УДК 658.51:519.8

К. А. Кузнецов, кандидат физико-математических наук,
доцент кафедры математического обеспечения ЭВМ
Днепропетровского национального университета
имени Олеся Гончара
В. А. Громов, кандидат физико-математических наук,
доцент кафедры вычислительной математики
и математической кибернетики Днепропетровского
национального университета имени Олеся Гончара

ПОДХОД К РЕШЕНИЮ ЗАДАЧИ ОПТИМИЗАЦИИ СТРУКТУРЫ ДИСТРИБЬЮТОРСКОЙ КОМПАНИИ

Представлены результаты численного анализа задачи оптимизации работы дистрибьюторской компании, математическая модель которой обсуждалась в работе [1]. Предложен эвристический поход к декомпозиции задачи, разработаны алгоритмы решения полученных подзадач.

Ключевые слова: *системный анализ; маршрутизация транспортных средств; целочисленное программирование.*

The paper analyzes numerically logistics of large distribution company. The analysis is based upon company logistics model presented in the previous paper of the authors (published in the present edition). The problem is heuristically decomposed into subproblems; methods to solve these subproblems are presented.

Key words: *system analysis; vehicle routing problems; integer programming.*

Постановка проблемы. Эффективность работы крупной дистрибьюторской компании напрямую зависит от качества и оптимальности принимаемых управленческих решений. Обычно здесь необходимы системный анализ и оптимизация по следующим направлениям деятельности компании: управление запасами, транспортная логистика, складская логистика, оптимизация работы торговых агентов.

Настоящая работа посвящена решению задачи сокращения расходов компании на оплату труда агентов и доставку товаров потребителю. С математической точки зрения, здесь возникают задачи кластеризации, теории составления расписаний и маршрутизации движения транспортных средств. Многообразие ситуаций, встречающихся при решении подобных практических задач исследования операций по всему миру, приводит к различным сочетаниям задач указанных классов во всевозможных постановках.

Несмотря на разнообразие постановок и методов решения в большинстве случаев практически значимых ситуаций, может быть найдено эффективное решение, что позволяет надеяться на результативность применения моделей и методов исследования операций и в случае, рассмотренном в данной работе.

Подчеркнем, что во всех известных авторам работах для поиска решения использовалась та или иная эвристика. Это обстоятельство не является случайностью, что связано с вычислительной сложностью решения задач данного класса для реальных размерностей.

Анализ последних исследований и публикаций. Сформулированная в [1] задача предполагает многократное решение подзадач маршрутизации транспортных средств (*VRP*) и определение оптимального маршрута движения агентов (*TSPMTW*). *NP*-трудность указанных подзадач делает невозможным нахождение оптимального решения задачи за реальное время. Необходимость нахождения решения подобных задач на практике обычно ведет к формулировке эвристических подходов к решению, учитывающих специфику задачи [2].

Естественным предположением в рассматриваемой задаче является предположение об относительной компактности зон обслуживания для агентов всех групп. На аналогичном предположении основывается часто встречающаяся в приложениях постановка кластерной задачи маршрутизации (*Clustered VRP, CluVRP*).

Постановка кластерной задачи маршрутизации была впервые сформулирована Сево и Соренсеном [3] для решения задач оптимизации службы курьерской доставки. Доставка товаров клиентам осуществляется в специальных контейнерах. Товары, которые должны быть доставлены близко расположенным клиентам (кластер), грузятся в отдельный контейнер. Курьер возвращается за товаром на склад только после того, как все товары из контейнера доставлены клиентам.

В работе [4] приведен метод отыскания точного решения задачи *CluVRP*, сформулированной как задача целочисленного программирования. Метод использует технику препроцессинга, существенно упрощающую структуру решаемой задачи. Эта техника предполагает решение квадратичного числа *TSP*-задач малой размерности. Полученные маршруты используются для уменьшения размерности графа, который и используется для отыскания точного решения задачи различными вариантами метода ветвей и границ.

В работе [5] для кластеризации множества клиентов используется подход, основанный на геометрических принципах определения понятия кластера. Для решения задачи маршрутизации транспортных средств с несколькими складами использован генетический алгоритм. Эффективность предложенного авторами подхода была оценена путем сравнения с результатами, полученными методом ближайшего соседа.

Еще один подход к решению задачи *CluVRP*, имплементирующий идею геометрической кластеризации, приведен в [6]. Предложенная монгольским автором эвристика состоит из трех этапов: на начальном этапе выбираются наиболее удаленные точки множества как центры будущих кластеров. На втором этапе осуществляется кластеризация остальных точек множества с учетом меры близости к центрам, а затем, на третьем этапе, отыскивается решение задачи *TSP* для каждого из полученных кластеров.

Процесс предварительной кластеризации множества точек был использован в работе [7] для решения задачи маршрутизации транспортных средств с временными окнами (*VRPTW*).

Цель статьи. В настоящем исследовании зона ответственности агента определяется как выпуклая оболочка множества торговых точек, обслуживаемых этим агентом. При построении выпуклой оболочки торговая точка $i \in V$ рассматривается как точка на карте с заданной широтой и долготой $lat_i, long_i$. Рассматриваемый в настоящей работе подход к решению базируется на предположении о том, что зоны ответственности агентов одной группы не пересекаются. Таким образом, решение задачи предполагает разбиение области, которой принадлежат точки, обслуживаемые агентами одной группы, на непересекающиеся выпуклые подмножества.

Указанное предположение позволило сформулировать следующий алгоритм отыскания эффективного решения оптимизационной задачи при ограничениях на переменные задачи. Детальное описание целевой функции и ограничений приведено в работе [1].

Изложение основного материала. При описании алгоритма будет использован оператор ***Repair***, который минимизирует суммарную меру нарушения ограничений на продолжительность рабочего времени агентов. Подробное описание данного оператора, а также других операторов, используемых при описании алгоритма, будет дано ниже.

Алгоритм. Поиск оптимального решения

$n \leftarrow 0$

for $m = \overline{1, M}$ ***do***

$V_{mk}^{(n)} \leftarrow$ ***Initial_Clustering*** (V_m)

$\Xi_{mk}^{(n)} \leftarrow$ ***Neighbors*** $(V_{mk}^{(n)}), k = 1,..., k_m^{(n)}$

endfor

$U^{(n)} \leftarrow$ ***Infeasibility*** $(V_{mk}^{(n)})$

$I^{(n)} \leftarrow$ ***Objective*** $(V_{mk}^{(n)})$

Repeat

for $m = \overline{1, M}$ ***do***

for $\left(all\ pairs\ (i, j) : i = 1,..., k_m^{(n)}, j \in \Xi_{mi}^{(n)}\right)$ ***do***

$V_{mi}^{(n)} \leftarrow$ ***Join*** $(V_{mi}^{(n)}, V_{mj}^{(n)})$

$\Xi_{mi}^{(n)}(i, j) \leftarrow \Xi_{mi}^{(n)} \cup \Xi_{mj}^{(n)}, \quad \Xi_{mk}^{(n)}(i, j) : k \in \Xi_{mj}^{(n)} \leftarrow \Xi_{mk}^{(n)} \cup i \setminus j$

$\tilde{V}_{ms}^{(n)}(i, j) \leftarrow$ ***Repair*** $(V_{ms}^{(n)}), s = 1,2...$

$U_m^{(n)}(i, j) \leftarrow$ ***Infeasibility*** $(\tilde{V}_{ms}^{(n)})$

$I_m^{(n)}(i, j) \leftarrow$ ***Objective*** $(\tilde{V}_{ms}^{(n)})$

endfor

$(i_m^*, j_m^*) = \arg \min_{U_m^{(n)}(i, j) = 0} I_m^{(n)}(i, j)$

$V_{mk}^{(n+1)} \leftarrow \tilde{V}_{mk}^{(n)}(i_m^*, j_m^*), k = 1,..., k_m^{(n+1)}$

$\Xi_{mk}^{(n+1)} \leftarrow \Xi_{mk}^{(n)}(i_m^*, j_m^*)$

endfor

$U^{(n+1)} \leftarrow \max_m U_m^{(n)}(i_m^*, j_m^*)$

$I^{(n+1)} \leftarrow$ ***Objective*** $(V_{mk}^{(n+1)})$

$n \leftarrow n+1$

Until $I^{(n)} < I^{(n-1)}$ & $U^{(n)} = 0$

return $V_{mk}^{(n-1)}$.

Остановимся теперь подробнее на описании операторов, используемых в приведенном алгоритме.

Получение начального разбиения (Initial_Clustering). Для построения начального приближения для зон ответственности агентов *j*-й группы необходимо разбить множество торговых точек, обслуживаемых агентами данной группы, V_j на подмножества V_{jk}, выпуклые оболочки которых не пересекаются

$$\bigcup_{k=1}^{k_j^0} V_{jk} = V_j, \quad Conv\,V_{jk} \cap Conv\,V_{ji} = \varnothing,\ k \neq i.$$

Здесь и далее под *Conv(A)* подразумевается выпуклая оболочка множества точек $\{lat_i, long_i, i \in A\}$.

Для получения указанного разбиения случайным образом выбирается $k_j^0 = 0,2 \times |V_j|$ торговых точек $G_j = \{g_{jk}, k = 1, \ldots, k_j\}$, которые служат центрами кластеров V_{jk}. Для множества G_j строится диаграмма Вороного [8], и торговые точки, принадлежащие множеству Вороного с центром g_{jk}, считаются принадлежащими кластеру V_{jk}.

Определение соседства для начального разбиения (Neighbors). Диаграмма Вороного позволяет естественным образом определить понятие соседних кластеров, а именно соседними считаются кластеры, множества Вороного которых имеют общее ребро. Введя в рассмотрение двойственный объект к диаграмме Вороного – триангуляцию Делоне [8], мы можем найти соседние кластеры как такие, центры которых соединены ребром в графе Делоне. Обозначим множество соседних кластеров для кластера V_{jk} как Ξ_{jk}.

На рис. 1 представлен фрагмент начальной кластеризации для торговых точек первой группы агентов. Точки множества G_j выделены кружками, границы множеств Вороного обозначены сплошными линиями, также сплошными линиями изображен соответствующий граф Делоне. Видно, что ребра графа Делоне соединяют центры соседних кластеров.

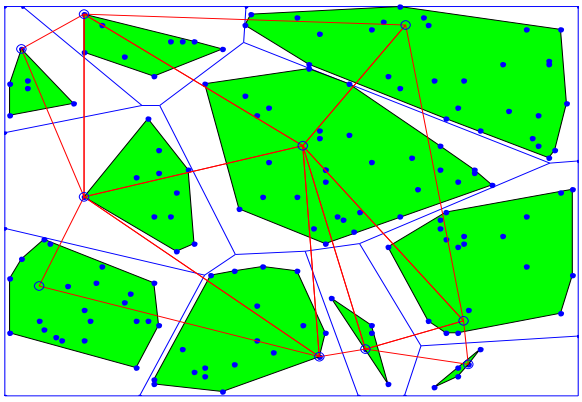

Рис. 1. Начальная кластеризация и отношение соседства

Для оценки качества текущего решения будем оперировать двумя характеристиками.

Мера недопустимости разбиения (Infeasibility). Характеристикой, описывающей меру недопустимости построенного расписания для *k*-го кластера *m*-й группы, является величина

$u_{mk} = \max\limits_{t=1,T} \max\ 0, \tau_{mkt} - 480$, где τ_{mkt} – время работы *k*-го агента *m*-й группы в *t*-й день горизонта планирования [1]. Суммарное нарушение ограничений на длительность рабочего дня агентов формализуется как мера недопустимости разбиения

$$U = \sum_{m=1}^{M} \sum_{k=1}^{k_m} u_{mk} .$$

Для допустимых решений *U=0*. Для нахождения оптимальных значений величин τ_{mkt} необходимо распределить точки кластера V_{mk} (точки, посещаемые *k*-м агентом *m*-й группы) по дням в пределах горизонта планирования. С использованием введенных в [1] векторов возможных сценариев посещения торговых точек $c^i = c_1^i, c_2^i, ..., c_M^i$, $i \in V_{mk}$ решение рассматриваемой подзадачи можно представить в виде целочисленного вектора

$$\xi_{mk} = c_{p_1}^1, c_{p_2}^2, \ldots, c_{p_{|V_{mk}|}}^{|V_{mk}|}, \quad c_{p_i}^i \in c^i .$$

Компоненты вектора ξ_{mk} отвечают сценариям посещения торговых точек. Для заданного вектора ξ_{mk} легко определить множество торговых точек, посещаемых *k*-м агентом *m*-й группы в *t*-й день горизонта планирования, V_{mkt}, $m = \overline{1,M}$, $k = \overline{1,k_m}$, $t = \overline{1,T}$ (в алгоритме для указанной операции используется обозначение ***Day_points***). Для каждого из множеств V_{mkt} решается задача коммивояжера с множественными временными окнами (*TSPMTW*), что позволяет определить значения τ_{mkt}.

Для сравнения двух расписаний *A* и *B*, построенных для точек множества V_{mk}, использовался лексикографически упорядоченный список величин τ_{mkt}, $t = \overline{1,T}$. Расписание *A* считается лучше расписания *B*, если:

$$A \prec B \Leftrightarrow \tau_{mkt_1}^A \geq \tau_{mkt_2}^A \geq \ldots \geq \tau_{mkt_T}^A \ \wedge\ \tau_{mkt_1}^B \geq \tau_{mkt_2}^B \geq \ldots \geq \tau_{mkt_T}^B \ \wedge$$

$$\exists t = \overline{1,T} : \tau_{mks}^A = \tau_{mks}^B, \forall s < t, \quad \tau_{mkt}^A < \tau_{mkt}^B .$$

Тем самым вводится отношение порядка на множестве всех возможных векторов ξ_{mk}. Для отыскания минимального в смысле введенного отношения порядка вектора ξ_{mk}^* использовался простой локальный поиск (*HillClimbing*), при этом вектор $\psi_{mk} = c_{r_1}^1, c_{r_2}^2, \ldots, c_{r_{|V_{mk}|}}^{|V_{mk}|}$, $c_{r_i}^i \in c^i$ принадлежит окрестности $N_1(\xi_{mk})$ текущего решения ξ_{mk}, если $\exists i^* : r_i = p_i, \forall i = \overline{1,|V_{mk}|}, i \neq i^*$. В алгоритме построения расписания агента использовалась версия алгоритма локального поиска с выбором наилучшего элемента окрестности (*SteepestAscent*).

Алгоритм. Построение расписания агента

$\xi_{mkp} \leftarrow$ ***random*** $(p, p = \overline{1, |V_{mk}|})$

$V_{mkt} \leftarrow$ ***Day_points*** $(\psi_{mk}, t = \overline{1, T})$

$\tau_{mkt} \leftarrow$ ***TSPMTW*** $(V_{mkt}, t = \overline{1, T})$

$found \leftarrow$ *false*

Repeat

Forall $\psi_{mk} \in N_1(\psi_{mk})$ ***do***

$\tilde{V}_{mkt} \leftarrow$ ***Day_points*** $(\psi_{mk}, t = \overline{1, T})$

$\tilde{\tau}_{mkt} \leftarrow$ ***TSPMTW*** $(\tilde{V}_{mkt}, t = \overline{1, T})$

If $\tilde{\tau}_{mk} \prec \tau_{mk}^{\min}$

$\tau_{mk}^{\min} \leftarrow \tilde{\tau}_{mk}, \quad V_{mkt}^{\min} \leftarrow \tilde{V}_{mkt}, t = \overline{1, T}$

End If

End For

If $\tau_{mk}^{\min} \prec \tau_{mk}$

$\tau_{mk} \leftarrow \tau_{mk}^{\min}, \quad V_{mkt} \leftarrow V_{mkt}^{\min}$

Else

$found \leftarrow$ *false*

End If

Until *found*

Return τ_{mk}.

Для отыскания решения подзадачи коммивояжера с временными окнами использовался метод, предложенный в [9].

Эмпирический коэффициент 0,2 в формуле для определения начального количества агентов $k_m^{(0)}$ был выбран из соображения обеспечения допустимости начального разбиения V_{mk}. Для рассматриваемых данных такой выбор начального количества агентов практически гарантировал получение допустимого решения. Однако даже возможное получение недопустимого начального разбиения не влияет на дальнейшую работу алгоритма, так как далее решение корректируется (ниже дано описание оператора *Repair*).

Мера качества решения (Objective). Для оценки качества решения использовалось значение составного критерия, вычисленного для данного разбиения. Детально критерий описан в [1]. Для фиксированного разбиения вычисление значения компоненты I_1 (затраты компании на оплату труда агентов) очевидно. Для нахождения значения компоненты I_2 (затраты компании на движение агентов по маршрутам) необходимо составить расписание для каждого агента (k=1,...,k_j) каждой группы (j=1,...,M), что предполагает многократное решение задачи коммивояжера с несколькими временными окнами (*TSPMTW*). Построение расписания по всем агентам всех групп для дня t (t=1,...,T) позволяет определить множество точек, в которые необходимо произвести завоз товара в (t+1)-й день. Зная величины заказов d_i по соответствующим торговым точкам, можно определить значение компоненты I_3 (затраты на аренду транспортных средств). Для вычисления последнего слагаемого в функционале I (затраты на доставку товаров в торговые точки) необходимо для каждого дня t (t=1,...,T) решить задачу маршрутизации транспортных средств. Методы решения указанных задач будут рассмотрены ниже.

Объединение кластеров (Join). Для минимизации количества торговых агентов, обслуживающих торговые точки, относящиеся к одной группе, в рамках алгоритма осуществляется пробное объединение соседних кластеров. При этом условие непересечения выпуклых оболочек кластеров должно по-прежнему выполняться. Вследствие этого слияние двух кластеров V_{ji} и V_{jk} может привести к изменению конфигурации других кластеров в разбиении. Точки других кластеров V_{js}^{n} $s \neq i,k$, оказавшиеся внутри выпуклой оболочки объединенного кластера, считаются принадлежащими указанному кластеру

$$s \in V_{ji} \cup V_{jk} \Leftrightarrow lat_s, long_s \in Conv\left(V_{ji} \cup V_{jk}\right).$$

При этом возможна ситуация, когда объединенный кластер целиком поглотит некоторый третий кластер.

Рис. 2 иллюстрирует процесс объединения двух кластеров. Пунктирными линиями обозначены границы кластеров до слияния. Точки принадлежат поглощаемому кластеру, звездочки – точки других кластеров, изменившие свою принадлежность вследствие объединения.

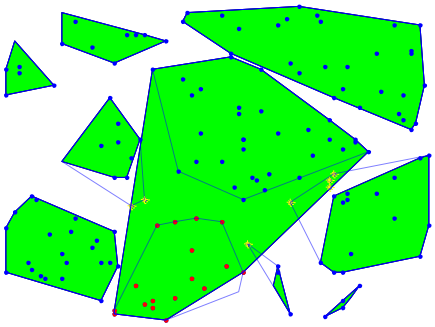

Рис. 2. Объединение кластеров

Процесс объединения кластеров завершается процедурой пересчета отношения соседства между кластерами

$$\tilde{V}_{ji} = V_{ji} \cup V_{jk} \Leftrightarrow \tilde{\Xi}_{ji} = \Xi_{ji} \cup \Xi_{jk}.$$

Построение допустимого решения (Repair). В результате слияния кластеров или неудачного начального разбиения некоторые кластеры могут оказаться слишком большими: для них невозможно построить расписание агента, удовлетворяющее всем ограничениям задачи [1]. Целью алгоритма *Repair* является восстановление допустимости решения или установление того факта, что такого решения не существует для данного количества кластеров.

Для построения допустимого решения алгоритм *Repair* осуществляет перенос точек из кластера с максимальным временем работы агента, превышающим допустимый предел в 480 минут, в соседний с ним кластер. В качестве точки – кандидата на перенос выступает точка, принадлежащая выпуклой оболочке “перегруженного” кластера, расстояние от которой до кластера соседа минимально. При этом для выполнения требования выпуклости кластеров

осуществляется перестройка выпуклых оболочек (рис. 3). Указанная перестройка может изменить принадлежность некоторых точек других кластеров.

Наиболее простой стратегией поиска допустимого решения является разгрузка кластера с максимальным значением меры недопустимости u_{mk} с перекладыванием точки в соседний кластер с минимальным значением меры недопустимости. Однако в рамках этой стратегии организация процесса переноса точек между кластерами, основанная на требовании монотонного убывания меры недопустимости решения, потенциально приводит к преждевременной остановке алгоритма на решении, далеком от оптимального.

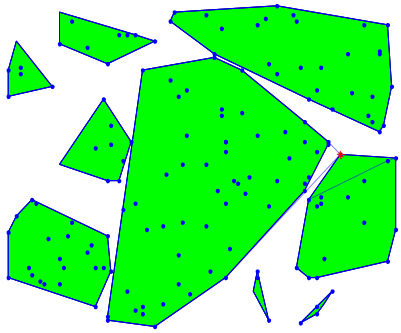

Рис. 3. Перенос точек из кластера в кластер

Возможным путем решения указанной проблемы является увеличение ширины и/или глубины поиска. Под шириной здесь понимается множество кластеров – кандидатов для удаления и добавления точки, а под глубиной – количество пробных шагов, которые необходимо осуществить для окончательного принятия решения о перекладывании. Оба варианта ведут к существенному росту вычислительных затрат, что, в конечном итоге, не позволяет получить решение задачи за приемлемое время.

Поэтому было предложено остаться в рамках вышеупомянутой простой стратегии, но для предотвращения возникновения ситуации преждевременной остановки использовать парадигму метода поиска с запретами (*TabuSearch*) [10].

В рамках алгоритма для каждой из групп в качестве “разгружаемого” кластера выбирается кластер k^* с максимальным значением меры недопустимости u_{mk}. Если среди соседей кластера k^* есть разрешенные кластеры, то в качестве “догружаемого” кластера j^* выбирается тот из них, для которого значение величины меры недопустимости минимально; если же все соседние кластеры помечены как запрещенные, то в рамках стратегии преодоления запрета (*aspiration criterion*) поиск осуществляется по всем соседним кластерам.

Далее определяется точка p^* для переноса из “разгружаемого” в “догружаемый” кластер. Перенос точки p^*, как и в случае объединения кластеров, может сопровождаться изменением принадлежности точек других кластеров, оказавшихся внутри новой выпуклой оболочки кластера j^*. Также осуществляется пересчет мер недопустимости для всех измененных кластеров.

В качестве инструмента для предотвращения преждевременной остановки алгоритма вводятся квадратные матрицы $S^{(m)}$ размера k_m. Ненулевой элемент s_{ij} указывает на количество итераций алгоритма, в течение которых запрещено перекладывание из i-го в j-й кластер m-й группы. После переноса точки из i-го в j-й кластер обратная операция становится

запрещенной на ζ итераций. Указанная организация вычислительного процесса полностью соответствует концепции неявной памяти (*implicit memory*) алгоритма поиска с запретами. Значение $\zeta = 3$ обеспечило хороший баланс между качеством решения и скоростью сходимости алгоритма.

Алгоритм *Repair*. Восстановление допустимости решения

for $m = \overline{1, M}$ ***do***

$S_{k_m j} \leftarrow 0$

$n \leftarrow 0$

Repeat

$k^* \leftarrow \arg\max_{k=1,k_m} u_{mk}$

If $\exists j \in \Xi_{mk^*} : S_{k^* j} = 0$

$j^* \leftarrow \arg\min_{j \in \Xi_{mk^*} : S_{k^* j} = 0} u_{mj}$

Else

$j^* \leftarrow \arg\min_{j \in \Xi_{mk^*}} u_{mj}$

EndIf

$p^* \leftarrow \arg\min_{p \in Conv(V_{mk^*})} d(p, Conv(V_{mj^*}))$

$V_{mk^*} \leftarrow V_{mk^*} \setminus p^*$

$u_{mk^*} \leftarrow$ ***Infeasibility*** (V_{mk^*})

$V_{mj^*} \leftarrow V_{mj^*} \cup p^*$

Forall $k \neq k^*, j^*$

If $Conv(V_{mj^*}) \cap V_{mk} \neq \varnothing$

$V_{mj^*} \leftarrow V_{mj^*} \cup (Conv(V_{mj^*}) \cap V_{mk})$

$V_{mk} \leftarrow V_{mk} \setminus (Conv(V_{mj^*}) \cap V_{mk})$

$u_{mk} \leftarrow$ ***Infeasibility*** (V_{mk})

EndIf

EndFor

$u_{mj^*} \leftarrow$ ***Infeasibility*** (V_{mj^*})

$S_{ij} \leftarrow S_{ij} - 1, \quad \forall i, j : S_{ij} > 0$

$S_{j^* k^*} \leftarrow \zeta$

$n \leftarrow n + 1$

Until $(n > Maxiter) \vee \left(\max_{k=1,k_m} u_{mk} = 0 \right)$

endfor

return V_{mk} .

Критерием завершения рассматриваемого итерационного процесса является построение допустимого расписания агентов или установление факта невозможности построения указанного расписания за *Maxiter* итераций алгоритма.

Следует отметить, что метод не предполагает монотонного уменьшения интегральной меры недопустимости, заданной как $\max_{k=1,k_m} u_{mk}$, так и в виде лексикографически упорядоченной перестановки u_{mk}.

Выводы из данного исследования и перспективы дальнейших разведок в этом направлении. Сформулированная в [1] задача была декомпозирована с помощью иерархической последовательности подзадач: кластеризации множества торговых точек и определения зон ответственности агентов, построения расписания агентов по дням горизонта планирования, дневных маршрутов агентов, а также оптимальных маршрутов транспортных средств. Для каждой подзадачи предложен метод решения, что позволило найти приближенное решение всей задачи за приемлемое время.

Список использованной литературы:

1. Кузнецов К. А. Системный анализ и математическая модель работы дистрибьюторской компании / К. А. Кузнецов, В. А. Громов // Вестник Академии таможенной службы Украины. Серия : "Технические науки". – 2013. – № 2 (50). – С. 119–129.
2. Toth P. The vehicle routing problem / P. Toth, D. Vigo. – Philadelphia : SIAM, 2001. – 200 p.
3. Sevaux M. Hamiltonian paths in large clustered routing problems / M. Sevaux, K. Sörensen // In Proceedings of the EU/Meeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing, Troyes, France. – 2008. – № 1. – P. 11–12.
4. Battarra M. Exact Algorithms for the Clustered Vehicle Routing Problem / M. Battarra, G. Erdoğan, D. Vigo // Operation Research. – 2014. – Vol. 62. – № 1. – P. 58–71.
5. Yücenur G. N. A new geometric shape-based genetic clustering algorithm for the multi-depot vehicle routing problem / G. N. Yücenur, N. Ç. Demirel // Expert Systems with Applications. – 2011. – Vol. 38. – № 9. – P. 11859–11865.
6. Shin K. A Centroid-based Heuristic Algorithm for the Capacitated vehicle routing problem / K. Shin // Computing and Informatics. – 2011. – Vol. 30. – P. 721–732.
7. Dondo R. A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows / R. Dondo, J. Cerda // European Journal of Operational Research. – 2007. – Vol. 176. – P. 1478–1507.
8. Computational Geometry (2nd revised ed.) / M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. – N.-Y. : Springer-Verlag, 2000. – 230 p.
9. Focacci F. A hybrid exact algorithm for the TSPTW (Technical Report OR/01/2, D.E.I.S., University of Bologna) / Focacci F., Lodi A., Milano M. – Bologna : University of Bologna, 2001. – 250 p.
10. Glover F. Tabu Search / F. Glover, M. Laguna. – N.-Y. : Kluwer Academic Publishers, 1997. – 200 p.