Russian Technological Journal.
2023;11(1):60–69

ISSN 2782-3210 (Print)
ISSN 2500-316X (Online)

Математическое моделирование

Mathematical modeling

УДК 004.021: 65.012.26
https://doi.org/10.32362/2500-316X-2023-11-1-60-69

НАУЧНАЯ СТАТЬЯ

Алгоритм поиска подкритических путей на сетевых графиках

М.А. Анфёров @

МИРЭА – Российский технологический университет, Москва, 119454 Россия
@ *Автор для переписки, e-mail: anfyorov@inbox.ru*

Резюме

Цели. Информационная поддержка процессов планирования и управления проектами использует в качестве модели сетевые графики, помогающие в формировании структуры планируемых работ и расчете характеристик эффективности проекта. С целью оптимизации и выравнивания ресурсов, используемых в проектах, возникает необходимость нахождения на этих моделях не только критического пути максимальной взвешенной длины, но и ближайших к нему подкритических путей с меньшей по отношению к нему длиной. Цель работы – синтез и анализ алгоритма поиска *k*-кратчайших путей между вершинами входа и выхода сети, позволяющего идентифицировать вышеназванные подкритические пути.

Методы. Использованы положения теории графов и теории групп, а также метод динамического программирования.

Результаты. Разработан алгоритм поиска *k*-кратчайших путей на ориентированных графах без контуров с отношением строгого порядка. С использованием теории групп на графах были определены абстрактные элементы – *p*-контуры, между которыми была установлена многоуровневая структура отношений, позволившая реализовывать необходимый поиск путей. В рамках обоснования работоспособности построенного алгоритма доказана справедливость основных положений: во-первых, многоуровневая система отношений является исчерпывающей; во-вторых, не происходит потерь в окончательном решении в процессе работы алгоритма; в-третьих, пути, найденные в результате работы алгоритма, удовлетворяют основному требуемому соотношению между ними. Численно алгоритм реализован методом динамического программирования, который был расширен за счет использования дополнительного функционального соотношения, предполагающего наличие подоптимальных политик.

Выводы. Проведенная серия вычислительных экспериментов подтвердила работоспособность и эффективность программно реализованного алгоритма. Выполненный анализ показал хорошие характеристики сходимости предложенного алгоритма в сравнении с алгоритмами данного класса, применяемыми к сетевым графикам. Это позволяет рекомендовать его к практическому использованию в информационных системах управления проектами.

Ключевые слова: управление проектами, сетевой график, критический путь, алгоритм, вычислительный эксперимент

• **Поступила:** 27.03.2022 • **Доработана:** 06.06.2022 • **Принята к опубликованию:** 24.10.2022

Для цитирования: Анфёров М.А. Алгоритм поиска подкритических путей на сетевых графиках. *Russ. Technol. J.* 2023;11(1):60–69. https://doi.org/10.32362/2500-316X-2023-11-1-60-69

Прозрачность финансовой деятельности: Автор не имеет финансовой заинтересованности в представленных материалах или методах.

Автор заявляет об отсутствии конфликта интересов.

RESEARCH ARTICLE

Algorithm for finding subcritical paths on network diagrams

Mikhail A. Anfyorov @

MIREA – Russian Technological University, Moscow, 119454 Russia
@ *Corresponding author, e-mail: anfyorov@inbox.ru*

Abstract

Objectives. Network diagrams are used as an information support element in planning and project management processes for structuring planned work and calculating project efficiency characteristics. In order to optimize and balance resources used in projects, it becomes necessary to locate in these models not only the critical path of the maximum weighted length, but also the subcritical paths closest to it having a shorter length in relation to it. The aim of the work is to synthesize and analyze an algorithm for finding *k*-shortest paths between the input and output network vertices, on which basis the above-mentioned subcritical paths can be identified.

Methods. The provisions of graph theory and group theory, as well as the method of dynamic programming, were used.

Results. An algorithm for finding *k*-shortest paths in contourless directed graphs having a strict order relation was developed. Abstract elements were defined according to group theory in graphs as *p*-contours, between which a multilevel structure of relations for implementing the necessary search of paths was then established. For substantiating the efficiency of the constructed algorithm, the validity of the main provisions was demonstrated as follows: firstly, the multilevel system of relations is exhaustive; secondly, there is no loss in the final solution during the operation of the algorithm; thirdly, the paths obtained as a result of the work of the algorithm satisfy the main required relation between them. Numerically, the algorithm was implemented by the dynamic programming method extended by means of an additional functional relationship, implying the presence of suboptimal policies.

Conclusions. The conducted runs of computational experiments confirmed the operability and efficiency of the software-implemented algorithm. The performed analysis demonstrated the good convergence characteristics of the proposed algorithm as compared with other algorithms of this class applied to network diagrams. On this basis, it can be recommended for practical use in project management information systems.

Keywords: project management, network diagram, critical path, algorithm, computational experiment

• **Submitted:** 27.03.2022 • **Revised:** 06.06.2022 • **Accepted:** 24.10.2022

For citation: Anfyorov M.A. Algorithm for finding subcritical paths on network diagrams. *Russ. Technol. J.* 2023;11(1):60–69. https://doi.org/10.32362/2500-316X-2023-11-1-60-69

Financial disclosure: The author has no a financial or property interest in any material or method mentioned.

The author declares no conflicts of interest.

ВВЕДЕНИЕ

Как известно, проектный менеджмент в качестве основной модели, отображающей структуру взаимной зависимости работ проекта и используемой для расчета временных характеристик этих работ и всего проекта в целом, использует сетевой ориентированный граф без контуров [1–3]. Данная модель в технологии сетевого планирования и управления получила название сетевого графика, построенного по одному из двух принципов. В первом случае работы в модели отображаются дугами графа (принцип «события-работы»), что характерно для методологии PERT (Project Evaluation and Review Technique), во втором – вершинами графа (принцип «работы-связи»). Однако любой из этих подходов предполагает определение на сетевом графике критического пути, имеющего максимальную взвешенную длину и соединяющего висячую вершину с тупиковой, которые соответствуют началу и окончанию проекта. При этом длина рассчитывается, исходя из значения времени выполнения критических работ, отображаемых активными элементами пути (дугами или вершинами). Использование критического пути является неотъемлемой частью технологии сетевого планирования и управления проектами, что отражено в исследованиях на этапе

ее теоретического становления (обширная библиография данного периода представлена в [4]) и последующего развития [5–8].

Что касается реальных проектов в различных предметных областях, то они реализуются в условиях ограничения материальных, трудовых и финансовых ресурсов. Это предполагает развитие теории проектного менеджмента в направлении оптимизации структуры проекта по критерию минимальной его стоимости при определенных ресурсных ограничениях [7–12]. В этом случае встает вопрос о необходимости поиска на сетевом графике не только критического, но и подкритических путей для осуществления вышеназванной оптимизации и выравнивания ресурсов. Если критический путь Ω_0 можно определить как

$$\Omega_0 : L(\Omega_0) = \max_{\Omega_j \in \mathbf{\Omega}} \{L(\Omega_j)\}, \quad (1)$$

где $L(\Omega_0)$ – длина критического пути, а $\mathbf{\Omega}$ – множество всех полных путей на сетевом графике, то подкритические пути составят упорядоченное множество полных путей графа $\{\Omega_1, \Omega_2, ...\}$, характеризуемое следующим образом:

$$L(\Omega_0) \geq L(\Omega_1); L(\Omega_j) \geq L(\Omega_{j+1}). \quad (2)$$

Для нахождения критического пути достаточно использовать любой из алгоритмов для поиска кратчайшего взвешенного пути между вершинами, отображающими начало и окончание проекта на сетевом графике, предварительно поменяв значения весов его активных элементов (времени выполнения работ) на отрицательные. Это могут быть алгоритмы, традиционно использующие динамическое программирование [13, 14], либо эвристические методы [15]. Что касается нахождения подкритических путей, то для этого используются так называемые алгоритмы поиска *k*-кратчайших путей между вершинами графа. Это могут быть алгоритмы, опять-таки использующие динамическое программирование непосредственно [16–22] либо с многократным применением алгоритма поиска кратчайшего пути [23–25], или использующие другие современные подходы [26–28], а также подходы, ориентированные на более сложные конструкции сетей [29–32].

В данной работе описывается алгоритм поиска *k*-кратчайших путей на сетевых графиках, основанный на выделении в них элементов более высокого порядка и структуризации отношений между ними.

СТРУКТУРИЗАЦИЯ ОТНОШЕНИЙ НА ГРАФАХ

Выполненные исследования ориентированы на применение сетевых графиков типа «работы-связи», использующих в качестве модели сетевые ориентированные графы $\mathbf{G}(\mathbf{X}, \mathbf{U})$ ($\mathbf{X}$ – множество вершин, $\mathbf{U}$ – множество дуг) без контуров с отношением строгого порядка с разбивкой на слои [33] (рис. 1).

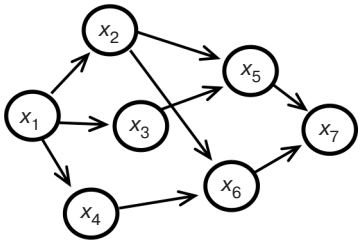

Рис. 1. Сетевой график

Построим свободную абелеву группу $\mathbf{P}(\mathbf{U})$ над порождающим множеством $\mathbf{U}$ всех дуг графа $\mathbf{G} = (\mathbf{X}, \mathbf{U})$ следующим образом. В качестве элементов $\mathbf{P(U)}$ будем рассматривать множество формальных линейных комбинаций элементов из $\mathbf{U}$ с целыми коэффициентами вида $p_j = \sum_{i=1}^{n} (\gamma_i \cdot u_i),\ p_j \in \mathbf{P}(\mathbf{U});$ $u_i \in \mathbf{U}; \gamma_i = 0, \pm 1, \ldots,$ где n – количество дуг графа.

В качестве бинарной аддитивной операции определим сумму элементов из множества $\mathbf{P}(\mathbf{U})$ формулой $p_j = \sum_{i=1}^{n} (\gamma_i \cdot u_i) + \sum_{i=1}^{n} (\gamma'_i \cdot u_i) = \sum_{i=1}^{n} (\gamma_i + \gamma'_i) \cdot u_i.$

Аналогично построим группу $\mathbf{H}(\mathbf{X})$ над множеством $\mathbf{X}$ всех вершин графа, определив ее элементы как $h_j = \sum_{i=1}^{m} (\gamma_i \cdot x_i),\ h_j \in \mathbf{H}(\mathbf{X});\quad x_i \in \mathbf{X}; \gamma_i = 0, \pm 1, \ldots,$ где m – количество вершин. В дальнейшем при записи элементов групп $\mathbf{P(U)}$ и $\mathbf{H(X)}$ будем опускать составляющие их элементы, имеющие нулевые коэффициенты.

Определение 1. Дифференциалом d группы $\mathbf{P}(\mathbf{U})$ называется гомоморфизм d: $\mathbf{P}(\mathbf{U}) \rightarrow \mathbf{H}(\mathbf{X})$, определенный следующим образом:

1) если $u_t = (x_i, x_j)$, то $du_t = x_j - x_i$;
2) если $p_q = \Sigma\gamma_t u_t$, то $dp_q = \Sigma\gamma_t du_t$

при $p_q \in \mathbf{P}(\mathbf{U})$; $u_t \in \mathbf{U}$; $x_i, x_j \in \mathbf{X}$.

Определение 2. Элемент $r \in \mathbf{P}(\mathbf{U})$ называется p-контуром, если $dr = 0$. Подгруппа $\mathbf{R} = Ker\ d \subset \mathbf{P}(\mathbf{U})$ называется подгруппой p-контуров.

Лемма 1. Сумма нескольких p-контуров есть p-контур.

Доказательство. Во-первых, сумма Σr_i есть элемент группы $\mathbf{P}(\mathbf{U})$ как результат аддитивного сложения ее элементов. Во-вторых, дистрибутивность отображения d согласно *определению 1* позволяет записать $d\Sigma r_i = \Sigma dr_i = 0$. Утверждение леммы доказано.

Используя общепринятое понятие пути на графе как связанной конечной последовательности дуг, будем обозначать его

$$\Omega = \{u_1, u_2, \ldots, u_w\}. \quad (3)$$

При этом, если $u = (x_1, x_2)$, то $x_1 = u^-$, $x_2 = u^+$. Длину этого пути будем определять через вышеупомянутые численные весовые оценки вершин $\varepsilon(x)$ выражением

$$L(\Omega) = \varepsilon(u_1^-) + \sum_{i=1}^{w} \varepsilon(u_1^+). \quad (4)$$

Если путь соединяет вершину входа x' с вершиной выхода x'' сети, то будем называть такой путь полным.

Зададим отображение φ следующим образом:

$$\varphi(\{\gamma_1 u_1, \gamma_2 u_2, ..., \gamma_n u_n\}) = \sum_{i=1}^{n} \gamma_i u_i, \gamma_i > 0, \quad (5)$$

а также обратное ему

$$\varphi'\left(\sum_{i=1}^{n} \gamma_i u_i\right) = \{|\gamma_1| u_1, |\gamma_2| u_2, ..., |\gamma_n| u_n\}. \quad (6)$$

Определение 3. Образом пути $\Omega \subset \mathbf{U}$ на графе $\mathbf{G} = (\mathbf{X}, \mathbf{U})$ называется отображение $\varphi(\Omega)$.

Лемма 2. Дифференциал образа любого полного пути на сетевом графе определится как $d\varphi(\Omega) = x'' - x'$.

Доказательство. Поскольку полный путь Ω на рассматриваемых графах является простым путем, не имеющим кратных дуг, то его образ, с учетом (3), можно записать как $\varphi(\Omega) = u_1 + u_2 + \ldots + u_w$. В соответствии с *определением 1* получим $d\varphi(\Omega) = du_1 + du_2 + \ldots + du_w$ или $d\varphi(\Omega) = (u_1^+ - u_1^-) + (u_2^+ - u_2^-) + ... + (u_w^+ - u_w^-)$ или

$$d\varphi(\Omega) = -u_1^- + (u_1^+ - u_2^-) + ... + (u_{w-1}^+ - u_w^-) + u_w^+. \quad (7)$$

Свойство инцидентности дуг пути предполагает $u_i^+ = u_{i+1}^-, i \in [1, w-1]$. Следовательно, выражение (7) запишется в виде $d\varphi(\Omega) = u_w^+ - u_1^-$ или $d\varphi(\Omega) = x'' - x'$.

Лемма 3. Разность образов двух полных путей на сетевом графе есть p-контур.

Доказательство. Разность образов полных путей $\varphi(\Omega_i)$ и $\varphi(\Omega_j)$, являясь результатом аддитивной функции сложения двух элементов группы $\mathbf{P}(\mathbf{U})$, также принадлежит этой группе. С другой стороны, дифференциал этой разности определится как $d[\varphi(\Omega_i) - \varphi(\Omega_j)] = d\varphi(\Omega_i) - d\varphi(\Omega_j)$. Однако в силу утверждения *леммы 2* $d\varphi(\Omega_i) = d\varphi(\Omega_j)$. Следовательно, $d[\varphi(\Omega_i) - \varphi(\Omega_j)] = 0$, что доказывает утверждение леммы.

Зададим отображение α следующим образом:

$$\alpha(p) = \sum_{u_i \in p}^{n} \gamma_i \cdot \varepsilon(u_i^+). \quad (8)$$

Определение 4. Значением p-контура r называется величина $\alpha(r)$.

Лемма 4. Разность длин двух полных путей на сетевом графе равна значению p-контура, образованного разностью их образов.

Доказательство. Пусть $\Omega_i = \{u_{i1}, u_{i2}, \ldots, u_{iw}\}$ и $\Omega_j = \{u_{j1}, u_{j2}, \ldots, u_{js}\}$ – два любых полных пути графа. Тогда их длины, в соответствии с (4), можно записать как $L(\Omega_i) = \varepsilon(x') + \sum_{t=1}^{w} \varepsilon(u_{it}^+)$, $L(\Omega_j) = \varepsilon(x') + \sum_{t=1}^{s} \varepsilon(u_{jt}^+)$, а разность длин как

$$L(\Omega_i) - L(\Omega_j) = \sum_{t=1}^{w} \varepsilon(u_{it}^+) - \sum_{t=1}^{s} \varepsilon(u_{jt}^+). \quad (9)$$

С другой стороны, в соответствии с *леммой 3*, p-контур, определяемый образами этих полных путей, запишется с учетом выражения (5) в виде $r = \varphi(\Omega_i) - \varphi(\Omega_j) = u_{i1} + u_{i2} + ... + u_{iw} - u_{j1} - u_{j2} - ... - u_{js}$, а значение этого p-контура, с учетом (8), как

$$\alpha(r) = \sum_{t=1}^{w} \varepsilon(u_{it}^+) - \sum_{t=1}^{s} \varepsilon(u_{jt}^+). \quad (10)$$

Сравнивая выражения (9) и (10), приходим к утверждению леммы.

Определение 5. Элементарным p-контуром $g^{(ab)}$ относительно дуги графа $u = (a, b) \in \mathbf{U}$ назовем элемент группы $\mathbf{P}(\mathbf{U})$, определяемый в виде суммы этой дуги с разностью между образами кратчайших взвешенных путей, соединяющих вершины x' и a, а также вершины x' и b (см. рис. 2). При этом элементарные p-контуры ориентированы на кратчайшие пути ввиду того, что основной целью является поиск k-кратчайших путей графа.

Определение 6. Дуга u_i называется инцидентной пути Ω_0 в вершине b, если выполняются следующие условия:

1) $u_i \notin \Omega_0$;
2) $\exists u_i \in \Omega_0, u_i^+ = u_j^+ = b$.

Вершина b называется при этом вершиной сечения пути Ω_0 (рис. 2).

Определение 7. Элементарный p-контур относительно дуги, инцидентной пути Ω в вершине сечения $b \in \mathbf{X}$, называется базирующимся по этому пути и обозначается как[1] r^b (рис. 2).

В привязке к рассмотрению полных путей графа в рамках задачи сетевого планирования базирующийся по пути Ω_0 p-контур, позволяющий

[1] Следует иметь ввиду, что по вершине b может базироваться несколько элементарных контуров при наличии нескольких инцидентных дуг. [Several elementary contours can be based on the vertex b in the presence of several incident arcs.]

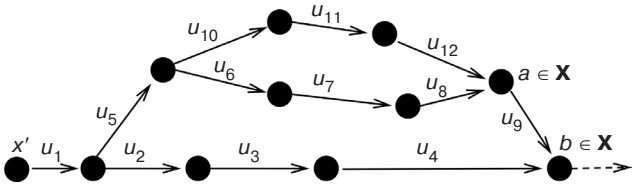

Рис. 2. Графическое пояснение к определениям. $\{u_1, u_5, u_6, u_7, u_8\}$ – кратчайший путь между вершинами x' и a; $\{u_1, u_2, u_3, u_4\}$ – кратчайший путь между вершинами x' и b; $\{u_5 + u_6 + u_7 + u_8 + u_9 - u_2 - u_3 - u_4\}$ – элементарный p-контур ($g^{(ab)}$), базирующийся по пути $\{u_1, u_2, u_3, u_4, ...\}$ по инцидентной дуге u_9 в вершине сечения b: $g^{(ab)} \equiv r^{bi}$

определить другой полный путь Ω_s назовем производящим. На основании лемм 3 и 4 запишем соотношения

$$\varphi(\Omega') = \varphi(\Omega_0) + r^b,\ L(\Omega') = L(\Omega_0) + \alpha(r^b), \quad (11)$$

которые в дальнейшем используются в алгоритме поиска k-кратчайших путей графа.

Введенные математические объекты позволяют строить упорядоченную структуру отношений между полными путями сетевого графа. Это позволяет эффективно решать численные задачи на данных моделях.

Применительно к разработке описываемого алгоритма рассматривается задача поиска упорядоченного множества полных путей

$$\{\Omega_0, \Omega_1, \ldots, \Omega_k\},\ L(\Omega_i) \le L(\Omega_{i+1}),\ i \in [0, k-1]. \quad (12)$$

Система отношений между путями строится относительно полного пути Ω_0, имеющего минимальную длину $\Omega_0 : L(\Omega_0) = \min\limits_{\Omega_j \in \mathbf{\Omega}} \{L(\Omega_j)\}$, который может быть легко найден одним из известных алгоритмов, например [13, 14]. Данная система описывается иерархической многоуровневой структурой производящих p-контуров кратчайшего пути Ω_0 в виде графа $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$, $\mathbf{V} = \mathbf{R}_0 \times \mathbf{R}_0$. Здесь $\mathbf{R}_0$ – полное множество производящих p-контуров, а $\mathbf{V}$ – множество отношений между ними, связывающих производящие p-контуры смежных уровней. Так для верхнего 0-го уровня связь p-контуров[2] $\mathbf{R}_{00} = \{r_0^b\}$ с p-контурами следующего 1-го уровня $\mathbf{R}_{01}$ представлена соотношением

$$r_1^{be} = r_0^b + r^e;\ \alpha(r_1^{be}) = \alpha(r_0^b) + \alpha(r^e), \quad (13)$$

где b – вершина сечения пути Ω_0; e – вершина сечения пути, образованного p-контуром r_0^b (11), расположенная на этом пути левее вершины b (т.е. $e < b$)[3]. Объединение по уровням определяет все множество производящих p-контуров $\mathbf{R}_0 = \bigcup\limits_Z \mathbf{R}_{0Z}$.

ОПИСАНИЕ АЛГОРИТМА

Ниже приведен алгоритм в менее компактном виде, исключающем циклы, с целью наглядного показа конечности числа выполняемых шагов.

Шаг 1. Найти полный путь Ω_0, имеющий минимальную длину.

Шаг 2. Упорядочить множество $\mathbf{R}_{00} = \{r_1, r_2, r_3, ...\}$ производящих p-контуров 0-го уровня в порядке возрастания их значений[4] и исключить из дальнейшего рассмотрения p-контуры, стоящие в полученной последовательности ниже k-го места; $\bar{\mathbf{R}} = \mathbf{R}_{00}$. Найти полный путь Ω_1 через соотношения (11), используя p-контур r_1, стоящий в последовательности первым.

Шаг 3. Включить в множество $\bar{\mathbf{R}}$ p-контуры следующего уровня, определяемые p-контуром r_1 через соотношения (13) по всем вершинам сечения, исключив сам p-контур r_1. Упорядочить множество $\bar{\mathbf{R}}$ согласно соотношению $\alpha(r_i) \le \alpha(r_{i+1})$ и исключить из дальнейшего рассмотрения p-контуры, стоящие в полученной последовательности ниже $(k-1)$ места.

Найти полный путь Ω_2 через соотношения (11), используя p-контур r_1.

. . . . . .

Шаг (s). Включить в множество $\bar{\mathbf{R}}$ p-контуры следующего уровня, определяемые p-контуром r_1 через соотношения (13) по всем вершинам сечения, исключив сам p-контур r_1. Упорядочить множество $\bar{\mathbf{R}}$ согласно соотношению $\alpha(r_i) \le \alpha(r_{i+1})$ и исключить из дальнейшего рассмотрения p-контуры, стоящие в полученной последовательности ниже $[k - (s - 2)]$ места[5].

---

[2] Дополнительный нижний индекс введен для обозначения номера уровня в системе производящих p-контуров. [An additional subscript is introduced to denote the level number in the system of generating p-contours.]

[3] Это условие связано с ориентацией на вершину входа x' при построении элементарных p-контуров (см. *определение 5*). [This condition is related to the orientation to the input vertex x' when constructing elementary p-contours (see *Definition 5*).]

[4] Упрощенное обозначение производящих контуров введено с целью лучшего понимания работы алгоритма. Нижний индекс показывает место контура в упорядоченном множестве. [A simplified designation of generating contours was introduced in order to better understand the operation of the algorithm. The subscript shows the place of the contour in the ordered set.]

[5] Случай, когда количество элементов множества $\mathbf{R}_0$ меньше $[k - (i - 2)]$ является частным. В этом случае не происходит исключения контуров из множества, что не влияет на ход всех рассуждений и на конечный результат. [The particular case is when the number of elements of the set $\mathbf{R}_0$ is less than $[k - (i - 2)]$. In this case, contours are not excluded from the set, which does not affect the course of all reasoning and the final result.]

Найти полный путь Ω_{s-1} через соотношения (11), используя p-контур r_1.

. . . . . .

Шаг $(k + 1)$. Включить в множество $\bar{\mathbf{R}}$ p-контуры следующего уровня, определяемые p-контуром r_1 через соотношения (13) по всем вершинам сечения, исключив сам p-контур r_1. Упорядочить множество $\bar{\mathbf{R}}$ согласно соотношению $\alpha(r_i) \le \alpha(r_{i+1})$ и исключить из дальнейшего рассмотрения p-контуры, стоящие в полученной последовательности ниже $[k - (k - 1)] = 1$ места.

Найти полный путь Ω_k через соотношения (11), используя p-контур r_1.

Реализация алгоритма опирается на поиск производящих p-контуров и полного пути Ω_0, имеющего минимальную длину. Данная задача эффективно решается методом динамического программирования. Основное рекуррентное функциональное соотношение Беллмана для задачи поиска кратчайшего пути[6] $\Omega^{1t}_{\min}$, соединяющего вершину входа (x_1) с любой вершиной x_t, запишется в виде

$$L(\Omega^{1t}_{\min}) = \min_i \{\varepsilon(x_t) + L(\Omega^{1i}_{\min})\}, i \in \mathbf{I}_t, \quad (14)$$

где $\mathbf{I}_t$ – подмножество номеров вершин графа, определяемое условием $\forall x_i$ $(i \in \mathbf{I}_t)$ $\exists u_y$ такая, что $x_i = u_y^-, x_t = u_y^+$. Элемент из $\mathbf{I}_t$, являющийся оптимальной политикой, определяемой (14), обозначим как i_o.

Для поиска производящих p-контуров, базирующихся по вершине сечения t необходимо задать еще одно функциональное соотношение в виде

$$L(\Omega^{1t}_j) = \varepsilon(x_t) + L(\Omega^{1j}_{\min}), j \in \mathbf{J}_t, \mathbf{J}_t = \mathbf{I}_t / i_o, \quad (15)$$

где Ω^{1t}_j – путь, соединяющий вершины x_1 и x_t и проходящий через вершину x_j (рис. 3).

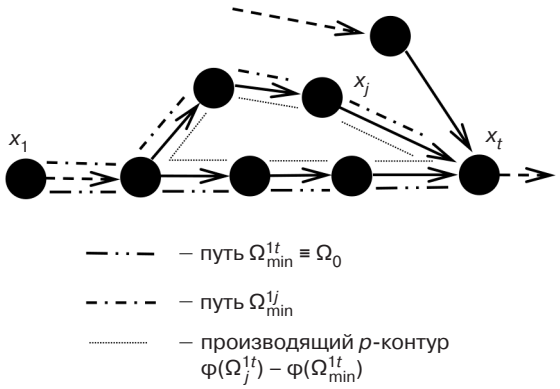

Рис. 3. Графическое пояснение к приведенным выкладкам

[6] Верхние индексы показывают номера соединяемых вершин. [The superscripts show the numbers of connected vertices.]

Соотношение (15) предполагает наличие в методе динамического программирования подоптимальных политик, наличие которых было показано в свое время Р. Беллманом [34]. Данное соотношение определяет множество инцидентных дуг $\{(x_j, x_t)\}$, $j \in \mathbf{J}_t$ в вершине сечения x_t и соответствующих им производящих p-контуров через знание путей $\Omega^{1j}_{\min}$ (см. *определение 5*).

В результате была выполнена программная реализация алгоритма на языке программирования Delphi. Выполненное тестирование программы подтвердило работоспособность и эффективность представленного алгоритма.

ОБОСНОВАНИЕ АЛГОРИТМА

Для доказательства работоспособности построенного алгоритма необходимо убедиться в справедливости следующих положений:

- во-первых, любой производящий p-контур полного пути Ω_0 описывается системой $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$;
- во-вторых, не происходит потерь в окончательном решении в процессе работы алгоритма;
- в-третьих, пути, найденные в результате работы алгоритма, удовлетворяют условию (12).

Справедливость первого положения подтверждается следующей теоремой.

Теорема 1. Любой производящий p-контур полного пути Ω_0 описывается системой $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$.

Доказательство. Пусть r_x – произвольно выбранный производящий p-контур пути Ω_0. Покажем, что этот p-контур описывается одним из уровней системы $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$.

Обязательно существует такая дуга u_0, инцидентная пути Ω_0 с вершиной сечения b_0, что $u_0 \in \varphi'(r_x)$. Этой дуге соответствует базирующийся по Ω_0 элементарный p-контур $g^{u_0} = r^{b_0}$.

Если $r^{b_0} = r_x$, то p-контур r_x описывается системой $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$ на 0-м уровне. Если равенство не выдерживается, тогда $\alpha(r^{b_0}) \le \alpha(r_x)$, что следует из *определения 5*, а следовательно справедливо неравенство $L(\Omega_1) \le L(\Omega_x)$, где Ω_1 и Ω_x – полные пути, определяемые p-контурами r^{b_0} и r_x через соотношения (11), т.е.

$$\varphi(\Omega_1) = \varphi(\Omega_0) + r^{b_0}, L(\Omega_1) = L(\Omega_0) + \alpha(r^{b_0}), \quad (16)$$

$$\varphi(\Omega_x) = \varphi(\Omega_0) + r_x, L(\Omega_x) = L(\Omega_0) + \alpha(r_x). \quad (17)$$

Вычитая почленно равенства (16) и (17) и преобразовав, получим $\varphi(\Omega_x) = \varphi(\Omega_1) + (r_x - r^{b_0})$, где $(r_x - r^{b_0})$ – производящий p-контур пути Ω_1. Причем его значение (получается тем же вычитанием равенств (16) и (17)) равно $\alpha(r_x) - \alpha(r^{b_0}) \ge 0$.

Так как $u_0 \in \Omega_1$, то вершина сечения b_1 p-контура $(r_x - r^{b_0})$ находится левее b_0.

Переходим к следующему аналогичному шагу рассуждений. Существует такая дуга u_1, инцидентная пути Ω_1 с вершиной сечения b_1, что $u_1 \in \varphi'(r_x - r^{b_0})$. Этой дуге соответствует базирующийся по Ω_0 элементарный p-контур $g^{u_1} = r^{b_1}$.

Если $r^{b_1} = r_x - r^{b_0}$, тогда p-контур $r_x = r^{b_0} + r^{b_1}$, т.е. описывается системой $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$ на 1-м уровне. Если равенство не выдерживается, тогда $\alpha(r^{b_1}) \le \alpha(r_x - r^{b_0})$ или $\alpha(r_x) \ge \alpha(r^{b_0}) + \alpha(r^{b_1})$, а, следовательно, $L(\Omega_2) \le L(\Omega_x)$, где Ω_2 – полный путь, определяемый p-контуром r^{b_1} через соотношения (11) как $\varphi(\Omega_2) = \varphi(\Omega_1) + r^{b_1}$ или с учетом (16)

$$\varphi(\Omega_2) = \varphi(\Omega_0) + r^{b_0} + r^{b_1},$$

$$L(\Omega_2) = L(\Omega_0) + \alpha(r^{b_0} + r^{b_1}). \qquad (18)$$

Вследствие этого по аналогии существует производящий p-контур пути Ω_2 в виде $(r_x - r^{b_0} - r^{b_1})$, $\alpha(r_x - r^{b_0}) - \alpha(r^{b_1}) \ge 0$.

Таким образом, на любом N-м шаге рассуждений невыполнение условия равенства

$$r^{b_N} = r_x - r^{b_0} - r^{b_1} - \ldots - r^{b_{N-1}} \qquad (19)$$

влечет переход к следующему шагу. Но, поскольку количество шагов ограничено, то на определенном шаге равенство (19) будет выдержано (т.е. r^{b_N} будет элементарным p-контуром). Это означает, что p-контур r_x пути Ω_0 опишется системой $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$ на $(N-1)$ уровне.

Ограниченность числа шагов подтверждается следующими соображениями. Во-первых, выбираемая на любом шаге инцидентная дуга принадлежит пути Ω_x, а, во-вторых, вершина сечения, как было отмечено выше, находится левее выбранной на предыдущем шаге. Поскольку количество дуг пути Ω_x ограничено, то и ограничено количество шагов основного рассуждения.

Поскольку p-контур r_x был выбран произвольным, то *условие теоремы можно считать доказанным*.

Для проверки второго положения введем в рассмотрение вектор $\mathbf{Q}_j = [\alpha(r_{1j}), \alpha(r_{2j}), \ldots, \alpha(r_{(k+1-j)j})]$, элементами которого являются значения p-контуров, составляющих множество $\bar{\mathbf{R}}$ при работе алгоритма на этапе определения пути Ω_j (номер пути определяет второй нижний индекс в обозначении этих p-контуров). Напомним, что элементы вектора $\mathbf{Q}_j$ образуют неубывающую последовательность.

Введем также вектор $\mathbf{W} = [\alpha(r_{k1}), \alpha(r_{(k-1)2}), \ldots, \alpha(r_{1k})]$ (смысловое значение индексов соответствует вектору $\mathbf{Q}_j$), элементы которого $\alpha(r_{(k+1-j)j})$ – значения p-контуров, стоящих на последнем месте во множестве $\bar{\mathbf{R}}$ на этапе определения пути Ω_j.

Теорема 2. Элементы вектора $\mathbf{W}$ образуют невозрастающую последовательность.

Доказательство. Рассмотрим произвольно выбранный j-й элемент вектора $\mathbf{W}$, т.е. $\alpha(r_{(k+1-j)j})$, который одновременно является последним элементом вектора $\mathbf{Q}_j$. Нас интересует процесс перехода при работе алгоритма от вектора $\mathbf{Q}_j$ к вектору $\mathbf{Q}_{j+1}$ с точки зрения появления нового элемента $\alpha(r_{(k-j)(j+1)})$ вектора $\mathbf{W}$. Он состоит из следующих этапов:

- исключается первый элемент вектора $\mathbf{Q}_j$;
- к оставшимся элементам добавляются новые, соответствующие включаемым в рассмотрение p-контурам нижнего уровня;
- упорядочение полученного множества;
- исключение из дальнейшего рассмотрения лишних p-контуров.

Исключение первого элемента вектора $\mathbf{Q}_j$ не может повлиять на выбор элемента $\alpha(r_{(k-j)(j+1)})$.

Результат работы последующих процедур не изменится, если их производить несколько в ином порядке. К оставшимся элементам вектора $\mathbf{Q}_j$ (их количество теперь соответствует необходимому количеству элементов вектора $\mathbf{Q}_{j+1}$) будем добавлять по одному элементу нового множества значений p-контуров. После каждого такого добавления будем осуществлять упорядочение множества $\bar{\mathbf{R}}$ и исключение лишнего элемента.

Элементы вектора $\mathbf{Q}_{j+1}$ перед добавлением новых элементов определятся следующим образом: $\alpha(r_{1(j+1)}) = \alpha(r_{2j})$, $\alpha(r_{2(j+1)}) = \alpha(r_{3j})$, ..., $\alpha(r_{(k-j)(j+1)}) = \alpha(r_{(k+1-j)j})$. Для добавляемого элемента $\alpha(r_y)$ существуют две альтернативы: либо $\alpha(r_y) \ge \alpha(r_{(k+1-j)j})$, либо $\alpha(r_{(t+1)j}) \ge \alpha(r_y) \ge \alpha(r_{tj})$, $t \in [2, k-j]$. В первом случае исключается из дальнейшего рассмотрения именно элемент $\alpha(r_y)$, что не изменит упорядоченного положения оставшихся элементов вектора $\mathbf{Q}_j$. Во втором случае после упорядочения элемент $\alpha(r_y)$ займет место $\alpha(r_{(t+1)j})$, элемент $\alpha(r_{(t+1)j})$ – место $\alpha(r_{(t+2)j})$ и т.д.

В конечном счете, элемент $\alpha(r_{(k-j)j})$ займет место элемента $\alpha(r_{(k+1-j)j})$, который будет удален. Но, поскольку $\alpha(r_{(k-j)j}) \le \alpha(r_{(k+1-j)j})$ (в силу определения вектора $\mathbf{Q}_j$), то и в этом случае не произойдет увеличения значения последнего элемента, т.е. элемент $\alpha(r_{(k-j)(j+1)})$ не может быть больше элемента $\alpha(r_{(k+1-j)j})$.

Так как в качестве j-го был выбран произвольный элемент вектора $\mathbf{W}$, то *утверждение теоремы можно считать доказанным*.

С помощью *теоремы 2* нетрудно доказать, что не происходит потерь в окончательном решении при

работе алгоритма. В самом деле, потери могут произойти только в процедуре исключения из дальнейшего рассмотрения *p*-контуров после их упорядочения на каждом шаге[7]. Однако на любом шаге работы алгоритма значения исключаемых *p*-контуров не меньше элемента $\alpha(r_{(k+1-j)j})$ вектора $\mathbf{Q}_j$. На основании *теоремы 2* можно заключить, что эти значения также заведомо не меньше последующих элементов вектора $\mathbf{W}$: $\alpha(r_{(k-j)(j+1)})$, $\alpha(r_{(k-j-1)(j+2)})$, ..., $\alpha(r_{1k})$, т.е. они не должны подлежать рассмотрению на оставшихся шагах работы алгоритма и не могут входить в окончательное решение.

Для проверки последнего положения введем в рассмотрение вектор, состоящий из первых элементов векторов $\mathbf{Q}_j, j \in [1,k]$: $\bar{\mathbf{W}} = [\alpha(r_{11}), \alpha(r_{12}), \ldots, \alpha(r_{1k})]$, участвующих в формировании окончательного решения через соотношения (11). Тогда справедливость выполнения условия (12) подтверждает следующая теорема.

Теорема 3. Элементы вектора $\bar{\mathbf{W}}$ образуют неубывающую последовательность.

Доказательство. Рассмотрим произвольно выбранный *j*-й элемент вектора $\bar{\mathbf{W}}$ – $\alpha(r_{1j})$. Этот элемент одновременно является первым элементом вектора $\mathbf{Q}_j$. В процессе перехода от вектора $\mathbf{Q}_j$ к вектору $\mathbf{Q}_{j+1}$ добавляемое новое множество *p*-контуров характеризуется тем, что их значения заведомо не меньше $\alpha(r_{1j})$, т.к. они находятся на следующем уровне после *p*-контура r_{1j} и определяются этим *p*-контуром через соотношения (13).

После упорядочения множества $\bar{\mathbf{R}}$ элементом $\alpha(r_{1(j+1)})$ станет либо элемент $\alpha(r_{2j})$, либо наименьший из добавляемого множества $\alpha_{\min}(r_{\text{add}})$, т.е. $\alpha(r_{1(j+1)}) = \min\{\alpha(r_{2j}), \alpha_{\min}(r_{\text{add}})\}$. Но, так как $\alpha(r_{2j}) \geq \alpha(r_{1j})$, $\alpha_{\min}(r_{\text{add}}) \geq \alpha(r_{1j})$, то $\alpha(r_{1(j+1)}) \geq \alpha(r_{1j})$.

Так как в качестве *j*-го был выбран любой элемент вектора $\bar{\mathbf{W}}$, то последнее неравенство *доказывает справедливость утверждения теоремы*.

Оценка производительности алгоритма проводилась в сравнении с алгоритмом двойного поиска («double sweep»), по которому автором данного алгоритма были проведены серьезные вычислительные эксперименты [35]. Причем аналитическая оценка производительности по алгоритмам данной категории далека от реальных результатов, т.к. время вычислений сильно зависит от конфигурации используемых сетей, а выполняемые обобщенные операции при выполнении алгоритма не совсем однозначно отображаются в элементарные операции сложения и сравнения. Так для упомянутого алгоритма, который относится к классу наиболее производительных алгоритмов данной категории, при аналитической оценке время вычислений имеет порядок $O(kN^3)$ [19, 20], в то время как вычислительные эксперименты показывают другие результаты [35] – вместо линейной наблюдается полиноминальная зависимость времени вычислений τ от количества кратчайших путей *k*:

$$\tau = 0.8457 + 0.1616\,k + 0.0260\,k^2.$$

При этом следует заметить, что исследования [35] проводились не на сетевых графиках, а на ориентированных графах, вершины которых образовывали решетчатую структуру с наличием контуров. Размерность графов учитывалась через размеры и конфигурацию решетки, образованной вершинами. Веса дуг генерировались случайными целыми числами в диапазоне до 100. Кроме того, значительно было ограничено значение *k*.

Поэтому применительно к поиску подкритических путей на сетевых графиках для сравнения полученного в работе алгоритма с алгоритмом двойного поиска была проведена серия вычислительных экспериментов на микропроцессоре Intel с тактовой частотой 1.7 ГГц. Результаты зависимости времени вычислений (*t*) от количества подкритических путей (*k*) и размерной характеристики сетевых графиков (*z*) показаны на рис. 4. Размерная характеристика

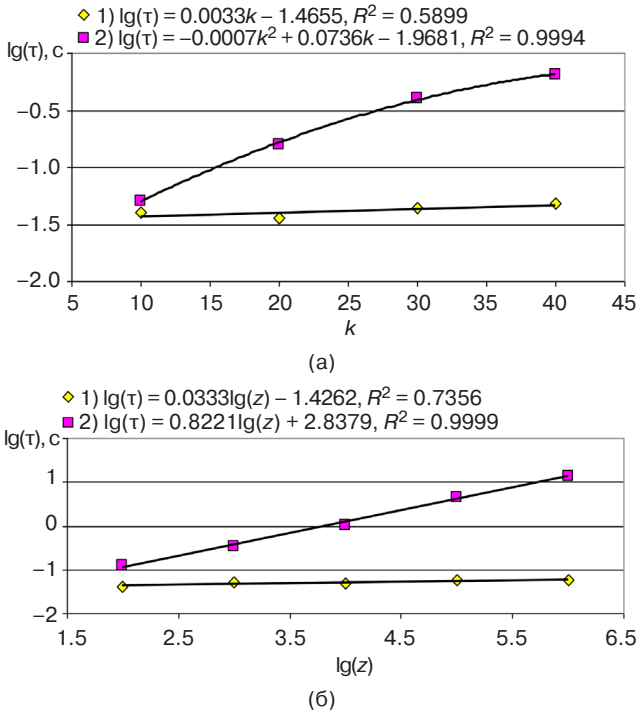

Рис. 4. Сравнительный анализ алгоритмов:
(а) $z = 1600$; (б) $k = 40$.
1 – описываемый алгоритм; 2 – алгоритм двойного поиска

[7] Исключение контура, стоящего первым, не приводит к потерям, поскольку с одной стороны он используется для получения решения, а с другой – в рассмотрение попадают все порождаемые им контуры нижнего уровня. [The exclusion of the contour that is the first does not lead to losses, since on the one hand it is used to obtain a solution, and on the other hand, all lower-level contours generated by it fall into consideration.]

представляет собой произведение общего количества дуг графа на количество дуг в пути максимальной длины, соединяющем вершины входа и выхода сети.

Более высокая производительность представленного в работе алгоритма объясняется его специальной ориентацией на рассматриваемый класс сетевых графов, в то время как другие алгоритмы, в т.ч. и двойного поиска, являются более универсальными в отношении просчитываемых графов.

ЗАКЛЮЧЕНИЕ

В статье представлен алгоритм, реализующий поиск *k*-кратчайших путей на ориентированных графах без контуров с отношением строгого порядка, используемых в качестве моделей (так называемых сетевых графиков) в задачах сетевого планирования и управления проектами. Это позволяет находить подкритические пути на данных моделях с целью выравнивания и оптимизации используемых в проекте ресурсов.

Вышеназванные особенности используемых графов позволили выстроить многоуровневую структуру отношений между специально введенными абстракциями (*p*-контурами), отображающими структурные элементы графов. Это послужило основой для разработки алгоритма, который был численно реализован методом динамического программирования, расширенного за счет использования дополнительного функционального соотношения.

Узкая направленность алгоритма на графы, используемые в управлении проектами, определила его высокую производительность, подтвержденную серией сравнительных вычислительных экспериментов. Эффективность алгоритма устойчива к размерам просчитываемых графов и количеству подкритических путей. Это позволяет рекомендовать его к практическому использованию в информационных системах управления проектами.

СПИСОК ЛИТЕРАТУРЫ / REFERENCES

1. Hagstrom J.N. Computing the probability distribution of project duration in a PERT network. *Networks*. 1990; 20(2):231–244. https://doi.org/10.1002/net.3230200208
2. Kamburowski J., Michael D.J., Stallmann M.F.M. Minimizing the complexity of an activity network. *Networks*. 2000;36(1):47–52. https://doi.org/10.1002/1097-0037(200008)36:1%3C47::AID-NET5%3E3.0.CO;2-Q
3. Bianco L., Caramia M. A new formulation of the resource-unconstrained project scheduling problem with generalized precedence relations to minimize the completion time. *Networks*. 2010;56(4):263–271. https://doi.org/10.1002/net.20388
4. Brennan M. *PERT and CPM: a selected bibliography.* Monticello, Ill.: Council of Planning Librarians, 1968. 11 p.
5. Alagheband A., Soukhakian M.A. An efficient algorithm for calculating the exact overall time distribution function of a project with uncertain task durations. *Indian Journal of Science and Technology*. 2012;5(9):3310–3316. https://doi.org/10.17485/ijst/2012/v5i9.20
6. Damci A., Polat G., Akin F.D., et al. Use of float consumption rate in resource leveling of construction projects. *Front. Eng. Manag.* 2022;9:135–147. https://doi.org/10.1007/s42524-020-0118-0
7. Willis R.J. Critical path analysis and resource constrained project scheduling – Theory and practice. *Eur. J. Oper. Res*. 1985;21(2):149–155. https://doi.org/10.1016/0377-2217(85)90026-8
8. Bowers J.A. Criticality in resource constrained networks. *J. Oper. Res. Soc*. 1995;46(1):80–91. https://doi.org/10.1057/jors.1995.9
9. Shtub A. The trade-off between the net present cost of a project and the probability to complete it on schedule. *J. Oper. Manag*. 1986;6(3–4):461–470. https://doi.org/10.1016/0272-6963(86)90017-3
10. Shtub A. The integration of CPM and material management in project management. *Constr. Manag. Econ*. 1988;6(4):261–272. https://doi.org/10.1080/01446198800000023
11. Ananthanarayanan P.S. Project Technology and Management. In: S. Seetharaman (Ed.). *Treatise on Process Metallurgy*. V. 3. *Industrial Processes.* Elsevier; 2014. P. 1145–1191. https://doi.org/10.1016/B978-0-08-096988-6.00038-9
12. García-Heredia D., Molina E., Laguna M., et al. A solution method for the shared resource-constrained multi-shortest path problem. *Expert Systems with Applications*. 2021;182:115193. https://doi.org/10.1016/j.eswa.2021.115193
13. Fulkerson D.R. Expected critical path lengths in PERT networks: *Oper. Res*. 1962;10(6):745–912. https://doi.org/10.1287/opre.10.6.808
14. Bellman R. On a routing problem. *Quart. Appl. Math.* 1958;16(1):87–90. https://doi.org/10.1090/qam/102435
15. Stern R., Goldenberg M., Saffidine A., et al. Heuristic search for one-to-many shortest path queries. *Ann. Math. Artif. Intell.* 2021;89:1175–1214. https://doi.org/10.1007/s10472-021-09775-x
16. Dreyfus S.E. An appraisal of some shortest-path algorithms. *Operations Research.* 1969;17(3):395–412. https://doi.org/10.1287/opre.17.3.395
17. Clarke S., Krikorian A., Rausan J. Computing the *N* best loopless paths in a network. *J. Soc. Indust. Appl. Math*. 1963;11(4):1096–1102. https://doi.org/10.1137/0111081
18. Pollack M. Solutions of the kth best route through a network – A review. *J. Math. Anal. Appl*. 1961;3(3): 547–559. https://doi.org/10.1016/0022-247X(61)90076-2
19. Shier D.R. Iterative methods for determining the k shortest paths in a network. *Networks*. 1976;6(3):205–229. https://doi.org/10.1002/net.3230060303
20. Shier D.R. On algorithms for finding the *k* shortest paths in a network. *Networks*. 1979;9(3):195–214. https://doi.org/10.1002/net.3230090303

21. Minieka E., Shier D.R. A note on an algebra for the *k* best routes in a network. *IMA J. Appl. Math*. 1973;11(2): 145–149. https://doi.org/10.1093/imamat/11.2.145
22. Eppstein D. Finding the *k* shortest paths. *SIAM J. Comput*. 1999;28(2):652–673. https://doi.org/10.1137/S0097539795290477
23. Yen J.Y. Finding the *K* shortest loopless paths in a network. *Manag. Sci.* 1971;17(11–Theory Series):712–716. Available from URL: http://www.jstor.org/stable/2629312
24. Hershberger J., Maxel M., Suri S. Finding the *k* shortest simple paths: A new algorithm and its implementation. *ACM Trans. Algor.* 2007;3(4): Art. 45(19 p.) https://doi.org/10.1145/1290672.1290682
25. Chen B.Y., Chen X.-W., Chen H.-P., et al. A fast algorithm for finding *K* shortest paths using generalized spur path reuse technique. *Transactions in GIS.* 2021;25(1): 516–533. https://doi.org/10.1111/tgis.12699
26. Hu X.-B., Zhang C., Zhang G.-P., et al. Finding the *k* shortest paths by ripple-spreading algorithms. *Eng. Appl. Artif. Intell*. 2020;87:Art.103229. https://doi.org/10.1016/j.engappai.2019.08.023
27. Hamed A.Y. A genetic algorithm for finding the *k* shortest paths in a network. *Egypt. Inform. J*. 2010;11(2):75–79. https://doi.org/10.1016/j.eij.2010.10.004
28. Hu X.-B., Zhou J., Li H., et al. Finding the *k* shortest paths for co-evolutionary path optimization. In: *IEEE Symposium Series on Computational Intelligence*. November 18–21, 2018; Bangalore, India. P. 1906–1912. https://doi.org/10.1109/SSCI.2018.8628928
29. Liu G., Qiu Z., Chen W. An iterative algorithm for single-pair *K* shortest paths computation. *WSEAS Transactions on Information Science and Applications*. 2015;12(Art. 30):305–314. Available from URL: https://wseas.org/wseas/cms.action?id=10185
30. Guo J., Jia L. A new algorithm for finding the *K* shortest paths in a time-schedule network with constraints on arcs. *J. Algorithms Comput. Technol*. 2017;11(2):170–177. https://doi.org/10.1177/1748301816680470
31. Xu W., He S., Song R., et al. Finding the *K* shortest paths in a schedule-based transit network. *Comput. Oper. Res.* 2012;39(8):1812–1826. https://doi.org/10.1016/j.cor.2010.02.005
32. Jin W., Chen S., Jiang H. Finding the *K* shortest paths in a time-schedule network with constraints on arcs. *Comput. Oper. Res.* 2013;40(12):2975–2982. https://doi.org/10.1016/j.cor.2013.07.005
33. Kaufmann A., Debazeille G. *La méthode du chemin critique*. Paris: Dunod; 1964. 170 p.
34. Bellman R., Kalaba R. On *k*th best policies. *J. Soc. Ind. Appl. Math.* 1960;8(4)582–588. Available from URL: https://www.jstor.org/stable/2099049
35. Shier D.R. Computational Experience with an algorithm for finding the *k* shortest paths in a network. *J. Res. Natl. Bureau Stand*. 1974;78B(3)139–165. https://doi.org/10.6028/JRES.078B.020

Об авторе

Анфёров Михаил Анисимович, д.т.н., профессор, профессор кафедры «Предметно-ориентированные информационные системы» Института кибербезопасности и цифровых технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: anfyorov@inbox.ru. https://orcid.org/0000-0003-2853-6184

About the author

Mikhail A. Anfyorov, Dr. Sci. (Eng.), Professor, Domain-Specific Information Systems Department, Institute of Cybersecurity and Digital Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: anfyorov@inbox.ru. https://orcid.org/0000-0003-2853-6184