Специализированный алгоритм ветвей и границ для обобщённой задачи коммивояжера с ограничениями предшествования

М.Ю. Хачай, С.С. Уколов, А.А. Петунин

28 октября 2021 г.

Аннотация

Обобщенная задача коммивояжера (Generalized Traveling Salesman Problem, GTSP) – это хорошо известная задача комбинаторной оптимизации, имеющая множество важных практических приложений при исследовании операций. В GTSP с ограничением предшествования (PCGTSP) накладываются дополнительные ограничения на порядок посещения кластеров в соответствии с некоторым заранее заданным частичным порядком. В отличие от классической GTSP, PCGTSP все еще слабо исследована с точки зрения разработки и реализации алгоритмов. Насколько нам известно, все известные алгоритмические подходы для этой проблемы исчерпываются общей структурой ветвления Салмана, несколькими моделями MILP и недавно предложенной авторами метаэвристикой PCGLNS. В данной работе мы представляем первый специализированный алгоритм ветвей и границ, разработанный с расширением подхода Салмана и использующий PCGLNS в качестве мощной первичной эвристики. Используя общедоступную тестовую библиотеку PCGTSPLIB, мы оцениваем производительность предложенного алгоритма по сравнению с классической схемой динамического программирования Хелда-Карпа, дополненный стратегией ветвления и границ, и современным решателем Gurobi, использующим нашу недавнюю моделью MILP и горячий старт на основе PCGLNS.

1 Введение

Обобщенная задача коммивояжера (GTSP) – это хорошо известная задача комбинаторной оптимизации, представленная в основополагающей статье [1] Сриваставы и др. и привлекшая внимание многих исследователей (см. обзор в [2]).

В GTSP для данного взвешенного орграфа $G = (V, E, c)$ и разбиения $V_1 \cup \ldots \cup V_m$ набора узлов V на непустые взаимно непересекающиеся кластеры требуется найти замкнутый тур с минимальной стоимостью T, который посещает каждый кластер V_i ровно один раз.

В этой статье мы рассматриваем обобщенную задачу коммивояжера с ограничением предшествования (PCGTSP), в которой кластеры следует посещать в соответствии с некоторым заданным частичным порядком. Эта расширенная версия GTSP имеет множество практический применений, включая

- оптимизация траектории инструмента для станков с числовым программным управлением (ЧПУ) [3]
- минимизация времени *холостого хода* при раскрое листового металла [4; 5]
- координатно-измерительное оборудование [6]
- оптимизация траектории при многоствольном бурении [7].

1.1 Связанные работы

GTSP - это расширение классической задачи коммивояжера (TSP). Следовательно, если оценивать размер задачи количеством кластеров m, задача становится NP-сложной даже на евклидовой плоскости [8]. С другой стороны, хорошо известная схема динамического программирования Хелда и Карпа [9], адаптированная к GTSP, имеет временную сложность $O(n^3 m^2 \cdot 2^m)$, то есть этот алгоритм принадлежит FPT, будучи параметризован количеством кластеров. Следовательно, оптимальное решение GTSP может быть найдено за полиномиальное время при условии $m = O(\log n)$.

Обзор литературы показывает, что алгоритмическое проектирование GTSP развивалось по нескольким направлениям.

Первый подход основан на сведении исходной задачи к некоторой задаче асимметричной TSP, после чего этот вспомогательный экземпляр может быть решен с помощью алгоритмов, разработанного для ATSP ([10; 11]). Несмотря на математическую элегантность, этот подход страдает несколькими недостатками:

1. полученные экземпляры ATSP устроены довольно необычно, что затрудняет их решение даже для современных решателей MIP, таких как Gurobi и CPLEX.

2. близкие к оптимальным решения задачи ATSP могут соответствовать недопустимым решениям исходной задачи [12].

Другой подход связан с разработкой точных алгоритмов для частных случаев и алгоритмов аппроксимации с теоретическими гарантиями производительности. Среди них есть алгоритмы ветвей и границ и ветвей и разрезов (см., например, [13; 14]) и приближенные схемы полиномиального времени (PTAS) для некоторых специальных случаев [15; 16].

Наконец, третий подход заключается в разработке различных эвристик и метаэвристик. Так, Г. Гутин и Д. Карапетян [17] предложили эффективный меметический алгоритм, в [18] знаменитый эвристический решатель Лина-Кернигана-Хельсгауна был расширен до GTSP, а в [19] была разработана мощная метаэвристика Adaptive Large Neighborhood Search (ALNS), которая на сегодняшний день является наиболее эффективной.

К сожалению, в случае PCGTSP алгоритмические результаты все еще остаются довольно малочисленными. Насколько нам известно, в открытых источниках доступны только

1. эффективные алгоритмы для специальных ограничений предшествования типа Баласа [20—22] и ограничения предшествования, приводящие к квази- и псевдопирамидальным оптимальным обходам [23; 24]

2. общие идеи о специализированном (PCGTSP) алгоритме ветвей и границ [25]

3. недавно разработанный авторами данной статьи метаэвристический солвер PCGLNS [26; 27], развивающий результаты, полученные в [19] для GTSP.

В этой статье мы пытаемся восполнить этот пробел.

1.2 Новизна данной работы

– расширяя идею, предложенную в [25], мы разрабатываем и реализуем первый специализированный алгоритм для PCGTSP

– расширяя классический подход к ветвлению [28], мы реализуем схему динамического программирования Хелда и Карпа, дополненную оригинальной ограничивающей стратегией

– проведенные численные эксперименты показывают, что производительность предложенных алгоритмов сравнима как в смысле скорости, так и точности получаемых решений с современным решателем Gurobi, использующим лучшую в настоящее время MILP-модель и стартовое решение MIP

2 Постановка задачи

Мы рассматриваем общую постановку обобщённой задачи коммивояжера с ограничениями предшествования (PCGTSP). Задача определяется тройкой $(G, \mathcal{C}, \Pi)$, где

– взвешенный ориентированный граф $G = (V, E, c)$ определяет веса $c(u, v)$ для всех путей $(u, v) \in E$

– разбиение $\mathcal{C} = \{V_1, \ldots, V_m\}$ делит множество вершин V графа G на m непустых попарно непересекающихся *кластеров*

– ориентированный ациклический граф $\Pi = (\mathcal{C}, A)$ определяет частичный порядок (*ограничения предшествования*) на множестве кластеров $\mathcal{C}$.

Для каждой вершины $v \in V$, за $V(v)$ обозначим (единственный) кластер $V_p \in \mathcal{C}$, такой что $v \in V_p$. Далее, без ограничения общности, полагаем Π *транзитивно замкнутым* (то есть из $(V_i, V_j) \in A$ и $(V_j, V_k) \in A$ следует $(V_i, V_k) \in A$) и что $(V_1, V_p) \in A$ для каждого $p \in \{2, \ldots, m\}$.

замкнутый m-тур T называется *допустимым* решением задачи PCGTSP, если он

– начинается и заканчивается в некоторой вершине $v_1 \in V_1$

– посещает каждый кластер $V_p \in \mathcal{C}$

– каждое ребро (v_i, v_j) в T (кроме ребра (v_m, v_1)) удовлетворяет ограничению предшествования, то есть $(V(v_i), V(v_j)) \in A$.

Каждому решению $T = v_1, v_2, \ldots, v_m$, мы назначаем стоимость

$$cost(T) = c(v_m, v_1) + \sum_{i=1}^{m-1} c(v_i, v_{i+1}).$$

Требуется найти допустимый тур T с минимальной стоимостью $cost(T)$.

3 Общие соображения

Оба алгоритма, разработанные и реализованные в данной работе, используют общие основные идеи.

3.1 Разбиение задачи

В каждой вершине дерева поиска, мы разделяем исходную задачу на две (более простых) подзадачи следующим образом:

1. Рассмотрим подмножество $\mathcal{C}' \subset \mathcal{C}$, такое что $V_1 \in \mathcal{C}'$, зафиксируем некоторый кластер $V_l \in \mathcal{C}'$ и вершины $v \in V_1$ и $u \in V_l$ соответственно

2. пусть $c_{\min}$ – нижняя граница стоимости $v - u$-путей, проходящих через все кластеры $\mathcal{C}'$ и удовлетворяющих ограничению предшествования[1]

3. исключая из $\mathcal{C}'$ все внутренние кластеры и соединяя V_1 с V_l напрямую ребром нулевого (0) веса, мы тем самым создаём подзадачу $\mathcal{P}$, имеющую все те же остальные веса путей, разбиение на кластеры и ограничения предшествования, что и исходная

4. принимая
$$\mathrm{LB} = c_{\min} + \mathrm{OPT}(\mathcal{P}_{rel}) \tag{1}$$
за нижнюю границу, мы отсекаем все узлы, для которых $\mathrm{LB} > \mathrm{UB}$. Здесь, $\mathrm{OPT}(\mathcal{P}_{rel})$ это вес некоторого эффективно находимого решения упрощённой задачи $\mathcal{P}$ и UB – стоимость наилучшего известного допустимого решения исходной задачи.

3.2 Нижние границы

В этом разделе мы сравним разные способы получения нижних границ для вспомогательной проблемы $\mathcal{P}$. Рассмотрим несколько способов упростить $\mathcal{P}$, используя двухэтапный подход, предложенный в [25].

На первом этапе мы упрощаем $\mathcal{P}$, превращая ее в задачу ATSP одним из следующих способов:

1. ослабляя исходное ограничение предшествования, исключаем все ребра $(v', v'') \in E$, для которых $(V(v''), V(v')) \in A$. Затем, сводим полученную задачу к ATSP, используя классическую трансформацию Нуна-Бина [11]

2. тем же способом ослабив исходное ограничение предшествования, сводим полученную задачу к ATSP, определённой на вспомогательном графе *кластеров* $H_1 = (\tilde{\mathcal{C}}', A_1, c_1)$, где

$$\tilde{\mathcal{C}}' = \mathcal{C} \setminus \mathcal{C}' \cup \{V_1, V_l\},$$

$$A_1 = \{(V_1, V_l)\} \cup \{(V_i, V_j) \mid i > 2, \{V_i, V_j\} \subset \tilde{\mathcal{C}}', \exists (v' \in V_i, v'' \in V_j) \colon (v', v'') \in E\},$$

$$c_1(V_1, V_l) = 0,\ c_1(V_i, V_j) = \min\{c(v', v'') \colon v' \in V_i, v'' \in V_j, (v', v'') \in E\}$$

[1] В нашем варианте динамического программирования эта граница будет точной

3. сводим исходную задачу к ATSP, определённой на ориентированном графе $H_2 = (\tilde{\mathcal{C}}', A_2, c_2)$, где

$$\begin{aligned} A_2 = \{(V_1, V_l)\} \cup \{(V_i, V_k) \mid i > 2,\\ \exists (j > 1) : \{V_i, V_j, V_k\} \subset \tilde{\mathcal{C}}' \wedge (\{(V_j, V_i), (V_k, V_j), (V_k, V_i)\} \cap A = \varnothing)\\ \wedge \exists (v' \in V_i, v'' \in V_j, v''' \in V_k) \colon (\{(v', v''), (v'', v''')\} \subset E)\}\\ \cup \{(V_i, V_k) \mid i > 2, (\{V_i, V_k\} \subset \tilde{\mathcal{C}}') \wedge ((V_k, V_i) \notin A)\\ \wedge \exists (v' \in V_i, v_1 \in V_1, v'' \in V_k) \colon \{(v', v_1), (v_1, v'')\} \subset E\}, \end{aligned}$$

то есть, для любого $V_i \in \tilde{\mathcal{C}}' \setminus \{V_1\}$, упорядоченная пара $(V_i, V_k) \in A_2$, если существует $V_j \in \tilde{\mathcal{C}}'$ и вершины $v' \in V_i, v'' \in V_j$ и $v''' \in V_k$, такие, что путь $\pi = v', v'', v'''$ не запрещён исходным ограничением предшествования Π. Далее,

$$c_2(V_1, V_l) = 0,\ c_2(V_i, V_k) = \min\{c(v', v'') + c(v'', v''') \colon \pi = v', v'', v''' \text{ – допустимый путь}\}.$$

На втором этапе, мы находим приближенное решение полученной задачи ATSP, путем нахождения либо минимального остовного дерева (Minimum Spanning Arborescence Problem, MSAP), либо решения задачи о назначениях (Assignment Problem, AP) и тем самым получаем значение нижней границы по формуле (1). Кроме того, мы можем посчитать ещё более точную нижнюю границу, прямо решая вспомогательную задачу ATSP при помощи солвера Gurobi (на практике только для задач, полученных способом 2). Для удобства все способы получения нижних границ сведены в табл. 1(a).

Таблица 1: Нижние границы

	Noon-Bean	$\mathbf{H_1}$	$\mathbf{H_2}$
AP	E_1	$\mathbf{L_1}$	$\mathbf{L_2}$
MSAP	E_2	E_3	E_4
Gurobi	E_5	$\mathbf{L_3}$	E_6

(a) Обозначения

E_1	$E_2 = E_3$	E_4
0.48 ± 0.03	0.54 ± 0.01	0.60 ± 0.002
L_1	L_2	L_3
0.91 ± 0.02	0.97 ± 0.02	1.00

(b) Оценки по сравнению с L_3

На основе результатов численных экспериментов, мы сократили полный список методов оценки нижней границы, см. табл. 1(b). На практике оценки L_1–L_3 оказываются почти всегда наиболее строгими, статистически значимо с доверительным интервалом 95%. Мы также отказались от использования оценок E_5 и E_6 ввиду того, что они требуют гораздо большего времени счета. Таким образом, в разделе 6 мы ограничились использованием оценок

$$\mathrm{LB}_i = c_{\min} + L_i,\ i \in \{1, 2, 3\}.$$

4 Алгоритм ветвей и границ

Для решения задачи PCGTSP $(G, \mathcal{C}, \Pi)$, мы обходим дерево поиска в ширину (Breadth First Search), см. Алгоритм 1. Каждый узел этого дерева связан с префиксом $\sigma = (V_{i_1}, V_{i_2}, \dots V_{i_r})$, где $V_{i_j} \in \mathcal{C}$, $V_{i_1} = V_1$, и $r \in \{1, \dots m\}$.

К каждому узлу дерева поиска мы применяем процедуру отсечения Bound (Алгоритм 2), которая выполняет следующие действия:

– для префикса σ, мы находим кортеж

$$\mathcal{T}(\sigma) = (V_{i_1}, \{V_{i_1}, V_{i_2}, \dots V_{i_r}\}, V_{i_r})$$

– на шаге 4, мы вычисляем матрицу $D(\sigma)$ минимальных попарных весов по формуле:

$$D(\sigma)_{vu} = \min\left\{cost(P_{v,u}) \colon v \in V_{i_1}, u \in V_{i_r}, P_{v,u} \text{ путь } v\text{-}u \text{ в порядке } \sigma\right\}.$$

Эта матрица удобно вычисляется инкрементально на основе матрицы $D(\sigma')$ родительского узла дерева поиска

– если, для некоторого σ_1, $\mathcal{T}(\sigma) = \mathcal{T}(\sigma_1)$ и

$$D(\sigma)_{vu} \geqslant D(\sigma_1)_{vu}, \quad (v \in V_{i_1}, u \in V_{i_r}),$$

то, префикс σ имеет веса в матрице$D(\sigma)$ больше, чем для префикса σ_1 и подлежит отсечению

Алгоритм 1 BnB :: Главная процедура

Вход: орграф G, кластеры $\mathcal{C}$, частичный порядок Π
Выход: маршрут и его стоимость

инициализация Q = empty queue
начинаем с $Root = V_1$
Q.push($Root$)
while not Q.empty() **do**
  берём следующий префикс для обработки: $\sigma = Q$.pop()
  $process = Bound(\sigma)$
  if not $process$ **then**
    префикс отсекается; **continue**
  end if
  $UpdateLowerBound(\sigma)$
  for all $child \in Branch(\sigma)$ **do**
    помещаем префикс в очередь на обработку Q.push($child$)
  end for
end while

Алгоритм 2 BnB :: Bound

Вход: префикс σ
Выход: признак того, что префикс подлежит обработке

global $D_{ij}^{\mathcal{T}}$
global $Opt^{\mathcal{T}}$
вычисляем кортеж $\mathcal{T} = (V_{i_1}, \{V_{i_1}, V_{i_2}, \dots V_{i_r}\}, V_{i_r})$
$D_{ij} = MinCosts(\sigma)$
if $D_{ij}^{(\sigma)} \geqslant D_{ij}^{\mathcal{T}}[\mathcal{T}], \forall i, j$ **then**
  return false
end if
обновляем веса маршрутов $D_{ij}^{\mathcal{T}}[\mathcal{T}] = \min\left(D_{ij}^{\mathcal{T}}[\mathcal{T}], D_{ij}\right), \forall i, j$
$c_{min} = \min\limits_{i,j} D_{ij}$
if $\mathcal{T} \notin Opt^{\mathcal{T}}$ **then**
  вычисляем нижнюю границу $Opt^{\mathcal{T}}[\mathcal{T}] = \max\left(L_1(\sigma), L_2(\sigma)\right))$
end if
LB $= c_{min} + Opt^{\mathcal{T}}[\mathcal{T}]$
if LB $>$ UB **then**
  return false
end if
return true

– на шаге 11, мы рассчитываем оценки L_1 и L_2, см. табл. 1 и сохраняем их в глобальной переменной $Opt^{\mathcal{T}}$, используя формулу

$$Opt^{\mathcal{T}(\sigma)} = \max(L_1, L_2)$$

– для текущего узла σ, рассчитываем нижнюю границу по формуле

$$\mathrm{LB}(\sigma) = \min_{vu} D(\sigma)_{vu} + Opt^{\mathcal{T}(\sigma)}$$

на шаге 13

– наконец, узел σ отсекается, если LB > UB.

Алгоритм 3 BnB :: Branch

Вход: префикс σ
Выход: список потомков префикса для обработки

```
1: инициализация R = empty queue
2: for all V ∈ C do
3:    valid = true
4:    for all W ∈ σ do
5:       if W = V or (V, W) ∈ Π then
6:          valid = false
7:          break
8:       end if
9:    end for
10:   if valid then
11:      добавляем новый префикс R.push(σ + V)
12:   end if
13: end for
14: return R
```

Префиксы, которые избежали отсечения, обрабатываются процедурой *Branch* (Алгоритм 3), которая пытается удлинить префикс σ на один кластер, соблюдая при этом ограничение предшествования Π.

5 Динамическое программирование

Алгоритм ветвей и границ, описанный в разделе 4, оказывается сильно связан с классической схемой, использующей динамическое программирование (DP) и носящей имя Хелда-Карпа [9], адаптированной для учёта ограничения предшествования и дополненной стратегией отсечения, представленной в основополагающей статье [28].

В данной работе мы реализуем уточненную версию этой схемы для численной оценки производительности нашего алгоритма ветвей и границ. Подобно классическому DP, наш алгоритм состоит из двух этапов.

1. На этом этапе таблица поиска строится инкрементально, в прямом направлении, слой за слоем. Оптимальная стоимость для решаемой задачи находится после вычисления последнего m-го слоя.

2. Оптимальный маршрут реконструируется обратным просмотром на основе данных, хранящихся в таблице поиска.

Каждое состояние DP (запись в таблице поиска) соответствует частичному v-u-пути и индексируется кортежем $(\mathcal{C}', V_l, v, u)$, где

1. $\mathcal{C}' \subset \mathcal{C}$ представляет собой *идеал* частично упорядоченного множества кластеров $\mathcal{C}$, то есть

$$\forall(V \in \mathcal{C}', V' \in \mathcal{C})\ (V', V) \in A) \Rightarrow (V' \in \mathcal{C}');$$

очевидно, в наших условиях, V_1 принадлежит произвольному идеалу $\mathcal{C}' \subset \mathcal{C}$

2. $V_l \subset \mathcal{C}'$, для которого нет $V \in \mathcal{C}'$, такого, что $(V_l, V) \in A$

Алгоритм 4 DP :: индуктивное построение таблицы поиска

Вход: орграф G, частичный порядок П, слой таблицы поиска $\mathcal{L}_k$, верхняя граница UB
Выход: $(k+1)$-ый слой $\mathcal{L}_{k+1}$

1: инициализация $\mathcal{L}_{k+1} = \varnothing$
2: **for all** $\mathcal{C}' \in \mathfrak{I}_k$ **do**
3: **for all** кластер $V_l \in \mathcal{C} \setminus \mathcal{C}'$, s.t. $\mathcal{C}' \cup \{V_l\} \in \mathfrak{I}_{k+1}$ **do**
4: **for all** $v \in V_1$ и $u \in V_l$ **do**
5: **if** есть состояние $S = (\mathcal{C}', U, v, w) \in \mathcal{L}_k$, s.t. $(w, u) \in E$ **then**
6: создаем новое состояние $S' = (\mathcal{C}' \cup \{V_l\}, V_l, v, u)$
7: $S'[cost] = \min\{S[cost] + c(w, u) \colon S = (\mathcal{C}', U, v, w) \in \mathcal{L}_k\}$
8: $S'[pred] = \arg\min\{S[cost] + c(w, u) \colon S = (\mathcal{C}', U, v, w) \in \mathcal{L}_k\}$
9: $S'[LB] = S'[cost] + \max\{L_1, L_2, L_3\}$
10: **if** $S'[\text{LB}] \leqslant \text{UB}$ **then**
11: добавляем S' к $\mathcal{L}_{k+1}$
12: **end if**
13: **end if**
14: **end for**
15: **end for**
16: **end for**
17: **return** $\mathcal{L}_{k+1}$

3. $v \in V_1$, $u \in V_l$.

Содержимое каждой записи DP S состоит из ссылки $S[pred]$ на предшествующее состояние, локальной нижней границы $S[LB]$ и стоимости $S[cost]$ соответствующего частичного v-u-пути.

Пусть $\mathfrak{I}_k$ – подмножество идеалов одного размера $k \in \{1, \ldots, m\}$. Очевидно, $\mathfrak{I}_1 = \{\{V_1\}\}$, а значит первый слой $\mathcal{L}_1$ таблицы поиска строится тривиально. Индуктивное построение остальных слоев описано в Алгоритме 4.

5.1 Замечания

1. Оптимум для решаемой задачи дается классическим уравнением Беллмана

$$\text{OPT} = \min_{v \in V_1} \min\{S[cost] + c(u, v) \colon S = (\mathcal{C}', V_l, v, u) \in \mathcal{L}_m\}$$

2. По построению, размер таблицы поиска $O(n^2 m \cdot |\mathfrak{I}|)$. Значит, время работы нашего алгоритма $O(n^3 m^2 \cdot |\mathfrak{I}|)$. В частности, в случае частичного порядка фиксированной *ширины* w, $|\mathfrak{I}| = O(m^w)$ [29]. Следовательно, оптимальное решение PCGTSP может быть найдено в этом случае за полиномиальное время даже без применения отсечения на шагах 10–12.

3. После построения любого из слоев $\mathcal{L}_k$, мы обновляем глобальное значение нижней границы, что приводит к сокращению общего разрыва.

4. В нашей реализации, для повышения быстродействия мы вычисляем оценку L_3 на шаге 9 только для небольшого количества состояний с наименьшей нижней границей.

6 Численные эксперименты

В данном разделе приводятся результаты численных экспериментов по оценке производительности предлагаемого алгоритма ветвей и границ в сравнении со схемой динамического программирования а также решателем Gurobi, использующим нашей недавней MILP-моделью [26].

6.1 Условия эксперимента

Все алгоритмы тестировались на общедоступной библиотеке PCGTSPLIB [25]. Во всех случаях для теплого старта, всем алгоритмам предоставляется одно и то же допустимое решение, полученное эвристическим решателем PCGLNS [27]. Для алгоритмов ветвей и границ и динамического программирования, все

Таблица 2: Результаты экспериментов

Задача					*Gurobi*			*Ветвей и границ*			*DP*		
№	*ID*	n	m	UB_0	*Время*	*LB*	*gap, %*	*Время*	*LB*	*gap, %*	*Время*	*LB*	*gap, %*
1	br17.12	92	17	43	82.00	43	0.00	**11.2**	**43**	**0.00**	27.3	43	0.00
2	ESC07	39	8	1730	0.24	1730	0.00	1.3	1726	0.23	8.37	1730	0.00
3	ESC12	65	13	1390	3.35	1390	0.00	4.3	1385	0.36	14.99	1390	0.00
4	ESC25	133	26	1418	10.61	1383	0.00	32	1383	0.00	60.69	1383	0.00
5	ESC47	244	48	1399	3773	1064	4.93	36000	980	42.76	36000	981	42.61
6	ESC63	349	64	62	25.35	62	0.00	1.3	62	0.00	**0.52**	**62**	**0.00**
7	ESC78	414	79	14872	1278.45	14630	1.66	1.3	14594	1.63	**0.68**	**14594**	**1.63**
8	ft53.1	281	53	6194	36000	5479	13.04	36000	4839	28.27	36000	4839	28.27
9	ft53.2	274	53	6653	36000	5511	20.7	36000	4934	34.84	36000	4940	34.68
10	ft53.3	281	53	8446	36000	6354	32.92	36000	5465	54.55	36000	5465	54.55
11	ft53.4	275	53	11822	20635	11259	5.00	35865	11274	4.86	**2225**	**11290**	**4.71**
12	ft70.1	346	70	32848	83.70	31521	4.21	36000	31153	5.44	36000	31177	5.36
13	ft70.2	351	70	33486	36000	31787	5.35	36000	31268	7.09	36000	31273	7.08
14	ft70.3	347	70	35309	36000	32775	7.73	36000	32180	9.72	36000	32180	9.72
15	ft70.4	353	70	44497	36000	41160	8.11	36000	38989	14.13	**36000**	**41640**	**6.86**
16	kro124p.1	514	100	33320	36000	29541	12.79	36000	27869	19.56	36000	27943	19.24
17	kro124p.2	524	100	35321	36000	29983	17.80	36000	28155	25.45	36000	28155	25.45
18	kro124p.3	534	100	41340	36000	30669	34.79	36000	28406	45.53	36000	28406	45.53
19	kro124p.4	526	100	62818	36000	46033	36.46	36000	38137	64.72	36000	38511	63.12
20	p43.1	203	43	22545	4691	21677	4.00	36000	738	2954.88	36000	788	2761.04
21	p43.2	198	43	22841	36000	21357	6.94	36000	749	2949.53	36000	877	2504.45
22	p43.3	211	43	23122	36000	15884	45.57	36000	898	2474.83	36000	906	2452.10
23	p43.4	204	43	66857	36000	45198	47.92	4470	66846	0.00	**333.02**	**66846**	**0.00**
24	prob.100	510	99	1474	36000	805	83.10	36000	632	133.23	36000	632	133.23
25	prob.42	208	41	232	13310	196	4.86	36000	149	55.70	36000	153	51.63
26	rbg048a	255	49	282	24.22	282	0.00	0.9	272	3.68	**0.25**	**272**	**3.68**
27	rbg050c	259	51	378	13.83	378	0.00	**0.2**	**372**	**1.61**	0.25	372	1.61
28	rbg109a	573	110	848	6	848	0.00	2407	812	4.43	682	809	4.82
29	rbg150a	871	151	1415	15	1382	2.38	**0.4**	**1353**	**4.58**	0.53	1353	4.58
30	rbg174a	962	175	1644	27	1605	2.43	**0.4**	**1568**	**4.85**	0.67	1568	4.85
31	rbg253a	1389	254	2376	61	2307	2.99	**0.8**	**2269**	**4.72**	1.42	2269	4.72
32	rbg323a	1825	324	2547	416	2490	2.29	**2.0**	**2448**	**4.04**	3.59	2448	4.04
33	rbg341a	1822	342	2101	18470	2033	4.97	36000	1840	14.18	36000	1840	14.18
34	rbg358a	1967	359	2080	17807	1982	4.95	36000	1933	7.60	36000	1933	7.60
35	rbg378a	1973	379	2307	32205	2199	4.91	36000	2032	13.53	36000	2031	13.59
36	ry48p.1	256	48	13135	36000	11965	9.78	36000	10739	22.31	36000	10764	22.03
37	ry48p.2	250	48	13802	36000	12065	14.39	36000	10912	26.48	36000	11000	25.47
38	ry48p.3	254	48	16540	36000	13085	26.40	36000	11732	40.98	36000	11822	39.91
39	ry48p.4	249	48	25977	36000	22084	17.62	18677	25037	3.75	**14001**	**25043**	**3.73**

вычисления проводятся на одном и том же оборудовании (16-ядерный Intel Xeon, 128G RAM) с предельным временем счета 10 часов. В качестве критерия остановки мы используем понижение разрыва ниже 5%, где разрыв определяется по формуле

$$gap = \frac{\mathrm{UB} - \mathrm{LB}}{\mathrm{LB}}. \tag{2}$$

В качестве базы сравнения, мы воспроизвели численные эксперименты, представленные в [26] в условиях, описанных выше, включая время счёта 10 часов и критерий остановки (2).

Исходный код предложенных алгоритмов и вспомогательные скрипты доступны в [30].

6.2 Обсуждение

Полученные результаты эксперимента представлены в табл. 2, которая организована следующим образом: первая группа столбцов описывает задачу, включая её обозначение ID, количество вершин n и кластеров m, а также стоимость стартового решения UB_0, полученного эвристикой PCGLNS. Затем следуют три группы столбцов для решателя Gurobi и двух предлагаемых алгоритмов. Каждая группа содержит время счета в секундах, наилучшее значение нижней границы LB и наилучший разрыв gap в процентах. Задачи, в которых один из предлагаемых алгоритмов сработал лучше Gurobi, выделены жирным шрифтом.

Как следует из табл. 2, для 13 из 39 задач (33%) один из наших алгоритмов показал лучшую производительность. Из них в 12 случаях лучше время счёта, а в 7 – точность.

Заметим, что предложенные алгоритмы смогли найти оптимальное решение в 6 из 39 случаях (хотя это не было целью эксперимента). Для 10 (15) задач, включая одни из самых больших *rbg323a* и *rbg358a* (1825 и 1967 вершин соответственно) было получено решение с точностью 5% (10%).

С другой стороны, для некоторых задач (например, *p43.1, p43.2* и *p43.3*), результаты наших алгоритмов оказались крайне слабы по сравнению с Gurobi, что по видимому объясняется очень грубыми оценками нижней границы. В то же время для задач *p43.4* и *ry48p.4* наши алгоритмы сработали гораздо лучше Gurobi.

В целом, хотя Gurobi демонстрирует в среднем чуть лучшую производительность, предложенные алгоритмы за редким исключением, показывают вполне сопоставимые результаты. Считаем нужным добавить, что в наших экспериментах решателю Gurobi было предоставлено, так же как и тестируемым алгоритмам, хорошее стартовое решение, что является не очень обычным способом организации эксперимента.

7 Заключение

В данной работе разработан и реализован первый специализированный алгоритм ветвей и границ для обобщенной задачи коммивояжера с ограничениями предшествования. Он развивает идеи классической схемы динамического программирования Хелда-Карпа и схемы Салмана.

Для оценки производительности предложенных алгоритмов, проведены численные эксперименты, в качестве базы сравнения использован решатель Gurobi. Эксперименты продемонстрировали, что наши алгоритмы вполне конкурентноспособны на уровне современных MIP-решателей.

В качестве направления дальнейших исследований мы предполагаем разработку более точных нижних оценок. Кроме того, мы полагаем, что дальнейшая оптимизация и распараллеливание могут существенно улучшить производительность наших алгоритмов.

В настоящее время разработанное программное обеспечение интегрируется с системой автоматизированного проектирования СИРИУС [31], предназначенной для оптимизация раскроя листового материала на фигурные заготовки и подготовки управляющих программ для машин листовой резки с ЧПУ.

Благодарность

Работа выполнена в ходе исследований Уральского Математического Центра при финансовой поддержке Министерства науки и высшего образования РФ, соглашение № 075-02-2021-1383.

Список литературы

1. *Srivastava S.*, *Kumar S.*, *Garg R.*, *Sen P.* Generalized Traveling Salesman Problem through n sets of nodes // CORS journal. — 1969. — Т. 7, вып. 2, № 2. — С. 97—101.

2. *Gutin G.*, *Punnen A. P.* The Traveling Salesman Problem and Its Variations. — Boston, MA : Springer US, 2007. — ISBN 978-0-387-44459-8.

3. *Castelino K.*, *D'Souza R.*, *Wright P. K.* Toolpath optimization for minimizing airtime during machining // Journal of Manufacturing Systems. — 2003. — Т. 22, № 3. — С. 173—180. — DOI: 10.1016/S0278-6125(03)90018-5. — URL: http://www.sciencedirect.com/science/article/pii/S0278612503900185.

4. *Chentsov A. G.*, *Chentsov P. A.*, *Petunin A. A.*, *Sesekin A. N.* Model of megalopolises in the tool path optimisation for CNC plate cutting machines // International Journal of Production Research. — 2018. — Т. 56, № 14. — С. 4819—4830. — URL: https://doi.org/10.1080/00207543.2017.1421784.

5. *Makarovskikh T.*, *Panyukov A.*, *Savitskiy E.* Mathematical models and routing algorithms for economical cutting tool paths // International Journal of Production Research. — 2018. — Т. 56, № 3. — С. 1171—1188. — DOI: 10.1080/00207543.2017.1401746.

6. *Salman R.*, *Carlson J. S.*, *Ekstedt F.*, *Spensieri D.*, *Torstensson J.*, *Söderberg R.* An Industrially Validated CMM Inspection Process with Sequence Constraints // Procedia CIRP. — 2016. — Т. 44. — С. 138—143. — DOI: 10.1016/j.procir.2016.02.136. — URL: http://www.sciencedirect.com/science/article/pii/S2212827116004182 ; 6th CIRP Conference on Assembly Technologies and Systems (CATS).

7. *Dewil R.*, *Küçükoğlu İ.*, *Luteyn C.*, *Cattrysse D.* A Critical Review of Multi-hole Drilling Path Optimization // Archives of Computational Methods in Engineering. — 2019. — Т. 26, № 2. — С. 449—459. — URL: https://doi.org/10.1007/s11831-018-9251-x.

8. *Papadimitriou C.* Euclidean TSP is NP-complete // Theoret. Comput. Sci. — 1977. — Т. 4, вып. 3. — С. 237—244.

9. *Held M.*, *Karp R. M.* A Dynamic Programming Approach to Sequencing Problems // Journal of the Society for Industrial and Applied Mathematics. — 1962. — Т. 10, № 1. — С. 196—210. — ISSN 03684245. — URL: http://www.jstor.org/stable/2098806.

10. *Laporte G.*, *Semet F.* Computational Evaluation Of A Transformation Procedure For The Symmetric Generalized Traveling Salesman Problem // INFOR: Information Systems and Operational Research. — 1999. — Т. 37, № 2. — С. 114—120. — DOI: 10.1080/03155986.1999.11732374.

11. *Noon C. E.*, *Bean J. C.* An Efficient Transformation Of The Generalized Traveling Salesman Problem // INFOR: Information Systems and Operational Research. — 1993. — Т. 31, № 1. — С. 39—44. — DOI: 10.1080/03155986.1993.11732212.

12. *Karapetyan D.*, *Gutin G.* Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem // European Journal of Operational Research. — 2012. — Т. 219, № 2. — С. 234—251. — ISSN 0377-2217. — DOI: 10.1016/j.ejor.2012.01.011. — URL: https://www.sciencedirect.com/science/article/pii/S0377221712000288.

13. *Fischetti M.*, *González J. J. S.*, *Toth P.* A Branch-and-Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem // Operations Research. — 1997. — Т. 45, № 3. — С. 378—394. — DOI: 10.1287/opre.45.3.378.

14. *Yuan Y.*, *Cattaruzza D.*, *Ogier M.*, *Semet F.* A branch-and-cut algorithm for the generalized traveling salesman problem with time windows // European Journal of Operational Research. — 2020. — Т. 286, № 3. — С. 849—866. — ISSN 0377-2217. — DOI: 10.1016/j.ejor.2020.04.024. — URL: https://www.sciencedirect.com/science/article/pii/S0377221720303581.

15. *Feremans C.*, *Grigoriev A.*, *Sitters R.* The geometric generalized minimum spanning tree problem with grid clustering // 4OR. — 2006. — Т. 4, № 4. — С. 319—329. — ISSN 1614-2411. — DOI: 10.1007/s10288-006-0012-6. — URL: https://doi.org/10.1007/s10288-006-0012-6.

16. *Khachai M. Y.*, *Neznakhina E. D.* Approximation Schemes for the Generalized Traveling Salesman Problem // Proceedings of the Steklov Institute of Mathematics. — 2017. — Т. 299, № 1. — С. 97—105. — ISSN 1531-8605. — DOI: 10.1134/S0081543817090127. — URL: https://doi.org/10.1134/S0081543817090127.

17. *Gutin G.*, *Karapetyan D.* A Memetic Algorithm for the Generalized Traveling Salesman Problem // Natural Computing. — 2010. — Т. 9, № 1. — С. 47—60. — DOI: 10.1007/s11047-009-9111-6.

18. *Helsgaun K.* Solving the equality Generalized Traveling Salesman Problem using the Lin–Kernighan–Helsgaun Algorithm // Mathematical Programming Computation. — 2015. — С. 1—19.

19. *Smith S. L.*, *Imeson F.* GLNS: An effective large neighborhood search heuristic for the Generalized Traveling Salesman Problem // Computers & Operations Research. — 2017. — Т. 87. — С. 1—19. — DOI: 10.1016/j.cor.2017.05.010.

20. *Balas E.*, *Simonetti N.* Linear Time Dynamic-Programming Algorithms for New Classes of Restricted TSPs: A Computational Study // INFORMS J. on Computing. — Institute for Operations Research, the Management Sciences (INFORMS), Linthicum, Maryland, USA, 2001. — Февр. — Т. 13, № 1. — С. 56—75. — ISSN 1526-5528. — DOI: 10.1287/ijoc.13.1.56.9748. — URL: http://dx.doi.org/10.1287/ijoc.13.1.56.9748.

21. *Chentsov A. G.*, *Khachai M. Y.*, *Khachai D. M.* An exact algorithm with linear complexity for a problem of visiting megalopolises // Proceedings of the Steklov Institute of Mathematics. — 2016. — Т. 295, № 1. — С. 38—46. — ISSN 1531-8605. — DOI: 10.1134/S0081543816090054. — URL: https://doi.org/10.1134/S0081543816090054.

22. *Chentsov A.*, *Khachay M.*, *Khachay D.* Linear time algorithm for Precedence Constrained Asymmetric Generalized Traveling Salesman Problem // IFAC-PapersOnLine. — 2016. — Т. 49, № 12. — С. 651—655. — ISSN 2405-8963. — DOI: https://doi.org/10.1016/j.ifacol.2016.07.767. — URL: http://www.sciencedirect.com/science/article/pii/S2405896316310485 ; 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016.

23. *Khachay M.*, *Neznakhina K.* Towards Tractability of the Euclidean Generalized Traveling Salesman Problem in Grid Clusters Defined by a Grid of Bounded Height // Optimization Problems and Their Applications. Т. 871 / под ред. A. Eremeev, M. Khachay, Y. Kochetov, P. Pardalos. — Cham : Springer International Publishing, 2018. — С. 68—77. — (Communications in Computer and Information Science). — ISBN 978-3-319-93799-1. — DOI: 10.1007/978-3-319-93800-4{_}6. — URL: https://doi.org/10.1007/978-3-319-93800-4%7B%5C_%7D6.

24. *Khachay M.*, *Neznakhina K.* Complexity and approximability of the Euclidean Generalized Traveling Salesman Problem in grid clusters // Annals of Mathematics and Artificial Intelligence. — 2020. — Т. 88, № 1. — С. 53—69. — DOI: 10.1007/s10472-019-09626-w.

25. *Salman R.*, *Ekstedt F.*, *Damaschke P.* Branch-and-bound for the Precedence Constrained Generalized Traveling Salesman Problem // Operations Research Letters. — 2020. — Т. 48, № 2. — С. 163—166. — ISSN 0167-6377. — DOI: 10.1016/j.orl.2020.01.009.

26. *Khachay M.*, *Kudriavtsev A.*, *Petunin A.* PCGLNS: A Heuristic Solver for the Precedence Constrained Generalized Traveling Salesman Problem // Optimization and Applications. Т. 12422 / под ред. N. Olenev, Y. Evtushenko, M. Khachay, V. Malkova. — Cham : Springer International Publishing, 2020. — С. 196—208. — (Lecture Notes in Computer Science). — ISBN 978-3-030-62867-3. — DOI: 10.1007/978-3-030-62867-3_15.

27. *Kudriavtsev A.*, *Khachay M.* PCGLNS: adaptive heuristic solver for the Precedence Constrained GTSP. — 2020. — URL: https://github.com/AndreiKud/PCGLNS/.

28. *Morin T. L.*, *Marsten R. E.* Branch-And-Bound Strategies for Dynamic Programming // Operations Research. — 1976. — Т. 24, № 4. — С. 611—627. — ISSN 0030364X, 15265463. — URL: http://www.jstor.org/stable/169764.

29. *Steiner G.* On the complexity of dynamic programming for sequencing problems with precedence constraints // Annals of Operations Research. — 1990. — Т. 26, № 1. — С. 103—123.

30. *Ukolov S.*, *Khachay M.* Branch-and-Bound algorithm for the Precedence Constrained GTSP. — 2021. — URL: https://github.com/ukoloff/PCGTSP-BnB.

31. *Tavaeva A.*, *Petunin A.*, *Ukolov S.*, *Krotov V.* A Cost Minimizing at Laser Cutting of Sheet Parts on CNC Machines // Mathematical Optimization Theory and Operations Research. — Cham, Switzerland : Springer, 10.2019. — С. 422—437. — ISBN 978-3-030-33393-5. — DOI: 10.1007/978-3-030-33394-2_33.