УДК 519.854

**Программное обеспечение для решения обобщённой задачи коммивояжера с ограничениями предшествования**

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

А.А. Петунин, д.т.н., профессор, Уральский федеральный университет имени первого Президента России Б.Н. Ельцина, [a.a.petunin@urfu.ru](mailto:a.a.petunin@urfu.ru), [aapetunin@gmail.com](mailto:aapetunin@gmail.com);

С.С. Уколов, м.н.с., Уральский федеральный университет имени первого Президента России Б.Н. Ельцина, s.s.ukolov@urfu.ru;

М.Ю. Хачай, д.ф.-м.н., профессор РАН, зав. отделом, Институт математики и механики им. Н. Н. Красовского Уральского отделения Российской академии наук, [mkhachay@imm.uran.ru](mailto:mkhachay@imm.uran.ru)

**Аннотация**: Программное обеспечение для решения многих задач дискретной оптимизации использует математическую модель обобщенной задачи коммивояжера (Generalized Traveling Salesman Problem, GTSP), обладающей множеством важных практических приложений в области исследовании операций. Как и в классической задаче коммивояжера (TSP), цель задачи состоит в построении циклического маршрута минимальной стоимости. Принципиальное отличие GTSP от TSP состоит в том, что постановка задачи GTSP наряду с весами ребер графа содержит разбиение множества его вершин на непустые попарно непересекающиеся подмножества – кластеры, и каждый допустимый маршрут обязан посетить каждый из кластеров в единственной вершине. В исследуемой задаче GTSP с ограничениями предшествования (PCGTSP) множество допустимых маршрутов стеснено дополнительным ограничением на порядок посещения кластеров. В отличие от базовой постановки GTSP, задача PCGTSP является слабо исследованной как теоретически, так и с точки зрения проектирования и реализации алгоритмов. Насколько нам известно, все алгоритмические результаты в области решения этой задачи исчерпываются общей схемой ветвления Салмана, несколькими моделями MILP и недавно предложенным авторами метаэвристическим солвером PCGLNS. В данной работе предлагаются первые специализированные алгоритмы ветвей и границ, использующие PCGLNS в качестве мощной первичной эвристики. Алгоритмы реализованы в виде открытого (open source) программного обеспечения 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 нашу недавно предложенную MILP модель, и представляется вполне конкурентноспособной 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

**Software for solving of the Precedence Constrained Generalized Traveling Salesman Problem**

The work was performed with the financial support of the Ministry of Science and Higher Education of the Russian Federation (Agreement number 075-02-2021-1383).

A.A. Petunin, Doctor of Technical Sciences, Professor, Ural Federal University, [aapetunin@gmail.com](mailto:aapetunin@gmail.com);

S.S. Ukolov, Junior researcher, Ural Federal University, s.s.ukolov@urfu.ru;

M.Yu. Khachay, Doctor of Physical and Mathematical Sciences, Professor, N.N. Krasovskii Institute of Mathematics and Mechanics of the Ural Branch of the Russian Academy of Sciences, [mkhachay@imm.uran.ru](mailto:mkhachay@imm.uran.ru)

**Abstract**: Software for solving many discrete optimization problems uses a mathematical model of the Generalized Traveling Salesman Problem (GTSP) having numerous valuable practical applications in operations research. As in the classic Traveling Salesman Problem (TSP), the goal of the problem is to construct a minimum cost cyclic route. The fundamental difference between GTSP and TSP is that the statement of the GTSP problem, along with a weighted directed graph, contains a partition of the set of its vertices into nonempty pairwise disjoint subsets – clusters, and each admissible route must visit each cluster at a single vertex. In the GTSP problem with precedence constraints (PCGTSP) investigated in this paper, the set of feasible routes is constrained by an additional constraint on the order of visiting clusters. Unlike the common setting of the GTSP, the PCGTSP appears still weakly studied in terms of algorithmic design and implementation. To the best of our knowledge, all the known algorithmic results for this problem can be exhausted by Salman's general branching framework, a few MILP models, and the PCGLNS meta-heuristic proposed by the authors recently. In this paper, we present the first problem-specific branch-and-bound algorithm exploiting PCGLNS as a powerful primal heuristic. Using the public PCGTSPLIB testbench, we evaluate the performance of the proposed algorithm against the classic Held-Karp dynamic programming scheme with branch-and-bound node fathoming strategy and Gurobi state-of-the-art solver armed by our recently proposed MILP model and PCGLNS-based warm start. The algorithms developed as open-source software using the Python 3 programming language and a specialized NetworkX library.

**Keywords**: GTSP, precedence constraint, branch and bound, dynamic programming, Held–Karp scheme.

1. Введение

Программное обеспечение для решения многих задач дискретной оптимизации использует математическую модель обобщенной задачи коммивояжера (GTSP), широко известной задачи комбинаторной оптимизации, впервые сформулированной в основополагающей статье Шриваставы и др. [1] и привлекшей внимание многих исследователей, (см., напр. обзор в [2]). В GTSP для заданного взвешенного орграфа *G=(V, E, c)* и разбиения *V1* ∪ … ∪ *Vm* набора узлов *V* графа *G* на непустые взаимно непересекающиеся кластеры требуется найти замкнутый тур с минимальной стоимостью, который посещает каждый кластер *Vi* в точности один раз. В этой статье мы рассматриваем обобщенную задачу коммивояжера с ограничениями предшествования (PCGTSP), каждому допустимому маршруту которой необходимо посещать кластеры в соответствии с заданным частичным порядком. Эта модификация задачи GTSP имеет множество практических применений, среди которых задачи

* оптимизации траектории инструмента для станков с числовым программным управлением (ЧПУ) [3],
* минимизации времени и стоимости резки в процессе раскроя листового металла [4, 5],
* настройки координатно-измерительного оборудования [6],
* оптимизации траектории при множественном сверлении отверстий [7].
  1. Обзор текущего состояния исследований

Задача GTSP является обобщением классической задачи коммивояжера (TSP). Как следствие, задача остается NP-трудной даже на евклидовой плоскости всякий раз, когда число кластеров является частью ее условия [8]. С другой стороны, хорошо известная схема динамического программирования Хелда и Карпа [9], адаптированная к GTSP, обладает трудоемкостью *O(n3m22m)* Тем самым, задача принадлежит классу FPT относительно параметризации количеством кластеров. Более того, при *m = O(log n)* оптимальное решение GTSP может быть найдено за полиномиальное время. Как следует из приведенного ниже 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 (ATSP) и последующем решении полученной вспомогательной задачи ([10; 11]). Несмотря на математическое изящество, этот подход не свободен от ряда известных недостатков:

1. получаемые в результате такого сведения постановки задачи ATSP обладают специфической структурой, затрудняющей их численное решение даже на современных MIP решателях, таких как Gurobi и CPLEX;
2. 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 (см., например, [13; 14]) и полиномиальные приближенные схемы (PTAS) [15; 16].

Наконец, третий подход заключается в разработке новых и адаптации известных эвристик и метаэвристик. Среди известных результатов в этом направлении 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.

Отметим, что алгоритмические результаты для рассматриваемой в статье обобщенной задачи коммивояжера с ограничениями предшествования (PCGTSP) до сих пор остаются немногочисленными и исчерпываются, по нашим сведениям, приведенным ниже списком.

1. Эффективные алгоритмы для специальных ограничений предшествования типа Баласа [20—22] и ограничений предшествования, приводящих к квази- и псевдопирамидальным оптимальным маршрутам [23, 24],
2. Общий подход к выводу нижних оценок в методе ветвей и границ [25],
3. Недавно разработанный авторами данной статьи метаэвристический солвер PCGLNS [26, 27], развивающий результаты, полученные в [19] для GTSP.

Пытаясь восполнить данный пробел, в этой статье мы предлагаем два новых точных алгоритма для общей постановки задачи PCGTSP, практическую эффективность которых подтверждаем результатами сравнительных численных экспериментов.

* 1. Результаты данной статьи
* первый специализированный алгоритм ветвей и границ, объединяющий идеи, предложенные в [25] и авторские результаты в области эффективных эвристик [26];
* ускоренный алгоритм динамического программирования, основанный на классической схеме Хелда и Карпа (см. напр., [28]) и оценках Салмана [25];
* численные эксперименты, подтверждающие сравнимость предложенных в статье алгоритмов с наилучшим коммерческим решателем Gurobi, как по точности, так и по производительности вычислений.

2. Постановка задачи

Мы рассматриваем общую постановку обобщённой задачи коммивояжера с ограничениями предшествования (PCGTSP). Условие задачи определяется тройкой *(G, C,* Π*)*, где

* *G = (V, E, c)* – взвешенный ориентированный граф, вес произвольной дуги *(u, v)* ∈ *E* которого задается соотношением *c(u, v)*,
* *C = {V1, …, Vm}* – разбиение множества *V* вершин графа *G* на *m* непустых попарно непересекающихся *кластеров,*
* ориентированный ациклический граф Π = (*C, A*) задает частичный порядок (*ограничения предшествования*) на множестве кластеров *C*.

Каждой вершине *v* ∈ *V* графа *G* сопоставим (единственный) кластер *V(v)*, содержащий данную вершину. Далее, без ограничения общности, полагаем

* - орграф Π транзитивно замкнутым (в котором соотношения *(Vi, Vj)* ∈ *A* и *(Vj, Vk)* ∈ *A* влекут *(Vi, Vk)* ∈ *A* для произвольных индексов *i*, *j* и *k*);
* - верным включение *(V1, Vp)* ∈ *A* для каждого *p* ∈ *{2, …, m}*.

Договоримся замкнутый маршрут *T = v1, v2, …, vm* называть *допустимым решением* задачи PCGTSP, если

* Маршрут *T* начинается и заканчивается в произвольной вершине *v1* ∈ *V1*,
* произвольный кластер *Vp* ∈ *C* посещается маршрутом *T* в точности один раз,
* маршрут *T* *соответствует* частичному порядку Π, то есть любой кластер *Vq* посещается маршрутом *T* только *после* всех кластеров, предшествующих ему в Π.

Стоимость решения *T* определяется соотношением

Цель задачи состоит в построении допустимого решения минимальной стоимости.

3. Предварительные соображения

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

3.1. Декомпозиция задачи

В каждой вершине дерева поиска мы сопоставляем исходной задаче две (более простые) вспомогательные подзадачи следующим образом:

1. Рассмотрим подмножество *C'* ⊂ *C*, такое что *V1* ∈ *C'*, зафиксируем некоторый кластер *Vl* ∈ *C'* и вершины *v* ∈ *V1* и *u* ∈ *Vl* соответственно.
2. Пусть *cmin* – нижняя граница стоимости *v*-*u*-путей, проходящих через все кластеры *C'* и удовлетворяющих ограничениям предшествования[[1]](#footnote-1).
3. Исключая из *C'* все внутренние кластеры и соединяя *V1* с *Vl* непосредственно ребром нулевого (0) веса, мы тем самым создаём подзадачу *P*, наследующую у исходной задачи веса остальных дуг, разбиение на кластеры и ограничения предшествования.
4. Используя соотношение

(1)

в качестве нижней границы, отсекаем текущий узел всякий раз, когда *LB > UB*. Здесь OPT(*Prel*) это вес некоторого эффективно находимого решения релаксации *Prel* задачи *P*, а *UB* – стоимость наилучшего найденного допустимого решения исходной задачи.

3.2. Нижние границы

В этом разделе мы сравниваем нижние оценки, получаемые применением различных релаксаций вспомогательной задачи *P*. Для построения такой релаксации *P* используем двухэтапный подход, предложенный в [25]. На первом этапе мы сводим задачу *P* к подходящей постановке задачи ATSP одним из следующих способов:

1. Ослабляя исходные ограничения предшествования, исключаем ребра *(v', v'')* ∈ *E*, для которых *(V(v''), V(v'))* ∈ *A*. Затем, сводим полученную задачу к ATSP, используя классическое преобразование Нуна и Бина [11].
2. Ослабляя аналогичным способом исходные ограничения предшествования, сводим полученную задачу к ATSP, определённой на вспомогательном *графе кластеров* , где
3. Сводим исходную задачу к ATSP, определённой на ориентированном графе , где

то есть, для любого , упорядоченная пара *(Vi, Vk)* ∈ *A*, если существует и вершины *v’* ∈ *Vi*, *v’’* ∈ *Vj*, *v’’’* ∈ *Vk*, такие, что путь π = *v’, v’’, v’’’* согласован с исходными ограничениями предшествования. Далее,

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

Опираясь на результаты численных экспериментов, мы сократили список используемых методов построения нижних оценок (см. табл. 1б). Согласно нашим наблюдениям, оценки L1–L3 оказывались наиболее точными с доверительной вероятностью 95%. Кроме того, мы отказались от использования оценок E5 и E6 в силу повышенной трудоемкости вычисления последних. Таким образом, всюду ниже мы ограничиваемся оценками

Таблица 1

Нижние границы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | |  | Нун и Бин |  |  | | AP | E1 | **L1** | **L2** | | MSAP | E2 | E3 | E4 | | Gurobi | E5 | **L3** | E6 | | |  |  |  | | --- | --- | --- | | E1 | E2=E3 | E4 | | 0.48±0.03 | 0.54±0.01 | 0.60±0.002 | | L1 | L2 | L3 | | 0.91±0.02 | 0.97±0.02 | 1.00 | |
| а) Обозначения | б) Сравнение с оценкой L3 |

4. Алгоритм ветвей и границ

Для обхода дерева поиска в процессе решения задачи PCGTSP *(G, C,* Π*)*, мы используем метод поиска в ширину, см. Алгоритм 1. Каждый узел этого дерева связан с префиксом σ = (*Vi1, Vi2, … Vir*), где *Vij* ∈ *C*, Vi1=V1, и *r* ∈ {1*, …, m*}. Кластеры *Vij* посещаются в порядке, задаваемом последовательностью σ, все же остальные – в произвольном порядке (с соблюдением ограничений предшествования Π), тем самым образуя вспомогательную задачу *P* из раздела 3.1.

|  |  |
| --- | --- |
| **Алгоритм 1**. BnB :: Главная процедура | |
| Вход: | орграф *G*, кластеры *G*, частичный порядок Π |
| Выход: | маршрут и его стоимость |
|  | Инициализация *Q* = empty queue |
|  | Начинаем с *Root = V*1 |
|  | *Q.push(Root)* |
|  | **while** **not** *Q*.empty() |
|  | Берем следующий префикс для обработки σ = *Q*.pop() |
|  | *process=Bound*(σ) |
|  | **if** **not** *process* **then** |
|  | Префикс отсекается; **continue** |
|  | **end** **if** |
|  | *UpdateLowerBound*(σ) |
|  | **for all** *child* ∈ *Branch*(σ) **do** |
|  | Помещаем префикс в очередь на обработку *Q*.push(*child*) |
|  | **end** **for** |
|  | **end while** |

К каждому узлу дерева поиска мы применяем процедуру построения нижней оценки *Bound* (Алгоритм 2), выполняющую следующие действия:

* для префикса σ мы сопоставляем кортеж *T*(σ) = (*Vi1*, {*Vi1, Vi2, … Vir*}, *Vir*)
* на шаге 4, вычисляем матрицу *D*(σ) минимальных попарных расстояний по формуле:

Эта матрица легко вычисляется инкрементально с использованием матрицы *D*(σ’) родительского узла дерева поиска

* если, для некоторого σ1, *T*(σ) = *T*(σ1) и *D*(σ)*uv* ⩾ *D*(σ1)*uv*, *v* ∈ *Vi*1, *u* ∈ *Vir*

то префикс σ доминируется префиксом σ1 и подлежит отсечению

* на шаге 11, мы вычисляем оценки L1 и L2 (см. табл. 1) и сохраняем их в глобальной переменной *OptT*, используя формулу
* для текущего узла σ, рассчитываем на шаге 13 нижнюю границу по формуле
* наконец, узел σ отсекается, если *LB > UB*. Префиксы, которые избежали 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 обработке1 |
|  | **global** *DTij* |
|  | **global** *OptT* |
|  | вычисляем кортеж *T* = (*Vi1*, {*Vi1, Vi2, … Vir*}, *Vir*) |
|  | *Dij=MinCosts*(σ) |
|  | **if** *Dij* ⩾ *DTij* [*T*], ∀*i, j* **then** |
|  | **return** **false** |
|  | **end** **if** |
|  | обновляем веса маршрутов *DTij* [*T*] = min(*DTij* [*T*], *Dij*), ∀*i, j* |
|  | *cmin = minij Dij* |
|  | **if** *T* ∉ *OptT* **then** |
|  | вычисляем нижнюю границу *OptT*[*T*] = max(*L*1(σ), *L*2(σ)) |
|  | **end** **if** |
|  | *LB = cmin + OptT[T]* |
|  | **if** *LB > UB* **then** |
|  | **return false** |
|  | **end if** |
|  | **return true** |

5. Динамическое программирование

Основная идея алгоритма ветвей и границ, описанного в разделе 4, представляется близкой к классической схеме динамического программирования (DP) Хелда и Карпа [9], адаптированной к учету ограничений предшествования и дополненной стратегией оценивания, представленной в основополагающей статье [28].

В данной работе мы реализуем усовершенствованную версию этой схемы для проведения численного оценивания производительности описанного выше алгоритма ветвей и границ. Подобно классической схеме, наш 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 (ячейка таблицы динамического программирования) соответствует частичному *v*-*u*-пути и индексируется кортежем (*C’, Vl, v, u*), где

1. *C’* ⊂ *C* представляет собой *идеал* частично упорядоченного множества кластеров *C*, то есть ∀(*V*∈*C*’, *V*’∈*C*, (*V*’, *V*)∈*A*) ⇒ (*V*’∈*C*’); очевидно, в наших условиях, *V*1 принадлежит произвольному идеалу *C’* ⊂ *C*
2. *Vl ⊂ C’,* для которого нет *V∈C’,* такого, что *(Vl, V)∈A*
3. *v∈V1, u∈Vl*

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 ℑ*k* – подмножество идеалов размера *k* ∈ {1, …, *m*}. Очевидно, ℑ1 = {{*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 |
|  | Инициализация *R* = empty queue |
|  | **for** **all** *V* ∈ *C* **do** |
|  | *valid* = **true** |
|  | **for all** *W* ∈ *C* **do** |
|  | **if** *W=V* **or** (*V, W*) ∈ Π **then** |
|  | *valid* = **false** |
|  | **break** |
|  | **end** **if** |
|  | **end for** |
|  | **if** *valid* **then** |
|  | добавляем новый префикс *R*.push(σ + *V*) |
|  | **end if** |
|  | **end for** |
|  | **return** *R* |

5.1. Замечания

1. Оптимум для решаемой задачи дается классическим уравнением Беллмана
2. По построению, таблица DP имеет размер *O*(*n2m*|ℑ|), и значит, время работы нашего алгоритма *O*(*n3m2*|ℑ|). В частности, для частичного порядка фиксированной ширины *w*, |ℑ| = *O*(*mw*) [29]. Следовательно, в этом случае оптимальное решение PCGTSP может быть найдено за полиномиальное время даже без применения нижних оценок на шагах 10–12.
3. После построения произвольного слоя *Lk*, мы обновляем глобальное значение нижней оценки, что улучшает точность аппроксимации.
4. В нашей реализации для повышения быстродействия мы вычисляем оценку L3 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 , слой таблицы DP *Lk*, верхняя граница *UB* |
| Выход: | (*k*+1)-ый слой *Lk*+1 |
|  | Инициализация *Lk*+1 = ∅ |
|  | **for all** *C*’ ∈ℑk  **do** |
|  | **for** **all** кластер *Vl* ∈*C* \ *C*’ s.t. *C*’ ∪{*Vl*} ∈ ℑ*k*+1 **do** |
|  | **for** **all** *v* ∈ *V*1 и *u* ∈ *Vl* **do** |
|  | **if** есть состояние *S = (C’, U, v, w)* ∈ *Lk* s.t. (*w, u*) ∈ *E* **then** |
|  | создаем новое состояние *S’ = (C’ ∪ {Vl}, Vl, v, u)* |
|  | *S*’[*cost*]=min{*S*[*cost*] + *c*(*w, u*): *S* = (*C’, U ,v ,w*)∈*Lk*} |
|  | *S*’[*pred*]=argmin{*S*[*cost*] + *c*(*w, u*): *S* = (*C’, U, v, w*)∈*Lk*} |
|  | *S*’[*LB*]=*S*’[*cost*] + max{*L*1, *L*2, *L*3} |
|  | **if** *S*’[*LB*] ⩽ *UB* **then** |
|  | Добавляем *S*’ к *Lk*+1 |
|  | **end** **if** |
|  | **end** **if** |
|  | **end for** |
|  | **end for** |
|  | **end for** |
|  | **return** *Lk*+1 |

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 солвером Gurobi, применяемой к модели целочисленной линейной оптимизации (MILP), предложенной в работе [26].

6.1. Условия эксперимента

Все алгоритмы тестировались на общедоступной библиотеке PCGTSPLIB [25]. В качестве начального приближения всем алгоритмам задавалось одно и то же допустимое решение, полученное эвристикой PCGLNS [27]. Для всех алгоритмов вычисления проводятся на одном и том же оборудовании (16-ядерный Intel Xeon, 128G RAM) с предельным временем счета 10 часов. Для оценки точности вычислений мы используем верхнюю оценку относительной погрешности, вычисляемую по формуле

(2)

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

Таблица 2

Результаты экспериментов

(некоторые строки таблицы удалены для сокращения размера)

| Задача | | | | | Gurobi | | | Ветвей и границ | | | DP | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **ID** | **n** | **m** | **UB0** | **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** |

6.2. Обсуждение

Результаты экспериментов 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 *UB0*, 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 оценку погрешности, заданную в процентах. Задачи, в которых один из предлагаемых алгоритмов превосходит Gurobi по производительности, выделены жирным шрифтом. Как следует из табл. 2, для 13 из 39 задач (33%) один из наших алгоритмов показал рекордную производительность, в том числе в 12 случаях – по быстродействию, и в 7 – по точности.

Заметим, что предложенные алгоритмы смогли найти оптимальное решение в 6 из 39 случаях (хотя это не являлось целью данного эксперимента). Для 10 (15) постановок, 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 значительно уступают Gurobi, что, по-видимому, объясняется недостаточно точными нижними оценками. В то же время для задач *p43.4* и *ry48p.4* наши алгоритмы значительно превзошли 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 солверу Gurobi, как и тестируемым алгоритмам, было предоставлено хорошее стартовое решение, полученное нашей эвристикой, что, вообще говоря, не является обязательным при организации подобных сравнительных экспериментов.

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 и Карпа и подход Салмана к построению нижних оценок. Предложенные в данной работе алгоритмы реализованы в виде программного обеспечения 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, продемонстрировали их конкурентоспособность.

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 оптимизации 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.

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

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, No 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, No 3. — С. 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 // International Journal of Production Research. — 2018. — Т. 56, No 14. — С. 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 // International Journal of Production Research. — 2018. — Т. 56, No 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; 6th CIRP Conference on Assembly Technologies and Systems (CATS).
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. — Т. 26, No 2. — С. 449—459. — DOI: 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, No 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, No 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, No 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, No 2. — С. 234—251. — ISSN 0377-2217. — 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. — Т. 45, No 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, No 3. — С. 849—866. — ISSN 0377-2217. — 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. — Т. 4, No 4. — С. 319—329. — DOI: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, No 1. — С. 97—105. — ISSN 1531-8605. — DOI:10.1134/S0081543817090127.
17. Gutin G., Karapetyan D. A Memetic Algorithm for the Generalized Traveling Salesman Problem // Natural Computing. — 2010. — Т. 9, No 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. — Т. 13, No 1. — С. 56—75. — ISSN 1526-5528. — 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 // Proceedings of the Steklov Institute of Mathematics. — 2016. — Т. 295, No 1. — С. 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. — Т. 49, No 12. — С. 651—655. — ISSN 2405-8963. — DOI: /10.1016/j.ifacol.2016.07.767; 8thIFAC 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.
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, No 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, No 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, No 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, No 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.

References

1. Generalized Traveling Salesman Problem through N sets of nodes / Srivastava S., Kumar S, Garg R, and Sen P. // CORS journal. — 1969. — Vol. 7, no. 2. — P. 97–101.
2. Gutin Gregory, Punnen Abraham P. The Traveling Salesman Problem and Its Variations. — Boston, MA : Springer US, 2007. — ISBN:978-0-387-44459-8.
3. Castelino Kenneth, D’Souza Roshan, Wright Paul K. Toolpath optimization for minimizing airtime during machining //Journal of Manufacturing Systems. — 2003. — Vol. 22, no. 3. — P. 173 – 180. — Access mode: http://www.sciencedirect.com/science/article/pii/S0278612503900185.
4. Model of megalopolises in the tool path optimisation for CNC plate cutting machines / Chentsov Alexander G., Chentsov Pavel A., Petunin Alexander A., and Sesekin Alexander N. // International Journal of Production Research. — 2018. — Vol. 56, no. 14. — P. 4819–4830. — Access mode: https://doi.org/10.1080/00207543.2017.1421784.
5. Makarovskikh T.A., Panyukov A.V., Savitskiy E.A. Mathematical models and routing algorithms for economical cutting tool paths //International Journal of Production Research. — 2018. — Vol. 56, no. 3. — P. 1171–1188.
6. An Industrially Validated CMM Inspection Process with Sequence Constraints / Salman Raad, Carlson Johan S., Ekstedt Fredrik, Spensieri Domenico, Torstensson Johan, and Söderberg Rikard //Procedia CIRP. —2016. — Vol. 44. — P. 138 – 143. — 6th CIRP Conference on Assembly Technologies and Systems (CATS).
7. A Critical Review of Multi-hole Drilling Path Optimization / Dewil Reginald, Küçükoğlu ̇Ilker, Luteyn Corrinne, and Cattrysse Dirk // Archives of Computational Methods in Engineering. — 2019. — Vol. 26, no. 2. —P. 449–459.
8. Papadimitriou Christos. Euclidean TSP is NP-complete // Theoret. Comput. Sci. — 1977. — Vol. 4. —P. 237–244.
9. Held Michael, Karp Richard M. A Dynamic Programming Approach to Sequencing Problems // Journal ofthe Society for Industrial and Applied Mathematics. — 1962. — Vol. 10, no. 1. — P. 196–210. — Access mode: http://www.jstor.org/stable/2098806.
10. Laporte Gilbert, Semet Frédéric. Computational Evaluation Of A Transformation Procedure For The Sym-metric Generalized Traveling Salesman Problem //INFOR: Information Systems and Operational Research. —1999. — Vol. 37, no. 2. — P. 114–120.
11. Noon Charles E., Bean James C. An Efficient Transformation Of The Generalized Traveling Salesman Problem //INFOR: Information Systems and Operational Research. — 1993. — Vol. 31, no. 1. — P. 39–44.
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. — Vol. 219, no. 2. — P. 234–251.
13. Fischetti Matteo, González Juan José Salazar, Toth Paolo. A Branch-and-Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem //Operations Research. — 1997. — Vol. 45, no. 3. — P. 378–394.
14. A branch-and-cut algorithm for the generalized traveling salesman problem with time windows / Yuan Yuan, Cattaruzza Diego, Ogier Maxime, and Semet Frédéric //European Journal of Operational Research. — 2020. — Vol. 286, no. 3. — P. 849–866.
15. Feremans Corinne, Grigoriev Alexander, Sitters René. The geometric generalized minimum spanning tree problem with grid clustering //4OR. — 2006. — Vol. 4, no. 4. — P. 319–329. — Access mode: https://doi.org/10.1007/s10288-006-0012-6.
16. Khachai M. Yu., Neznakhina E. D. Approximation Schemes for the Generalized Traveling Salesman Problem //Proceedings of the Steklov Institute of Mathematics. — 2017. — Vol. 299, no. 1. — P. 97–105. — Access mode: https://doi.org/10.1134/S0081543817090127.
17. Gutin Gregory, Karapetyan Daniel. A Memetic Algorithm for the Generalized Traveling Salesman Problem //Natural Computing. — 2010. — Vol. 9, no. 1. — P. 47–60.
18. Helsgaun Keld. Solving the equality Generalized Traveling Salesman Problem using the Lin–Kernighan–Helsgaun Algorithm // Mathematical Programming Computation. — 2015. — P. 1–19.
19. Smith Stephen L., Imeson Frank. GLNS: An effective large neighborhood search heuristic for the Generalized Traveling Salesman Problem //Computers & Operations Research. — 2017. — Vol. 87. — P. 1–19
20. Balas Egon, Simonetti Neils. Linear Time Dynamic-Programming Algorithms for New Classes of Restricted TSPs: A Computational Study //INFORMS J. on Computing. — 2001. — Feb. — Vol. 13, no. 1. — P. 56–75. —Access mode: http://dx.doi.org/10.1287/ijoc.13.1.56.9748.
21. Chentsov A. G., Khachai M. Yu., Khachai D. M. An exact algorithm with linear complexity for a problemof visiting megalopolises //Proceedings of the Steklov Institute of Mathematics. — 2016. — Vol. 295, no. 1. —P. 38–46. — Access mode: https://doi.org/10.1134/S0081543816090054.
22. Chentsov Alexander, Khachay Michael, Khachay Daniel. Linear time algorithm for Precedence Constrained Asymmetric Generalized Traveling Salesman Problem //IFAC-PapersOnLine. — 2016. — Vol. 49, no. 12. —P. 651 – 655. — 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016.
23. Khachay Michael, Neznakhina Katherine. Towards Tractability of the Euclidean Generalized Traveling Salesman Problem in Grid Clusters Defined by a Grid of Bounded Height // Optimization Problems and Their Applications / ed. by Eremeev Anton, Khachay Michael, Kochetov Yury, Pardalos Panos. — Cham : Springer International Publishing, 2018. — Vol. 871 of Communications in Computer and Information Science. — P. 68–77. — ISBN:978-3-319-93799-1. — Access mode: https://doi.org/10.1007/978-3-319-93800-46.
24. Khachay Michael, Neznakhina Katherine. Complexity and approximability of the Euclidean Generalized Traveling Salesman Problem in grid clusters //Annals of Mathematics and Artificial Intelligence. — 2020. — Vol. 88, no. 1. — P. 53–69.
25. Salman Raad, Ekstedt Fredrik, Damaschke Peter. Branch-and-bound for the Precedence Constrained Generalized Traveling Salesman Problem //Operations Research Letters. — 2020. — Vol. 48, no. 2. — P. 163–166.
26. Khachay Michael, Kudriavtsev Andrei, Petunin Alexander. PCGLNS: A Heuristic Solver for the Precedence Constrained Generalized Traveling Salesman Problem// Optimization and Applications / ed. by Olenev Nicholas, Evtushenko Yuri, Khachay Michael, Malkova Vlasta. — Cham: Springer International Publishing. — 2020. —Vol. 12422 of Lecture Notes in Computer Science. — P. 196–208.
27. Kudriavtsev Andrei, Khachay Michael. PCGLNS: adaptive heuristic solver for the Precedence Constrained GTSP. — 2020. — Access mode: https://github.com/AndreiKud/PCGLNS/.
28. Morin Thomas L., Marsten Roy E. Branch-And-Bound Strategies for Dynamic Programming // Operations Research. — 1976. — Vol. 24, no. 4. — P. 611–627. — Access mode: http://www.jstor.org/stable/169764.
29. Steiner George. On the complexity of dynamic programming for sequencing problems with precedence constraints // Annals of Operations Research. — 1990. — Vol. 26, no. 1. — P. 103–123.
30. Ukolov Stanislav, Khachay Michael. Branch-and-Bound algorithm for the Precedence Constrained GTSP. —2021. — Access mode: <https://github.com/ukoloff/PCGTSP-BnB>.
31. A Cost Minimizing at Laser Cutting of Sheet Parts on CNC Machines/ Tavaeva Anastasia, Petunin Alexander, Ukolov Stanislav, and Krotov Vladimir // Mathematical Optimization Theory and Operations Research. — Cham, Switzerland: Springer, 2019. — P. 422–437.

1. В нашем варианте динамического программирования эта граница будет точной. [↑](#footnote-ref-1)