Ватутин Э.И., Мартынов И.А., Титов В.С.

Анализ результатов использования метода перебора с ограничением глубины в задаче поиска кратчайшего пути в графе

Существует обширный класс задач дискретной комбинаторной оптимизации, нахождение оптимального решения которых требует реализации комбинаторного перебора с целью анализа качества всех возможных решений с последующим выбором наилучшего. К ним относятся многие задачи из области теории графов, исследования операций, математического программирования с ограничениями на целочисленность решений и пр. Асимптотическая временная сложность подобного перебора обычно факториально или экспоненциально зависит от размерности задачи *N*, что на практике ограничивает возможность получения оптимальных решений лишь для задач малой размерности (за исключением задач, имеющих быстрые алгоритмы с полиномиальной сложностью) и вынуждает использовать различные эвристические подходы. Одним из широко известных подходов является жадный подход, суть которого сводится к выбору такого направления движения (ветви) в соответствующем дереве комбинаторного перебора, которое соответствует локально минимальному приращению минимизируемой целевой функции на каждом шаге. Данный подход является последовательным (обеспечивает получение одного субоптимального решения) и, как следствие, обеспечивает получение решения с малыми затратами вычислительного времени ввиду обхода лишь одной из ветвей комбинаторного дерева. В некоторых задачах (например, в задаче построения минимального остовного дерева [1]) он гарантирует получение оптимальных решений, однако в большинстве задач качество решений, получаемых с его использованием, существенно отстает как от оптимума, так и от решений, получаемых с использованием других эвристических подходов (как специализированных, например [2–4], так и универсальных [5, 6]). Данная особенность является следствием локального характера процедуры выбора направления движения в комбинаторном дереве. Так, например, несколько начальных шагов, характеризующихся малым приращением целевой функции, не гарантируют отсутствия на более глубоких ярусах дерева б**о**льших приращений, существенно ухудшающих результирующее качество найденного решения, которых можно было бы избежать путем организации движения по другой ветви дерева, зная о поджидающих неприятностях.

Выбор направления движения (ветви дерева) желательно производить на основании более глубокого анализа (в предельном случае – вплоть до листьев комбинаторного дерева), однако это требует организации рекуррентных спусков с последующими возвратами и фактически соответствует стратегии полного перебора. Кажущееся противоречие между низким качеством быстро формируемых жадных решений и необходимостью существенно б**о**льших временн**ы**х затрат на глубокий анализ можно разрешить путем введения ограничения на глубину *S* анализа дерева. При этом для выбора направления движения из текущего узла необходимо произвести анализ возможных направлений, производя серию рекуррентных спусков и возвратов не более чем на *S* ярусов дерева в глубину (фактически, обход поддерева с корнем в текущей вершине и высотой *S*), оценить полученные приращения и выбрать направление движения, характеризующееся минимальным из них. Данному подходу по прежнему свойственна определенная жадность, однако с ростом глубины *S* следует ожидать уменьшения ее влияния в угоду более глубокому анализу качества решений при полиномиальном росте затрат вычислительного времени. Данный подход является определенным компромиссом между жадным и полным перебором: при значении глубины $S=1$ он соответствует жадному, а при значениях *S*, превышающих высоту дерева *V*, – полному перебору.

Рассмотрим применение описанного подхода на примере решения задачи поиска кратчайшего пути между заданной парой вершин $a_{нач}$ и $a_{кон}$ в графе $G=\langle A,V\rangle$, где $A=\{a_1,a_2,...,a_N\}$ – множество вершин, $N=|A|$ – число вершин, $V=\{v_1,v_2,...,v_M\}\subseteq A\times A$ – множество дуг, $M=|V|$ – число дуг, причем дуги взвешены значениями длины перехода $l(a_i,a_j)$. Необходимо найти такой путь $P=[a_{i_1},a_{i_2},...,a_{i_Q}]$ между заданной парой вершин $a_{i_1}=a_{нач}$ и $a_{i_K}=a_{кон}$, что $L=\sum_{j=1}^{Q-1}l(a_{i_j},a_{i_{j+1}})\to\min$. В некоторых случаях граф может не быть полносвязным, что соответствует отсутствию переходов между некоторыми парами вершин, при этом соответствующая длина перехода обычно условно принимается равной бесконечности, а граф характеризуется значением «плотности» $d(G)=\frac{M}{N(N-1)}\in[0;1]$ [6]. Для решения данной задачи известен быстрый алгоритм Дейкстры [7], обеспечивающий нахождение оптимального решения за время $O(N^2)$. Это позволят выполнение достаточно простой оценки качества оптимального решения с целью сопоставления с ним качества решений, получаемых с использованием различных эвристических методов [5, 6], и формулировке последующих рекомендаций о целесообразности их применимости на практике.

Алгоритм перебора с ограничением глубины может быть представлен в следующем виде.

1. (инициализация) Положить текущий путь состоящим из начальной вершины $P:=[a_{нач}]$, множество нерассмотренных вершин $\tilde{A}:=A\setminus\{a_{нач}\}$.
2. Найти очередной участок пути P', начинающегося в последней вершине пути P, характеризующегося минимальной длиной $L(P')\to\min$ среди множества других возможных путей и включающего не более S вершин. Если нахождение очередного участка невозможно $(P'=[])$, перейти к п. 5.
3. Добавить найденный участок пути P' к пути P: $P:=P\odot P'$, где «$\odot$» – обозначение операции конкатенации, исключить вершины $a_i\in P'$ из множества нерассмотренных: $\tilde{A}:=\tilde{A}\setminus\{a_i\},\forall a_i\in P'$.
4. Если последняя вершина пути P равна конечной $a_{кон}$, перейти к п. 5, в противном случае перейти к п. 2.
5. Конец алгоритма.

Он базируется на алгоритме нахождения очередного участка пути, начинающегося в указанной вершине a_{i_0}, характеризующегося минимальной длиной среди множества других возможных путей и включающего не более S вершин. Данный алгоритм производит обход в глубину соответствующего поддерева с корнем в вершине a_{i_0} и высотой не более S и представлен ниже.

1. (инициализация) Положить текущий участок пути $P':=[a_{i_0}]$, наилучший участок пути $P^+:=[]$, $L(P^+):=\infty$, текущую глубину рекурсии $D:=0$.
2. (условие завершения рекурсии) Если достигнута максимальная глубина рекурсии $(D=S)$ или последняя вершина текущего участка пути равна $a_{кон}$, то

   2.1. Если:

   - очередной участок пути еще не найден $(L(P^+)=\infty)$ или

- найденный участок приводит к вершине $a_{кон}$ или
- найденный участок короче предыдущего $\left(L\left(P'\right) < L\left(P^{+}\right)\right)$, то

запомнить найденный участок пути как наилучший: $P^{+} := P'$.

2.2. Произвести рекуррентный возврат.

3. (рекуррентный спуск) Для всех нерассмотренных вершин $a_j \in \tilde{A}$, имеющих дуги связи с последней вершиной пути P': осуществить рекуррентный спуск (переход к п. 2 алгоритма) с увеличением значения глубины рекурсии $\left(D := D+1\right)$ и добавлением вершины к текущему пути $P' := P' \odot \left[a_j\right]$.
4. Если $L\left(P^{+}\right) \neq \infty$, вернуть P^{+} в качестве найденного пути, в противном случае решение не найдено.
5. Конец алгоритма.

С целью пояснения деталей предложенного подхода рассмотрим решение задачи на примере нахождения кратчайшего пути в графе (рис. 1, а) между парой вершин a_1 и a_9. Соответствующее дерево комбинаторного перебора приведено на рис. 2, б.

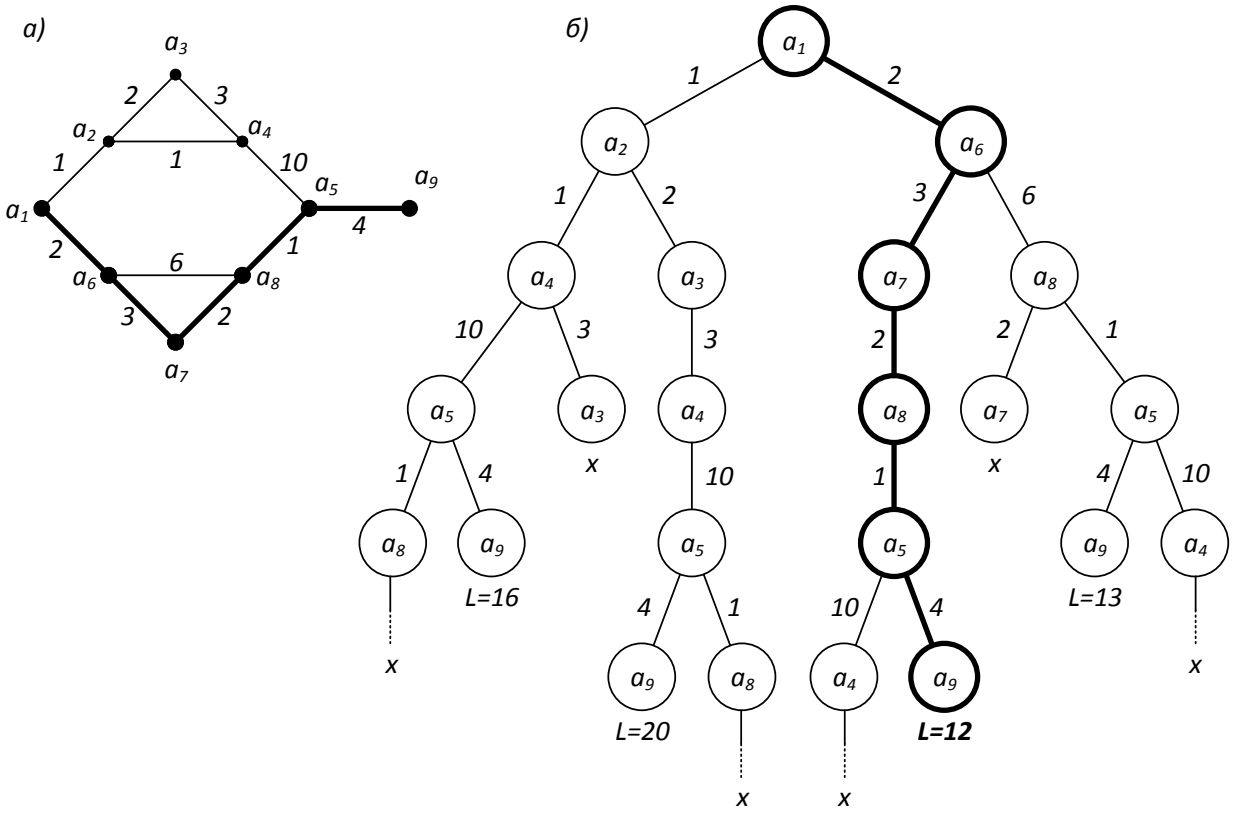

Рис. 1. Пример графа (а) и соответствующее ему дерево возможных путей между вершинами a_1 и a_9 (б). Крестами отмечены пути и направления, приводящие в тупик, жирным выделено оптимальное решение (кратчайший путь)

Следуя жадной стратегии поиска пути $\left(S=1\right)$, производится построение пути $\left[a_1, a_2, a_4, a_3\right]$, заканчивающегося тупиком. При увеличении глубины анализа дерева *S* до 2 при движении из вершины a_1 производится анализ поддерева, образованного путями

$P_1=[a_1,a_2,a_4]$, $P_2=[a_1,a_2,a_3]$, $P_3=[a_1,a_6,a_7]$, $P_4=[a_1,a_6,a_8]$ не более чем из двух дуг, которым соответствуют длины $L(P_1)=2$, $L(P_2)=3$, $L(P_3)=5$, $L(P_4)=8$. Из полученного множества путей выбирается кратчайший (P_1), для него повторяются рассмотренные выше действия, что в итоге приводит к формированию результирующего пути $[a_1 \to a_2 \to a_4 \to a_5 \to a_8 \to ...]$, также приводящего в тупик (рис. 2).

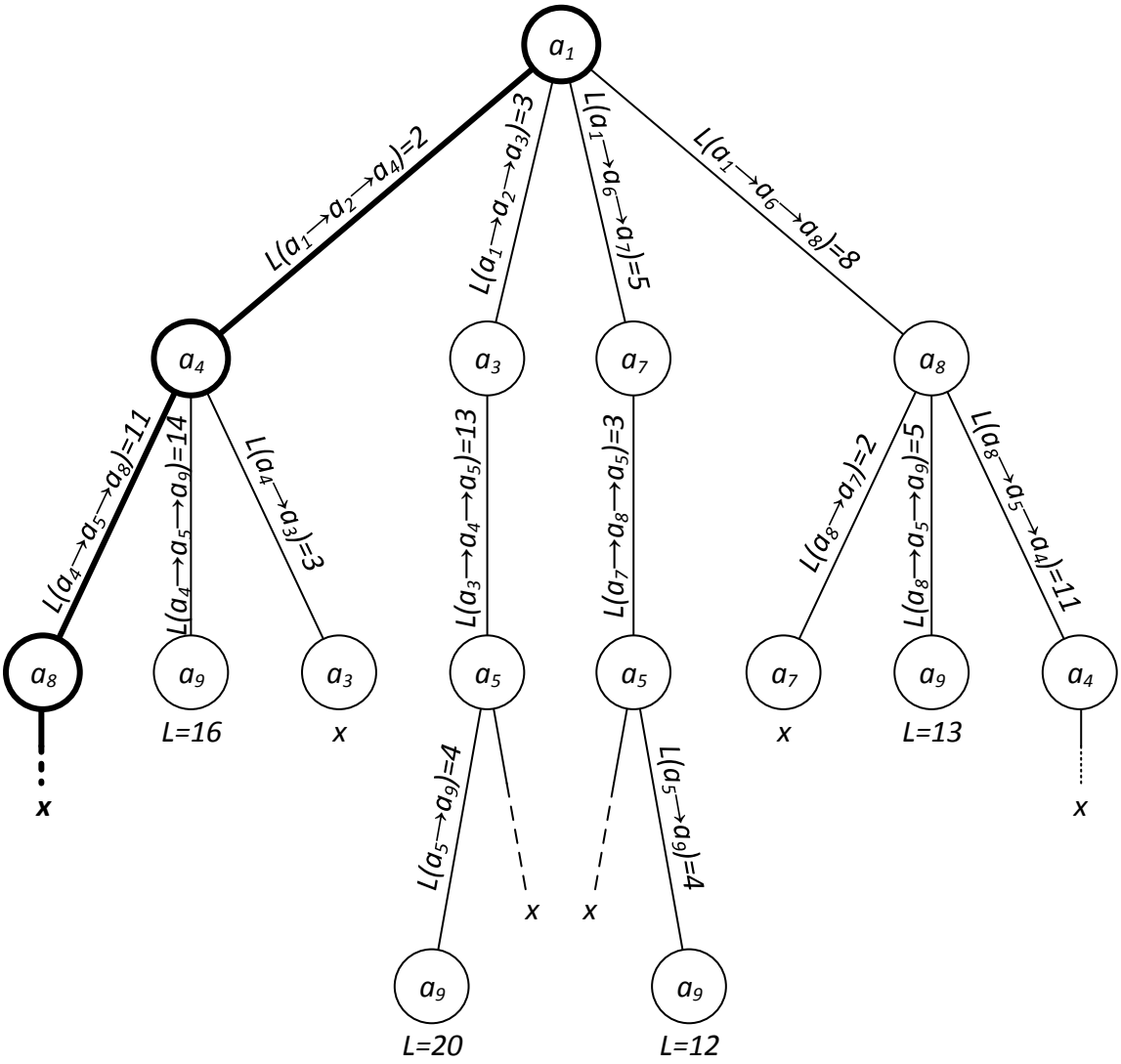

Рис. 2. Дерево возможных путей при ограничении на глубину перебора $S=2$: найденный путь (выделен жирным) $a_1 \underbrace{\to a_2 \to a_4}_{1\ ярус} \underbrace{\to a_5 \to a_8}_{2\ ярус} \to ...$ заходит в тупик

Случаи для ограничения глубины $S=3$ и $S=4$ приведены соответственно на рис. 3 и 4. В них ввиду более глубокого анализа найденный путь сперва перестает быть тупиковым, а затем становится равным оптимальному (следует отметить, что при этом глубина $S=4$ не достигает высоты дерева *V*, равной в приведенном примере 7). Также приведенные примеры наглядно демонстрируют рост арности узлов дерева, что требует б**о**льших временных затрат на их обход с ростом глубины *S*.

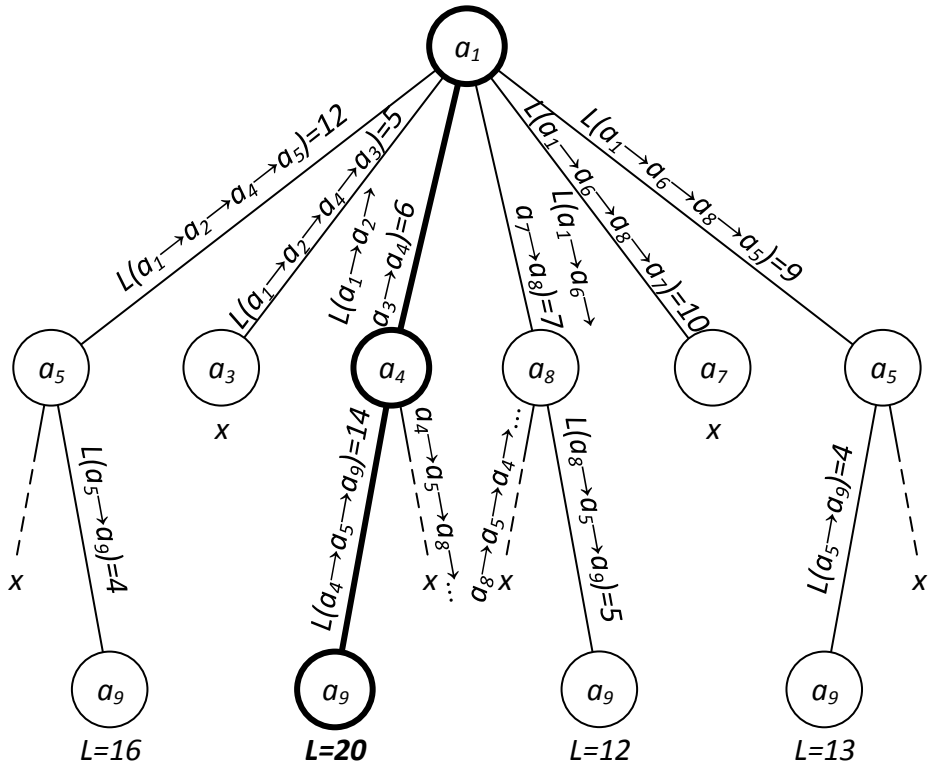

Рис. 3. Дерево возможных путей при ограничении на глубину перебора $S=3$: найденный путь $a_1 \underbrace{\rightarrow a_2 \rightarrow a_3 \rightarrow a_4}_{1\ ярус} \underbrace{\rightarrow a_5 \rightarrow a_9}_{2\ ярус}$ имеет длину $L=20$ и не является оптимальным

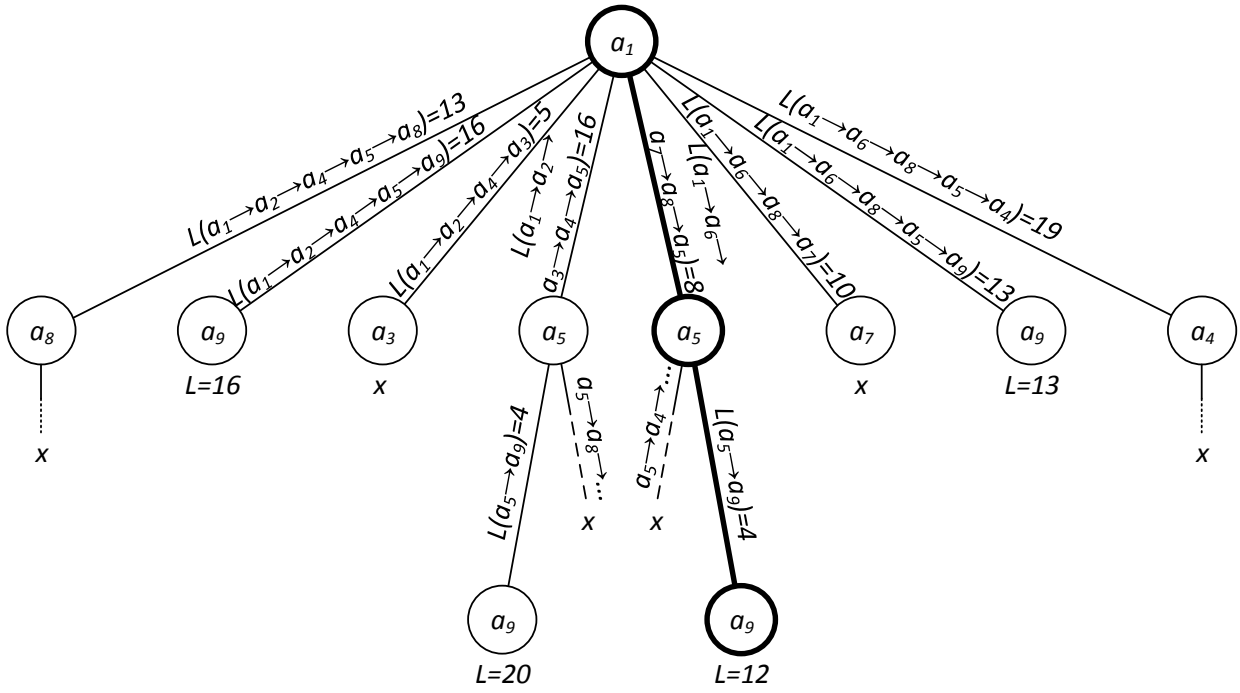

Рис. 4. Дерево возможных путей при ограничении на глубину перебора $S=4$: найденный путь $a_1 \underbrace{\rightarrow a_6 \rightarrow a_7 \rightarrow a_8 \rightarrow a_5}_{1\ ярус} \underbrace{\rightarrow a_9}_{2\ ярус}$ имеет длину $L=12$ и является оптимальным

При анализе комбинаторного дерева фактически производится его разбиение на $n=\left\lceil \frac{V}{S} \right\rceil$ ярусов, причем время построения результирующего пути линейно зависит от *n*. На каждом ярусе производится обход в глубину поддерева высотой *S*, число путей в

котором ограничено сверху значением $\prod_{i=V_0}^{V_0+S} d_i$, где V_0 – высота корневого элемента рассматриваемого поддерева в объемлющем дереве, d_i – средняя арность узлов объемлющего дерева в ярусе с высотой *i*. Так в рассматриваемой задаче $d_i \le i$, число путей в объемлющем дереве $(S=V)$ составляет $V\cdot(V-1)\cdot...\cdot 1=V!$, а в одном из поддеревьев с корневой вершиной на уровне V_0 – $V_0\cdot(V_0-1)\cdot...\cdot(V_0-S)=\frac{V_0!}{(V_0-S-1)!}\le V_0^S\le V^S$. С учетом того, что $V\simeq N$ (высота определяется диаметром графа *G*, а он в свою очередь не превышает числа его вершин *N*), соответствующая асимптотическая временная сложность алгоритма в целом составляет не более $O\left(\left\lceil\frac{V}{S}\right\rceil V^S\right)\simeq O\left(V^{S+1}\right)\simeq O\left(N^{S+1}\right)$. С линейным ростом глубины S происходит полиномиальный рост асимптотической временной сложности ($O\left(N^2\right)$ для $S=1$, $O\left(N^3\right)$ для $S=2$ и т.д.). При построении участка пути используется не более $O(S)$ ячеек памяти на его хранение, а при построении общего пути – не более $O(N)$ ячеек, что определяет линейную асимптотическую емкостную сложность алгоритма: $O(N+S)\simeq O(N)$. Процесс поиска очередного участка пути допускает распараллеливание, в то время как процесс построения пути из участков является сугубо последовательным.

С целью апробации предложенного подхода на практике была разработана программная реализация [8], с использованием которой организован соответствующий вычислительный эксперимент, в ходе которого была сформирована выборка $\Lambda=\{G_1,G_2,...,G_K\}$ из *K* графов с псевдослучайной структурой и заданными параметрами (число вершин *N* и плотность графа *d*, длина дуги – псевдослучайное число с равномерным распределением на отрезке $[0;1]$), для которых осуществлялся поиск путей между случайно выбранной парой вершин. Следуя работе [6], производилось определение следующих средневыборочных параметров: средняя длина пути $\overline{L}$; среднее отклонение $\Delta\overline{L}$ длины пути от минимальной $L_{\min}(G_i)$, получаемой с использованием алгоритма Дейкстры; вероятность нахождения решения $\overline{p}$ (отсутствия попадания в тупик); вероятность нахождения оптимального решения $\overline{p}_{opt}$ и среднее время получения решения *t*, для оценки которого использован компьютер с процессором Intel Core i7 4770 @ 3,4 ГГц (Haswell). Полученные результаты приведены в табл. 1 и 2.

Таблица 1. Результаты вычислительного эксперимента при $N=10$, $d=0{,}5$, $K=1\,000$

Метод		$\overline{L}$	$\Delta\overline{L}$	$\overline{p}$	$\overline{p}_{opt}$	t
Дейкстры		0,4804	0	1	1	3,7 мкс
Перебор с ограничением глубины	$S=1$	0,6418	0,1621	0,997	0,533	2,6 мкс
	$S=2$	0,5770	0,0766	1	0,737	3,8 мкс
	$S=3$	0,4920	0,0117	1	0,915	8,1 мкс
	$S=4$	0,4822	0,0018	1	0,982	18,3 мкс
	$S=5$	0,4806	0,0002	1	0,997	41,4 мкс
	$S=6$	0,4804	0,0001	1	0,999	60,7 мкс
	$S=7$	0,4804	0	1	1	73,9 мкс

Таблица 2. Результаты вычислительного эксперимента при $N=100$, $d=0{,}5$, $K=1\,000$

Метод		$\overline{L}$	$\Delta\overline{L}$	$\overline{p}$	$\overline{p}_{opt}$	t
Дейкстры		0,0967	0	1	1	0,16 мс
Перебор с ограничением глубины	$S=1$	0,5169	0,4202	1	0,080	0,01 мс
	$S=2$	0,2025	0,1058	1	0,198	0,22 мс
	$S=3$	0,1276	0,0309	1	0,414	11,3 мс
	$S=4$	0,1073	0,0106	1	0,628	553 мс

Полученные результаты позволяют сделать ряд выводов. На задачах малой размерности при достаточно большой глубине S метод обеспечивает получение оптимальных или очень близких к ним по качеству решений при умеренных затратах вычислительного времени. На задачах большой размерности рост глубины S приводит к существенному увеличению затрат вычислительного времени и не позволяет получения решений высокого качества за приемлемое время, в отличие от других известных эвристических методов.

Библиографический список

1. http://ru.wikipedia.org/wiki/Минимальное_остовное_дерево
2. Ватутин Э.И., Зотов И.В. Метод формирования субоптимальных разбиений параллельных управляющих алгоритмов // Параллельные вычисления и задачи управления (PACO'04). М.: ИПУ РАН, 2004. С. 884–917.
3. Комбинаторно-логические задачи синтеза разбиений параллельных алгоритмов логического управления при проектировании логических мультиконтроллеров / Э.И. Ватутин, И.В. Зотов, В.С. Титов и др. Курск: изд-во КурскГТУ, 2010. 200 с.
4. Ватутин Э.И. Проектирование логических мультиконтроллеров. Синтез разбиений параллельных граф-схем алгоритмов. Saarbrücken: Lambert Academic Publishing, 2011 г. 292 с.
5. Ватутин Э.И., Дремов Е.Н., Мартынов И.А., Титов В.С. Метод взвешенного случайного перебора для решения задач дискретной комбинаторной оптимизации // Известия ВолГТУ. Серия: Электроника, измерительная техника, радиотехника и связь. № 10 (137). Вып. 9. 2014. с. 59–64.
6. Ватутин Э.И., Титов В.С. Анализ результатов применения алгоритма муравьиной колонии в задаче поиска пути в графе при наличии ограничений // Известия Южного федерального университета. Технические науки. 2014. № 12 (161). С. 111–120.
7. Dijkstra E. W. A note on two problems in connexion with graphs // Numerische Mathematik. V. 1 (1959), PP. 269–271.
8. Ватутин Э.И., Валяев С.Ю., Дремов Е.Н., Мартынов И.А., Титов В.С. Расчетный модуль для тестирования комбинаторных оптимизационных алгоритмов в задаче поиска кратчайшего пути в графе с использованием добровольных распределенных вычислений // Свидетельство о государственной регистрации программы для ЭВМ № 2014619797 от 22.09.14.