено, то некоторые из масс увеличивались до тех пор, пока такая упаковка не становилась допустимой. Полученное решение оптимально по построению. Все предметы этого решения включались в итоговый набор прямоугольников. После этого в набор добавлялись предметы со случайными параметрами. Значения длины и ширины выбирались случайным образом с равномерным распределением на отрезке $[5, 30]$, массы — на отрезке $[1, 100]$. Затем все предметы итогового набора перемешивались.

Таблица 2

Вычислительные результаты работы алгоритма имитации отжига

		Алгоритм SA		Станд. алг. SA	
$\|I\|$	$\sigma_x = \sigma_y$	E(time)	D(time)	E(time)	D(time)
20	25	0,128	0,013	0,0844	0,0044
20	15	0,251	0,047	0,1061	0,0083
20	5	0,220	0,036	0,1205	0,0209
50	25	1,967	1,892	2,745	3,045
50	15	1,996	2,562	4,035	6,502
50	5	2,319	4,853	4,445	3,499
100	25	3,225	3,305	4,126	8,091
100	15	1,585	1,491	4,425	15,874
100	5	5,401	8,661	7,088	31,536
200	25	0,473	0,386	1,224	0,566
200	15	0,469	0,316	1,167	0,422
200	5	1,054	0,722	1,898	1,987
300	25	0,546	1,123	1,399	1,187
300	15	1,435	1,313	17,237	12,353
300	5	2,540	3,094	3,569	6,140

В табл. 2 представлены результаты работы алгоритма имитации отжига и стандартного алгоритма имитации отжига (когда перестановка π' всегда выбирается из множества $\mathrm{swap}(\pi)$). Алгоритмы получали оптимальное решение за приемлемое время. Алгоритмы тестировались 30 раз для каждого тестового набора и подсчитывалось среднее значение времени поиска оптимального решения и дисперсия. Третий и четвёртый столбец показывают информацию о новом алгоритме имитации отжига, последние два — о стандартном алгоритме имитации отжига. Заметим, что новый алгоритм имитации отжига находит оптимальное решение быстрее и с меньшей дисперсией.

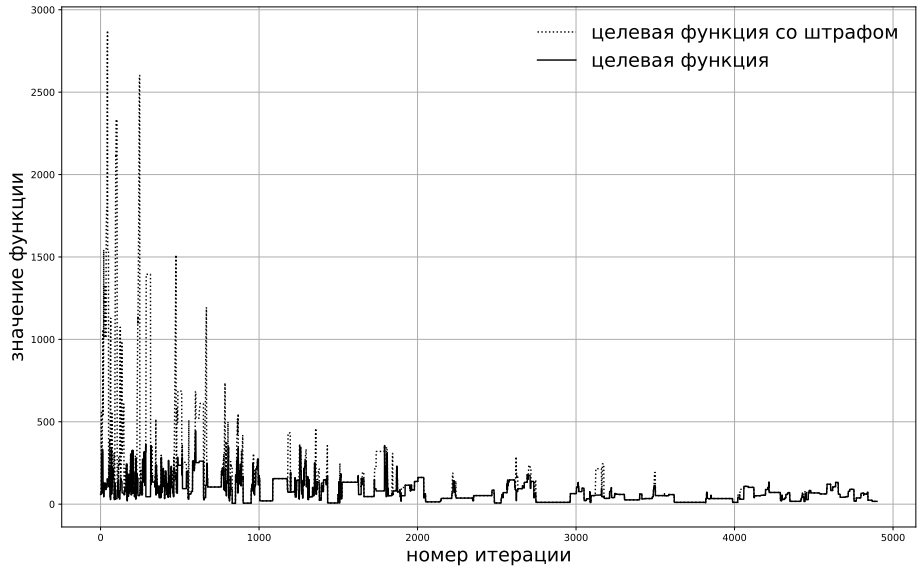

Рис. 2. Типичное поведение целевой функции со штрафом и без него

Рис. 2 показывает поведение целевой функции во время работы алгоритма имитации отжига. Пунктирная линия соответствует целевой функции со штрафом, сплошная линия — значения целевой функции без штрафа. Короткие горизонтальные промежутки на этих линиях отражают те случаи, когда алгоритм не смог перейти к соседнему решению.

Рис. 3 демонстрирует полученные оптимальные решения на примерах со 100 прямоугольниками. Интересно заметить, что иногда получаются решения, которые не совпадают с исходными гильотинными упаковками, используемыми при генерации примеров. Другими словами, задача может иметь несколько оптимальных решений и алгоритм получает одно из них, причём новой структуры. В частности, исходное гильотинное решение было получено только при $\sigma = 25$.

Заключение

Рассмотрена новая постановка двумерной задачи о рюкзаке для прямоугольников при ограничениях на расположение общего центра тяжести с поворотами предметов на 90°. Для этой NP-трудной задачи разработана модель частично целочисленного линейного программирования. Показано, что такой подход позволяет коммерческому решателю Gurobi эффективно решать задачу для входных данных с не более чем 50 предметами. Для больших размерностей разработан алгоритм имитации отжига, который получает оптимальные решения вплоть до 300 предметов.

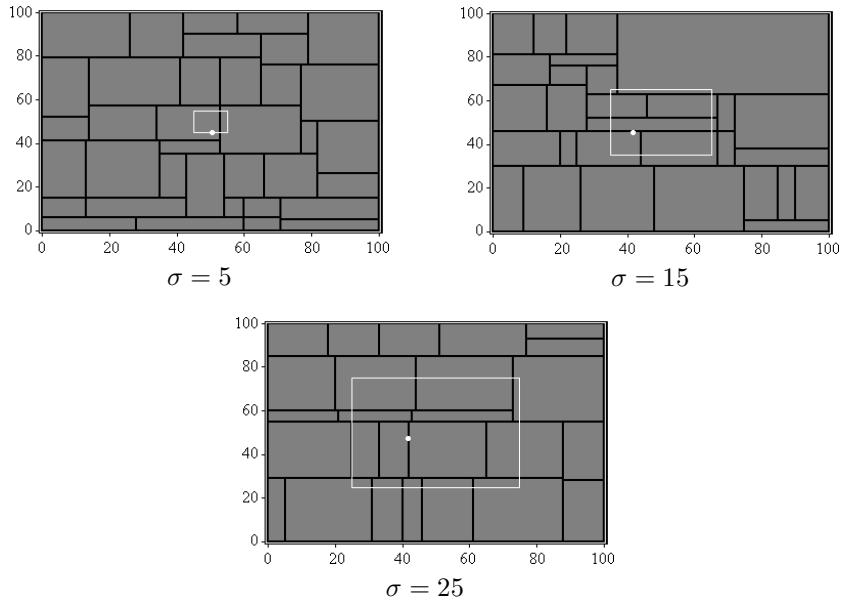

Рис. 3. Оптимальные решения при $\sigma = 5, 15, 25, 100$ прямоугольников в начальном наборе

Интересным направлением дальнейших исследований является изучение задачи упаковки в контейнеры для прямоугольников при ограничениях на расположение центра тяжести каждого контейнера. Такая задача имеет важные применения в логистике. Другим направлением дальнейших исследований является трёхмерная задача о рюкзаке или об упаковке в контейнеры. В настоящее время это направление очень востребовано на практике.

ЛИТЕРАТУРА

1. **Gallo G., Hammer P. L., Simeone B.** Quadratic knapsack problems // Math. Program. Stud. 1980. V. 12. P. 132–149.
2. **Martello S., Toth P.** Knapsack problems: Algorithms and computer implementations. Chichester: John Wiley & Sons, 1990. 296 p.
3. **Davies A. P., Bischoff E. E.** Weight distribution considerations in container loading // Eur. J. Oper. Res. 1999. V. 114, No. 3. P. 509–527.
4. **Ratcliff M. S. W., Bischoff E. E.** Allowing for weight considerations in container loading // OR Spectrum. 1998. V. 20, No. 1. P. 65–71.
5. **Baldi M. M., Perboli G., Tadei R.** The three-dimensional knapsack problem with balancing constraints // Appl. Math. Comput. 2021. V. 218. P. 9802–9818.
6. **Kaluzny B. L., Shaw R. H. A. D.** Optimal aircraft load balancing // Int. Trans. Oper. Res. 2009. V. 16, No. 6. P. 767–787.

7. **Mongeau M., Bès C.** Optimization of aircraft container loading // IEEE Trans. Aerosp. Electron. Syst. 2003. V. 39, No. 1. P. 140–150.
8. **Trivella A., Pisinger D.** The load-balanced multi-dimensional bin-packing problem // Comput. Oper. Res. 2016. V. 74. P. 152–164.
9. **Fernández A., Gil C., Baños R., Montoya M. G.** A parallel multi-objective algorithm for two-dimensional bin packing with rotations and load balancing // Expert Syst. Appl. 2013. V. 40, No. 13. P. 5169–5180.
10. **Liu D. S., Tan K. C., Huang S. Y., Goh C. K., Ho W. K.** On solving multiobjective bin packing problems using evolutionary particle swarm optimization // Eur. J. Oper. Res. 2008. V. 190, No. 2. P. 357–382.
11. **Kirkpatrick S., Gelatt C. D., Jr., Vecchi M. P.** Optimization by simulated annealing // Science. 1983. V. 220, No. 4598. P. 671–680.
12. **Dowsland K. A.** Some experiments with simulated annealing techniques for packing problems // Eur. J. Oper. Res. 1993. V. 68, No. 3. P. 389–399.
13. **Vasilyev I., Ushakov A. V., Barkova M. V., Zhang D., Ren J., Chen J.** Fast heuristic algorithms for the multiple strip packing problems // Mathematical Optimization Theory and Operations Research: Recent Trends. Rev. Sel. Pap. 20th Int. Conf. (Irkutsk, Russia, July 5–10, 2021). Cham: Springer, 2021. P. 285–297. (Commun. Comput. Inf. Sci.; V. 1476).
14. Gurobi optimizer reference manual. Beaverton: Gurobi Optimization, 2021. Available at `www.gurobi.com/documentation/9.5/refman/index.html` (accessed May 16, 2022).

Шперлинг Софья Михайловна
Кочетов Юрий Андреевич

Статья поступила
16 мая 2022 г.
После доработки —
23 мая 2022 г.
Принята к публикации
24 мая 2022 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
July–September 2022. Vol. 29, No. 3. P. 102–115

UDC 519.8+518.25 DOI 10.33048/daio.2022.29.741

A KNAPSACK PROBLEM FOR RECTANGLES UNDER THE GRAVITY CENTER CONSTRAINTS

S. M. Shperling[1, a] and *Yu. A. Kochetov*[2, b]

[1] Novosibirsk State University,
2 Pirogov Street, 630090 Novosibirsk, Russia
[2] Sobolev Institute of Mathematics,
4 Acad. Koptyug Avenue, 630090 Novosibirsk, Russia
E-mail: [a] s.shperling@g.nsu.ru, [b] jkochet@math.nsc.ru

Abstract. We have a set of rectangles with pre-defined widths, lengths, and masses and a knapsack with known width and length. Our goal is to select a subset of items and find their packing into the knapsack without overlapping to minimize the total empty space in the knapsack, while deviation of the total gravity center of such items from the geometric center of the knapsack does not exceed some threshold in both axes. We use the item permutations to represent the solutions and the skyline heuristic as a decoding procedure. The gravity center constraint is relaxed and included into the objective function with a penalty. To find the best permutation, we apply the simulated annealing algorithm with swap neighborhood and a special rule for returning into the feasible domain. Computational results for test instances with known optimal solutions are discussed. Tab. 2, illustr. 3, bibliogr. 14.

Keywords: 2D packing, skyline heuristic, gravity center constraint, local search.

REFERENCES

1. **G. Gallo, P. L. Hammer,** and **B. Simeone,** Quadratic knapsack problems, *Math. Program. Stud.* **12**, 132–149 (1980).
2. **S. Martello** and **P. Toth,** *Knapsack Problems: Algorithms and Computer Implementations* (John Wiley & Sons, Chichester, 1990).
3. **A. P. Davies** and **E. E. Bischoff,** Weight distribution considerations in container loading, *Eur. J. Oper. Res.* **114** (3), 509–527 (1999).

The research is carried out within the framework of the state contract of the Sobolev Institute of Mathematics (Project FWNF–2022–0019) .

English version: Journal of Applied and Industrial Mathematics **16** (3) (2022).

4. **M. S. W. Ratcliff** and **E. E. Bischoff,** Allowing for weight considerations in container loading, *OR Spectrum* **20** (1), 65–71 (1998).
5. **M. M. Baldi, G. Perboli,** and **R. Tadei,** The three-dimensional knapsack problem with balancing constraints, *Appl. Math. Comput.* **218**, 9802–9818 (2021).
6. **B. L. Kaluzny** and **R. H. A. D. Shaw,** Optimal aircraft load balancing, *Int. Trans. Oper. Res.* **16** (6), 767–787 (2009).
7. **M. Mongeau** and **C. Bés,** Optimization of aircraft container loading, *IEEE Trans. Aerosp. Electron. Syst.* **39** (1), 140–150 (2003).
8. **A. Trivella** and **D. Pisinger,** The load-balanced multi-dimensional bin-packing problem, *Comput. Oper. Res.* **74**, 152–164 (2016).
9. **A. Fernández, C. Gil, R. Baños** and **M. G. Montoya,** A parallel multi-objective algorithm for two-dimensional bin packing with rotations and load balancing, *Expert Syst. Appl.* **40** (13), 5169–5180 (2013).
10. **D. S. Liu, K. C. Tan, S. Y. Huang, C. K. Goh,** and **W. K. Ho,** On solving multiobjective bin packing problems using evolutionary particle swarm optimization, *Eur. J. Oper. Res.* **190** (2), 357–382 (2008).
11. **S. Kirkpatrick, C. D. Gelatt, Jr.,** and **M. P. Vecchi,** Optimization by simulated annealing, *Science* **220** (4598), 671–680 (1983).
12. **K. A. Dowsland,** Some experiments with simulated annealing techniques for packing problems, *Eur. J. Oper. Res.* **68** (3), 389–399 (1993).
13. **I. Vasilyev, A. V. Ushakov, M. V. Barkova, D. Zhang, J. Ren,** and **J. Chen,** Fast heuristic algorithms for the multiple strip packing problems, in *Mathematical Optimization Theory and Operations Research: Recent Trends* (Rev. Sel. Pap. 20th Int. Conf., Irkutsk, Russia, July 5–10, 2021) (Springer, Cham, 2021), pp. 285–297 (Commun. Comput. Inf. Sci., Vol. 1476).
14. Gurobi Optimizer Reference Manual (Gurobi Optimization, Beaverton, 2021). Available at `www.gurobi.com/documentation/9.5/refman/index.html` (accessed May 16, 2022).

Sofia M. Shperling
Yury A. Kochetov

Received May 16, 2022
Revised May 23, 2022
Accepted May 24, 2022