БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *S. Martello, P. Toth.* Knapsacks problems: algorithms and computer implementations. Chichester/England: John Wiley and sons Ltd, 1990.
2. *Емельянов В.В., Курейчик В.В., Курейчик В.М.* Теория и практика эволюционного моделирования. – М.: Физматлит, 2003.
3. *E. Falkenauer.* Genetic Algorithms and Grouping Problems, Wiley, Chichester, 1998.
4. *Гладков Л.А., Курейчик В.В., Курейчик В.М.* Генетические алгоритмы: Учебное пособие. Под ред. В.М. Курейчика. – Ростов-на-Дону: ООО «Ростиздат», 2004. – 400 с.
5. *A.R Brown.* Optimal Packing and Depletion. American Elsevier, New York, 1971.
6. *Потарусов Р.В., Курейчик В.М.* Проблема одномерной упаковки элементов // Известия ТРТУ. – Таганрог: Изд-во ТРТУ, №8, 2006. – С. 88-93.
7. *Потарусов Р.В., Курейчик В.М.* Модифицированные генетические алгоритмы // Сборник трудов международной научно-практической конференции «Интегрированные модели и мягкие вычисления в искусственном интеллекте», 28-30 мая, Коломна, 2007.
8. *J. Levine, F. Ducatelle.* Ant Colony Optimization and Local Search for Bin Packing and Cutting Stock Problems. - Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh, 2003.
9. *E. Falkenauer.* A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30, 1996.
10. *C. Reeves.* Hybrid genetic algorithms for bin-packing and related problems. *Annals of Operations Research*, 63:371–396, 1996.
11. *M. Vink.* Solving combinatorial problems using evolutionary algorithms, 1997. Available from http://citeseer.nj.nec.com/vink97solving.html.
12. *F. Vanderbeck.* Computational study of a column generation algorithm for bin-packing and cutting stock problems, Math. Programming A 86(1999) 565–594.
13. *R. Potarusov, V. Kureychik, G. Goncalves, H.* Allaoui. Solving the Bin Packing Problem with Algorithm of Genetic Search with Migration // Сборник трудов Международных научно-технических конференций «AIS'07», «CAD'07». – М.: Физматлит, 2007, – С. 34-45.

УДК 004.023

Т.С. Емельянова

ОБ ОДНОМ ГЕНЕТИЧЕСКОМ АЛГОРИТМЕ РЕШЕНИЯ ТРАНСПОРТНОЙ ЗАДАЧИ С ОГРАНИЧЕНИЕМ ПО ВРЕМЕНИ*

Введение. Данная статья посвящена новому генетическому алгоритму (ГА) решения транспортных задач (ТЗ) с ограничениями по времени (VRPTW –vehicle routing problems with time windows) [1]. Транспортные задачи или задачи маршрутизации транспортных средств возникают в различных областях деятельности человека: доставка товаров от поставщика к клиенту, доставка сырья на производство, сбор промышленных отходов, почтовая доставка и т.д. Так как цена перевозки различного рода товаров явно или не явно присутствует в их стоимости, то сокращение транспортных расходов является важной и насущной экономической задачей. Целью решения всех ТЗ является составление маршрутов транспортных средств минимальных по ценовым затратам. ТЗ с ограничением по времени являются подклассом ТЗ, в них учитывается время, в течение которого должен быть обслужен клиент. Являясь более сложными по постановке, данные задачи более полно описывают реальный процесс, т.к. во многих практических задачах доставки товаров время прибытие к клиенту и время обслуживания клиента играют сущест-

* Работа выполнена при поддержке: РФФИ (грант № 07-01-00174), РНП 2.1.2.3193, РНП 2.1.2.2238, г/б № Т.1.04.01, г/б № Т.12.8.08

венную роль. К данным задачам относятся задачи доставки людей, груза, сбор промышленных отходов и пр. [1-2]. Предложенный ГА позволяет успешно решать ТЗ с ограничением по времени, что продемонстрировано результатами решения тестовых задач. Наиболее успешно решаются задачи, в которых клиенты расположены группами (сгруппированы). Данное расположение соответствует практическому расположению клиентов, например, расположение клиентов (магазинов) в определенных городах относительно центрального завода или склада. В данном случае клиенты в определенном городе будут сгруппированы относительно центра.

В данной статье дана математическая формулировка ТЗ с ограничением по времени, описаны все генетические операторы, используемые в данном ГА, сам ГА решения ТЗ, а также приведены результаты решения тестовых задач Соломона данным ГА.

1. Постановка транспортной задачи с ограничением по времени. ТЗ с ограничением по времени относится к классу задач маршрутизации автотранспорта (VRP – Vehicle Routing Problem). Задачи данного типа можно описать следующим образом. Имеется некоторое количество автотранспорта, один склад (депо) и некоторое количество клиентов. Для каждого транспортного средства требуется составить маршрут, на протяжении которого транспортное средство посещает ряд клиентов (например, с целью доставки какого-либо груза). На маршрут каждого транспортного средства накладывается ряд ограничений. Каждый маршрут должен начинаться и заканчиваться в депо. Общее количество товаров, требуемых для доставки клиентам на данном маршруте данного транспортного средства не должно превышать его грузоподъемность. Каждый клиент обслуживается лишь одним транспортным средством и лишь единожды, т.е. не допускается посещение одного клиента двумя и более транспортными средствами. Каждый клиент должен быть обслужен в определенный промежуток времени, этот промежуток определяется двумя значениями, первое значение определяет время прибытии транспортного средства к клиенту, второе время отправления. Для данной задачи формулируются следующие цели (целевые функции): первичная цель – минимизировать общее количество транспортных средств, необходимых для обслуживания всех клиентов; вторичные – минимизировать общее время обслуживания всех клиентов и общее расстояние, пройденное всеми транспортными средствами [2].

Согласно [1, 2] математически ТЗ с ограничением по времени можно представить в виде графа $G = (N, A)$, где N – множество вершин, которые соответствуют набору клиентов (customers) и обозначаются $1, 2, \ldots, n$, и вершины 0 и $n + 1$ соответствуют исходному депо (depot) из которого начинают и в котором заканчивают свой маршрут все автомобили; A – набор дуг соединяющих соответствующие вершины графа (соответствующих клиентов), если i – один клиент, а j – другой, то дуга их соединяющая обозначается $(i, j) \in A$. Обозначим C – множество клиентов $|C| = n$. Каждый клиент характеризуется определенным спросом d_i, $i \in C$. Каждой дуге $(i, j) \in A$ соответствуют время t_{ij} – время перемещения от клиента i, $i \in C$ к клиенту j, $j \in C$, это время включает в себя время обслуживание клиента i и c_{ij} – стоимость пути автомобиля из i в j. Верхним индексом k, далее будет обозначаться соответствующий автомобиль (где $k \in V$, V – количество идентичных автомобилей грузоподъемностью q). Каждый клиент должен быть обслужен в определенный промежуток времени, так называемое “временное окно” (time window), обозначаемый $[a_i, b_i]$, $i \in C$. Время прибытии соответствующего автомобиля в определенную вершину графа обозначается S_i^k для $\forall i \in N$, $\forall k \in V$. Время отправ-

ления из депо для всех автомобилей равно 0, т. е. $S_0^k = 0$ (для $\forall k \in V$). Полная математичесая формулировка ТЗ с ограничением по времени приводится в [1, 2].

2. Генетический алгоритм решения транспортной задачи с ограничением по времени. Генетические алгоритмы (ГА) являются случайно направленными поисковыми методами и успешно применялись для решения задач транспортного типа [3, 4]. В общем случае структура ГА состоит из следующих пунктов [5].

1. Конструирование начальной популяции определенного размера (оператор инициализации).
2. Выбор двух родительских хромосом из популяции (оператор селекции).
3. Копирование выбранных хромосом и применение генетических операторов для создания новых хромосом (операторы кроссинговера, мутации).
4. Отбор и последующее удаление хромосом из популяции для восстановления ее первоначального размера.
5. До тех пор пока не пройдено заданное число шагов, возврат к шагу 2, в противном случае конец работы алгоритма.

Далее будут описаны основные генетические операторы: инициализации, кроссинговера, мутации, а также способ кодирования решения применяемый в данном ГА.

Кодирование решения. Для эффективной работы генетического алгоритма требуется простой и удобный способ представления решения или кодирование решения. От выбора метода кодирования зависит работа операторов кроссинговера и мутации. В [6-9] часто используют представление решения в виде набора маршрутов транспортных средств, где количество маршрутов совпадает с количеством транспортных средств, а каждый маршрут представлен в виде набора клиентов (узлов графа), которые последовательно посещает данное транспортное средство. Любой маршрут начинается и заканчивается клиентом номер 0 – депо. Это очевидно, в связи с ограничением задачи, согласно которому все маршруты должны начинаться и заканчиваться в депо. В данном алгоритме это представление будет расширено и каждому клиенту, кроме номера, будут соответствовать пять переменных, описывающих состояние машины во время обслуживания данного клиента, которые будут записываться рядом в круглых скобках, это – время прибытия к данному клиенту at_{ik}, время отправления от данного клиента после его обслуживания wt_{ik}, загрузка автомобиля после обслуживания данного клиента l_{ik}, общее расстояние, пройденное автомобилем к этому моменту D_{ik}, и общее время простоя (задержки) данной машины dt_{ik}. В данных обозначениях индексами i и k обозначаются номер клиента в маршруте и номер машины (маршрута) соответственно. Т. к. каждый маршрут начинается в депо, то $at_{0k} = 0$; $wt_{0k} = S_0^k$; $l_{0k} = q$ – грузоподъемность автомобиля; $D_{0k} = 0$; $dt_{0k} = 0$ для любых k. Представление решений в виде набора маршрутов позволит в последующем оперировать как маршрутами целиком, рассматривая их в качестве хромосом, так и отдельными клиентами представляя их набором генов. Данное представление удобно для описанных ниже операторов кроссинговера и мутации.

Оператор инициализации. Оператор инициализации это ключевой оператор в данном генетическом алгоритме, т.к. он будет использоваться в операторах мутации и кроссинговера. Данный оператор использует метод конструирования маршрутов – метод Соломона [6, 7]. Алгоритм данного оператора состоит из следующих этапов.

1. Если все клиенты обслужены, переход к пункту 6.
2. Выбираем клиента k^* из числа ещё не обслуженных, для этого берем случайное число в пределах от 1 до N – количество клиентов.

3. Последовательно перебираем клиентов имеющегося решения для вставки клиента k^* в имеющийся маршрут текущего решения. Если такие допустимые вставки существуют, выбираем, ту в которой добавочное расстояние (в связи с появлением нового клиента k^*) меньше. Если существуют две вставки с одинаковым добавочным расстоянием, выбираем ту, в которой меньше общая задержка (время простоя машины).
4. Если допустимых вставок в имеющиеся маршруты текущего решения не существует, то начинаем новый маршрут, в который и вставляется клиент k^*. Данная вставка будет допустима всегда, если нет ограничения на количество машин.
5. Переходим к пункту 1.

Все клиенты обслужены, решение построено. Выход.

Преимущество данного оператора в том, что он позволяет не только строить новые решения при инициализации популяции решений, но и достраивать решения (решения в которых часть клиентов обслужена, а часть еще нет), что будет использоваться в описанных ниже операторах.

Оператор кроссинговера. Оператор кроссинговера важный и критичный оператор в ГА, т. к. позволяет закрепить в решениях-потомках полезные свойства родителей [10]. Применительно к данной транспортной задаче с ограничением по времени это означает, что оператор кроссинговера позволяет решениям-потомкам наследовать наиболее удачные маршруты решений-родителей. В данном операторе кроссинговера количество родителей, участвующих в скрещивании определяется числом N. Опытным путем было определено, что эффективнее всего выбирать N равным *2, 3* и *4*.

Алгоритм оператора кроссинговера описан ниже:

1. Выбираем число N – количество решений (особей) участвующих в кроссинговере.
2. Выбираем N решений из популяции. Для этого применяем оператор **Селекции**.
3. Маршруты из выбранных решений объединяем в одно решение. Данное решение назовем *объединенным решением*.
4. Пока в объединенном решении есть маршруты, делаем следующее:
   - выбираем маршрут и вставляем его в новое решение, для этого берем случайное число в пределах от 0 до количества маршрутов в объединенном решении, данное число и будет указывать номер маршрута в объединенном решении;
   - удаляем выбранный маршрут в объединенном решении;
   - удаляем в объединенном решении все маршруты, в которых есть клиенты из выбранного решения.
5. Вставляем не обслуженных клиентов в новое решение с использованием оператора **Инициализации.**
6. Новое решение является потомков выбранных N решений родителей.

Данный оператор кроссинговера реализован по аналогии с адаптивной памятью в методе поиска с запретами [8]. Объединенное решение аналогично адаптивной памяти. Отличительная особенность состоит в том, что количество решений формирующих объединенное решение гораздо меньше, чем в методе поиска с запретами, также маршруты выбираются произвольным образом, тогда как маршруты из адаптивной памяти выбираются с вероятностью тем выше, чем меньше ЦФ решения соответствующего данному маршруту. ЦФ решения учитывается и оценивается при выборе решений для скрещивания (оператор селекции). Следует от-

метить, что преимуществом данного оператора кроссинговера является то, что в результате получается допустимое решение, не требующее дальнейшей обработки.

Операторы мутации. В данном ГА используются два оператора мутации *OM_1* и *OM_2*. Первый оператор работает с маршрутами целиком и является менее затратным по времени, второй работает с отдельными клиентами и затрачивает большие временные ресурсы, чем первым. В общем случае, операторы мутации используются для расширения пространства поиска [10]. В данном ГА в обычном процессе поиска применяется оператор мутации *OM_1*. При "застаивании" процесса поиска (ЦФ не меняется в течение определенного, заранее заданного числа итераций) используется оператор мутации *OM_2*. Ниже приводятся алгоритмы данных операторов мутации. Описание работы оператора мутации *OM_1*.

1. Имеется некоторое решение задачи, к которому будет применяться оператор мутации. Задаем *M* – число удаляемых маршрутов.
2. Удаляем *M* маршрутов из решения. Маршруты для удаления выбираются из решения вероятностным образом.
3. Заново вставляем клиентов из удаленных маршрутов в решение с использованием оператора инициализации описанного выше.

Экспериментальным путем было установлено, что число должно *M* меняться в пределах от *2* до *5* в зависимости от типа задачи. В результате применения оператора мутации *OM_1* получаем допустимое решение, которое не увеличивает количество транспортных средств и расширяет область поиска за счет нового набора маршрутов.

Далее приведем описание оператора мутации *OM_2*.

1. Имеется некоторое решение задачи, к которому будет применяться оператор мутации. Задаем *K* – число удаляемых клиентов.
2. Удаляем *K* клиентов из решения. Клиенты для удаления выбираются из решения вероятностным образом.
3. Заново вставляем *K* клиентов из удаленных маршрутов в решение с использованием оператора инициализации описанного выше.

В данном алгоритме число *K = количество клиентов / 4*. В результате применения оператора мутации *OM_2* получаем новое решение, которое расширяет область поиска за счет нового набора маршрутов. Решение, получаемые в результате применения данного оператора мутации, являются допустимым и также как при применении оператора кроссинговера не требуют дополнительной обработки.

Результаты экспериментов. Данный алгоритм проходил тестирование на тестовых задачах Соломона [10]. Эти задачи разбиты на группы R1 (12 задач), R2 (11 задач), C1 (9 задач), C2 (8 задач), RC1 (8 задач), RC2 (8 задач). В задачах группы R клиенты географически распределены равномерно вероятностным образом, в задачах группы C клиенты расположены группами, в задачах группы RC часть клиентов расположена группами, а остальная часть распределена равномерно вероятностным образом. В каждой задаче 100 клиентов и одно депо, задачи различаются по временным ограничениям. Задачи Соломона предназначены для тестирования алгоритмов решения ТЗ с ограничением по времени. На Internet ресурсе [11] находятся лучшие решения для данных тестовых задач найденные с помощью других алгоритмов.

Следует отметить, что лучшие решения для различных типов тестовых задач найдены различными алгоритмами, т. е. нет универсального алгоритма. Поэтому для оценки алгоритма сравнивается общее количество машин и общее пройденное расстояние в каждой группе тестовых задач. В таблице показаны усредненные результаты по каждой из шести групп тестовых задач, а также общее количество полученных машин и общее пройденное расстояние.

Таблица

R1	R2	C1	C2	RC1	RC2	Суммарное количество машин/суммарное расстояние
12.41 1209.3	**3.09** 976.4	**10.00** 828.4	**3.00** 589.9	**12.13** 1368.9	**3.38** 1169.5	**420** 53682

При решении тестовых задач с использованием данного ГА было получено, что общее пройденное расстояние, в каждой отдельной задаче близко к лучшему, а в задачах группы С совпадает с лучшим значение. Общее пройденное расстояние при решении всех тестовых задач является наименьшим среди других алгоритмов решения данного типа задач. Количество машин в полученных решениях совпадает с количеством машин лучших решений или превышает это количество, но не более, чем на одну машину. Из приведенных результатов решения тестовых задач Соломона можно сделать вывод, что данный алгоритм позволяет эффективно решать ТЗ с ограничением по времени.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Tobias Babb.* Pickup and Delivery Problem with Time Windows, Coordinated Transportation Systems: The State of the Art. Department of Computer Science University of Central Florida Orlando, Florida, 2005.
2. *J.-F. Cordeau, Guy Desaulniers, Jacques Gesrosiers, Marius M. Solomon, Francois Soumis.* The VRP with Time Windows. Chapter 7, Paolo Toth and Daniel Vigo (eds), SIAM, Monographs on Discrete Mathematics and Applications, 2001.
3. *Емельянова Т.С.* Применение генетических алгоритмов для решения транспортной задачи линейного программирования // Перспективные информационные технологии и интеллектуальные системы. – Таганрог, №3(27), 2006. – С. 15-29.
4. *Емельянова Т.С.* Об одном генетическом алгоритме решения транспортной задачи. // Известия ТРТУ. – Таганрог, №1(73), 2007. – С. 65-70.
5. Генетические алгоритмы: Учебное пособие. Под ред. В. М. Курейчика. – Ростов-на-Дону: ООО «Ростиздат», 2004.
6. *K.C. Tan, L. H. Lee, K. Q. Zhu.* Heuristic Methods for Vehicle Routing Problem with Time Windows. Proceedings of the 6th Internayional Symposium on Artifical Intelligence & Mathematics, Ft. Lauderdale, Florida, 2000.
7. Olli Bräysy, Michel Gendreau. Route Construction and Local Search Algorithms for the Vehicle Routing Problem with Time Windows. Internal Report STF42 A01024, SINTEF Applied Mathematics, 2001.
8. *EricTaillard, Philippe Badeau, Michel Gendreau, Francois Guertin and Jean-Yves Potvin.* A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. Transportation Science 31, pp. 170 – 186, 1997.
9. *EricTaillard, Philippe Badeau, Michel Gendreau, Francois Guertin and Jean-Yves Potvin.* A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. Transportation Research – C 5, pp. 109 – 122, 1997.
10. http://web.cba.neu.edu/~msolomon/problems.htm.
11. http://web.cba.neu.edu/~msolomon/heuristi.htm.