УДК 005.8: 004.891

О.С. Тимчук, Я.А. Проценко

МУРАВЬИНЫЙ АЛГОРИТМ РАЗРАБОТКИ АДАПТИВНОГО РАСПИСАНИЯ ПРОЕКТА

Предложен алгоритм разработки адаптивного расписания проекта в условиях неопределенности проектной информации. В качестве инструментального аппарата предложено использовать концепцию мягких вычислений, а именно алгоритм муравьиной оптимизации. Предлагаемый алгоритм выполняет оптимальное разбиение операций на кластеры, а затем оптимизирует выполнение операций внутри кластеров. Рис. 8, табл. 1, ист. 11.

Ключевые слова: управление проектом, адаптивное расписание, мягкие вычисления, муравьиный алгоритм, ACO, mTSP.

JEL C61

Постановка проблемы в общем виде. Согласно PMBoK [1], управление расписанием проекта является одной из основных задач, которая решается проектным менеджером на протяжении всего жизненного цикла проекта. Управление расписанием включает в себя все необходимые процессы, которые напрямую влияют на своевременность выполнения проекта и косвенно на его стоимость и качество конечного продукта. Для составления модели расписания проекта традиционно в PMBoK и Practice standard for scheduling [2] предлагается использовать следующие методы: метод анализа сети расписания, метод оптимизации ресурсов и метод критического пути. Перечисленные методы позволяют определить запланированные даты старта / финиша операций и контрольных событий проекта, а также, в случае необходимости, внести изменения на последующих итерациях проекта. Типичная современная проектная среда отличается высоким уровнем неопределенности исходной информации [3, 4] и непредсказуемостью условий реализации проекта, что требует от проектного менеджера выполнять адаптивное управление расписанием проекта. Кроме этого, в управление проектами все чаще внедряются новые подходы, направленные на равномерное прозрачное распределение независимых / мало зависимых операций между исполнителями с учетом их приоритетности (например, «канбан» [5]). Поэтому задача разработки адаптивных инструментов разработки расписания является актуальной.

Анализ исследований и публикаций и выделение нерешенных ранее частей общей проблемы. Разработка адаптивного расписания в крупном проекте – сложная оптимизационная NP-полная задача, которая не может быть решена менеджером в ручном режиме. Решение задач такого класса со слабо структурированной проектной информацией возможно с использованием технологий мягких вычислений, в частности метаэвристических алгоритмов роевого интеллекта. В [6] применяется алгоритм роя частиц для составления расписания проекта с ограниченными ресурсами. В [7, 8] предлагаются методы составления расписания для IT-проектов, построенные на базе муравьиного алгоритма. В [9] предложена параллельная реализация «муравьиной» эвристики Лагранжа для решения задачи разработки расписания проекта. Несмотря на наличие некоторого количества публикаций, посвященных исследуемому вопросу, все еще остается потребность в новых моделях, методах и алгоритмах для разработки адаптивного расписания проектов.

Цель статьи. Разработать алгоритм составления адаптивного расписания проекта с использованием технологий мягких вычислений.

Постановка задачи. Пусть задано множество операций $T = \{1,2, \dots, n\}$ и проектная команда $E = \{1,2, \dots, m\}$. Каждый исполнитель может выполнить одну или несколько операций из *T*. С учетом особенностей адаптивных проектных сред, будем считать, что операция может выполняться только одним сотрудником, а операции независимы. Необходимо найти такое допустимое расписание выполнения операций, которое минимизирует сроки и стоимость их выполнения, т. е.

$$\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}x_{ij} \to min, \tag{1}$$

$$x_{ij} = \begin{cases} 1, i - \text{й исполнитель вовлечен в выполнение } j - \text{ой операции}, \\ 0, i - \text{й исполнитель не вовлечен в выполнение } j - \text{ой операции}, \end{cases} \tag{2}$$
$$i \in E, j \in T,$$

$$\sum_{j=1}^{n} x_{ij} > 0, i = 1,2, \dots, m, \tag{3}$$

$$\sum_{i=1}^{m} x_{ij} = 1, j = 1,2, \dots, n, \tag{4}$$

где d_{ij} – интегральная оценка выполнения *i*-м исполнителем *j*-ой операции, которая может содержать такие характеристики, как опыт и навыки сотрудника, стоимость 1-го часа работы и т. д.,

x_{ij} – бинарный признак вовлеченности исполнителя в выполнение операции.

(1) и (2) являются ограничениями задачи:

- (1) гарантирует, что каждому исполнителю назначена для выполнения минимум одна операция;
- (2) гарантирует, что каждую операцию выполняет только один исполнитель.

Для решения поставленной задачи авторами статьи предлагается использовать алгоритм оптимизации муравьиной колонии (ACO). В основу метаэвристики алгоритма ACO положены механизмами коллективного непрямого взаимодействия реальных муравьев, которые позволяют им найти оптимальные пути от муравейника к источникам пищи.

Традиционно муравьиный алгоритм формулируется через задачу коммивояжера (TSP), которая состоит в поиске самого выгодного замкнутого маршрута, проходящего через все указанные города ровно один раз. В классической постановке TSP определяется на взвешенном графе G(V, A), где множество вершин графа $V = \{1, \dots, n\}$ соответствует городам, а множество

ребер между вершинами $A = \{(i,j): i,j \in V\}$ – путям сообщения между этими городами. С А ассоциируется матрица $D = \|d_{ij}\|$, где $d_{ij} > 0$ является стоимостью перехода из i-й вершины в j-ю. Под стоимостью обычно понимают расстояние, время или стоимость поездки между городами i и j. Если выполняется условие $d_{ij} = d_{ji}, i,j \in V$, то TSP является симметричной, иначе – асимметричной, т. е. в симметричном случае количество возможных маршрутов вдвое меньше асимметричного случая. Если выполняется неравенство треугольника $d_{ij} + d_{jk} \geq d_{ik}, i,j,k \in V$, то TSP является метрической.

Сформулируем TSP в терминах целочисленного линейного программирования [10]:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \to min, \tag{5}$$

$$x_{ij} = \begin{cases} 1, \text{если } (i,j) \text{ входит в маршрут}, \\ 0, \text{иначе}, \end{cases} \quad \forall (i,j) \in A, \tag{6}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i \in V, i \neq j, \tag{7}$$

$$\sum_{i=1}^{n} x_{ij} = 1, j \in V, i \neq j, \tag{8}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, S \subset V, 2 \leq |S| \leq n-2, \tag{9}$$

$$x_{ij} \in \{0,1\}, (i,j) \in A, \tag{10}$$

где x_{ij} – бинарный признак включения перехода *(i, j)* в маршрут.

(7) – (10) являются ограничениями задачи:

- (7), (8) гарантируют, что коммивояжер посещает город только один раз;
- (9) гарантирует отсутствие подциклов;
- (10) гарантирует, что признак включения перехода в маршрут является бинарным.

Результатом решения задачи TSP является оптимальный путь, состоящий из n городов, которые должен посетить один коммивояжер.

Если спроецировать этот результат на поставленную задачу разработки расписания проекта, то на выходе будет получена оптимальная

последовательность выполнения независимых операций одним сотрудником. Так как в проектной среде практически невозможно встретить проектную команду в количестве одного исполнителя, который выполняет последовательно заданные операции, авторами данной статьи предлагается использовать модель задачи нескольких коммивояжеров (mTSP) без депо, которая состоит в поиске непересекающихся замкнутых маршрутов на графе, при этом каждая вершина должна принадлежать ровно одному маршруту, а стоимость самого длинного пути должна быть минимальной. mTSP, аналогично TSP, определяется на взвешенном графе *G(V, A)*. Сформулируем задачу mTSP без депо в терминах целочисленного линейного программирования [10]:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ijk} \to min, k = 1, \dots, m, \tag{11}$$

$$x_{ijk} = \begin{cases} 1, \text{если } (i,j) \text{ входит в маршрут } k-\text{го коммивояжера}, \\ 0, \text{иначе}, \end{cases} \quad \forall (i,j) \in A, k \in M, \tag{12}$$

$$\forall k \in M: \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij1} \geq \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ijk}, \tag{13}$$

$$\forall i \in V: \sum_{k=1}^{m} u_{ik} = 1, \tag{14}$$

$$\forall k \in M, i \in V: \sum_{j=1}^{n} x_{ijk} = 2u_{ik}, \tag{15}$$

+ ограничения, которые запрещают подциклы.

где $M = \{1, \dots, m\}$ – множество коммивояжеров,

u_{ik} – бинарный признак включения *i*-й вершины в маршрут *k*-го коммивояжера.

Также допускаем, не ограничивая всеобщность формулировки, что самый длинный путь прошел коммивояжер с индексом 1. (13) – (15) являются ограничениями задачи:

- (13) гарантирует, что путь первого коммивояжера не является короче путей остальных коммивояжеров;
- (14) гарантирует, что каждую вершину должен посетить ровно один коммивояжер;
- (13) и (14) не позволяют коммивояжерам посещать вершины, которые принадлежат маршрутам других коммивояжерам;
- (15) гарантирует, что каждый коммивояжер только один раз входит и выходит из вершины.

Проведем аналогию между mTSP и алгоритмом ACO: коммивояжер ассоциируется с муравьем, маршрут коммивояжера – с маршрутом муравья, а стоимостная характеристика d_{ij} – с эвристической информацией, которая

участвует в формировании уровня феромона, который оставляет муравей на пройденном пути.

Проведем аналогию между mTSP и поставленной задачей разработки расписания проекта: вершины графа ассоциируются с операциями; коммивояжеры — с члены проектной команды; *k*-й маршрут — последовательность операций, которые должен выполнить *k*-й исполнитель.

Справочная информация. В основе предлагаемого в данной работе алгоритма разработки расписания проекта лежит алгоритм ACO, предложенный M. Dorigo [11]. Для однозначного понимания и трактовки излагаемого в статье материала, рассмотрим основные положения классического ACO алгоритма для решения TSP.

В алгоритме ACO колония муравьев состоит из множества муравьев $E = \{1,2, \dots, m\}$. Последовательно муравьи колонии начинают строить маршрут, перемещаясь по всем вершинам графа *G*. Направление движения муравья определяется с помощью вероятностно-пропорционального правила перехода, которое определяет вероятность перехода *k*-го муравья из вершины $i \in V$ в вершину $j \in V$ на *t*-й итерации

$$p_k(i,j) = \begin{cases} \dfrac{\tau_{ij}(t) \cdot (\eta_{ij})^{\alpha}}{\sum_{u \in J_k(i)} \tau_{iu}(t) \cdot (\eta_{iu})^{\alpha}}, \text{если } s \in J_k(i) \\ 0 \qquad\qquad , \text{иначе,} \end{cases}, \tag{16}$$

где $\tau_{ij}(t)$ – уровень феромона на ребре *(i, j)* в момент времени *t*,

η_{ij} – величина, обратная d_{ij},

$J_k(i)$ –множество вершин, в которые муравей может перейти из вершины *i*,

$\alpha, \alpha > 0$ – регулируемый параметр, который определяет важность уровня феромона относительно расстояния.

Выбор города осуществляется по принципу рулетки, где сектора соответствуют вершинам из $J_k(i)$, а их площадь пропорциональна посчитанным вероятностям.

После того, как муравей построил маршрут, выполняется процедура обновления уровня феромона на каждом переходе *(i, j)* по формуле

$$\tau_{ij}(t) = (1 - \varepsilon) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_k(i,j), \tag{17}$$

$$\Delta\tau_k(i,j) = \begin{cases} \dfrac{1}{L_k}, \text{если } (i,j) \in \text{маршруту } k - \text{го муравья,} \\ 0, \qquad \text{иначе} \end{cases} \tag{18}$$

где $\varepsilon, 0 < \varepsilon < 1$ – регулируемый параметр, который определяет скорость испарения феромона,

L_k – стоимость пути *k*-го муравья.

После того, как все муравьи колонии построят свои маршруты, определяется оптимальный путь. При этом следует отметить, что ACO алгоритм дает приближенное решение TSP.

Алгоритм решения TSP с помощью алгоритма ACO представлен на рис. 1.

Описание алгоритма. Разработанный алгоритм решения mTSP основан на базовых принципах алгоритма ACO с некоторыми модификациями. Рассмотрим основные принципы предлагаемого алгоритма.

Маршрут. В классическом ACO алгоритме каждый муравей колонии решает задачу поиска оптимального маршрута, проходя через все вершина графа *G*. В предлагаемом алгоритме муравей имитирует работу нескольких коммивояжеров, и поэтому весь путь муравья разбивается на m непересекающихся фрагментов. Муравей последовательно проходит путь сначала одного коммивояжера, строя замкнутый маршрут, затем «перепрыгивает» в новую вершину и начинает проходить путь второго коммивояжера, и так далее, пока не будут пройдены все вершины графа *G*. Таким образом, маршрут муравья будет состоять из «пеших переходов» и «скачков», при этом скачков будет ровно m-1.

Феромон. В классическом ACO алгоритме реализуется идея феромона как способ непрямого взаимодействия между муравьями колонии. В TSP муравьи оставляют феромоны на переходах *(i, j)* и когда очередной муравей колонии оказывается в *i*-ой вершине, он выбирает один из имеющихся переходов с вероятностью, пропорциональной количеству феромона на домтупных переходах. В предлагаемом алгоритме идея феромона имеет такой же смысл – формирование степени «привлекательности» перехода для муравья, но при принятии решения осуществлять скачок или нет, значение уровня феромона используется несколько иным образом. Находясь в определенной вершине, муравей может продолжить путь коммивояжера, или замкнуть его в петлю и перейти к формированию маршрута следующего коммивояжера. Количество феромона между текущей вершиной и вершиной, в котором начался путь коммивояжера, и определяет «привлекательность» решения завершить путь коммивояжера и совершить скачок в другую вершину. Вершина, в которую осуществляется «скачок», выбирается без участия феромона, но с использованием эвристики, которая будет описана ниже. На рис. 2 приведен пример пути, который строит муравей.

На рис. 2 переходы (1, 2), (2, 3), (4, 5) и (5, 6) являются обычными «пешими переходами», а переход (3, 4) – «скачок», который означает завершение пути первого коммивояжера и начало пути следующего. Дуги (3, 1) и (6, 4) соединяют начало и конец сегментов непрерывного пешего пути муравья и замыкают пути коммивояжеров. Совершив такую последовательность переходов, муравей создал замкнутые пути 1-2-3 и 4-5-6, которые принадлежат двум различным коммивояжерам.

Вероятностно-пропорциональное правило перехода. В классическом ACO алгоритме, направление движения муравья определяется с помощью вероятностно-пропорционального правила перехода. В предлагаемом алгоритме на каждой итерации на основе рассчитанных вероятностей выбирается следующая вершина на пути текущего муравья или первая вершина на пути следующего коммивояжера.

1	**Инициализировать параметры алгоритма:**
2	• определить матрицу $D = \|d_{ij}\|$
3	• инициализировать матрицу феромонов $PH = \|\tau_{ij}\|$ начальными значениями
4	• определить количество муравьев в колонии (m)
5	• определить параметры α и ε
6	**Повторять**
7	Инициализировать параметры *k*-го муравья
8	**Повторять**
9	Для *i*-ой вершины сформировать множество $J_k(i)$
10	Рассчитать вероятности перехода муравья в вершины из $J_k(i)$
11	Определить для перехода вершину $j \in J_k(i)$
12	Перейти в вершину *j*
13	Добавить вершину *i* в «табу»-список
14	**Пока k-й муравей не посетит все вершины графа *G(V, A)***
15	Определить стоимость маршрута *k*-го муравья
16	Для каждого перехода *(i, j)* испарить уровень феромона
17	На маршруте *k*-го муравья увеличить уровень феромона
18	**Пока все муравьи колонии не построят маршрут**
19	**Определить оптимальный маршрут**

Рис. 1. Алгоритм ACO

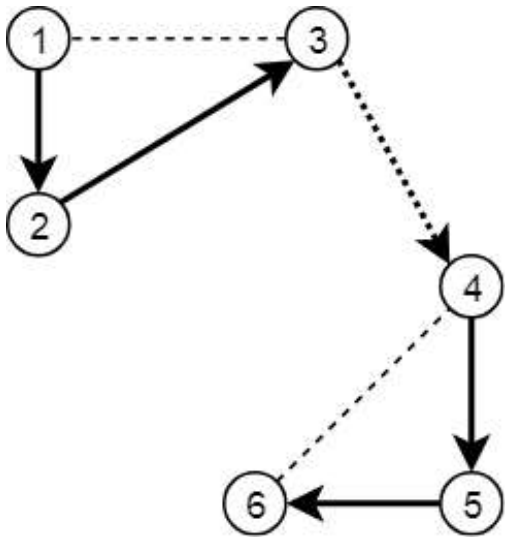


Рис. 2. Пример построения пути муравьем

Таким образом, если на определенной итерации имеем *N* не посещённых вершин, нужно выбрать одно из *2N* действий. Вероятностное распределение описывается следующими формулами:

$$P_{move}(j) = \frac{(\eta_{ij})^{\alpha} \cdot (\tau_{ij})^{\beta}}{(\eta_{ig})^{\alpha} \cdot (\tau_{ig})^{\beta} + \sum_{k \in N} (\eta_{ik})^{\alpha} \cdot (\tau_{ik})^{\beta}} \quad (19)$$

$$P_{jump}(j) = \frac{(\eta_{ig})^{\alpha} \cdot (\tau_{ig})^{\beta}}{(\eta_{ig})^{\alpha} \cdot (\tau_{ig})^{\beta} + \sum_{k \in N} (\eta_{ik})^{\alpha} \cdot (\tau_{ik})^{\beta}} \cdot \frac{(d_{ij})^{\gamma}}{\sum_{k \in N} (d_{ik})^{\gamma}} \quad (20)$$

где *g* – индекс вершины, в которой начался путь текущего коммивояжера,

α, *β* и *γ* – параметры алгоритма.

Несложно убедиться, что сумма всех вероятностей равна единице, а сумма вероятностей «скачков» равна вероятности перехода в вершину, которая является началом пути коммивояжера.

Применение такого распределения никак не гарантирует, что «скачков» будет ровно m-1. Поэтому необходимо рассмотреть два предельных случая:

1. Когда вершин осталось не больше, чем коммивояжеров – необходимо всегда осуществлять «скачок».
2. Когда совершено m-1 «скачков» – необходимо всегда осуществлять пеший переход.

Первый случай гарантирует, что в сгенерированном решении будет не менее m путей. При этом вершина для «скачка» выбирается произвольно, поскольку это не влияет на качество решения. Второй случай гарантирует, что в сгенерированном решении будет не более чем m путей. При этом вершина для последующего перехода выбирается с учетом вероятностей, которые определяются по формуле

$$P_{move}(j) = \frac{(\eta_{ij})^{\alpha} \cdot (\tau_{ij})^{\beta}}{\sum_{k \in N} (\eta_{ik})^{\alpha} \cdot (\tau_{ik})^{\beta}} \quad (21)$$

Алгоритм *Вероятностно-пропорционального правила перехода* приведен на рис. 3.

Обновления феромона. Обновление феромона в предлагаемом алгоритме осуществляется по формуле

$$\Delta\tau = K\frac{1}{L},$$

где *K* – константа,

L – стоимость самого длинного пути в сгенерированном решении.

Экспериментально установлено, что в качестве *K* эффективно использовать величину 1 / Q, где Q – количество муравьев на одну итерацию алгоритма. Матрица феромонов обновляется аналогично классическом ACO алгоритму – приращение получают феромоны на всех переходах между смежными вершинами всех путей решения, включая переход между первой и последней вершиной каждого пути. Поскольку во время выполнения алгоритма генерация решений выполняется последовательно, обновления феромона необходимо накапливать в отдельных буферных матрицах, и обновлять матрицы феромонов только после того, как все *Q* муравьев построят маршруты.

Параметры алгоритма:

- количество вершин (n)
- количество муравьев в колонии (m)
- матрица $D = \|d_{ij}\|$
- матрица феромонов $PH = \|\tau_{ij}\|$
- параметры β, α и γ

$S \leftarrow \emptyset$
$p \leftarrow [1]$
$C \leftarrow \{2, \dots, n\}$
WHILE *C* not empty:
    $i \leftarrow p[length(p)]$
    IF $|S| < m$:
        IF $|C| + |S| > m$:
            $g \leftarrow p[1]$
            jump, *j* ← выбрать согласно формулам (19, 20)
        ELSE:
            jump ← True
            j ← выбрать любое из *C*
        ENDIF
    ELSE:
        jump ← False
        j ← выбрать согласно формуле (21)
    ENDIF
    IF jump:
        $S \leftarrow S \cup \{p\}$
        $p \leftarrow [j]$
    ELSE:
        append(*p*, *j*)
    ENDIF
    $C \leftarrow C/\{j\}$
ENDWHILE
$S \leftarrow S \cup \{p\}$
RETURN S

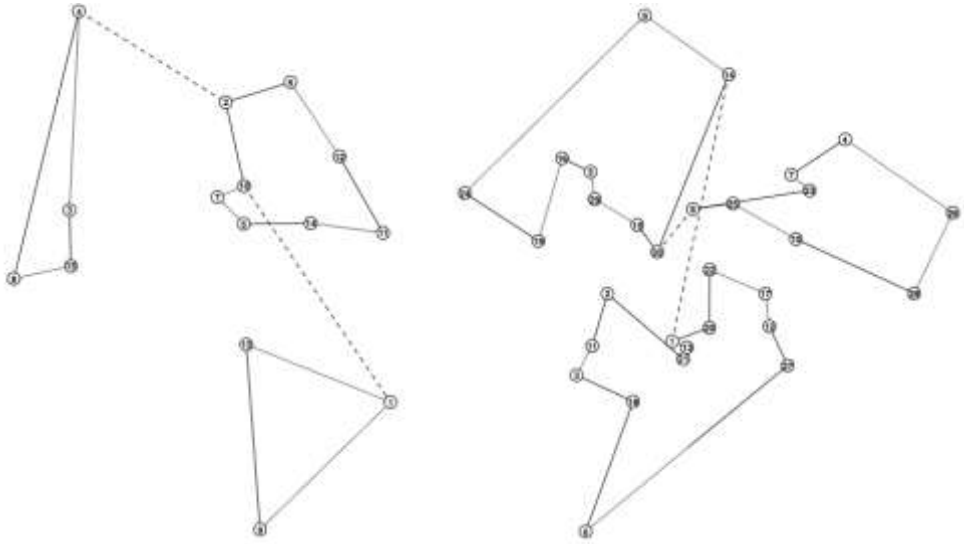
Рис. 3. Алгоритм *Вероятностно-пропорционального правила перехода*

Полный алгоритм. Объединяя идеи, изложенные в предыдущих подразделах, приведем полный алгоритм решения задачи mTSP без депо с помощью модифицированного ACO алгоритма (рис. 4). В качестве критерия завершения алгоритма выбрано выполнение заданного количества итераций. Результатом работы алгоритма является лучшее эвристически-случайное решение, которое было сгенерировано за все время работы алгоритма.

1	**Инициализировать параметры алгоритма:**
2	• определить количество вершин (*n*)
3	• определить количество муравьев в колонии (*m*)
4	• определить матрицу $D = \left\|d_{ij}\right\|$ размером *n* x *n*
5	• инициализировать матрицу феромонов $PH = \left\|\tau_{ij}\right\|$ размером n x n значениями $\frac{1}{n \cdot mean\ (\left\|d_{ij}\right\|)}$
6	• определить параметры β, α и γ
7	• определить лучший маршрут как $L_{best} \leftarrow +\infty$
8	• определить стоимость лучшего маршрута как $S_{best} \leftarrow \emptyset$
9	**REPEAT** n_{iter} times:
10	$\left\|\Delta\tau_{ij}\right\| \leftarrow$ нулевая матрица размером *n* x *n*
11	**REPEAT** *Q* times:
12	*S* ← сгенерировать решение по Алгоритму 1
13	*L* ← длина самого длинного пути в *S*
14	**IF** $L < L_{best}$:
15	$S_{best} \leftarrow S$
16	$L_{best} \leftarrow L$
17	**ENDIF**
18	**FOR** ***p*** **in** ***S*****:**
19	**FOR** *i*, *j* **in** все последовательные пары вершин в *p*:
20	$\Delta\tau_{ij} \leftarrow \Delta\tau_{ij} + \frac{K}{L}$
21	$\Delta\tau_{ji} \leftarrow \Delta\tau_{ji} + \frac{K}{L}$
22	**ENDFOR**
23	**ENDFOR**
24	**ENDREPEAT**
25	**FOR** *i* **in** $\{1, \dots, n\}$:
26	**FOR** *j* **in** $\{1, \dots, n\}/\{i\}$:
27	$\tau_{ij} \leftarrow (1 - \varepsilon) \cdot \tau_{ij} + \Delta\tau_{ij}$
28	**ENDFOR**
29	**ENDFOR**
30	**ENDREPEAT**
31	**RETURN** S_{best}

Рис. 4. Алгоритм решения задачи mTSP без депо

Анализ и обобщение полученных результатов. С целью экспериментальной оценки качества разработанного алгоритма и подбора эффективных параметров было сгенерировано несколько случайных наборов точек на двумерной плоскости. Координаты точек сгенерированы с помощью нормального распределения с нулевым средним и единичной дисперсией. Матрица расстояний между узлами графа определена как матрицу Евклидовых расстояний между точками. Алгоритм был протестирован на графах с 15 и 30 узлами для решения задачи с тремя коммивояжерами. На рис. 5 представлены найденные решения для графов с 15 и 30 узлами.



а) б)

Рис. 5. Результат решения задачи mTSP без депо с 3я коммивояжерами (Эксперимент 1)
(а) – граф с 15 вершинами, (б) – граф с 30 вершинами

В терминах поставленной в данной работе задачи, полученные результаты интерпретируются следующим образом: алгоритм выполнил оптимальное распределение 15 (рис. 5а) и 30 (рис. 5б) операций между 3 членами проектной команды; для каждого члена проектной команды построил оптимальную последовательность выполнения операций. Графическое представление построенного расписания в виде линейчатых диаграмм представлено на рис. 6а и 6б для 15 и 30 операций соответственно.

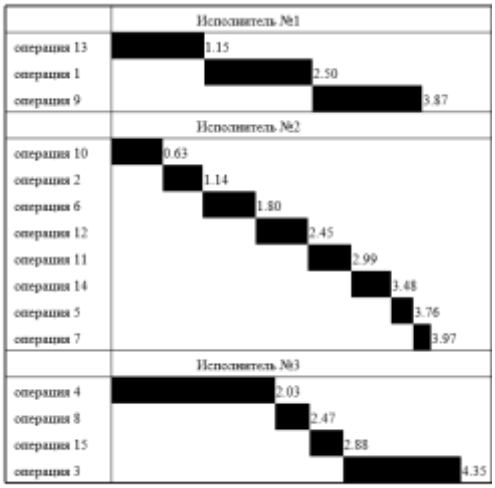
При проведении серии экспериментов были определены параметры алгоритма (табл. 1), которые позволили получить решения, близкие к оптимальным.

Таблица 1

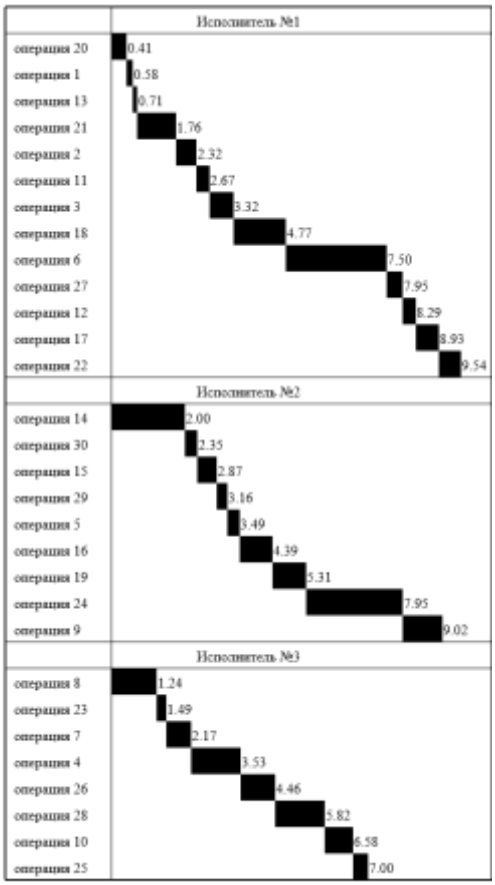
Рекомендуемые значения параметров разработанного алгоритма

Количество вершин в графе	α	β	γ	ε	n_{iter}	Q
15 вершин	2	1	10^{-2}	10^{-3}	500	100
30 вершин	2	1	10^{-2}	10^{-5}	300	500

Эффект от настройки параметров представлен на рис. 7. С целью сравнения результатов, на рис. 7а представлен результат без настройки параметров, а на рис. 7б – с учетом значений параметров, приведенных в табл. 1. Линейчатые диаграммы расписания представлены на рис. 8.

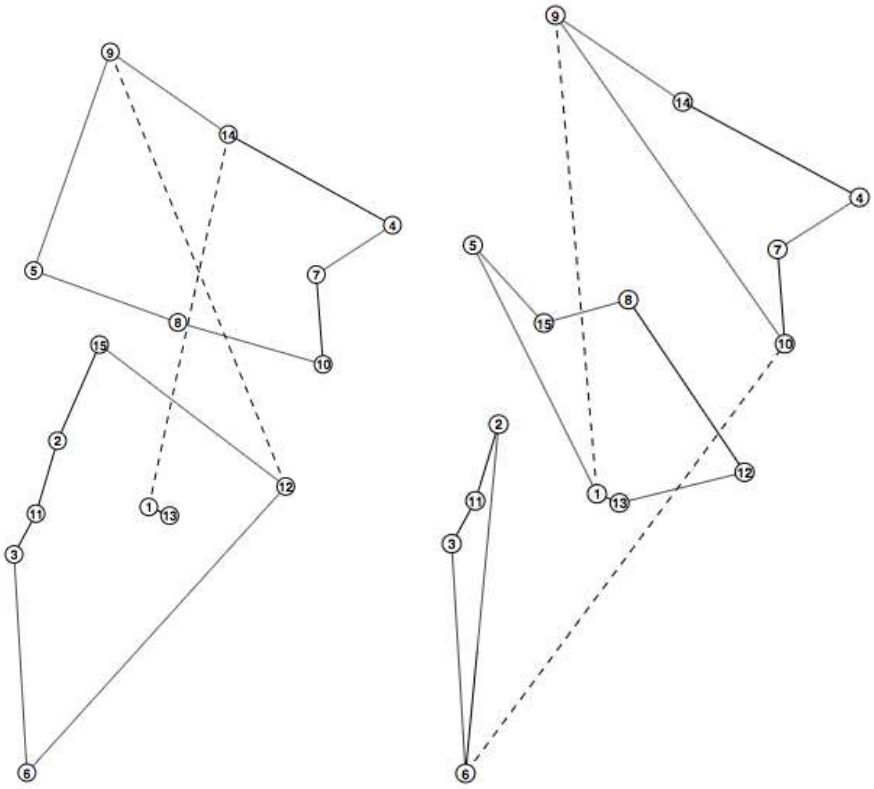


а)



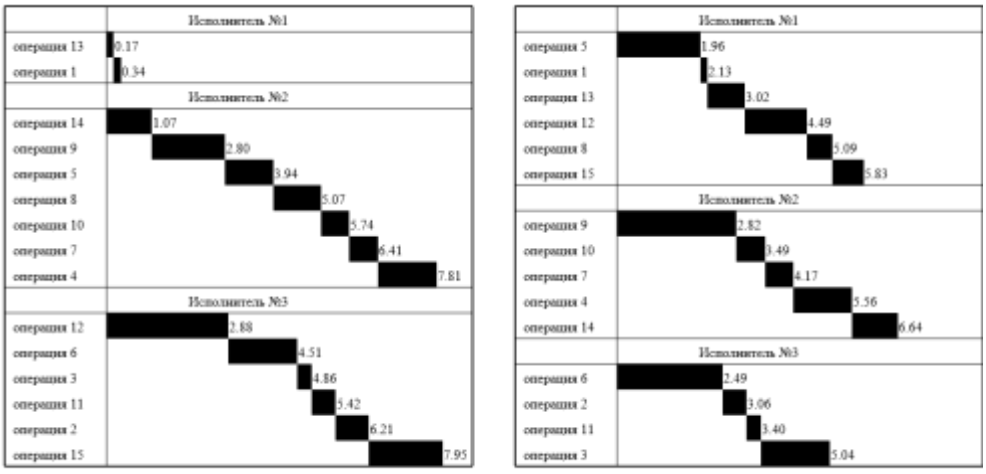
б)

Рис. 6. Расписание проекта (Эксперимент 1)



а) б)

Рис. 7. Результат решения задачи mTSP без депо с тремя коммивояжерами (Эксперимент 2)



а) б)

Рис. 8. Расписание проекта (Эксперимент 2)

Выводы. Проведенные исследования позволяют сделать следующие выводы:

1. По причине неопределенности проектной информации, непредсказуемости условий реализации проекта, а также появления новых практик управления проектами, установлена необходимость в разработке новых моделей и методов адаптивного управления расписанием проекта.
2. В качестве инструментального аппарата для построения адаптивных моделей и методов управления расписанием предлагается использовать концепцию мягких вычислений.
3. Разработан алгоритм составления расписания для параллельного выполнения не зависимых операций с учетом их приоритетности и доступных ресурсов на базе метаэвристического алгоритма оптимизации муравьиной колонии.
4. Выполнено тестирование алгоритма для выполнения 15 и 30 операций с тремя исполнителями. Даны рекомендации по подбору параметров алгоритма.

ЛІТЕРАТУРА

1. A Guide to the Project Management Body of Knowledge (PMBOK Guide). 6th. ed. / Project Management Institute, 2017. – 756 p.
2. Practice standard for scheduling. 2nd ed. / Project Management Institute, 2011. – 142 p.
3. Navigating complexity: a practice guide / Project Management Institute, 2014. – 102 p.
4. Morris Peter. Reconstructing Project Management / Peter Morris. – Wiley-Blackwell, 2013. – 342 p.
5. Hammarberg Marcus. Kanban in Action / M. Hammarberg, J. Sunden. – Manning Publications, 2014. – 360 p.
6. Cui J. An efficient discrete particle swarm optimization for solving multi-mode resource-constrained project scheduling problem / J. Cui, L. Yu // IEEE International Conference on Industrial Engineering and Engineering Management. – 2014. – Bandar Sunway, Malaysia. – pp. 858-862.
7. Han W. An Ant Colony Optimization Algorithm for Software Project Management / W. Han, X. Zhang, H. Jiang, W. Li // 7th International Conference on Control and Automation. – 2014. – Haikou, China. – pp. 19-23.
8. Rachman V. Comparative analysis of ant colony extended and mix-min ant system in software project scheduling problem / V. Rachman, M.A. Ma'sum // International Workshop on Big Data and Information Security (IWBIS). – 2017. – Jakarta, Indonesia. – pp. 85-91.
9. Brent O. A Parallel Lagrangian-ACO Heuristic for Project Scheduling / O. Brent, D. Thiruvady, A. Gómez-Iglesias, R. Garcia-Flores // 7th International Conference on Control and Automation. – 2014. – Beijing, China. – pp. 2985-2991.
10. Bektas T. The multiple traveling salesman problem: An overview of formulations and solution procedures / T. Bektas // Omega, 2006. - №34. – pp. 209-219.
11. Dorigo M., Gambardella, L.M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem / M. Dorigo, L.M. Gambardella // IEEE Transactions on Evolutionary Computation. – 1997. – Vol. 1, No. 1. – pp. 53-66.

Рецензент статті
к.т.н., проф. Морозов В.В.

Стаття рекомендована до
публікації 27.03.2019 р.