УДК 519.854
DOI: 10.15827/0236-235X.137.054-064
Дата подачи статьи: 04.10.21, после доработки: 18.11.21
2022. Т. 35. № 1. С. 054–064

Программное обеспечение для решения обобщенной задачи коммивояжера с ограничениями предшествования

А.А. Петунин [1], *д.т.н., профессор, a.a.petunin@urfu.ru, aapetunin@gmail.com*
С.С. Уколов [1], *м.н.с., s.s.ukolov@urfu.ru*
М.Ю. Хачай [2], *д.ф.-м.н., профессор, зав. отделом, mkhachay@imm.uran.ru*

[1] *Уральский федеральный университет им. первого Президента России Б.Н. Ельцина, г. Екатеринбург, 620002, Россия*
[2] *Институт математики и механики им. Н.Н. Красовского УрО РАН, г. Екатеринбург, 620108, Россия*

В статье рассматривается обобщенная задача коммивояжера с ограничениями предшествования (PCGTSP), в которой подобно классической задаче коммивояжера (TSP) ищется замкнутый цикл минимальной стоимости, при этом множество вершин разбито на непустые попарно непересекающиеся подмножества – кластеры и каждый допустимый маршрут обязан посетить каждый из кластеров в единственной вершине. Кроме того, множество допустимых маршрутов стеснено дополнительным ограничением на порядок посещения кластеров, то есть некоторые кластеры должны посещаться раньше других. Такая задача в отличие от TSP и обобщенной задачи коммивояжера (GTSP) является слабо исследованной как теоретически, так и с точки зрения проектирования и реализации алгоритмов.

В данной работе предлагаются первые специализированные алгоритмы ветвей и границ, использующие в качестве начального приближения решения, полученные при помощи недавно разработанной эвристики PCGLNS. Исходная задача PCGTSP подвергается нескольким релаксациям, в результате чего получаются несколько нижних оценок на решение исходной задачи, наибольшая из которых используется для отсечения ветвей дерева поиска и сокращения тем самым перебора. Алгоритмы реализованы в виде открытого ПО на языке программирования Python 3 с использованием специализированной библиотеки NetworkX. Производительность предложенных алгоритмов оценивается на тестовых примерах из общедоступной библиотеки PCGTSPLIB в сравнении с общеизвестным солвером Gurobi, использующим недавно предложенную авторами модель MILP, и представляется вполне конкурентоспособной даже в текущей реализации.

Разработанные алгоритмы могут применяться в широком классе практических задач, например, для оптимальной маршрутизации инструмента машин листовой резки с ЧПУ, а также для оценки качества решений, получаемых другими методами.

Ключевые слова*: GTSP, ограничения предшествования, метод ветвей и границ, динамическое программирование, схема Хелда–Карпа.*

Программное обеспечение для решения многих задач дискретной оптимизации основано на математической модели обобщенной задачи коммивояжера (Generalized Traveling Salesman Problem, GTSP), одной из самых известных задач комбинаторной оптимизации, впервые сформулированной в [1] и заинтересовавшей многих исследователей (например [2]). В GTSP для заданного взвешенного орграфа $G = (V, E, c)$ и разбиения $V_1 \cup \ldots \cup V_m$ набора узлов V графа G на непустые взаимно непересекающиеся кластеры требуется найти замкнутый тур с минимальной стоимостью, который посещает каждый кластер V_i точно один раз.

В данной статье рассматривается обобщенная задача коммивояжера с ограничениями предшествования (Precedence Constrained Generalized Traveling Salesman Problem, PCGTSP), каждому допустимому маршруту которой необходимо посещать кластеры в соответствии с заданным частичным порядком. Эта модификация задачи GTSP имеет множество практических применений, среди которых следующие: оптимизация траектории инструмента для станков с ЧПУ [3], минимизация времени и стоимости резки в процессе раскроя листового металла [4, 5], настройка координатно-измерительного оборудования [6], оптимизация траектории при множественном сверлении отверстий [7].

Обзор текущего состояния исследований

Задача GTSP является обобщением классической задачи коммивояжера (Traveling Sales-

man Problem, TSP). Как следствие, она остается NP-трудной даже на евклидовой плоскости всякий раз, когда число кластеров m является частью ее условия [8]. С другой стороны, хорошо известная схема динамического программирования Хелда–Карпа [9], адаптированная к GTSP, имеет трудоемкость $O(n^3m^2 2^m)$. Таким образом, задача принадлежит классу FPT (Fixed Parameter Tractability) относительно параметризации количеством кластеров. Более того, при $m = O(\log n)$ оптимальное решение GTSP может быть найдено за полиномиальное время. Как следует из приведенного далее обзора литературы, исследования в области алгоритмического анализа задачи GTSP развивались по нескольким основным направлениям.

Первый подход основан на сведении исходной задачи к подходящей постановке асимметричной задачи коммивояжера (Asymmetric Traveling Salesman Problem, ATSP) и к последующему решению полученной вспомогательной задачи [10, 11]. Несмотря на математическое изящество, этот подход не свободен от недостатков:

– получаемые задачи ATSP обладают специфической структурой, затрудняющей их численное решение и на современных MIP-решателях, таких как Gurobi и CPLEX;

– даже близкие по функционалу к оптимальным приближенные решения вспомогательной задачи ATSP могут соответствовать недопустимым решениям исходной задачи [12].

Другой известный подход связан с разработкой точных алгоритмов для частных случаев задачи GTSP и приближенных алгоритмов с теоретическими оценками, включая алгоритмы ветвей и границ [13, 14] и полиномиальные приближенные схемы (Polynomial-time Approximation Scheme, PTAS) [15, 16].

Третий подход заключается в разработке новых и адаптации известных эвристик и метаэвристик. В этом направлении среди известных результатов выделяются гибридный алгоритм Гутина и Карапетяна [17], адаптация известного солвера Лина–Кернигана–Хельсгауна [18] и метаэвристика адаптивного поиска в больших окрестностях (Adaptive Large Neighborhood Search, ALNS) [19], обладающая рекордной на сегодняшний день практической производительностью.

Однако алгоритмические результаты для рассматриваемой в статье задачи PCGTSP до сих пор остаются немногочисленными и, по мнению авторов, исчерпываются следующими:

– эффективные алгоритмы для специальных ограничений предшествования типа Баласа [20–22] и ограничений предшествования, приводящих к квази- и псевдопирамидальным оптимальным маршрутам [23, 24];

– общий подход к выводу нижних оценок в методе ветвей и границ [25];

– метаэвристический солвер PCGLNS, развивающий результаты [26, 27], полученные в [19] для GTSP.

В данной статье предлагаются два новых точных алгоритма для общей постановки задачи PCGTSP, практическая эффективность которых подтверждается сравнительными численными экспериментами.

Постановка задачи

Рассмотрим общую постановку задачи PCGTSP. Условие ее определяется тройкой (G, C, Π), где $G = (V, E, c)$ – взвешенный ориентированный граф, вес произвольной дуги $(u, v) \in E$ которого задается соотношением $c(u, v)$; $C = \{V_1, \ldots, V_m\}$ – разбиение множества V вершин графа G на m непустых попарно непересекающихся кластеров; $\Pi = (C, A)$ – ориентированный ациклический граф, задающий частичный порядок (ограничения предшествования) на множестве кластеров C.

Каждой вершине $v \in V$ графа G сопоставим (единственный) кластер $V(v)$, содержащий данную вершину. Далее без ограничения общности полагаем орграф Π транзитивно замкнутым (в котором соотношения $(V_i, V_j) \in A$ и $(V_j, V_k) \in A$ влекут $(V_i, V_k) \in A$ для произвольных индексов i, j и k) и верным включение $(V_1, V_p) \in A$ для каждого $p \in \{2, \ldots, m\}$.

Замкнутый маршрут $T = v_1, v_2, \ldots, v_m$ будет допустимым решением задачи PCGTSP при следующих условиях:

– маршрут T начинается и заканчивается в произвольной вершине $v_1 \in V_1$;

– произвольный кластер $V_p \in C$ посещается маршрутом T точно один раз;

– маршрут T соответствует частичному порядку Π, то есть любой кластер V_q посещается маршрутом T только после всех кластеров, предшествующих ему в Π.

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

Цель задачи состоит в построении допустимого решения T минимальной стоимости.

Предлагаемые в данной статье алгоритмы опираются на следующие общие основные идеи.

Декомпозиция задачи. В каждой вершине дерева поиска сопоставим исходной задаче две вспомогательные подзадачи следующим образом:

– рассмотрим подмножество $C' \subset C$, такое что $V_1 \in C'$, зафиксируем некоторый кластер $V_l \in C'$ и вершины $v \in V_1$ и $u \in V_l$ соответственно;

– пусть $c_{\min}$ – нижняя граница стоимости v-u-путей, проходящих через все кластеры C' и удовлетворяющих ограничениям предшествования (в рассматриваемом варианте динамического программирования эта граница будет точной);

– исключая из C' все внутренние кластеры и соединяя V_1 с V_l непосредственно ребром нулевого (0) веса, тем самым создаем подзадачу P, наследующую у исходной задачи веса остальных дуг, разбиение на кластеры и ограничения предшествования;

– используя соотношение

$$LB = c_{\min} + \mathrm{OPT}(\mathcal{P}_{rel}) \quad (1)$$

в качестве нижней границы, отсекаем текущий узел всякий раз, когда $LB > UB$; здесь $\mathrm{OPT}(P_{rel})$ – вес некоторого эффективно найденного решения релаксации P_{rel} задачи P, UB – стоимость наилучшего найденного допустимого решения исходной задачи.

Нижние границы. Сравним нижние оценки, получаемые применением различных релаксаций вспомогательной задачи P. Для построения релаксации P используем двухэтапный подход, предложенный в [25]. На первом этапе сведем задачу P к подходящей постановке задачи ATSP одним из следующих способов.

1. Ослабляя исходные ограничения предшествования, исключаем ребра $(v', v'') \in E$, для которых $(V(v''), V(v')) \in A$. Затем сведем полученную задачу к ATSP, используя классическое преобразование Нуна и Бина [11].

2. Ослабляя аналогичным способом исходные ограничения предшествования, сведем полученную задачу к ATSP, определенной на вспомогательном графе кластеров

$$H_1 = (\tilde{C}', A_1, c_1), \text{ где } \tilde{C}' = C \setminus C' \cup \{V_1, V_l\},$$
$$A_1 = \{(V_1, V_l)\} \cup \{(V_i, V_j) \mid i > 2, \{V_i, V_j\} \subset \tilde{C}',$$
$$\exists (v' \in V_i, v'' \in V_j) : (v', v'') \in E\},$$
$$c_1(V_1, V_l) = 0, c_1(V_i, V_j) =$$
$$= \min\{c(v', v'') : v' \in V_i, v'' \in V_j, (v', v'') \in E\}.$$

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

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

то есть для любого $V_i \in \tilde{C}' \setminus \{V_1\}$ упорядоченная пара $(V_i, V_k) \in A$, если существует $V_j \in \tilde{C}'$ и вершины $v' \in V_i$, $v'' \in V_j$, $v''' \in V_k$ такие, что путь $\pi = v', v'', v'''$ согласован с исходными ограничениями предшествования.

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

На втором этапе повторно релаксируем полученную задачу ATSP путем либо нахождения минимального остовного дерева (Minimum Spanning Arborescence Problem, MSAP), либо решения подходящей задачи о назначениях (Assignment Problem, AP). Тем самым находим искомую нижнюю границу по формуле (1). Кроме того, в некоторых случаях для уточнения нижних оценок находим оптимальное значение вспомогательной задачи ATSP, применяя солвер Gurobi. Для удобства все способы получения нижних оценок приведены в таблице 1, столбцы которой соответствуют способам релаксации исходной задачи на первом этапе предлагаемой процедуры, а строки – способам построения оценок на втором.

Таблица 1

Нижние границы

Table 1

Lower bounds

Метод релаксации	Нун и Бин	H_1	H_2
Цикловое покрытие AP	E_1	L_1	L_2
Минимальное остовное дерево MSAP	E_2	E_3	E_4
Прямое решение при помощи Gurobi	E_5	L_3	E_6

Опираясь на результаты численных экспериментов, сократим список используемых методов построения нижних оценок. По наблюде-

ниям, оценки L_1–L_3 оказывались наиболее точными с доверительной вероятностью 95 %. На тестовых задачах L_1 в среднем оказывается в интервале 0.91±0.02 по отношению к оценке L_3, оценка L_2 соответственно 0.97±0.02, а оценки E_1–E_4 систематически ниже: E_1 0.48±0.03, E_2 и E_3 0.54±0.01, E_4 0.60±0.002. Кроме того, не используются оценки E_5 и E_6 в силу повышенной трудоемкости их вычисления. Таким образом, далее ограничиваемся оценками $LB_i = c_{\min} + L_i$, $i \in \{1, 2, 3\}$.

Алгоритм ветвей и границ

Для обхода дерева поиска в процессе решения задачи PCGTSP (G, C, П) используем метод поиска в ширину (алгоритм 1). Каждый узел этого дерева связан с префиксом $\sigma = (Vi_1, Vi_2, \ldots, Vi_r)$, где $Vi_j \in C$, $Vi_1 = V_1$ и $r \in \{1, \ldots, m\}$. Кластеры Vi_j посещаются в порядке, задаваемом последовательностью σ, все остальные – произвольно (с соблюдением ограничений предшествования П), тем самым образуя вспомогательную задачу P.

Алгоритм 1. BnB :: Главная процедура
Вход: орграф G, кластеры G, частичный порядок П
Выход: маршрут и его стоимость
1. Инициализация Q = empty queue
2. Начинаем с $Root = V_1$
3. *Q.push(Root)*
4. **while not** *Q*.empty()
5. Берем следующий префикс для обработки σ = *Q*.pop()
6. *process=Bound*(σ)
7. **if not** *process* **then**
8. Префикс отсекается; **continue**
9. **end if**
10. *UpdateLowerBound*(σ)
11. **for all** *child* ∈ *Branch*(σ) **do**
12. Помещаем префикс в очередь на обработку *Q*.push(*child*)
13. **end for**
14. **end while**

К каждому узлу дерева поиска применяем процедуру построения нижней оценки Bound (алгоритм 2), заключающуюся в выполнении следующих действий:

– для префикса σ сопоставляем кортеж $T(\sigma) = (Vi_1, \{Vi_1, Vi_2, \ldots, Vi_r\}, Vi_r)$;

– на шаге 4 вычисляем матрицу $D(\sigma)$ минимальных попарных расстояний по формуле

$$D(\sigma)_{vu} = \min\{\cos t(P_{v,u}) : v \in V_{i_1}, u \in V_{i_r},$$

$P_{v,u}$ – частичный путь в порядке σ}

(эта матрица легко вычисляется инкрементально с использованием матрицы $D(\sigma')$ родительского узла дерева поиска);

– если для некоторого σ_1, $T(\sigma) = T(\sigma_1)$ и $D(\sigma)_{uv} \geq D(\sigma_1)_{uv}$, $v \in Vi_1$, $u \in Vi_r$, то префикс σ доминируется префиксом σ_1 и подлежит отсечению;

– на шаге 11 вычисляем оценки L_1 и L_2 (табл. 1) и сохраняем их в глобальной переменной Opt^T, используя формулу

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

– для текущего узла σ на шаге 13 рассчитываем нижнюю границу по формуле

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

– наконец, узел σ отсекается, если $LB > UB$; префиксы, избежавшие отсечения, обрабатываются процедурой ветвления Branch (алгоритм 3), которая пытается удлинить префикс σ на один кластер с соблюдением ограничений предшествования.

Алгоритм 2. BnB :: Bound
Вход: префикс σ
Выход: признак того, что префикс подлежит обработке
1. **global** D^T_{ij}
2. **global** Opt^T
3. вычисляем кортеж $T = (Vi_1, \{Vi_1, Vi_2, \ldots, Vi_r\}, Vi_r)$
4. D_{ij}=MinCos*ts*(σ)
5. **if** $D_{ij} \geqslant D^T_{ij}[T]$, $\forall i, j$ **then**
6. **return false**
7. **end if**
8. обновляем веса маршрутов $D^T_{ij}[T] =$
$= \min(D^T_{ij}[T], D_{ij})$, $\forall i, j$
9. $c_{\min} = \min_{ij} D_{ij}$
10. **if** $T \notin Opt^T$ **then**
11. вычисляем нижнюю границу
$Opt^T[T] = \max(L_1(\sigma), L_2(\sigma))$
12. **end if**
13. $LB = c_{\min} + Opt^T[T]$
14. **if** $LB > UB$ **then**
15. **return false**
16. **end if**
17. **return true**

Динамическое программирование

Основная идея алгоритма ветвей и границ представляется близкой к классической схеме динамического программирования (Dynamic Programming, DP) Хелда–Карпа [9], адаптированной к учету ограничений предшествования и дополненной стратегией оценивания, представленной в [28].

В данной работе реализуется усовершенствованная версия этой схемы для численной оценки производительности описанного выше алгоритма ветвей и границ. Подобно классической схеме представленный алгоритм состоит из двух этапов.

1. Строится таблица динамического программирования в прямом направлении инкрементально, слой за слоем. Оптимальное значение решаемой задачи находится после построения последнего, m-го слоя.

2. Оптимальный маршрут восстанавливается обратным ходом по таблице, построенной на первом этапе.

Каждое состояние DP соответствует частичному *v*-*u*-пути и индексируется кортежем (C', V_l, v, u), где $C' \subset C$ представляет собой идеал частично упорядоченного множества кластеров C, то есть $\forall\ (V \in C', V' \in C, (V', V) \in \mathrm{A}) \Rightarrow$ $\Rightarrow (V' \in C')$; очевидно, в предлагаемых условиях V_1 принадлежит произвольному идеалу $C' \subset C$; $V_l \subset C'$, для которого нет такого $V \in C'$, что $(V_l, V) \in A$; $v \in V_1$, $u \in V_l$.

Значение каждой ячейки DP S содержит ссылку S[*pred*] на предшествующее состояние, локальное значение нижней оценки S[*LB*] и стоимость S[*cost*] соответствующего частичного *v*-*u*-пути.

Пусть $\Im_k$ – подмножество идеалов размера $k \in \{1, \ldots, m\}$. Очевидно, что $\Im_1 = \{\{V_1\}\}$, поэтому первый слой L_1 таблицы динамического программирования строится тривиально. Индуктивное построение остальных слоев описано в алгоритме 4.

Алгоритм 3. BnB :: Branch
Вход: префикс σ
Выход: список потомков префикса для обработки
1. Инициализация R = empty queue
2. **for all** $V \in C$ **do**
3. *valid* = **true**
4. **for all** $W \in C$ **do**
5. **if** $W=V$ **or** $(V, W) \in \Pi$ **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

Замечания. Оптимум для решаемой задачи дается классическим уравнением Беллмана:

$$\mathrm{OPT} = \min_{v \in V_1} \min\{S[\cos t] + c(u,v) : S = (C', V_l, v, u) \in \mathcal{L}_m.$$

По построению таблица DP имеет размер $O(n^2 m|\Im|)$, значит, время работы алгоритма $O(n^3 m^2|\Im|)$, в частности, для частичного порядка фиксированной ширины w $|\Im| = O(m^w)$ [29]. Следовательно, оптимальное решение PCGTSP может быть найдено за полиномиальное время даже без применения нижних оценок на шагах 10–12.

После построения произвольного слоя L_k обновляется глобальное значение нижней оценки, что улучшает точность аппроксимации.

В данной реализации для повышения быстродействия вычисляется оценка L_3 на шаге 9 только для небольшого количества состояний с наименьшей нижней оценкой.

Алгоритм 4. DP :: индуктивное построение таблицы динамического программирования
Вход: орграф G, частичный порядок Π, слой таблицы DP L_k, верхняя граница *UB*
Выход: (k+1)-й слой L_{k+1}
1. Инициализация $L_{k+1} = \varnothing$
2. **for all** $C' \in \Im_k$ **do**
3. **for all** кластер $V_l \in C \setminus C'$ s.t. $C' \cup \{V_l\} \in \Im_{k+1}$ **do**
4. **for all** $v \in V_1$ и $u \in V_l$ **do**
5. **if** есть состояние $S = (C', U, v, w) \in L_k$ s.t. $(w, u) \in E$ **then**
6. создаем новое состояние $S' = (C' \cup \{V_l\}, V_l, v, u)$
7. $S'[cost] = \min\{S[cost] + c(w, u): S = (C', U, v, w) \in L_k\}$
8. $S'[pred] = \mathrm{argmin}\{S[cost] + c(w, u): S = (C', U, v, w) \in L_k\}$
9. $S'[LB] = S'[cost] + \max\{L_1, L_2, L_3\}$
10. **if** $S'[LB] \leqslant UB$ **then**
11. Добавляем S' к L_{k+1}
12. **end if**
13. **end if**
14. **end for**
15. **end for**
16. **end for**
17. **return** L_{k+1}

Численные эксперименты

Все алгоритмы тестировались на общедоступной библиотеке PCGTSPLIB [25]. В качестве начального приближения им задавалось

одно и то же допустимое решение, полученное эвристикой PCGLNS [27]. Вычисления проводились на одном и том же оборудовании (16-ядерный Intel Xeon, 128G RAM) с предельным временем счета 10 часов. Для оценки точности вычислений использовалась верхняя оценка относительной погрешности, вычисляемая по формуле

$$gap = \frac{UB - LB}{LB}. \quad (2)$$

Критерием остановки является условие $gap < 5$ %. Разработанные в данном исследовании алгоритмы реализованы на кроссплатформенном языке Python и могут исполняться на всех современных операционных системах, включая Linux, MacOS и Microsoft Windows. Код оптимизирован за счет использования библиотек NumPy и SciPy для обработки матриц и NetworkX для работы с графами, для параллельного исполнения использован модуль multiprocessing стандартной библиотеки Python. Исходный код доступен в [30].

Обсуждение

Результаты экспериментов иллюстрирует таблица 2, которая организована следующим образом: первая группа столбцов описывает задачу, включая ее 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

Таблица 2

Результаты экспериментов

Table 2

Experimental results

Задача					Gurobi			Метод ветвей и границ			DP		
№	**ID**	**n**	**m**	**UB_0**	**Время, с**	**LB**	**gap, %**	**Время, с**	**LB**	**gap, %**	**Время, с**	**LB**	**gap, %**
1	br17.12	92	17	43	82	43	0	**11.2**	**43**	**0**	27.3	43	0
4	ESC25	133	26	1418	10.61	1383	0	32	1383	0	60.69	1383	0
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	1.3	62	0	**0.52**	**62**	**0**
7	ESC78	414	79	14872	1278.45	14630	1.66	1.3	14594	1.63	**0.68**	**14594**	**1.63**
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	35865	11274	4.86	**2225**	**11290**	**4.71**
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**
20	p43.1	203	43	22545	4691	21677	4	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	2452.1
23	p43.4	204	43	66857	36000	45198	47.92	4470	66846	0	**333.02**	**66846**	**0**
26	rbg048a	255	49	282	24.22	282	0	0.9	272	3.68	**0.25**	**272**	**3.68**
27	rbg050c	259	51	378	13.83	378	0	**0.2**	**372**	**1.61**	0.25	372	1.61
28	rbg109a	573	110	848	6	848	0	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**	**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	7.6	36000	1933	7.6
38	ry48p.3	254	48	16540	36000	13085	26.4	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**

счета в секундах, наилучшее значение нижней границы *LB* и оценку погрешности, заданную в процентах. Задачи, в которых один из предлагаемых алгоритмов превосходит Gurobi по производительности, выделены жирным шрифтом. Следует отметить, что для 13 из 39 задач (33 %) один из построенных авторами алгоритмов показал рекордную производительность, в том числе в 12 случаях по быстродействию и в 7 по точности.

Предложенные алгоритмы смогли найти оптимальное решение в 6 из 39 случаев (хотя такая цель не ставилась). Для 10 (15) постановок, включая одни из самых больших – rbg323a и rbg358a (1 825 и 1 967 вершин соответственно), было получено решение с точностью 5 % (10 %). С другой стороны, для некоторых задач (например, p43.1, p43.2 и p43.3) результаты предлагаемых алгоритмов значительно уступают Gurobi, что, по-видимому, объясняется недостаточно точными нижними оценками. В то же время для задач p43.4 и ry48p.4 алгоритмы значительно превзошли Gurobi. В целом, хотя Gurobi демонстрирует в среднем чуть лучшую производительность, почти все предложенные алгоритмы показывают вполне сопоставимые результаты. Необходимо добавить, что в проводимых экспериментах солверу Gurobi, как и тестируемым алгоритмам, было предоставлено хорошее стартовое решение, полученное эвристикой, что не является обязательным при организации подобных сравнительных экспериментов.

Заключение

В данной работе описан разработанный первый специализированный алгоритм ветвей и границ для обобщенной задачи коммивояжера с ограничениями предшествования. Он развивает идеи классической схемы динамического программирования Хелда–Карпа и подход Салмана к построению нижних оценок. Предложенные авторами алгоритмы реализованы в виде ПО на языке программирования Python, что обеспечивает мультиплатформенность программного продукта.

Для оценки производительности алгоритмов проведены численные эксперименты, в которых для сравнения использовался передовой коммерческий солвер Gurobi, продемонстрирована их конкурентоспособность.

В качестве направления дальнейших исследований предполагается разработка более точных нижних оценок. Кроме того, по мнению авторов, дальнейшая оптимизация и распараллеливание могут существенно улучшить производительность представленных алгоритмов.

В настоящее время разработанное ПО интегрируется с САПР СИРИУС [31], предназначенной для оптимизации раскроя листового материала на фигурные заготовки и подготовки управляющих программ для машин листовой резки с ЧПУ.

Численные эксперименты проводились на суперкомпьютере «Уран» Института математики и механики им. Н.Н. Красовского Уральского отделения Российской академии наук.

Работа выполнена при финансовой поддержке Министерства науки и высшего образования РФ, соглашение № 075-02-2021-1383.

Литература

1. Srivastava S., Kumar S., Garg R., Sen P. Generalized traveling salesman problem through n sets of nodes. CORS J., 1969, vol. 7, no. 2, pp. 97–101.

2. Gutin G., Punnen A.P. The Traveling Salesman Problem and Its Variations. Boston, MA: Springer US, 2007, 38 p.

3. Castelino K., D'Souza R., Wright P.K. Toolpath optimization for minimizing airtime during machining. J. of Manufacturing Systems, 2003, vol. 22, no. 3, pp. 173–180. DOI: 10.1016/S0278-6125(03)90018-5.

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. Int. J. of Production Research, 2018, vol. 56, no. 14, pp. 4819–4830. DOI: 10.1080/00207543.2017.1421784.

5. Makarovskikh T., Panyukov A., Savitskiy E. Mathematical models and routing algorithms for economical cutting tool paths. Int. J. of Production Research, 2018, vol. 56, no. 3, pp. 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, vol. 44, pp. 138–143. DOI: 10.1016/j.procir.2016.02.136.

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, vol. 26, no. 2, pp. 449–459. DOI: 10.1007/s11831-018-9251-x.

8. Papadimitriou C. The Euclidean travelling salesman problem is NP-complete. Theoretical Computer Science, 1977, vol. 4, no. 3, pp. 237–244. DOI: 10.1016/0304-3975(77)90012-3.

9. Held M., Karp R.M. A dynamic programming approach to sequencing problems. J. Soc. Indust. Appl. Math., 1962, vol. 10, no. 1, pp. 196–210. URL: http://www.jstor.org/stable/2098806 (дата обращения: 16.09.2021).

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, vol. 37, no. 2, pp. 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, vol. 31, no. 1, pp. 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. EJOR, 2012, vol. 219, no. 2, pp. 234–251. DOI: 10.1016/j.ejor.2012.01.011.

13. Fischetti M., González J.J.S., Toth P. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Operations Research, 1997, vol. 45, no. 3, pp. 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. EJOR, 2020, vol. 286, no. 3, pp. 849–866. DOI: 10.1016/j.ejor.2020.04.024.

15. Feremans C., Grigoriev A., Sitters R. The geometric generalized minimum spanning tree problem with grid clustering. 4OR, 2006, vol. 4, no. 4, pp. 319–329. DOI: 10.1007/s10288-006-0012-6.

16. Khachai M.Y., Neznakhina E.D. Approximation schemes for the generalized traveling salesman problem. Proc. Steklov Inst. Math, 2017, vol. 299, no. 1, pp. 97–105. DOI: 10.1134/S0081543817090127.

17. Gutin G., Karapetyan D. A memetic algorithm for the generalized traveling salesman problem. Natural Computing, 2010, vol. 9, no. 1, pp. 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, vol. 7, no. 3, pp. 269–287. DOI: 10.1007/s12532-015-0080-8.

19. Smith S.L., Imeson F. GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. Computers and Operations Research, 2017, vol. 87, pp. 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, 2001, vol. 13, no. 1, pp. 56–75. DOI: 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. Proc. Steklov Inst. Math, 2016, vol. 295, no. 1, pp. 38–46. DOI: 10.1134/S0081543816090054.

22. Chentsov A., Khachay M., Khachay D. Linear time algorithm for precedence constrained asymmetric generalized traveling salesman problem. IFAC-PapersOnLine, 2016, vol. 49, no. 12, pp. 651–655. DOI: 10.1016/j.ifacol.2016.07.767.

23. Khachay M., Neznakhina K. Towards tractability of the Euclidean generalized traveling salesman problem in grid clusters defined by a grid of bounded height. In: Communications in Computer and Information Science, 2018, vol. 871, pp. 68–77. DOI: 10.1007/978-3-319-93800-4_6.

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, vol. 88, no. 1, pp. 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, vol. 48, no. 2, pp. 163–166. 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. In: Optimization and Applications, 2020, pp. 196–208. DOI: 10.1007/978-3-030-62867-3_15.

27. GitHub. AndreiKud. PCGLNS. URL: https://github.com/AndreiKud/PCGLNS/ (дата обращения: 16.09.2021).

28. Morin T.L., Marsten R.E. Branch-and-bound strategies for dynamic programming. Operations Research, 1976, vol. 24, no. 4, pp. 611–627. URL: http://www.jstor.org/stable/169764 (дата обращения: 16.09.2021).

29. Steiner G. On the complexity of dynamic programming for sequencing problems with precedence constraints. Annals of Operations Research, 1990, vol. 26, no. 1-4, pp. 103–123. DOI: 10.1007/BF02248587.

30. GitHub. Ukoloff. PCGTSP-BnB. URL: https://github.com/ukoloff/PCGTSP-BnB (дата обращения: 16.09.2021).

31. Tavaeva A., Petunin A., Ukolov S., Krotov V. A cost minimizing at laser cutting of sheet parts on CNC machines. In: Mathematical Optimization Theory and Operations Research, pp. 422–437. DOI: 10.1007/978-3-030-33394-2_33.

Software & Systems
DOI: 10.15827/0236-235X.137.054-064

Received 04.10.21, Revised 18.11.21
2022, vol. 35, no. 1, pp. 054–064

Software for solving the precedence constrained generalized traveling salesman problem

A.A. Petunin [1], *Dr.Sc. (Engineering), Professor, a.a.petunin@urfu.ru, aapetunin@gmail.com*
S.S. Ukolov [1], *Junior Researcher, s.s.ukolov@urfu.ru*
M.Yu. Khachay [2], *Dr.Sc. (Physics and Mathematics), Professor, Head of Department, mkhachay@imm.uran.ru*

[1] *Ural Federal University named after the First President of Russia B.N. Yeltsin, Ekaterinburg, 620002, Russian Federation*
[2] *N.N. Krasovskii Institute of Mathematics and Mechanics of the Ural Branch of the Russian Federation Academy of Sciences, Ekaterinburg, 620108, Russian Federation*

Abstract. The paper considers the generalized problem of the precedence constraint traveling salesman (PCGTSP). Like the classical traveling salesman problem (TSP), the authors search a minimum cost closed cycle in this problem, while the set of vertices is divided into nonempty pairwise disjoint subsets that are clusters; each feasible route must visit each cluster in a single vertex. In addition, the set of valid routes is constrained by an additional restriction on the order of visiting clusters, that is, some clusters must be visited earlier than others. In contrast to the TSP and the generalized traveling salesman problem (GTSP), this problem is poorly studied both theoretically and from the point of view of algorithm design and implementation.

The paper proposes the first specialized branch-and-bound algorithms using the solutions obtained using the recently developed PCGLNS heuristic as an initial guess. The original PCGTSP problem undergoes several relaxations, therefore there are several lower bounds for the original problem; the largest of them is used to cut off the branches of the search tree and thereby reduce the enumeration. The algorithms are implemented as open source software in the Python 3 programming language using the specialized NetworkX library. The performance of the proposed algorithms is evaluated on test examples from the PCGTSPLIB public library in comparison with the state-of-the-art Gurobi solver using the MILP model recently proposed by the authors, and seems to be quite competitive even in the current implementation.

The developed algorithms can be used in a wide class of practical problems, for example, for optimal tool routing for CNC sheet cutting machines, as well as for assessing the quality of solutions obtained using other methods.

Keywords: GTSP, precedence constraint, branch and bound, dynamic programming, Held-Karp algorithm.

Acknowledgements. *The work was financially supported performed by the Ministry of Science and Higher Education of the Russian Federation (Agreement no. 075-02-2021-1383).*

References

1. Srivastava S., Kumar S., Garg R., Sen P. Generalized traveling salesman problem through n sets of nodes. *CORS J.*, 1969, vol. 7, no. 2, pp. 97–101.

2. Gutin G., Punnen A.P. *The Traveling Salesman Problem and Its Variations*. Boston, MA: Springer US, 2007, 38 p.

3. Castelino K., D'Souza R., Wright P.K. Toolpath optimization for minimizing airtime during machining. *J. of Manufacturing Systems*, 2003, vol. 22, no. 3, pp. 173–180. DOI: 10.1016/S0278-6125(03)90018-5.

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. *Int. J. of Production Research*, 2018, vol. 56, no. 14, pp. 4819–4830. DOI: 10.1080/00207543.2017.1421784.

5. Makarovskikh T., Panyukov A., Savitskiy E. Mathematical models and routing algorithms for economical cutting tool paths. *Int. J. of Production Research*, 2018, vol. 56, no. 3, pp. 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, vol. 44, pp. 138–143. DOI: 10.1016/j.procir.2016.02.136.

7. Dewil R., Küçükoğlu I., Luteyn C., Cattrysse D. A critical review of multi-hole drilling path optimization. *Archives of Computational Methods in Engineering*, 2019, vol. 26, no. 2, pp. 449–459. DOI: 10.1007/s11831-018-9251-x.

8. Papadimitriou C. The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, 1977, vol. 4, no. 3, pp. 237–244. DOI: 10.1016/0304-3975(77)90012-3.

9. Held M., Karp R.M. A dynamic programming approach to sequencing problems. *J. Soc. Indust. Appl. Math.*, 1962, vol. 10, no. 1, pp. 196–210. Available at: http://www.jstor.org/stable/2098806 (accessed September 16, 2021).

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, vol. 37, no. 2, pp. 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, vol. 31, no. 1, pp. 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. *EJOR*, 2012, vol. 219, no. 2, pp. 234–251. DOI: 10.1016/j.ejor.2012.01.011.

13. Fischetti M., González J.J.S., Toth P. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 1997, vol. 45, no. 3, pp. 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. *EJOR*, 2020, vol. 286, no. 3, pp. 849–866. DOI: 10.1016/j.ejor.2020.04.024.

15. Feremans C., Grigoriev A., Sitters R. The geometric generalized minimum spanning tree problem with grid clustering. *4OR*, 2006, vol. 4, no. 4, pp. 319–329. DOI: 10.1007/s10288-006-0012-6.

16. Khachai M.Y., Neznakhina E.D. Approximation schemes for the generalized traveling salesman problem. *Proc. Steklov Inst. Math.*, 2017, vol. 299, no. 1, pp. 97–105. DOI: 10.1134/S0081543817090127.

17. Gutin G., Karapetyan D. A memetic algorithm for the generalized traveling salesman problem. *Natural Computing*, 2010, vol. 9, no. 1, pp. 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, vol. 7, no. 3, pp. 269–287. DOI: 10.1007/s12532-015-0080-8.

19. Smith S.L., Imeson F. GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. *Computers and Operations Research*, 2017, vol. 87, pp. 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*, 2001, vol. 13, no. 1, pp. 56–75. DOI: 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. *Proc. Steklov Inst. Math.*, 2016, vol. 295, no. 1, pp. 38–46. DOI: 10.1134/S0081543816090054.

22. Chentsov A., Khachay M., Khachay D. Linear time algorithm for precedence constrained asymmetric generalized traveling salesman problem. *IFAC-PapersOnLine*, 2016, vol. 49, no. 12, pp. 651–655. DOI: 10.1016/j.ifacol.2016.07.767.

23. Khachay M., Neznakhina K. Towards tractability of the Euclidean generalized traveling salesman problem in grid clusters defined by a grid of bounded height. In: *Communications in Computer and Information Science*, 2018, vol. 871, pp. 68–77. DOI: 10.1007/978-3-319-93800-4_6.

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, vol. 88, no. 1, pp. 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, vol. 48, no. 2, pp. 163–166. 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. In: *Optimization and Applications*, 2020, pp. 196–208. DOI: 10.1007/978-3-030-62867-3_15.

27. *GitHub. AndreiKud. PCGLNS.* Available at: https://github.com/AndreiKud/PCGLNS/ (accessed September 16, 2021).

28. Morin T.L., Marsten R.E. Branch-and-bound strategies for dynamic programming. *Operations Research*, 1976, vol. 24, no. 4, pp. 611–627. Available at: http://www.jstor.org/stable/169764 (accessed September 16, 2021).

29. Steiner G. On the complexity of dynamic programming for sequencing problems with precedence constraints. *Annals of Operations Research*, 1990, vol. 26, no. 1-4, pp. 103–123. DOI: 10.1007/BF02248587.

30. *GitHub. Ukoloff. PCGTSP-BnB.* Available at: https://github.com/ukoloff/PCGTSP-BnB (accessed September 16, 2021).

31. Tavaeva A., Petunin A., Ukolov S., Krotov V. A cost minimizing at laser cutting of sheet parts on CNC machines. In: *Mathematical Optimization Theory and Operations Research*, pp. 422–437. DOI: 10.1007/978-3-030-33394-2_33.

Для цитирования

Петунин А.А., Уколов С.С., Хачай М.Ю. Программное обеспечение для решения обобщенной задачи коммивояжера с ограничениями предшествования // Программные продукты и системы. 2022. Т. 35. № 1. С. 054–064. DOI: 10.15827/0236-235X.137.054-064.

For citation

Petunin A.A., Ukolov S.S., Khachay M.Yu. Software for solving the precedence constrained generalized traveling salesman problem. *Software & Systems*, 2022, vol. 35, no. 1, pp. 054–064 (in Russ.). DOI: 10.15827/0236-235X.137.054-064.