электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 519.173

Разработка программного средства для анализа подходов к выбору маршрута прокладки кабеля сети кольцевой архитектуры

***Парначёв А.С.**, студент*
кафедра «Системы обработки информации и управления»
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана

Научный руководитель: Постников В.М..,к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
chernen@bmstu.ru

В сети кольцевой архитектуры, например, такой как FDDI(*Fiber Distributed Data Interface — Волоконно-оптический интерфейс передачи данных*), кабель проходит через каждый узел только один раз, поэтому выбор маршрута прокладки кабеля может быть сведен к решению задачи коммивояжёра.

Целью работы является создание программного средства для анализа методов выбора маршрута прокладки кабеля сети кольцевой архитектуры и поиск наиболее эффективного метода.

Задача коммивояжёра (Travelling salesman problem — TSP) — одна из самых известных задач комбинаторной оптимизации, заключающаяся в отыскании самого выгодного маршрута, проходящего через указанные города по одному разу с последующим возвратом в исходный город. Поскольку коммивояжер в каждом из городов встает перед выбором следующего города из тех, что он еще не посетил, всего существует (n — 1)! маршрутов. Таким образом, размер пространства поиска зависит экспоненциально от количества городов. Поэтому оптимизационная постановка задачи относится к классу NP-трудных задач, и уже при относительно небольшом числе городов (66 и более) не может быть решена методом перебора вариантов современными компьютерными средствами.

Для решения задачи коммивояжера используют следующие методы:

Точные методы:

- Метод полного перебора
- Метод Литтла
- Метод ветвей и границ

Эвристические методы:

- Метод «Иди в ближний»
- Метод Прима-Эйлера

Для практического использования указанных методов необходима разработка программного средства, в основе которого лежит язык программирования. В настоящее время широко используют следующие языки программирования:

- B1 - Object Pascal
- B2 - C++
- B3 - Java
- B4 - Python
- B5 - C#

Для сравнительного анализа языков программирования был выбран следующий набор локальных критериев:

- К1 - Ресурсонезависимость
- К2 - Скорость компиляции
- К3 - Кроссплатформенность
- К4 - Производительность
- К5 - Удобные инструменты визуального программирования

Для выбора наиболее приемлимого языка с учетом предложенного набора локальных критериев целесообразно использовать метод анализа иерархии. Сравнительный анализ языков программирования с использованием метода анализа иерархии приведен в таблицах 1-6.

Проведем попарное сравнение локальных критериев.

Таблица 1

Попарное сравнение локальных критериев

	К1	К2	К3	К4	К5	C_i	α_i
К1	1	2	2	4	5	2,402	0,388
К2	0,5	1	2	4	5	1,821	0,294
К3	0,5	0,5	1	2	2	1	0,161
К4	0,25	0,25	0,5	1	2	0,574	0,093
К5	0,2	0,2	0,5	0,5	1	0,398	0,064

Определяем собственные значения векторов локальных критериев C_i:

$$C_i = (k_{i1}k_{i2} \dots k_{in})^{1/n}$$

Затем вычисляем коэффициенты важности локальных критериев:

$$\alpha_i = \frac{C_i}{\sum_{i=1}^{n} C_i}$$

Рассчитаем максимальное собственное значение матрицы парных сравнений локальных критериев $\boldsymbol{\lambda}_{max}$

Получаем, что

$$\boldsymbol{\lambda}_{max} = \sum_{j=1}^{5}\sum_{i=1}^{5} k_{ij}\alpha_j = 5{,}1074$$

Находим отношение согласованности полученной матрицы парных сравнений локальных критериев:

$$O_c = \frac{(\boldsymbol{\lambda}_{max} - n_F)}{(n_F - 1)R} = 0{,}024$$

где R - значение индекса согласованности для несимметричных матриц.

Поскольку $O_c = 0{,}024 < 0.1$, матрица парных сравнений локальных критериев является полностью согласованной и коэффициенты важности локальных критериев можно использовать для проведения дальнейших расчетов.

После этого сравним все варианты по каждому из критериев.

Таблица 2

Попарное сравнение вариантов по критерию К1

	B1	B2	B3	B5	B4	C_i	β_{1j}
B1	1	2	4	5	5	2,885	0,448608
B2	0,5	1	2	4	4	1,741	0,27072
B3	0,25	0,5	1	1	2	0,758	0,117867
B5	0,2	0,25	1	1	2	0,631	0,098118
B4	0,2	0,25	0,5	0,5	1	0,416	0,064687

Таблица 3

Попарное сравнение вариантов по критерию К2

	B4	B1	B2	B3	B5	C_i	β_{2j}
B4	1	1	4	5	5	2,511886	0,391192
B1	1	1	2	4	5	2,091279	0,325688
B2	0,25	0,5	1	2	2	0,870551	0,135576
B3	0,2	0,25	0,5	1	2	0,54928	0,085543
B5	0,2	0,2	0,5	0,5	1	0,398107	0,062

Таблица 4

Попарное сравнение вариантов по критерию К3

	B3	B2	B4	B5	B1	C_i	β_{3j}
B3	1	1	1	6	6	2,047673	0,299712
B2	1	1	1	6	6	2,047673	0,299712
B4	1	1	1	6	6	2,047673	0,299712
B5	0,166667	0,166667	0,166667	1	2	0,392026	0,05738
B1	0,166667	0,166667	0,166667	0,5	1	0,2971	0,043486

Таблица 5

Попарное сравнение вариантов по критерию К4

	B2	B1	B3	B5	B4	C_i	β_{4j}
B2	1	1	2	5	5	2,186724	0,347272
B1	1	1	2	5	5	2,186724	0,347272
B3	0,5	0,5	1	2	2	1	0,158809
B5	0,2	0,2	0,5	1	2	0,525306	0,083423
B4	0,2	0,2	0,5	0,5	1	0,398107	0,063223

Таблица 6

Попарное сравнение вариантов по критерию К5

	B5	B3	B1	B2	B4	C_i	β_{5j}
B5	1	2	4	5	5	2,8854	0,427252
B3	0,5	1	4	5	5	2,186724	0,323796
B1	0,25	0,25	1	2	2	0,757858	0,112219
B2	0,2	0,2	0,5	1	2	0,525306	0,077784
B4	0,2	0,2	0,5	0,5	1	0,398107	0,058949

Рассчитаем максимальное собственное значение матрицы парных сравнений вариантов по каждому критерию и отношение согласованности

$$\boldsymbol{\lambda}_{max\,1} = 5{,}0832;\ O_{c1} = 0{,}019 < 0{,}1$$

$$\boldsymbol{\lambda}_{max\,2} = 5{,}0813;\ O_{c2} = 0{,}018 < 0{,}1$$

$$\boldsymbol{\lambda}_{max\,3} = 5{,}0292;\ O_{c3} = 0{,}007 < 0{,}1$$

$$\boldsymbol{\lambda}_{max\,4} = 5{,}0416;\ O_{c4} = 0{,}009 < 0{,}1$$

$$\boldsymbol{\lambda}_{max\,5} = 5{,}1570;\ O_{c5} = 0{,}035 < 0{,}1$$

Из этого следует, что каждая матрица парных сравнений вариантов является согласованной и вычисленные коэффициенты важности β_{ij} по каждому из критериев являются достоверными.

Вычисление обобщённого коэффициента важности β_j для каждого варианта осуществляют по формуле:

$$\beta_j = \sum_{i=1}^{n} \alpha_i \beta_{ij}$$

После вычисления получаем следующие результаты:

$$\beta_1 = 0{,}3163$$

$$\beta_2 = 0{,}2304$$

$$\beta_3 = 0{,}1546$$

$$\beta_4 = 0{,}1980$$

$$\beta_5 = 0{,}1006$$

Выбираем наилучший вариант.

$$\beta_1 = \max_j \beta_j = 0{,}3163$$

Получаем, что в результате проведенных расчетов наилучшим языком программирования для решения задачи коммивояжёра является Object Pascal.

Программная реализация

Программная реализация алгоритмов была осуществлена на языке Object Pascal в среде разработки Borland Delphi 7. Object Pascal - высокоуровневый, строго типизированный язык общего назначения. Он характеризуется простотой чтения кода, быстрой компиляцией, и использует подключение модулей для модульного

программирования. Разработанная программа в среде Borland Delphi 7 может быть запущена на любом компьютере с ОС WIndows XP,7,8 без установки дополнительных приложений или библиотек. Программа представлена на рис. 1.

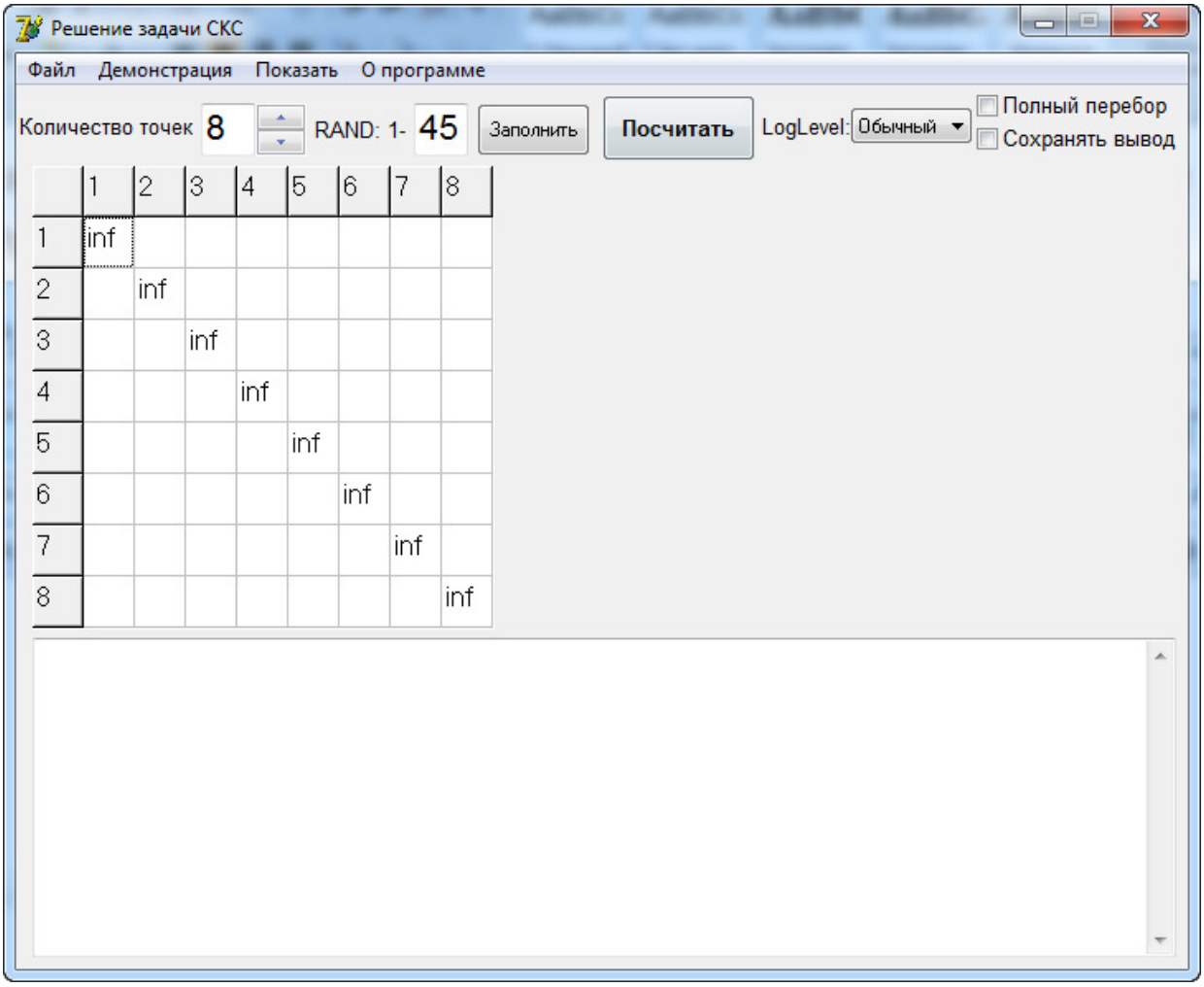


Рис. 1. Интерфейс программы SolveRoute

Описание методов

Метод полного перебора

1. Определяем множество узлов.
2. Определяем расстояния между парами узлов, то есть формируем матрицу расстояний между узлами.
3. Генерируем путь и вычисляем его длину.
4. Если значение длины пути:

a. Меньше текущего минимального, результирующий список очищается и обновляется минимальное значение. Текущий путь добавляем к результирующему списку.
b. Равно текущему минимальному, то добавляем путь к результирующему списку.

5. Повторяем шаги 3-4, пока не закончим расчет вариантов путей.

Метод "Иди в ближний"

1. Определяем множество узлов.
2. Определяем расстояния между парами узлов, то есть формируем матрицу расстояний между узлами.
3. Формируем очередь расстояний, в которой расстояния между узлами не уменьшаются.
4. Последовательно просматриваем очередь и добавляем в результирующий путь новые вершины и связи. Ни колец, ни петель в последовательности быть не должно, при этом к каждому узлу может быть подключено не более 2 узлов.
5. Завершаем просмотр очереди расстояний между узлами, когда все узлы включены в сформированный маршрут. Крайние вершины соединяются связью, образуя кольцо.

Метод "Прима-Эйлера"

1. Определяем множество узлов.
2. Определяем расстояния между парами узлов, то есть формируем матрицу расстояний между узлами.
3. С помощью алгоритма Прима строим остовое дерево, т. е., формируем очередь расстояний, в которой расстояния между узлами не уменьшаются, затем последовательно просматриваем очередь и добавляем в результирующий путь новые вершины и связи. Ни колец, ни петель в последовательности быть не должно, при этом к каждому узлу может быть подключено любое число узлов.
4. В полученным графе дублируем каждое ребро, получая Эйлеров граф(мультиграф).
5. В полученном мультиграфе строим маршрут Эйлера, соединяя конечные узлы в кольцо.

Метод Литтла

1. Определяем множество узлов.

2. Определяем расстояния между парами узлов, то есть формируем матрицу расстояний между узлами.

3. В каждой строке матрицы расстояний находим минимальный элемент и вычитаем его из всех элементов строки. Повторяем эту процедуру и для столбцов, не содержащих нуля. Получаем матрицу расстояний, каждая строка и каждый столбец которой содержат хотя бы один нулевой элемент.

4. Для каждого нулевого элемента матрицы cij рассчитываем коэффициент $\Gamma_{i,j}$, который равен сумме наименьшего элемента i строки (исключая элемент $C_{i,j}=0$) и наименьшего элемента j столбца. Из всех коэффициентов $\Gamma_{i,j}$ выбираем максимальный $\Gamma_{k,l}=\max\{\Gamma_{i,j}\}$. В гамильтонов контур вносим соответствующую дугу (k,l).

5. Удаляем k-тую строку и столбец l. Меняем на бесконечность значение элемента $C_{l,k}$(поскольку дуга (k,l) включена в контур, то обратный путь из l в k недопустим).

6. Повторяем алгоритм с шага 3, пока порядок матрицы расстояний не станет равным двум.

7. Затем в текущий граф вносим две недостающие дуги и получаем гамильтонов контур.

Метод ветвей и границ

1. Определяем множество узлов.

2. Определяем расстояния между парами узлов, то есть формируем матрицу расстояний между узлами.

3. В каждой строке матрицы расстояний находим минимальный элемент и вычитаем его из всех элементов строки. Повторяем эту процедуру и для столбцов, не содержащих нуля. Получаем матрицу расстояний, каждая строка и каждый столбец которой содержат хотя бы один нулевой элемент.

4. Для каждого нулевого элемента матрицы cij рассчитываем коэффициент $\Gamma_{i,j}$, который равен сумме наименьшего элемента i строки (исключая элемент $C_{i,j}=0$) и наименьшего элемента j столбца. Из всех коэффициентов $\Gamma_{i,j}$ выбираем максимальный $\Gamma_{k,l}=\max\{\Gamma_{i,j}\}$.

5. Процесс ветвления. Если существует несколько коэффициентов с максимальным значением, то каждый из вариантов запоминается и обрабатывается следующими шагами алгоритма отдельно.

6. В гамильтонов контур вносим соответствующую дугу (k,l).

7. Удаляем k-тую строку и столбец l. Меняем на бесконечность значение элемента $C_{l,k}$(поскольку дуга (k,l) включена в контур, то обратный путь из l в k недопустим).

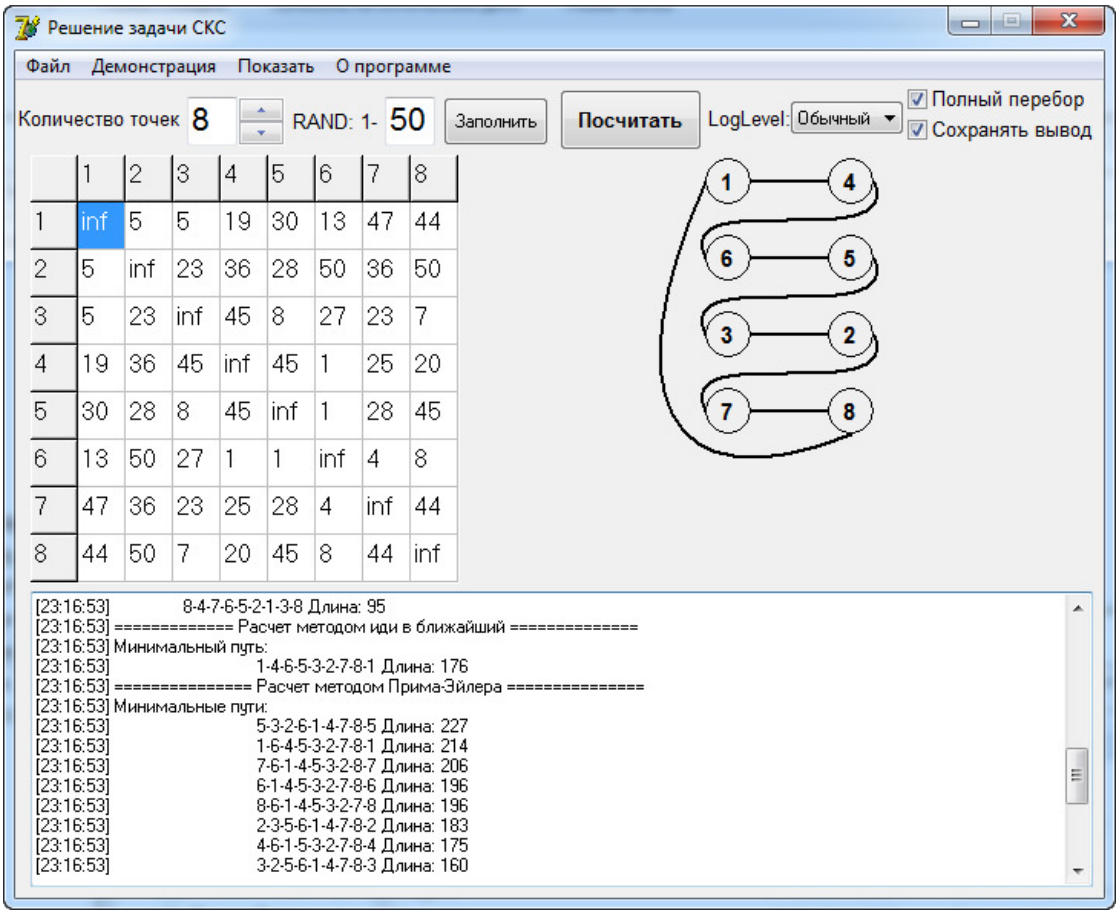
8. Повторяем алгоритм с шага 3, пока порядок матрицы расстояний не станет равным двум.

9. Затем в текущий граф вносим две недостающие дуги и получаем гамильтонов контур.

10. Из полученного списка путей выбираем путь с наименьшей длинной.

Пример выбора маршрута сети кольцевой архитектуры

Пример расчета для 8 узлов предоставлен на рисунке 2. На данном рисунке так же виден маршрут, вычисленный методом "Иди в ближний".



	1	2	3	4	5	6	7	8
1	inf	5	5	19	30	13	47	44
2	5	inf	23	36	28	50	36	50
3	5	23	inf	45	8	27	23	7
4	19	36	45	inf	45	1	25	20
5	30	28	8	45	inf	1	28	45
6	13	50	27	1	1	inf	4	8
7	47	36	23	25	28	4	inf	44
8	44	50	7	20	45	8	44	inf

Рис. 2. Пример решения задачи коммивояжёра программой SolveRoute

Отчёт программы

[23:16:52] _____ Запущены алгоритмы расчета _____
[23:16:52] Матрица:

[23:16:52]	0	1	2	3	4	5	6	7	8
[23:16:52]	1	-	5	5	19	30	13	47	44
[23:16:52]	2	5	-	23	36	28	50	36	50
[23:16:52]	3	5	23	-	45	8	27	23	7
[23:16:52]	4	19	36	45	-	45	1	25	20
[23:16:52]	5	30	28	8	45	-	1	28	45
[23:16:52]	6	13	50	27	1	1	-	4	8
[23:16:52]	7	47	36	23	25	28	4	-	44
[23:16:52]	8	44	50	7	20	45	8	44	-

[23:16:52] ========== Расчет методом полного перебора ==========
[23:16:53] Минимальные пути:
[23:16:53] 1-2-5-6-7-4-8-3-1 Длина: 95
[23:16:53] 1-3-8-4-7-6-5-2-1 Длина: 95
[23:16:53] 2-1-3-8-4-7-6-5-2 Длина: 95
[23:16:53] 2-5-6-7-4-8-3-1-2 Длина: 95
[23:16:53] 3-1-2-5-6-7-4-8-3 Длина: 95
[23:16:53] 3-8-4-7-6-5-2-1-3 Длина: 95
[23:16:53] 4-7-6-5-2-1-3-8-4 Длина: 95
[23:16:53] 4-8-3-1-2-5-6-7-4 Длина: 95
[23:16:53] 5-2-1-3-8-4-7-6-5 Длина: 95
[23:16:53] 5-6-7-4-8-3-1-2-5 Длина: 95
[23:16:53] 6-5-2-1-3-8-4-7-6 Длина: 95
[23:16:53] 6-7-4-8-3-1-2-5-6 Длина: 95
[23:16:53] 7-4-8-3-1-2-5-6-7 Длина: 95
[23:16:53] 7-6-5-2-1-3-8-4-7 Длина: 95
[23:16:53] 8-3-1-2-5-6-7-4-8 Длина: 95
[23:16:53] 8-4-7-6-5-2-1-3-8 Длина: 95
[23:16:53] ========== Расчет методом иди в ближайший ==========
[23:16:53] Минимальный путь:
[23:16:53] 1-4-6-5-3-2-7-8-1 Длина: 176
[23:16:53] ========== Расчет методом Прима-Эйлера ==========
[23:16:53] Минимальные пути:
[23:16:53] 5-3-2-6-1-4-7-8-5 Длина: 227
[23:16:53] 1-6-4-5-3-2-7-8-1 Длина: 214
[23:16:53] 7-6-1-4-5-3-2-8-7 Длина: 206
[23:16:53] 6-1-4-5-3-2-7-8-6 Длина: 196
[23:16:53] 8-6-1-4-5-3-2-7-8 Длина: 196
[23:16:53] 2-3-5-6-1-4-7-8-2 Длина: 183
[23:16:53] 4-6-1-5-3-2-7-8-4 Длина: 175
[23:16:53] 3-2-5-6-1-4-7-8-3 Длина: 160
[23:16:53] ========== Расчет методом Литтла ==========
[23:16:53] Минимальный путь:
[23:16:53] 1-2-7-6-5-3-8-4-1 Длина: 100
[23:16:53] ========== Расчет методом ветвей и границ==========
[23:16:53] Минимальные пути:
[23:16:53] 1-2-7-6-5-3-8-4-1 Длина: 100
[23:16:53] 1-2-5-7-6-4-8-3-1 Длина: 98
[23:16:53] 1-3-8-4-7-6-5-2-1 Длина: 95

Выводы

1. Проведен анализ методов решения задачи коммивояжёра и выбраны наиболее используемые методы: метод полного перебора, метод «иди в ближний», метод Литтла, метод Прима-Эйлера, метод ветвей и границ.

2. Для сравнительного анализа языков программирования предложен набор локальных критериев: ресурсонезависимость, скорость компиляции, кроссплатформенность, производительность, наличие удобных инструментов визуального программирования, отражающих практические аспекты реализации и использования программных средств.

3. На основе метода анализа иерархии выбран язык программирования Object Pascal.

4. На примере выбора маршрута сети кольцевой архитектуры показана практическая применимость разработанного программного средства.

5. Разработанное программное средство может быть использовано для сравнительного анализа рассмотренных методов решения задачи коммивояжёра и выдачи рекомендаций для их практического применения.

Список литературы

1. Белоусов А.И., Ткачев С.Б. Дискретная математика. Серия: Математика в техническом университете. М.: Изд-во: МГТУ им. Н. Э. Баумана, 2001. 744 с.
2. Мадера А.Г., Моделирование и принятие решений в менеджменте. Руководство для будующих топ-менеджеров. М.: ЛКИ, 2009. 688 с.
3. Зыков А.А. Основы теории графов. М.: Вузовская книга, 2004. 664 с.
4. Ляхович В.Ф. Руководство к решению задач по основам информатики и вычислительной техники. М.: Высшая школа, 1994. 256 с.
5. Фляйшнер Г. Эйлеровы графы и смежные вопросы. М.: Мир, 2002. 176 с.
6. Спирина М.С. Дискретная математика: учебник. М.: Академия, 2009. 368 с.
7. Новиков Ф.А. Дискретная математика для программистов. 2-е изд. СПб.: Питер, 2005. 364 с.
8. Гринвуд Б. Выбор перспективного языка программирования // Windows IS Pro/Re. 2014. № 9. С. 30-31.