Романовский Иосиф Владимирович,
Ейбоженко Дмитрий Анатольевич

УДК 519.176

МОДИФИКАЦИИ МЕТОДА ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ В ЗАДАЧАХ ШТЕЙНЕРА НА ОРИЕНТИРОВАННЫХ ГРАФАХ

Аннотация

Задача Штейнера на ориентированных графах – наиболее общая в семействе задач Штейнера. Известно, что она является NP-полной. Существует алгоритм для точного решения задачи, основанный на динамическом программировании, пригодный для задач маленького размера. В нашей статье приводятся специальные типы задач, на которых с помощью модификации названного метода точное решение может быть получено за полиномиальное время. Кроме того, представлен метод, предназначенный для приближенного решения произвольных задач Штейнера на ориентированных графах.

Ключевые слова: задача Штейнера, ориентированный граф, динамическое программирование, функция Беллмана, приближенные алгоритмы.

ВВЕДЕНИЕ

Эта статья связана со статьей И.В. Романовского [1], посвященной задаче Штейнера. В ней описывался алгоритм точного решения «графовой» версии задачи, причем варианту с ориентированными графами. Алгоритм в [1] основывался на методе динамического программирования и уравнении Беллмана. Как было отмечено, алгоритм динамического программирования имеет экспоненциальную сложность и малоприменим для задач больших размерностей. Здесь мы приведем некоторые специальные виды задач, а также модификации самого метода, которые позволяют применять его для задач с большим числом терминалов. Но прежде напомним некоторые факты, связанные с задачей Штейнера, а также приведем примеры использования ее на практике.

Якоб Штейнер (1796–1863) – известный немецкий геометр. Его научные интересы были связаны в первую очередь с геометрией. Штейнер достиг в этой области значительных успехов и заслуженно считается отцом-основателем современной синтетической геометрии, которая, принимая за базу некоторый набор аксиом, выводит из них другие утверждения с помощью последовательности логических выводов (всем известная геометрия Евклида является примером синтетической геометрии). В процессе своих изысканий он сформулировал задачу, которая впоследствии стала называться геометрической задачей Штейнера[1].

На данный момент распространены несколько видов задачи Штейнера:

1. ***Евклидова (геометрическая).*** Задан набор точек на евклидовой плоскости, требуется найти кратчайшую сеть, которая соединяет все точки. Сеть может иметь дополнительные точки пересечения, называемые точками Штейнера.

2. ***Прямоугольная*** **[8].** Это частный случай евклидовой задачи Штейнера. В ней для определения расстояний между точками используется прямоугольная метрика $L_1 : \rho((x_1, y_1),(x_2, y_2)) = (|x_1 - x_2| + |y_1 - y_2|)$. Ее еще называют *манхеттенской*[2].

[1] Существует, кстати, и другая задача Штейнера. Вот она: найти максимум функции $x^{1/x}$. Решением ее является $x = e$.

[2] Основная часть острова Манхеттен в Нью-Йорке расчерчена прямоугольной сеткой улиц и авеню. В Санкт-Петербурге подобным образом расчерчен Васильевский остров, так что задачу можно было бы называть и василеостровской.

3. ***В сетях (на неориентированных графах).*** Дан граф $G(M, N)$ подмножество $T \subseteq M$ вершин-*терминалов* и положительные длины дуг (можно говорить, что длина есть функция от дуги $d: N \to \mathbb{R}_+$). Необходимо найти сеть наименьшей длины в графе, соединяющую все терминалы из T.

4. ***На ориентированных графах.*** Пусть $G(M, N)$ – ориентированный граф с заданной на дугах функцией $d: N \to \mathbb{R}_+$. В M выделена начальная вершина b и множество терминальных вершин E. Ищется дерево минимальной длины с корнем в заданной вершине b, содержащее пути от b до любого терминала из E.

ПРИМЕНЕНИЕ

Различные виды задачи Штейнера широко применяются в настоящее время во многих областях. Расскажем лишь про некоторые из них.

VLSI-ПРОЕКТИРОВАНИЕ

Аббревиатура *VLSI* расшифровывается как *Very Large Scale Integration*. Это процесс создания сверхбольших интегральных схем (микросхем) путем комбинирования миллионов (а сейчас уже и миллиардов) транзисторов. Всем известные центральные процессоры (CPU) и графические платы (GPU) являются примерами VLSI-устройств.

Процесс создания таких микросхем трудоемок и состоит из многих этапов, среди которых можно так выделить ключевые:

- Проектирование логической схемы, которая решала бы задачи, определенные требованиями заказчика.[1]
- Физическое размещение транзисторов на плате наиболее эффективным образом.
- Проведение соединений между контактами схемы.

Именно на последнем этапе и используется задача Штейнера, а точнее, ее манхеттенская версия. Дело в том, что проектировщики предпочитают соединения, сопрягающиеся или поворачивающиеся под углом 90°. В последнее время все больше внимания уделяется нестандартным архитектурам: архитектуре, где соединения проводятся под углами в 120°[2], и архитектуре, в которой к углам в 90° добавляются также всевозможные промежуточные углы в 45°.

ВОССТАНОВЛЕНИЕ ФИЛОГЕНЕТИЧЕСКИХ ДЕРЕВЬЕВ

Филогенетическое дерево – это дерево, отражающее эволюционные взаимосвязи между различными видами или другими сущностями, имеющими общего предка [9]. Один из способов построения таких деревьев – техника, использующая данные о белковых последовательностях. В этом способе для конкретного набора биологических видов, каждому из которых сопоставлены одинаковые протеиновые последовательности, строится дерево. Символами в протеиновой последовательности обычно являются аминокислоты, каждая из упорядоченной тройки нуклеотидов. Природа использует 4 нуклеотида: A, C, G, U, так что в принципе может быть 64 таких тройки, но существует только 20 аминокислот. То есть каждая последовательность – это строка из символов алфавита размера 20. На множестве таких последовательностей задана метрика, называемая *расстоянием Хэмминга*: при условии, что число элементов в обеих строках равно, оно определяется как число отличающихся символов в них. В графе $G(M, N)$ множество M – множество нуклеотидных последовательностей, а N – $M \times M$ (полный граф), при этом длины дуг графа задаются расстояниями Хэмминга между соответствующими цепочками.

[1] Сейчас эта работа выполняется с помощью специальных языков программирования, таких как *Verilog*, похожий по синтаксису на обычный Си.

[2] Казалось бы, такая архитектура еще более ограничена, чем стандартная, зачем же ее использовать? Возможно, дело в некоторых свойствах оптимального решения геометрической задачи Штейнера: степень каждой точки Штейнера не должна превышать 3, и соединяться пути должны ровно под углами 120°.

НАПОМИНАНИЕ

Прежде чем перейти к основной части, напомним суть алгоритма динамического программирования из [1] для решения задачи Штейнера на ориентированных графах. Задача решается одновременно для всех возможных начальных вершин и всех возможных подмножеств терминального множества. Состояние процесса характеризуется текущей начальной точкой и подмножеством терминалов, и в каждом состоянии можно перейти по любой исходящей дуге, либо выбрать любое разбиение подмножества терминалов на два подмножества и решать задачи для них отдельно. Решением задачи для состояния (i, E_ρ) является $(i, E_\rho) = \min\{v_{cont}(i, E_\rho), v_{part}(i, E_\rho)\}$, где $v_{cont}(i, E_\rho) = \min\{l_j + v(end\, j, E_\rho) \mid beg\, j = i\}$ – наименьшие затраты при переходе по выбираемой исходящей дуге j,

$v_{part}(i, E_\rho) = \min\{v(i, A) + v(i, E_\rho \setminus A) \mid A \subset E_\rho\}$ – наименьшие затраты при выбираемом разбиении множества терминалов.

ЗАДАЧА НА КОНЦЕНТРИЧЕСКИХ ОКРУЖНОСТЯХ

Пусть на евклидовой плоскости имеется множество концентрических окружностей, на каждой из которых также задано произвольное множество точек. Рассмотрим граф, множество вершин которого состоит из выбранных точек, а также центральной точки системы окружностей. Любые две точки соседних (и только соседних) окружностей соединены попарно дугами, ведущими от внутренней окружности к внешней. Длина дуги определяется евклидовым расстоянием между ее вершинами на плоскости. Начальная точка задачи Штейнера — центральная точка системы, множество терминалов — некоторое подмножество множества вершин внешней окружности (см. рис. 1)

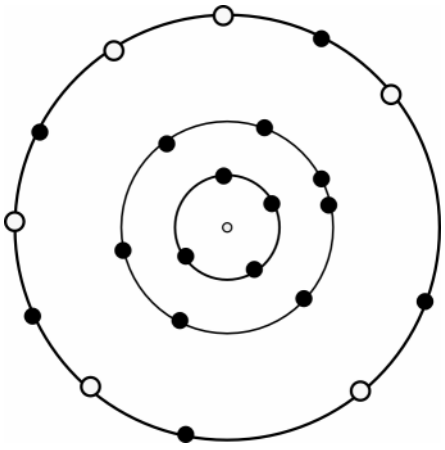

Рис. 1. Пример задачи

Покажем, что для такой задачи можно сделать алгоритм динамического программирования более быстрым, ограничив множество рассматриваемых подмножеств терминалов. А именно, будем рассматривать только такие подмножества, все элементы которых следуют один за другим на окружности. То есть не существует терминала, не вошедшего в множество, разделяющего два терминала из множества. В то же время, нетерминальных вершин между терминалами из такого множества может быть любое число (см. рис. 2).

В множестве из k терминалов содержится лишь $k(k-1) + 2 < k^2$ подмножеств, состоящих из вершин, идущих подряд. Это k одноэлементных, k двухэлементных и т. д., $k(k-1)$-элементных, а также пустое подмножество и множество всех терминалов. Это значит, что алгоритм динамического программирования с таким ограничением будет иметь не экспоненциальную, а полиномиальную сложность.

Однако нам необходимо удостовериться, что алгоритм динамического программирования с такими ограничениями для этой задачи по-прежнему будет выдавать точное решение. Для этого достаточно

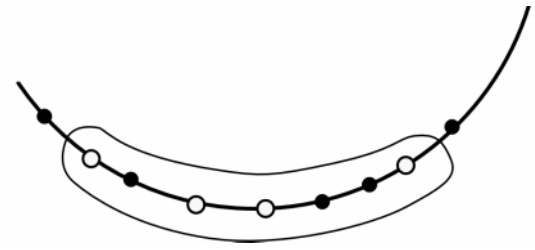

Рис. 2. Множество, удовлетворяющее условию

показать, что в любом состоянии (i, S_T) процесса (i – текущая начальная вершина, S_T – некоторое подмножество терминалов) для любого разбиения $\{S_A, S_{T\setminus A}\}$, не удовлетворяющего ограничениям, существует разбиение, удовлетворяющее ограничениям, которое дает не худший результат.

То, что разбиение не удовлетворяет ограничениям, означает, что существуют вершины $a, b \in S_A$, расположенные через одну: $c, d \in S_{T\setminus A}$. Рассмотрим решение, полученное из такого разбиения (см. рис. 3).

Существуют две соседние окружности, между которыми пути $i \to c$ и $i \to b$ пересекаются. Пусть пересечение происходит на дугах (x_1, y_2) и (x_2, y_1). Тогда, если выбросить из путей эти дуги и вставить вместо них дуги (x_1, y_1) и (x_2, y_2), получим некоторое решение для разбиения, в котором $a, b \in S_A$, $c, d \in S_{T\setminus A}$. Очевидно, что $|(x_1, y_1)| + |(x_2, y_2)| \le |(x_1, y_2)| + |(x_2, y_1)|$, а значит оптимальное решение для такого разбиения не больше, чем для исходного. Меняя таким образом местами все «перекрещивающиеся» вершины, придем к разбиению, удовлетворяющему наложенным ограничениям, и при этом получаемое с помощью данного разбиения решение будет не больше, чем для разбиения $\{S_A, S_{T\setminus A}\}$.

Как мы видим, в некоторых случаях алгоритм динамического программирования может применяться для получения точного решения и в задачах больших размеров. Однако нам хотелось бы использовать этот метод для решения задач на произвольных графах. Конечно, для общего случая нам не удастся создать алгоритм, основанный на динамическом программировании, полиномиальным, как это удалось выше, не ухудшив его точность (иначе мы бы доказали, что $P = NP$).

Ниже будет предложен основанный на динамическом программировании эвристический полиномиальный приближенный алгоритм, дающий, однако, достаточно точное решение на экспериментальных данных.

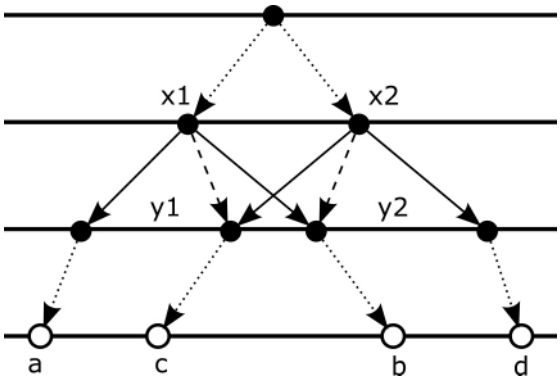

Рис. 3. Разбиение, не удовлетворяющее ограничениям

k-КЛАСТЕРНЫЙ МЕТОД

ОПИСАНИЕ МЕТОДА

Общая идея алгоритма состоит в том, чтобы разбить граф на некоторое количество подграфов, в каждый из которых попадает не слишком большое число терминалов, и на каждом построить дерево Штейнера для содержащихся в них терминалов точным методом. Затем, преобразовав исходный граф с учетом найденных частичных поддеревьев, с помощью точного метода найти решение.

Прежде всего построим на графе промежуточное дерево Штейнера, которое назовем *опорным*. Его ветвление мы используем, чтобы построить искомые подграфы, которые также будем называть *кластерами* (см. рис. 4).

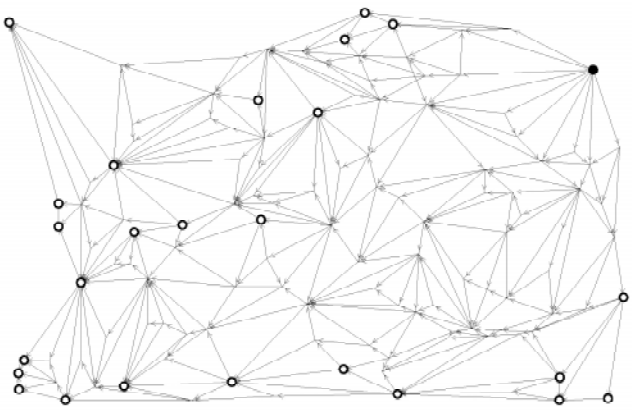

Рис. 4. Пример задачи. Постановка

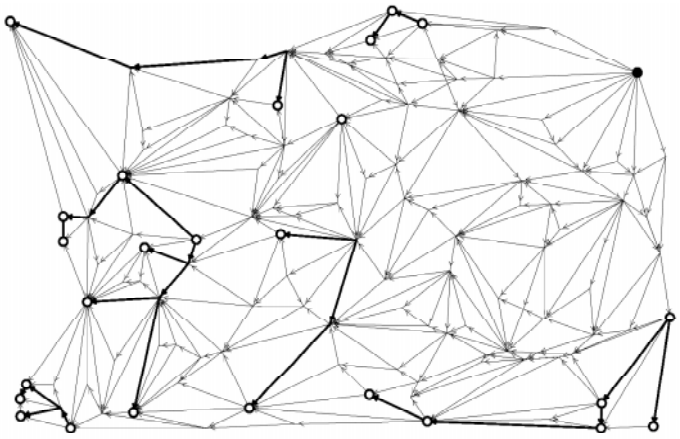

Рис. 5. Пример задачи. Частичные деревья Штейнера

Будем наращивать дерево постепенно. Вначале оно состоит из одной вершины b.

На каждом шаге найдем в множестве терминалов ближайший к уже построенной части дерева и добавим кратчайший путь до этого терминала к дереву. В результате, рассмотрев все терминалы, получим некоторое дерево Штейнера.[1] Далее будем разбивать граф на кластеры. Отметим, что параметр k, вынесенный нами в заглавие метода, определяет максимальное число терминалов в некотором графе, к которому мы согласны применять алгоритм динамического программирования.

Рассмотрим последовательно все вершины разветвления в опорном дереве, начиная с самой дальней от b. Найдем число терминалов, которое содержится в поддереве, ограниченном данной вершиной. Если их больше чем $|E|/k$, выделим подграф исходного графа на множестве вершин данного поддерева в отдельный кластер. Иначе найдем на пути от b до данной вершины следующую по удаленности вершину разветвления и таким же образом рассмотрим её.

Условие, что число терминалов в кластере должно быть больше чем $|E|/k$, введено, для того чтобы общее число кластеров было не более k.

Выделив кластеры, определим на каждом из них собственную задачу Штейнера: множество терминалов задается терминалами исходной задачи, попавшими в кластер, начальная вершина – вершина разветвления, на основе которой был построен кластер. Если число терминалов в кластере не превышает k, решаем задачу Штейнера на нем методом динамического программирования, иначе разбиваем его в свою очередь на более мелкие кластеры.

После того как все деревья Штейнера на кластерах найдены (см. рис. 5), преобразуем исходный граф с их учетом. Напомним, что общее число частичных деревьев Штейнера не превышает k. Просматривая все кластеры, выбросим из графа все дуги, ведущие извне кластера в любую вершину кластера, кроме начальной. Всем дугам, вошедшим в построенные деревья Штейнера, присвоим нулевую длину. Множеством терминалов в преобразованном графе будем считать множество начальных вершин деревьев Штейнера. Таким образом, число терминалов в построенной задаче также не превышает k, и мы можем вновь применить метод динамического программирования.

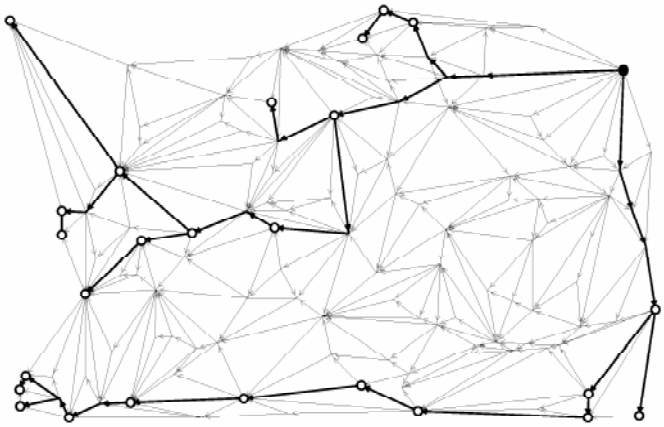

Рис. 6. Пример задачи. Решение

[1] Этот алгоритм в 1980 году придумали два японских математика, Такахаши и Мацуяма, как эвристический метод построения приближенного минимального дерева Штейнера.

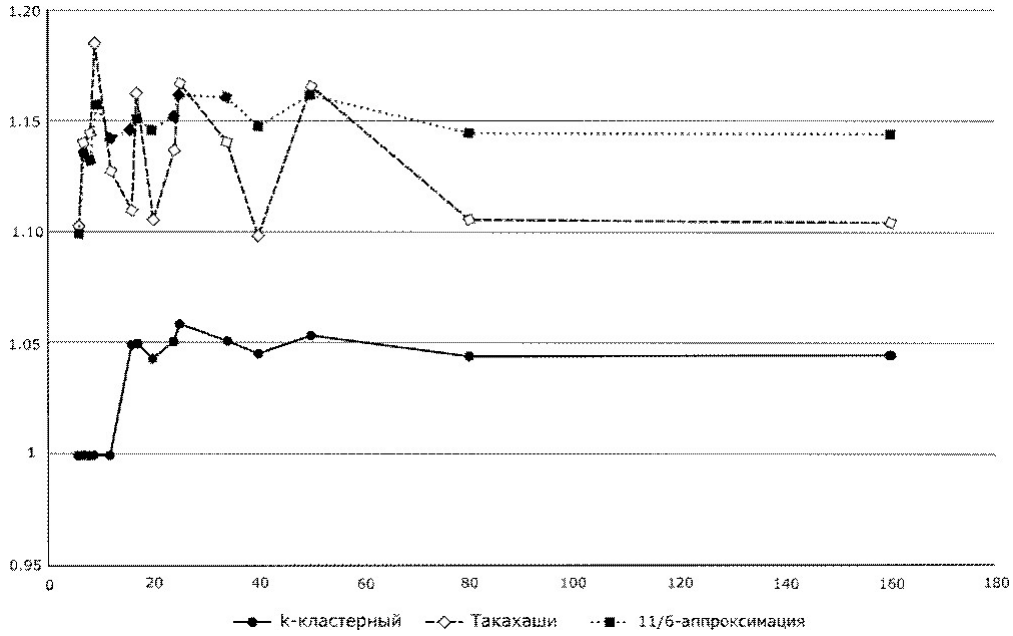

Рис. 7

В результате мы получим некоторое дерево Штейнера. Применим к нему метод локальных улучшений.

Будем рассматривать все вершины дерева, являющиеся либо ветвящимися, либо терминалами. Для каждой такой вершины i сделаем следующее: найдем в каждой ветви, начинающейся с нее, ближайшую подобную вершину j (и выбросим из дерева путь между ними). Таким образом, дерево разобьется на два, одно из которых (с корнем b) содержит i, а другое имеет корень j. Найдем кратчайший путь от первого дерева до вершины j (его длина заведомо не больше, чем длина выброшенного пути) и добавим его к дереву. В результате таких улучшений мы получим дерево Штейнера заведомо не большей длины, нежели промежуточное. Оно и является результатом работы алгоритма (см. рис. 6).

ОПЫТЫ

Прежде всего отметим, что сложность вышеприведенного алгоритма полиномиальная и составляет, как доказано,

$O(2^k t n \log m)$, где t – число терминалов, m – вершин, n – дуг. Как следует из оценки сложности, параметр k влияет только на коэффициент, однако увеличение k на единицу увеличивает его вдвое. В ходе экспериментов было испробовано несколько различных значений, и наиболее сбалансированными, с точки зрения отношения времени решения к точности аппроксимации, выглядят значения в диапазоне от 11 до 13. Стоит отметить, что эти результаты получены для персонального компьютера, на многопроцессорных системах наиболее благоприятный диапазон может быть сдвинут в сторону увеличения, что приведет к улучшению точности. Отметим к тому же, что структура самого алгоритма располагает к его распараллеливанию.

Результаты решения ста задач с разным числом терминалов приведены для $k = 11$. По оси абсцисс отложено количество терминалов в решенных задачах, по оси ординат – среднее для данного числа терминалов отклонение от известного точного решения (см. рис. 7).

Алгоритм кластерной оптимизации сравнивался со следующими алгоритмами:

- Модификация для ориентированных графов алгоритма Такахаши (1980), который использовался для построения опорного дерева в алгоритме кластерной оптимизации [4].

• Алгоритм 11/6-аппроксимации для неориентированных графов, представленный А. Зеликовским (1992), основанный на «жадном» добавлении к дереву звезд, созданных из троек терминалов, соединенных вершиной не из множества терминалов [5, 6].

Литература

1. Романовский И.В. Задача штейнера на графах и динамическое программирование // Компьютерные инструменты в образовании, 2004. № 2. С. 80–86.

2. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.

3. *Романовский И. В.* Дискретный анализ. СПб.: Невский Диалект, 2003.

4. *Takahashi H., Matsuyama A.* An Approximate Solution for the Steiner Problem In Graphs // Math. Japonica, No. 6: 573–577, 1980.

5. *Zelikovsky A.* The last achievements in Steiner tree approximations. // CSJM v.1, n.1 (1), 1993.

6. *Zelikovsky A.* 11/6-approximation algorithm fir the Steiner problem on graphs. // Fourth Czechoslovakian Symposium on Combinatorics, Graphs and Complexity. Ann. Disc. Math.: 351–354, 1992.

7. *Zachariasen M., Winter P.* Concatenation-Based Greedy Heuristics for the Euclidean Steiner Tree Problem. // Technical Report 97/20, DIKU, Department of Computer Science, University of Copenhagen, 1997.

8. *Hanan M.* On Steiner's problem with rectilinear distance. // SIAM Journal of Applied Mathematics, 30(1): 104–114, 1966.

9. *Chin L. L., Chuan Y. T., Lee R.* The Full Steiner Tree Problem in Phylogeny // Computing and Combinatorics, 2002.

Abstract

Steiner tree problem on oriented graphs is the most general in the family of Steiner tree problems. It is known to be *NP*-complete. There exists an algorithm to find an exact solution, based on dynamic programming, but applicable only for problems of modest size. Here special types of problems are presented, for which this algorithm is modified to find exact solution at polynomial time. Also approximate method, based on this algorithm for solving arbitrary problems is produced.

Keywords: Steiner problem, oriented graph, dynamic programming, Bellman function, approximation algorithms.

Романовский Иосиф Владимирович,
доктор физико-математических наук, профессор, кафедра исследования операций математико-механического факультета СПбГУ,
josephromanovsky@gmail.com,

Ейбоженко Дмитрий Анатольевич,
аспирант математико-механического факультета СПбГУ,
dmitry@eibozhenko.ru