УДК 519.1 DOI: 10.20998/2413-3000.2021.3.8

В. Ф. ПРОКОПЕНКОВ

ПОЛИНОМИАЛЬНЫЙ АЛГОРИТМ ПОИСКА ГАМИЛЬТОНОВА ЦИКЛА НА ГРАФЕ

Предметом исследований является решение задачи поиска гамильтонова цикла на графе, которая в дискретной математике относится к NP классу сложности и по-прежнему сохраняет к себе интерес. Целью работы является разработка нового алгоритма решения этой задачи, гарантирующего нахождение оптимального решения с полиномиальными затратами времени. В работе [1] был выполнен анализ современного состояния проблемы, отмечены недостатки существующих методов решения, изложены новые принципы и метод нахождения решения. Известные методы решения задачи основаны на реализации перебора возможных вариантов решений или на интуитивных эвристиках. Методы с перебором решений неприемлемы по затратам так как характеризуются неполиномиальными затратами времени. Эвристические методы удовлетворительны по времени, но не гарантируют нахождение оптимального решения. Популярность методов перебора объясняется простотой линейного поиска в заранее известном множестве допустимых решений задачи. Но факториальная зависимость мощности множества решений (n-1)! от числа вершин графа n делает невозможным применение таких методов для задач большого размера на практике. Желание существенно снизить временные затраты приводит к попыткам обоснованного редуцирования множества перебора либо к разработке различных эвристик, что фактически свидетельствует о невозможности сформулировать условия нахождения оптимального решения задачи в целом. В данной статье представлены описание условий, определяющих нахождение оптимального решения задачи и полиномиальный алгоритм решения задачи, воплощающий описанный метод. Поиск оптимального решения задачи сводится к поиску замкнутого пути в новом графе кратчайших путей. Граф кратчайших путей строится на основе исходного графа задачи, для чего используется алгоритм Дейкстры. Множество перебора для определения оптимального решения задачи состоит из решений, которые строятся из каждой вершины графа в графе кратчайших путей. Размер этого множества оценивается как n(n-1). Разработанный параллельный алгоритм гарантирует отыскание оптимального решения, значительно сокращает исходное пространство поиска, позволяет находить решение с полиномиальной сложностью. Тестирование программы показало работоспособность разработанного метода и алгоритма решения задачи.

Ключевые слова: граф, граф кратчайших путей, гамильтонов цикл, сложность, NP-полнота, алгоритм Дейкстры, пространство поиска, множество допустимых решений, множество перебора, параллельная обработка, полиномиальный алгоритм.

В. П. ПРОКОПЕНКОВ

ПОЛІНОМІАЛЬНИЙ АЛГОРИТМ ПОШУКУ ГАМІЛЬТОНОВА ЦИКЛУ НА ГРАФІ

Предметом досліджень є вирішення задачі пошуку гамільтонова циклу на графі, яка в дискретній математиці відноситься до NP класу складності та як і раніше зберігає до себе інтерес. Метою роботи є розробка нового алгоритму вирішення цієї задачі, що гарантує знаходження оптимального рішення з поліноміальними витратами часу. У роботі [1] був виконаний аналіз сучасного стану проблеми, відзначені недоліки існуючих методів вирішення, викладені нові принципи і метод знаходження рішення. Відомі методи вирішення задачі засновані на реалізації перебору можливих варіантів рішень або на інтуїтивних евристиках. Методи з перебором рішень неприйнятні за витратами бо характеризуються неполіноміальними витратами часу. Евристичні методи задовільні за часом, але не гарантують знаходження оптимального рішення. Популярність методів перебору пояснюється простотою лінійногопошуку в заздалегідь відомій безлічі допустимих рішень задачі. Але факторіальна залежність потужності множини рішень (n-1)! від числа вершин графа n унеможливлює застосування таких методів для задач великого розміру на практиці. Бажання істотно знизити часові витрати призводить до спроб обгрунтованого скорочення множини перебору або до розробки різних евристик, що фактично свідчить про неможливість сформулювати умови знаходження оптимального рішення задачі в цілому. У даній статті представлені опис умов, що визначають знаходження оптимального рішення задачі та поліноміальний алгоритм рішення задачі, що втілює описаний метод. Пошук оптимального рішення задачі зводиться до пошуку замкнутого шляху в новому графі найкоротших шляхів. Граф найкоротших шляхів будується на основі вихідного графа задачі, для чого використовується алгоритм Дейкстри. Множина перебору для визначення оптимального розв'язку задачі складається з розв'язків, які будуються з кожної вершини графа в графі найкоротших шляхів. Розмір цієї множини оцінюється як n(n-1). Розроблений паралельний алгоритм гарантує відшукання оптимального рішення, значно скорочує вихідний простір пошуку, дозволяє знаходити рішення з поліноміальною складністю. Тестування програми показало працездатність розробленого методу і алгоритму вирішення завдання.

Ключові слова: граф, граф найкоротших шляхів, гамільтонів цикл, складність, NP-повнота, алгоритм Дейкстри, простір пошуку, множина допустимих рішень, иножина перебору, паралельна обробка, поліноміальний алгоритм.

V. PROKOPENKOV

POLYNOMIAL ALGORITHM FOR FINDING A HAMILTONIAN CYCLE ON A GRAPH

The subject of research is the solution of the problem of finding a Hamiltonian cycle on a graph, which in discrete mathematics belongs to the NP complexity class and still retains interest. The aim of this work is to develop a new algorithm for solving this problem, which guarantees finding the optimal solution with polynomial time. In [1], an analysis of the current state of the problem was performed, the shortcomings of existing methods of solving were noted, and new principles and method of finding solution were presented. Known methods for solving the problem are based on the implementation of a search of possible solutions or on intuitive heuristics. Enumeration methods of solutions are unacceptable in terms of costs, since they characterized by non-polynomial time. Heuristic methods are satisfactory in time, but they do not guarantee finding the optimal solution. The popularity of enumeration methods is explained by the linear simplicity of searching in a pre-known set of acceptable solutions to the problem. But the factorial dependence of the power of the set of solutions (n-1)! from the number of vertices of the graph n makes it impossible to use such methods for large-size problems in practice. The desire to significantly reduce time costs leads to attempts to reasonably reduce the enumeration set or to the development of various heuristics, which actually indicates that it is impossible to formulate conditions for finding the optimal solution to the problem as a whole. This article describes the conditions that determine the optimal solution of the problem and a polynomial algorithm for solving the problem that embodies the described method. Finding the optimal solution to the problem is reduced to finding a closed path in a new graph of shortest paths. The shortest paths graph built on the original graph of the problem by using Dijkstra's algorithm. The enumeration set for determining the optimal solution of the problem consists of solutions that are constructed from each vertex of the source graph in the shortest paths graph. The size of this set is

В. Ф. Прокопенков, 2021

estimated as n(n-1). The developed parallel algorithm guarantees finding the optimal solution, significantly reduces the initial search space, and allows you to find a solution with polynomial complexity. Testing of the program showed the efficiency of the developed method and algorithm for solving the problem.

Keywords: graph, shortest path graph, Hamiltonian cycle, complexity, NP-completeness, Dijkstra algorithm, search space, set of feasible solutions, enumeration set, parallel processing, polynomial algorithm.

Введение. Задача поиска замкнутого пути на графе возникла как игра «Кругосветное путешествие» по додекаэдру [2], предложенная У. Гамильтоном, в честь которого в последующем искомый замкнутый путь в графе носит название гамильтонова цикла [3]. Если граф содержит Гамильтонов цикл, такой граф называется гамильтоновым. И поиск гамильтонова цикла в графе, и проверка, является ли граф гамильтонов – обе эти задачи относятся к NP полным задачам [4]. Такая сложность объясняется размером возможного пространства решений задачи. Если рассматривать полный граф из n вершин, в таком графе существует $n(n-1)!$ вариантов допустимых решений. Если мы не имеем чётких критериев, чтобы отличить оптимальное решение от допустимого, единственным качественным отличием остаётся длина замкнутого пути. В таком случае, для отыскания лучшего решения придётся перебрать все возможные решения. Решение этой задачи актуально и для сегодняшнего дня и для практических задач производства и для развития науки. Поэтому очевидна важность продолжения исследований.

Анализ последних публикаций. Возможно, решение задачи поиска гамильтонова цикла можно построить и так. Сначала определить содержит ли граф гамильтонов цикл, а потом, в случае положительного ответа, как-то найти этот цикл. В теории имеются критерии для проверки, является ли граф гамильтоновым [5-7], но они не применимы для графов произвольной структуры [8] для большинства практических задач, таких как [9]. Проблема усугубляется ещё и тем, что искомый гамильтонов цикл должен быть оптимальным, т.е. иметь наименьшую длину пути.

На текущий момент не существует эффективных полиномиальных алгоритмов для решения задачи в оптимальной постановке [10]. Любой алгоритм воплощает определённую логику решения задачи. Если алгоритм гарантирует получение оптимального решения, он называется точным, иначе эвристическим [11].

В группу точных методов для решения рассматриваемой задачи входят методы [12,13] динамического программирования со сложность 2n и все переборные алгоритмы с экспоненциальной сложностью. Для переборных алгоритмов допустимые решения можно получать, используя комбинаторные методы, для которых гамильтонов цикл рассматривается как перестановка вершин, или используя алгоритмы обхода графа, например, алгоритм Робертса и Флореса [14]. Большие затраты на перебор требуют каким-то способом сокращения пространства перебора, например используя метод ветвей и границ [15] для отбрасывания заведомо не оптимальных решений, но без потери оптимального.

Эвристические алгоритмы имеют полиномиальную сложность, строятся на логически обоснованных идеях, например, муравьиный алгоритм, генетические и гибридные [16-20], но их проблемой является сложность подбора настраивающих параметров. Появляются новые алгоритмы, которые ориентированы на графы определённой структуры, например, для кубических графов в [21] было предложено решение со сложностью 1.26n, а в [22] сложность решения понижена до 1.251n.

В результате изучения состояния проблемы можно сделать следующие выводы. Причиной неудачи в разработке полиномиального метода решения задачи стала невозможность сформулировать условия нахождения оптимального решения задачи. Как следствие, основным способом решения по-прежнему остаётся перебор всех или большой части допустимых решений и использование интуитивных приёмов (эвристик).

Цель работы. На сегодняшний день не существует алгоритма решения рассматриваемой проблемы, удовлетворяющего требованиям:

- полиномиальная сложность;
- гарантированность оптимального решения;
- универсальность.

Разработка такого алгоритма является целью работы

Постановка задачи. Пусть задан граф $G = \langle V, E \rangle$, где $V = \left\{ v_i \mid i = \overline{1,n} \right\}$ – это множество вершин, а $E = \left\{ e_{ij} \mid i, j = \overline{1,n}, i \neq j \right\}$ – множество дуг графа. Дуга e_{ij} определяет наличие соединения между вершинами v_i и v_j, характеризуется расстоянием d_{ij}. Пусть задана начальная вершина $v_s \in V$.

Необходимо найти гамильтонов цикл минимальной длины из вершины v_s, т.е. кортеж $GC = < v_1, v_2, ..., v_k, ... v_{n-1}, v_n, v_{n+1} >$ из вершин графа G, для которого выполняются условия:

1) $v_1 = v_s$;

2) $v_{n+1} = v_s$;

3) для любой пары вершин $v_i, v_j \mid i, j \in \{\overline{1,n}\}$ справедливо: если $i \neq j$, то $v_i \neq v_j$;

4) для $\forall v_k \mid k \in \{\overline{2,n}\}$ в графе G существуют дуги: $e_{k-1,k}$ – из вершины v_{k-1} в вершину v_k и $e_{k,k+1}$ – из вершины v_k в вершину v_{k+1}.

Метод решения задачи. Принципы и метод решения задачи в сделанной постановке рассмотрены в роботе [1], которую можно рассматривать как один из этапов разработки целевого алгоритма. В данной работе проанализируем эти результаты, выполним логическое обоснование и представим новый разработанный алгоритм.

Анализ состояния проблемы приводит нас к следующей последовательности логически связанных положений, обосновывающей метод решения задачи:

(1) Для отдельно взятого произвольного допустимого решения $gc^x =< v_1^x, v_2^x, ..., v_k^x, ...v_{n-1}^x, v_n^x, v_{n+1}^x >$ невозможно по каким-либо его параметрам сделать заключение является ли это решение оптимальным.

(2) Для отдельно взятого произвольного допустимого решения gc^x рассматривая его совместно с другим допустимым решением gc^y невозможно сделать заключение, является ли это решение оптимальным.

(3) Рассматривая пару допустимых решений gc^x и gc^y можно сделать только заключение о не оптимальности gc^x, если длины циклов находятся в отношении $|gc^x| > |gc^y|$

(4) Невозможность сформулировать условия нахождения оптимального решения оставляет единственно возможную схему решения задачи – перебор пространства допустимых решений.

(5) Множество всех допустимых решений задачи составляет пространство поиска оптимального решения $\{gc^x\}$, в котором $gc^x =< v_1^x, v_2^x, ..., v_k^x, ...v_{n-1}^x, v_n^x, v_{n+1}^x >$ – это отдельное допустимое решение.

(6) Размер пространства поиска равный $|\{gc^x\}|$ факториально зависит от количества вершин исходного графа n и делает неэффективным схему перебора для отыскания оптимального решения при большом значении n.

(7) Если для задачи путём перебора пространства допустимых решений не удаётся за приемлемое время найти оптимальное решение единственно возможным способом остаётся сокращение пространства поиска.

(8) Переборная схема решения будет эффективна только при значительном сокращении исходного пространства поиска $\{gc^x\}$ до множества перебора $\{gc^x\}^{opt}$, размер которого $|\{gc^x\}^{opt}|$ будет иметь полиномиальную зависимость от n.

(9) Реализация переборной схемы требует наличия процедуры генерации каждого допустимого решения gc^x пространства поиска, а сокращение пространства поиска $\{gc^x\}$ до множества перебора $\{gc^x\}^{opt}$ требует исключения максимально возможного количество таких решений gc^x, для которых длина $|gc^x| > |GC|$.

(10) В существующих методах решения задачи допустимые решения gc^x строятся либо как комбинаторные перестановки номеров вершин графа, через которые проходит путь, либо как пути в исходном графе, формируемые пошагово. На каждом очередном шаге включается очередная вершина (или дуга к ней). При такой процедуре формирования очередного допустимого решения невозможно оценить качество решения до момента завершения его формирования.

(11) Оптимальным решением задачи является кратчайший замкнутый путь, который не может не состоять из элементарных кратчайших путей исходного графа.

(12) Для каждого допустимого решения gc^x, составленного не из кратчайших путей, выполняется отношение $|gc^x| > |GC|$, которое и определяет возможности сокращения исходного пространства поиска до пространства перебора, за счёт исключения таких решений из пространства поиска.

(13) Нет необходимости строить все допустимые решения gc^x, формировать пространство поиска $\{gc^x\}$, а затем сокращать пространство поиска до пространства перебора.

(14) Каждое допустимое решение gc^x должно строиться не из отдельных дуг графа, а из сегментов кратчайших путей исходного графа, объединяющих несколько вершин (и, возможно, чем больше вершин в сегменте, тем лучше).

(15) Пространство перебора для нахождения оптимального решения должно включать только те допустимые решения gc^x, которые строятся на основе кратчайших путей между вершинами в исходном графе.

(16) При построении каждого решения gc^x, включаемого в множество перебора необходимо использовать не исходный граф, а производный от него граф кратчайших путей.

(17) Чтобы не потерять оптимальное решение при формировании множества перебора необходимо построить все возможные допустимые решения, формирующие это множество, на основе графа кратчайших путей исходного графа.

(18) Нахождение оптимального решения сводится к выполнения этапов:

- построение графа кратчайших путей исходного графа,
- формирование допустимых решений, которые составят множество перебора $\{gc^x\}^{opt}$ для определения оптимального решения.
- перебор $\{gc^x\}^{opt}$ и определение оптимального решения GC.

Руководствуясь сформулированными положениями, формально опишем предложенный в [1] метод и разработаем алгоритм решения задачи.

На первом этапе, применяя для реализации первого этапа, например алгоритм Дейкстры [23], для

каждой вершины v_s в графе G мы получим множество кратчайших путей $\{path_{sk} \mid k \in \overline{1,n},\ k != s\}$, из этой вершины в другие вершины графа. В целом для всех вершин мы получим множество $\{\{path_{sk} \mid k \in \overline{1,n},\ k != s\} \mid s \in \overline{1,n}\}$ всех критических путей между каждой парой вершин в графе G. Этот результат решения первого этапа приводит нас к новому понятию графа кратчайших путей.

Формально граф кратчайших путей можно определить как $G^* = \langle V^*, E^* \rangle$, где $V^* = V$ – это множество вершин, совпадающее с множеством вершин исходного графа G, а $E^* = \left\{ e^*_{ij} \mid i, j = \overline{1,n}, i \neq j \right\}$ множество дуг этого графа. Дуга e^*_{ij} существует, если существует путь $path_{ij}$ кратчайшей длины d^*_{ij} в графе G из вершины v_i в вершину v_j.

На втором этапе, из кратчайших путей между вершинами в графе G мы должны сформировать решения gc^x, которые составят пространство перебора $\{gc^x\}^{opt}$ для поиска оптимального решения. Допустимые решения gc^x, включаемые в множество $\{gc^x\}^{opt}$, строятся на основе сформированного графа кратчайших путей G^* как циклические пути gc^* из начальной вершины v_s.

Построение цикла в G^* из начальной вершины v_s сводится к формированию кортежа $gc^* =< v_1, v_2, ...v_k ..., v_m > | \, v_1 = v_m = v_s,\ k \in \overline{1,m},$ $m \leq n+1, v_i \in V^*$ как последовательности вершин в графе G^*. В этой последовательности для $\forall v_k \mid k \in \{\overline{2,m-1}\}$ в графе G^* должны существовать дуги $e^*_{k-1,k} = path_{k-1,k}$ (представляет кратчайший путь из вершины v_{k-1} в вершину v_k) и $e^*_{k,k+1} = path_{k,k+1}$ – (представляет кратчайший путь из вершины v_k в вершину v_{k+1}). Построенное в графе G^* решение gc^* определяет соответствующее решение gc^x в графе G, которое является допустимым, если gc^x является гамильтоновым циклом, что требует проверки дополнительных условий в процессе построения решения.

Третий этап реализует перебор элементов $\{gc^x\}^{opt}$ и определение оптимального решения GC.

Обоснование метода решения задачи. Сделанное описание метода решения задачи даёт обобщённое представление о нём, для ясности выполним его формальное обоснования.

Определение 1. Простым описателем пути в графе G из вершины v_i в вершину v_j называется кортеж $s_{ij} =< v^{ij}_1, v^{ij}_2, ...v^{ij}_s ..., v^{ij}_k >$ такой, что $v^{ij}_1 = v_i$ – начальная вершина пути, а $v^{ij}_k = v_j$ – конечная вершина пути и для $\forall v^{ij}_s \mid s \in \{\overline{2,k-1}\}$ в графе G существуют дуги из вершины v^{ij}_{s-1} в вершину v^{ij}_s и из вершины v^{ij}_s в вершину v^{ij}_{s+1}.

Множество вершин, через которые проходит путь s_{ij} будем обозначать как P^{ij}.

Простой описатель пути в графе G при ограниченном k определяет конечный путь из одной вершины в другую.

Определение 2. Путь $s_{ij} =< v^{ij}_1, v^{ij}_2, ...v^{ij}_s ..., v^{ij}_k >$ является бесконтурным, если для любой пары вершин v^{ij}_x, v^{ij}_z кортежа s_{ij} из утверждения $x \neq z$ следует выполнение условия $v^{ij}_x \neq v^{ij}_z$.

Определение 3. Путь $s_{ij} =< v^{ij}_1, v^{ij}_2, ...v^{ij}_s ..., v^{ij}_k >$ является замкнутым, если выполняется $v^{ij}_1 = v^{ij}_k$ и при удалении вершины v^{ij}_k он становится бесконтурным.

Определение 4. Структурным описателем пути в графе G из вершины v_i в вершину v_j называется кортеж $< s_1, s_2, ...s_p ..., s_r > | 1 \leq p \leq r, r \leq N$, в котором каждый сегмент $s_p =< v^p_1, v^p_2, ..., v^p_k >$ – это простой описатель пути и выполняются условия:

1) $v^1_1 = v_i$,

2) $v^r_k = v_j$,

3) для каждого сегмента $s_p \mid p \in \{\overline{2,r}\}$ выполняется $v^{p-1}_k = v^p_1$.

Поскольку и структурный и простой описатели определяют путь, должна существовать возможность замены структурного описателя на простой.

Пусть кортеж $c =< s_1, s_2, ...s_p ..., s_r > | 1 \leq p \leq r, r \leq N$ по форме является структурным описателем пути, тогда если это так, алгоритм 1 выполнит его преобразование в простой описатель пути.

Алгоритм 1.

П. 1. Положим текущий путь $path =< v^1_1, v^1_2, ..., v^1_k >$ как путь, соответствующий сегменту s_1 в кортеже c.

П. 2. Выполнить цикл по переменной $p = \overline{2,r}$:

Если к пути $path =< v^{path}_1, v^{path}_2, ..., v^{path}_{k-1}, v^{path}_k >$ можно добавить путь сегмента $s_p =< v^p_1, v^p_2, ..., v^p_k >$, т.е. выполняется условие $v^{path}_{k-1} = v^p_1$ – конечная вершина текущего пути совпадает с начальной вершиной добавляемого пути, то положить: $path =< v^{path}_1, v^{path}_2, ..., v^{path}_{k-1}, v^p_1, v^p_2, ..., v^p_k >$, иначе c – это не описатель пути и перейти к п. 3.

П. 3. Остановиться.

Утверждение 1. Для существования гамильтонова цикла в графе G необходимо, чтобы для любой пары вершин $v_i, v_i \mid i \neq j$ в графе G существовал путь из вершины v_i в вершину v_j.

Доказательство:

Гамильтоновым циклом является такой замкнутый путь, который проходит через все вершины этого графа. Если гамильтонов цикл существует, то существует и путь из любой вершины v_i в любую вершину v_j.

Предположим, что в графе G имеется гамильтонов цикл, и нет пути из вершины v_i в вершину v_j. Но в этом случае цикл размыкается, а это противоречит исходному предположению, т.е. утверждение доказано.

Сделанные определения определяют форму описания результата решаемой задачи и упрощают описание метода.

На первом этапе решения задачи формируется матрица простых описателей кратчайших путей между всеми возможными парами вершин $v_i, v_j \in V$ графа G:

$$M_{cp}^{G} = \{s_{ij} \mid i, j \in \{\overline{1, N}\}\},$$

где $s_{ij} =< v_1^{ij}, v_2^{ij}, ...v_s^{ij} ..., v_k^{ij} >$ – это простой описатель кратчайшего пути в графе G из вершины v_i в вершину v_j с расстоянием d_{ij}^{*}.

На втором этапе на основе матрицы M_{cp}^{G} будем конструировать структурный описатель искомого гамильтонова цикла:

$$c =< s_1, s_2, ...s_p ..., s_r > \mid 1 \le p \le r, r \le N,$$

в котором $s_p \in M_{cp}^{G}$.

Для действенности метода необходимо определить обоснованный способ конструирования структурного описателя гамильтонова цикла.

Утверждение 2. Для гамильтонова графа G матрица наикратчайших путей M_{cp}^{G} определяет полный граф G^{*} наикратчайших путей.

Доказательство:

Определим формально граф $G^{*} = \langle V^{*}, E^{*} \rangle$ так, что: $V^{*} = V$ – это множество вершин, совпадающее с множеством вершин исходного графа G, а $E^{*} = \left\{ e_{ij}^{*} \mid i, j = \overline{1, n}, i \neq j \right\}$ множество дуг. Дуга e_{ij}^{*} существует, если существует путь в графе G из вершины v_i в вершину v_j. Каждой дуге e_{ij}^{*} в графе G^{*} соответствует описатель $s_{ij} =< v_1^{ij}, v_2^{ij}, ..., v_k^{ij} >$ матрицы M_{cp}^{G} с длинной d_{ij}^{*}. Если гамильтонов цикл в графе G существует, то как следует из утверждения 1 существуют пути для любой пары отличных вершин графа G, и как результат – определённый граф G^{*} является полным.

Используя утверждение 2 можно говорить, что конструирование структурного описателя замкнутого пути для заданной начальной и конечной вершины v_s сводится к решению задачи поиска пути в графе G^{*}. Стартуя из начальной вершины, v_s будем искать замкнутый путь в графе G^{*} и формировать структурный описатель этого пути. Заметим, что конструирование структурного описателя процедурой поиска пути в графе G^{*} происходит шаг за шагом. Чтобы правильно построить эту процедуру сформулируем утверждение.

Утверждение 3. Чтобы структурный описатель $c =< s_1, s_2, ...s_p ..., s_r >$ определял гамильтонов цикл в графе G необходимо и достаточно выполнение четырёх условий:

1. Условие стыкуемых сегментов.

Для того чтобы сегмент s_l мог стыковаться в конец сегмента s_p $(l = p+1)$ для образования пути необходимо выполнение условия:

$$P^{p} \cap P^{l} = v_1^{l} = v_k^{p}.$$

Это условие выражает свойство соседних сегментов пути. Соседние сегменты имеют единственную общую вершину – для одного из них s_p (сегмент, к которому стыкуется другой сегмент) эта вершина является конечной вершиной, а для другого s_{p+1} (сегмент, который стыкуется) эта вершина является начальной вершиной сегмента.

2. Условие не стыкуемых сегментов.

Для каждой пары не стыкуемых сегментов $s_p, s_l \mid l \neq p+1, p \neq l+1$ в $c =< s_1, s_2, ...s_p ..., s_r >$ необходимо выполнение условия:

$$P^{p} \cap P^{l} = \otimes.$$

Это условие утверждает, что сегменты-пути в графе G, которые не являются соседними в структурном описателе, определяющем гамильтонов цикл, должны проходить через разные вершины.

3. Условие полноты пути.

Для всех сегментов описателя пути $c =< s_1, s_2, ...s_p ..., s_r >$ образующих гамильтонов цикл необходимо выполнение условия:

$$\bigcup_{p=1}^{r} P^{p} = V.$$

Это условие утверждает, что путь, определяемый описателем $c =< s_1, s_2, ...s_p ..., s_r >$, проходит через все вершины графа G.

4. Условие замкнутости пути.

Для того, чтобы описатель $c =< s_1, s_2, ...s_p..., s_r >$ определял замкнутый путь (цикл) необходимо выполнение условия стыкуемых сегментов для пары сегментов s_r и s_1 (конечная вершина последнего сегмента описателя c, определяющего гамильтонов цикл, совпадает с начальной вершиной первого сегмента).

Доказательство:

Предположим, что в результате поиска мы сформировали структурный описатель предполагаемого гамильтонова цикла $c =< s_1, s_2, ...s_p..., s_r >$, в котором каждый сегмент $s_p =< v_1^p, v_2^p, ..., v_k^p >$ – это простой описатель пути в графе через множество вершин $P^p = \{v_1^p, v_2^p, ..., v_k^p\} \mid P^p \subset V$. Используя алгоритм 1, преобразуем описатель c в путь в графе G, который он определяет, получим:

$$path =< v_1^1, v_2^1, ..., v_{k-1}^1, v_1^2, v_2^2, ..., v_{k-1}^2, ...v_1^p, v_2^p, ..., v_{k-1}^p ..., v_1^r, v_2^r, ..., v_{k-1}^r > .$$

Заметим, что если условие стыкуемых сегментов было бы нарушено, то алгоритм 1 не выдал бы результат. А значит, необходимость выполнения условия 1 обоснована.

Полученный кортеж $path$ определяет последовательность вершин, составляющих путь. Чтобы полученный путь был гамильтоновым циклом из постановки задачи необходимо:

1) чтобы начальная и конечная вершина пути совпадали, т.е. необходимость условия замкнутости пути обоснована.

2) любая пара вершин (кроме начальной и конечной вершины для стыкуемых сегментов) должна включать разные вершины, а значит выполнения условия не стыкуемых сегментов обосновано.

3) условие наличия дуг в графе G, связывающих вершины в пути гамильтонова цикла обеспечивается формированием простых описателей кратчайших путей и условием стыкуемых сегментов.

4) необходимо чтобы путь проходил через все вершины графа, что обосновывает необходимость выполнения условия полноты пути.

Таким образом, необходимость выполнения перечисленных в утверждении условий доказана.

Доказательство достаточности этих условий для того чтобы структурный описатель определял гамильтонов цикл также не вызывает сомнений, так как указанные условия в совокупности соответствуют определению гамильтонова цикла.

Построение допустимых решений в графе G^* не проще чем в графе G. Но вводя в рассмотрение граф, G^* мы переводим синтез гамильтоновых циклов с уровня простых дуг графа G на уровень кратчайших путей в графе G^*, чем исключаем из пространства поиска заведомо не оптимальные решения.

Алгоритм решения задачи. Было бы логично сначала изложить последовательный алгоритм, воплощающий предложенный метод решения задачи, проанализировать его логику, выделить подзадачи, которые могут выполняться параллельно, а затем как результат представить параллельный алгоритм. Для сокращения объёма текста далее приводится только параллельная версия алгоритма, который воплощает изложенный метод поиска гамильтонова цикла.

Исходными данными для алгоритма 2, который формирует пространство перебора для определения оптимального решения, является исходный граф G. Результат работы алгоритма $\{gc^x\}^{opt}$ – найденные циклы, записываются в список res_list (изначально пуст).

Алгоритм 2.

П.1. Для каждой вершины $A \in \{\overline{1, N}\}$ организовать поток t_A для вычисления кратчайших путей из вершины A в любую другую вершину графа G и формирования для них простых описателей $M_{cp}^G = \{s_{Aj} \mid j \in \{\overline{1, N}\}\}$, используя алгоритм Дейкстры.

П.2. Запустить и ожидать завершения выполнения всех потоков t_A.

П.3. Если необходимое условие существования гамильтонова цикла не выполняется (см. утверждение 1), то гамильтонов цикл не существует, перейти к п. 7.

П.4. Для каждого элемента $M_{cp}^G[a,b] \mid a \neq b$ кратчайшего пути из вершины a в вершину b организовать поток t_b для нахождения возможно существующего гамильтонова цикла используя алгоритм 3.

П.5. Запустить и ожидать завершения выполнения всех потоков t_b.

П.6. В списке res_list найти цикл с наименьшим значением длины и вывести результат.

П.7. Остановиться.

На втором этапе реализуемого метода решения задачи для каждого элемента $M_{cp}^G[a,b] \mid a \neq b$ выполняется построение гамильтонова цикла в графе кратчайших путей G^* алгоритмом 3 как возможного решения задачи, включаемого в пространство перебора res_list. В качестве исходных данных этот алгоритм 3 получает:

a – номер начальной вершины,

b – номер конечной вершины кратчайшего пути из вершины a, используя которые будет строиться гамильтонов цикл.

Алгоритм 3.

П. 1. Проверить существует ли в M_{cp}^{G} путь s_{ba} из вершины b в вершину a такой, что вместе с путём s_{ab} из вершины a в вершину b образуется гамильтонов цикл. Если существует, объединить эти пути в результирующий гамильтонов цикл $c =< s_{ab}, s_{ba} >$, который добавить в список найденных решений res_list. Перейти к п. 5.

П. 2. Создать текущий путь $c =< s_{ab} >$.

П. 3. Пока $P^c \neq V^*$, т.е. не все вершины графа содержатся в пути c выполнять цикл:

3.1. Положить $a = v_1^c$ – начальная вершина пути c; $b = v_k^c$ – конечная вершина пути c.

3.2. Найти такую вершину $x \notin P^c$, не используемую в пути c такую, что в матрице M_{cp}^{G} существует путь s_{bx}, который можно добавить к пути c для формирования гамильтонова цикла.

3.3. Если вершина x не найдена, то перейти к п.5.

3.4. Если для описателей в c и s_{bx} выполняются необходимые условия формирования гамильтонова цикла, создать новый путь $c = c+ < s_{bx} >$ добавлением к c пути s_{bx}.

П. 4. Если путь c удовлетворяет всем условиям гамильтонова цикла, добавить цикл c в список найденных решений res_list.

П.5. Остановиться.

Реализация третьего этапа сводится к нахождению оптимального решения GC такого, что выполняется

$$GC \in res\_list = \{gc^x\}^{opt} \mid для \forall gc^x \in res\_list,\ gc^x \neq GC,\ GC < gc^x .$$

Доказательство корректности алгоритма. Для доказательства правильности алгоритма важно доказать правильность формирования множества перебора $\{gc^x\}^{opt}$. Попытаемся интуитивно обосновать правильность выбранного способа решения.

Чтобы не потерять оптимальное решение при формировании $\{gc^x\}^{opt}$, необходимо построить различные допустимые решения, определяемые графом G^*, не потеряв оптимального. Для заданной начальной вершины v_s эту задачу можно решить обходом графа G^* начиная с вершины v_s, реализующим перебор вариантов путей. Например, это можно сделать, используя схему с возвратами, сложность которого оценивается как $(n-1)!$.

Заметим, что в [1] было показано, что не для всякой выбранной вершины в качестве начальной для гамильтонова графа G, удаётся построить допустимое решение. Чтобы для гамильтонова графа не потерять решение, допустимые решения строятся, рассматривая каждую вершину как начальную. Но в этом случае затраты возрастут до значения $n!$.

Как способ сокращения затрат [24] для каждой начальной вершины и каждой исходящей из этой вершины дуги в графе G^* формируется единственное первое находимое решение (если оно существует). При таком способе в процессе решения задачи строится всего $n(n-1)$ решений и $|\{gc^x\}^{opt}$ имеет полиномиальную зависимость от числа вершин исходного графа. Но остаётся вопрос – попадает ли оптимальное решение задачи при такой схеме в $\{gc^x\}^{opt}$? Надежду, что оптимальное решение не теряется, дают следующие свойства разработанного метода и алгоритма решения.

Возможная потеря допустимых решений связана с тем, что при их формировании для начальной вершины и исходящей из неё дуги мы отказываемся от продолжения перебора вариантов, если получим (или не получим) первое допустимое решение. Проанализируем, как это влияет на результат.

Дуга e_{ij}^* в графе G^* существует, если в графе G из вершины v_i в вершину v_j .существует путь $path_{ij}$ кратчайшей длины d_{ij}^*. Каждой дуге e_{ij}^* в графе G^* соответствует описатель $s_{ij} =< v_1^{ij}, v_2^{ij}, ..., v_k^{ij} >$. Поэтому, можно записать $e_{ij}^* \equiv s_{ij} \equiv path_{ij} \equiv < v_1^{ij}, v_2^{ij}, ..., v_k^{ij} >$.

Дуги e_{ti}^* и e_{tj}^* ($i \neq j$), исходящие из вершины v_t, в графе G^* находятся в отношении зависимости, если путь $path_{ti}$ является частью пути $path_{tj}$ в графе G либо наоборот $path_{tj}$ является частью пути $path_{ti}$. Фактически, отношение зависимости между дугами исходящими из одной и той же вершины v_t можно проверить сравнением описателей путей, соответствующим этим дугам. Если описатель пути первой дуги полностью является частью описателя второй дуги, то дуги являются зависимые. При этом, путь, определяемый первой дугой является частью пути, определяемой второй дугой.

В графе кратчайших путей G^* для каждой его вершины v все исходящие из этой вершины кратчайшие пути делятся на зависимые и независимые дуги. При построении пути из вершины v_t выбор любого варианта продолжения пути из множества зависимых дуг, ведущих из вершины v_t, является эквивалентным. Это очевидно, если дуги зависимые, то путь, который определяет первая дуга, является частью пути, определяемого второй дугой. Как следствие путь в графе, определяемый второй дугой из вершины v_t можно реализовать либо этой дугой, либо первой и дополнительными другими дугами,

которые повторят этот путь. Таким образом, отказ от перебора на множестве зависимых дуг исходящих из вершины не влечёт потери допустимых решений.

Остаётся проанализировать последствия от отказа перебора на множестве независимых дуг исходящих из вершины. Если учесть, что при формировании множества перебора мы для каждой вершины, рассматривая её как начальную выполним формирование допустимого решения для каждой исходящей из неё дуги, то остаётся надежда, что отказ от перебора независимых дуг также не повлечёт потери допустимых решений.

Дополнительно необходимо отметить, что вариантность перебора при формировании замкнутого пути на графе G^* существенно сокращается естественным образом, благодаря требованию выполнения условий для стыкуемых и нестыкуемых сегментов при формировании пути как допустимого решения. При выборе очередного сегмента для продолжения пути, если не выполняются условия стыкуемых и нестыкуемых сегментов (вследствие пересечения на вершинах, через которые проходят пути), сегмент не выбирается для включения в путь, а значит, сразу отсекаются все возможные варианты продолжения перебора.

Сделанные рассуждения позволяют надеяться на то, что в множество перебора $\{gc^x\}^{opt}$ попадут все возможные допустимые решения построенные на графе G^* и оптимальное решение не будет потеряно.

Таким образом, на графе кратчайших путей G^* с n вершинами предложенным алгоритмом можно построить не более $n(n-1)$ различных допустимых решений, а значит, поиск можно выполнить за полиномиальное время.

Тестирование алгоритма. Для проверки разработанного решения была выполнена программная реализация на языке C#, которая подтвердила работоспособность разработанного метода решения. Тестирование было выполнено на полном и неполном графах, сгенерированных программным способом [25].

В качестве иллюстрации далее приводится результат работы разработанной программы, которая для полного графа G из 6 вершин (табл.1,2), заданного координатным способом на плоскости, в сформированном ею соответствующем графе кратчайших путей G^* (табл.1,3) построила множество допустимых решений $\{gc^x\}^{opt}$ (табл. 4) для нахождения оптимального решения, представленного на рисунке 1.

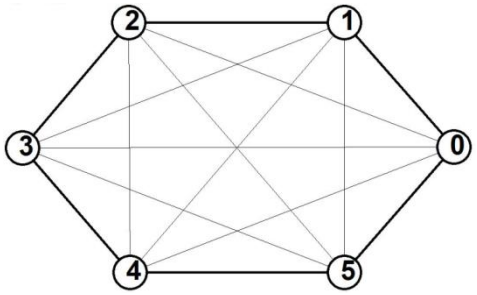

Рис. 1. Найденное оптимальное решение

Таблица 1 – Вершины графа G и G^*

Вершина		
№	Координаты	
	x	y
0	300	150
1	224	279
2	74	279
3	0	149
4	75	20
5	224	20

Таблица 2 – Дуги графа G

ij	0	1	2	3	4	5
0	∞	149	260	300	259	150
1	149	∞	150	258	298	259
2	260	150	∞	149	259	299
3	300	258	149	∞	149	258
4	259	298	259	149	∞	149
5	150	259	299	258	149	∞

Таблица 3 – Дуги графа G^*

ij	0	1	2	3	4	5
0	s_{00} =< >(∞)	s_{01} =< 0, 1 >(149)	s_{02} =< 0, 2 >(260)	s_{03} =< 0, 3 >(300)	s_{04} =< 0, 4 >(259)	s_{05} =< 0, 5 >(150)
1	s_{10} =< 1, 0 >(149)	s_{11} =< >(∞)	s_{12} =< 1, 2 >(150)	s_{13} =< 1, 3 >(258)	s_{14} =< 1, 4 >(298)	s_{15} =< 1, 5 >(259)
2	s_{20} =< 2, 0 >(260)	s_{21} =< 2, 1 >(150)	s_{22} =< >(∞)	s_{23} =< 2, 3 >(149)	s_{24} =< 2, 4 >(259)	s_{25} =< 2, 5 >(299)
3	s_{30} =< 3, 0 >(300)	s_{31} =< 3, 1 >(258)	s_{32} =< 3, 2 >(149)	s_{33} =< >(∞)	s_{34} =< 3, 4 >(149)	s_{35} =< 3, 5 >(258)
4	s_{40} =< 4, 0 >(259)	s_{41} =< 4, 1 >(298)	s_{42} =< 4, 2 >(259)	s_{43} =< 4, 3 >(149)	s_{44} =< >(∞)	s_{45} =< 4, 5 >(149)
5	s_{50} =< 5, 0 >(150)	s_{51} =< 5, 1 >(259)	s_{52} =< 5, 2 >(299)	s_{53} =< 5, 3 >(258)	s_{54} =< 5, 4 >(149)	s_{55} =< >(∞)

Таблица 4 – Найденное множество допустимых решений $\{gc^x\}^{opt}$

C_X	Множество сформированных допустимых решений $\{gc^x\}^{opt}$		Длина цикла
	Структурный описатель цикла $< s_1, s_2, \ldots s_p \ldots, s_r >$	$gc^x = < v_1, v_2, \ldots, v_k, \ldots, v_n, v_{n+1} >$	
1	$< s_{01}, s_{12}, s_{23}, s_{34}, s_{45}, s_{50} >$	< 0, 1, 2, 3, 4, 5, 0 >	896
2	$< s_{02}, s_{21}, s_{13}, s_{34}, s_{45}, s_{50} >$	< 0, 2, 1, 3, 4, 5, 0 >	1116
3	$< s_{03}, s_{31}, s_{12}, s_{24}, s_{45}, s_{50} >$	< 0, 3, 1, 2, 4, 5, 0 >	1266
4	$< s_{04}, s_{41}, s_{12}, s_{23}, s_{35}, s_{50} >$	< 0, 4, 1, 2, 3, 5, 0 >	1264
5	$< s_{05}, s_{51}, s_{12}, s_{23}, s_{34}, s_{40} >$	< 0, 5, 1, 2, 3, 4, 0 >	1116
6	$< s_{10}, s_{02}, s_{23}, s_{34}, s_{45}, s_{51} >$	< 1, 0, 2, 3, 4, 5, 1 >	1115
7	$< s_{14}, s_{40}, s_{02}, s_{23}, s_{35}, s_{51} >$	< 1, 4, 0, 2, 3, 5, 1 >	1483
8	$< s_{12}, s_{20}, s_{03}, s_{34}, s_{45}, s_{51} >$	< 1, 2, 0, 3, 4, 5, 1 >	1267
9	$< s_{13}, s_{30}, s_{02}, s_{24}, s_{45}, s_{51} >$	< 1, 3, 0, 2, 4, 5, 1 >	1485
10	$< s_{15}, s_{50}, s_{02}, s_{23}, s_{34}, s_{41} >$	< 1, 5, 0, 2, 3, 4, 1 >	1265
11	$< s_{21}, s_{10}, s_{03}, s_{34}, s_{45}, s_{52} >$	< 2, 1, 0, 3, 4, 5, 2 >	1196
12	$< s_{20}, s_{01}, s_{13}, s_{34}, s_{45}, s_{52} >$	< 2, 0, 1, 3, 4, 5, 2 >	1264
13	$< s_{23}, s_{30}, s_{01}, s_{14}, s_{45}, s_{52} >$	< 2, 3, 0, 1, 4, 5, 2 >	1344
14	$< s_{24}, s_{40}, s_{01}, s_{13}, s_{35}, s_{52} >$	< 2, 4, 0, 1, 3, 5, 2 >	1482
15	$< s_{25}, s_{50}, s_{01}, s_{13}, s_{34}, s_{42} >$	< 2, 5, 0, 1, 3, 4, 2 >	1264
16	$< s_{30}, s_{01}, s_{12}, s_{24}, s_{45}, s_{53} >$	< 3, 0, 1, 2, 4, 5, 3 >	1265
17	$< s_{32}, s_{20}, s_{01}, s_{14}, s_{45}, s_{53} >$	< 3, 2, 0, 1, 4, 5, 3 >	1263
18	$< s_{31}, s_{10}, s_{02}, s_{24}, s_{45}, s_{53} >$	< 3, 1, 0, 2, 4, 5, 3 >	1333
19	$< s_{34}, s_{40}, s_{01}, s_{12}, s_{25}, s_{53} >$	< 3, 4, 0, 1, 2, 5, 3 >	1264
20	$< s_{35}, s_{50}, s_{01}, s_{12}, s_{24}, s_{43} >$	< 3, 5, 0, 1, 2, 4, 3 >	1115
21	$< s_{40}, s_{01}, s_{12}, s_{23}, s_{35}, s_{54} >$	< 4, 0, 1, 2, 3, 5, 4 >	1114
22	$< s_{43}, s_{30}, s_{01}, s_{12}, s_{25}, s_{54} >$	< 4, 3, 0, 1, 2, 5, 4 >	1196
23	$< s_{42}, s_{20}, s_{01}, s_{13}, s_{35}, s_{54} >$	< 4, 2, 0, 1, 3, 5, 4 >	1333
24	$< s_{41}, s_{10}, s_{02}, s_{23}, s_{35}, s_{54} >$	< 4, 1, 0, 2, 3, 5, 4 >	1263
25	$< s_{45}, s_{50}, s_{01}, s_{12}, s_{23}, s_{34} >$	< 4, 5, 0, 1, 2, 3, 4 >	896
26	$< s_{50}, s_{01}, s_{12}, s_{23}, s_{34}, s_{45} >$	< 5, 0, 1, 2, 3, 4, 5 >	896
27	$< s_{52}, s_{20}, s_{01}, s_{13}, s_{34}, s_{45} >$	< 5, 2, 0, 1, 3, 4, 5 >	1264
28	$< s_{51}, s_{10}, s_{02}, s_{23}, s_{34}, s_{45} >$	< 5, 1, 0, 2, 3, 4, 5 >	1115
29	$< s_{53}, s_{30}, s_{01}, s_{12}, s_{24}, s_{45} >$	< 5, 3, 0, 1, 2, 4, 5 >	1265
30	$< s_{54}, s_{40}, s_{01}, s_{12}, s_{23}, s_{35} >$	< 5, 4, 0, 1, 2, 3, 5 >	1114

Поскольку алгоритм Дейкстры для $n=|V|$ имеет сложность $O(n^2)$, то сложность формирования матрицы критических путей составляет порядок $O(n^3)$. Для каждой из n вершин алгоритм 3 выполняется $n-1$ раз. Сложность алгоритма 3 не превышает $O(n^2)$. В итоге сложность алгоритма 2 в худшем случае имеет оценку $O(n^4)$. А значит, суммарная сложность алгоритма поиска гамильтонова цикла не превышает $O(n^4)$. Указанная оценка сделана в предположении последовательного выполнения вычислений. Параллельная реализация вычисления позволяет сократить время вычислений.

Выводы. В статье представлен новый алгоритм поиска гамильтонова цикла на графе, который программно реализован и протестирован. Алгоритм обеспечивает нахождение оптимального решения задачи за полиномиальное время и имеет сложность порядка $O(n^4)$.

Список литературы

1. Прокопенков В. Ф. Новый метод поиска гамильтонова цикла на графе. *Вісник Національного технічного університету «ХПІ». Серія: Стратегічне управління, управління портфелями, програмами та проектами.* 2020. № 2, С.43-49. doi.org/10.20998/2413-3000.2020.2.6
2. Акимов О. Е. *Дискретная математика. Логика, группы, графы, фракталы*. Москва, 2005. 656 с.
3. Емеличев В. А., Ковалев М. М., Кравцов М. К. *Многогранники, графы, оптимизация*. Москва. 1981. 341 с.
4. Гери М., Джонсон Д. *Вычислительные машины и трудноpешаемые задачи*. Москва, 1982. 416 с.
5. Дойбер В.А., Косточка А. В., Закс Х. Более короткое доказательство теоремы Дирака о числе ребер в хроматически критических графах. *Дискретный анализ и исследование операций.* Новосибирский гос.ун-т, 1996. с. 28–34.
6. Оре О. *Теория графов*. Москва, 1980. 336 с.
7. *Гамильтонов граф: сайт*. – URL : https://ru.wikipedia.org/wiki/Гамильтонов_граф. (дата обращения : 4.10.2019).
8. Павленко В. Б. Теоретические аспекты построения гамильтонова цикла. *Теорія оптимальних рішень*. 2011, №10. с. 150–155. сайт . URL : http://dspace.nbuv.gov.ua/bitstream/handle/123456789/46787/22-Pavlenko.pdf?sequence=1 (дата обращения : 4.10.2019).
9. Прокопенков В. Ф., Кожин Ю. Н., Малых О. Н. Определение оптимального кольцевого маршрута, проходящего через заданное множество пунктов на карте. *Innovative technologies and scientific solutions for industries*. 2019. No.1 № 7. С. 102–112. doi.org/10.30837/2522-9818.2019.7.102
10. Харари Ф. *Теория графов*. Москва, 1973. 300 с.
11. Стивенс Р. Алгоритмы. *Теория и практическое применение*. Москва, 2016. 544 с.
12. Bellman R. Dynamic Programming Treatment of the Travelling Salesman Problem. *Journal of the ACM*. 1962. Vol.9 № 1, p. 61–63. doi.org/10.1145/321105.321111
13. Held M. The travelling-salesman problem an minimum spanning trees. *Operations Research*. 1970. Vol. 18 № 6, p. 1138–1162. doi.org/10.1287/opre.18.6.1138
14. Roberts S. M., Flores B. An engineering approach to the travelling salesman problem. *Management Science*. 1967. Vol. 13 № 3, p. 269–288. doi.org/10.1287/mnsc.13.3.269
15. Little J. D. C. An Algorithm for the Traveling Salesman Problem. *Operations Research*. 1963. Vol.11. №6. p. 972–989. doi.org/10.1287/opre.11.6.972
16. *Муравьиный алгоритм : сайт*. URL : https://ru.wikipedia.org/wiki/Муравьиный_алгоритм. (дата обращения : 4.10.2019).
17. Pol R., Langdon W. B., McPhee N. F. A Field Guide to Genetic Programming. *Genetic Programming and Evolvable Machines*. 2009. Vol. 10 №2. p. 229 – 230. doi.org/10.1007/s10710-008-9073-y.
18. Прокопенков В. Ф. Модификация генетического алгоритма поиска гамильтонова цикла на графе. *Международная научная конференция MicroCAD 2016 : Секция №1 : Информационные и управляющие системы.* 2016. С. 32.
19. Прокопенков В. Ф. О возможности нахождения оптимального решения генетическим алгоритмом. *Международная научная конференция MicroCAD 2017 : Секция №1 : Информационные и управляющие системы.* 2017. С. 37.
20. Мартынов А. В., Курейчик В. М. Гибридный алгоритм решения задачи коммивояжера. *Известия ЮФУ. Технические науки*. – 2015. С.36–44.
21. Eppstein D. The travelling salesman problem for cubic graphs. *Lecture Notes in Computer Science*. 2003. P.307–318. doi.org/10.1007/978-3-540-45078-8_27
22. Iwama K., Nakashima T. An Improved exact algorithm for cubic graph TSP. *Lecture Notes in Computer Science, 2007.* p. 108 – 117. doi.org/10.1007/978-3-540-73545-8_13.
23. Dijkstra E. W. A note on two problems in connexion with graphs. *Numer. Math – Springer Science+Business Media*, 1959. Vol. 1, № 1. P. 269–271. doi.org/10.1007/bf01386390
24. Прокопенков В. Ф., Кожин Ю. Н. Новый подход поиска оптимального решения в переборных NP-задачах *Международная научная конференция MicroCAD 2019 : Секция №1 : Информационные и управляющие системы.* 2019. С. 38.
25. Прокопенков В. Ф. Параллельный алгоритм поиска гамильтонова цикла на графе. *Международная научная конференция MicroCAD 2015 : Секция №1 : Информационные и управляющие системы.* 2015. С. 25.

References

1. Prokopenkov, V. F. (2020), Novyy metod poiska gamil'tonova tsikla na grafe [A new method for searching a Hamilton cycle on a graf], "*Visnyk Natsionalnoho tekhnichnoho universytetu «KhPI». Seriia: Stratehichne upravlinnia, upravlinnia portfeliamy, prohramamy ta proektamy*." [Bulletin of the National Technical University "KHPI". Series: Strategic Management, portfolio management, programs and projects.], № 2, pp.43-49. doi.org/10.20998/2413-3000.2020.2.6
2. Akimov, O. E. (2005) *Diskretnaja matematika. Logika, gruppy, grafy, fraktaly* [*Discrete mathematics. Logic, groups, graphs, fractals*], Moscow, 656 p.
3. Emelichev, V. A., Kovalev, M. M., Kravcov, M. K. (1981), *Mnogogranniki, grafy, opitimizacija* [*Polyhedra, graphs, opitimization*]. Moscow, 341 p.
4. Hery, M., Dzhonson, D. (1982), *Vychyslytel'nye mashyny y trudnoreshaemye zadachy* [*Computational machines and difficult tasks*], Moscow, *419 p.*
5. Doiber, V. A., Kostochka, A. V., Sachs, H. "A shorter proof of Bolee korotkoe dokazatel'stvo teoremy Diraka o chisle reber hromaticheski kriticheskih grafah [Dirac's theorem on the number of edges of chromatically critical graphs], *Diskretnyj analiz i issledovanie operacij* [Discrete analysis and operations research] Novosibirsk state University, 1996. pp. 28–34.
6. Ore, O. (2009), Teoryya hrafov [*The theory of graphs*], Moscow, 354 p.
7. *Gamil'tonov_graf* ["Hamilton's graf"], available at : https://ru.wikipedia.org/wiki/Гамильтонов_граф. (last accessed: 04.10.2019)
8. Pavlenko, V. B. (2011) Teoreticheskie aspekty postroenija gamil'tonova cikla [Theoretical aspects of construction of the Hamiltonian cycle], *Teorija optimal'nih rishen'* [Theory of optimal solutions], №10, pp. 150–155. available at : http://dspace.nbuv.gov.ua/bitstream/handle/123456789/46787/22-Pavlenko.pdf?sequence=1 (last accessed 04.10.2019).
9. Prokopenkov, V. F., Kozhin, Ju., N., Malyh, *O. N.* (2019) Opredelenie optimal'nogo kol'cevogo marshruta, prohodjashhego cherez zadannoe mnozhestvo punktov na karte" [*"*Determination of the optimal circular route passing through a given set of points on the map"], *Innovative technologies and scientific solutions for industries*, No.1 № 7. pp. 102-112. doi.org/10.30837/2522-9818.2019.7.102
10. Harari, F. (1973), *Teorija grafov* [*Graph theory*], Moscow, 300 p.

11. Stivens, R. (2016), *Algoritmy. Teorija i prakticheskoe primenenie* [Algorithms. Theory and practical application], Moscow , 544 p.
12. Bellman, R. (1962), "Dynamic Programming Treatment of the Travelling Salesman Problem", *Journal of the ACM,* Vol.9 № 1, p. 61–63. doi.org/10.1145/321105.321111
13. Held, M. (1970), *"*The travelling-salesman problem an minimum spanning trees", *Operations Research*, Vol. 18 № 6, p. 1138 – 1162. doi.org/10.1287/opre.18.6.1138
14. Roberts, S. M., Flores, B. (1967), "An engineering approach to the travelling salesman problem", *Managment Science*, Vol. 13 № 3, p. 269–288. doi.org/10.1287/mnsc.13.3.269
15. Little, J. D. C. (1963) "An Algorithm for the Traveling Salesman Problem", *Operations Research*, Vol. 11, № 6, p. 972–989. doi.org/10.1287/opre.11.6.972
16. *Murav'inyj algoritm* ["Ant algorithm"], available at : https://ru.wikipedia.org/wiki/Муравьиный_алгоритм. (last accessed: 04.10.2019)
17. Pol, R., Langdon, W. B., McPhee, N. F. (2009), "A Field Guide to Genetic Programming", *Genetic Programming and Evolvable Machines*, Vol. 10, № 2, p. 229 – 230. doi.org/10.1007/s10710-008-9073-y.
18. Prokopenkov, V. F. (2016), Modifikacija geneticheskogo algoritma poiska gamil'tonova cikla na grafe" "[Modification of a genetic algorithm for finding a Hamiltonian cycle on a graph], *International Scientific Conference MicroCAD 2016: Section No. 1 – Information and Management Systems, p. 32.*
19. Prokopenkov, V. F. (2017), O vozmozhnosti nahozhdenija optimal'nogo reshenija geneticheskim algoritmom ["On the possibility of finding the optimal solution by genetic algorithm"], *International Scientific Conference MicroCAD 2017: Section No. 1 – Information and Management Systems, p. 37.*
20. Martynov, A. V., Kurejchik, V. M. (2015) "Gibridnyj algoritm reshenija zadachi kommivojazhera" ["Hybrid algorithm for solving the traveling salesman problem"], *SFU news. Technical science* [*Izvestija JuFU. Tehnicheskie nauki*]. p.36–44.
21. Eppstein, D. (2003), "The travelling salesman problem for cubic graphs", *Lecture Notes in Computer Science*, p. 307–318. doi.org/10.1007/978-3-540-45078-8_27
22. Iwama, K., Nakashima, T. (2007), "An Improved exact algorithm for cubic graph TSP", *Lecture Notes in Computer Science*, p. 108 – 117. doi.org/10.1007/978-3-540-73545-8_13.
23. Dijkstra, E. W. (1959), *"*A note on two problems in connexion with graphs", *Numer. Math, Springer Science+Business Media*, Vol. 1, № 1. p. 269–271. doi.org/10.1007/bf01386390
24. Prokopenkov, V. F., Kozhin, Ju. N. (2019), Novyj podhod poiska optimal'nogo reshenija v perebornyh NP-zadachah [A new approach to finding the optimal solution in iterative NP-problems], *International Scientific Conference MicroCAD 2019: Section No. 1 – Information and Management Systems, p. 38.*
25. Prokopenkov, V. F. (2015), Parallelniy algoritm poiska gamiltonova cikla na grafe [A parallel algorithm for finding a Hamiltonian cycle on a graph], *International Scientific Conference MicroCAD 2015: Section No. 1 – Information and Management Systems, p. 25.*

Поступила (Received) 08.01.2021

Відомості про авторів / Сведения об авторах / About the Authors

Прокопенков Володимир Пилипович (Прокопенков Владимир Филиппович, Prokopenkov Vladymyr) – Національний технічний університет "Харківський політехнічний інститут", старший викладач кафедри системний аналіз та інформаційно-аналітичні технології, Харків, Україна; e-mail: prokopenkov.vf@gmail.com; ORCID ID: https://orcid.org/0000-0003-0084-9832.