Эвристика для решения задачи маршрутизации тягачей с прицепами, возникающей в реальной практике

Михаил Бацын,
Александр Пономаренко
Национальный исследовательский университет «Высшая школа экономики», Лаборатория алгоритмов и технологий анализа сетевых структур, ул. Родионова, 136, Нижний Новгород 603093, Россия
mbatsyn@hse.ru, aponom84@gmail.com

Аннотация

В работе предложена итеративная жадная эвристика для задачи маршрутизации тягачей с прицепами, возникающей в реальной практике доставки товаров в магазины. Рассматриваемая задача включает такие особенности, как: ограничения каждого магазина на автомобили, которые могут осуществлять доставку; автомобили разного формата и грузоподъемности; наличие прицепа у каждого автомобиля; возможность разбиения доставки для одного магазина на два и более автомобиля; мягкие и жесткие временные окна. Такая задача возникает при доставке товаров со склада в розничные магазины крупной компании. При этом на один склад приходится до 400 магазинов и 100 автомобилей. В основе нашей эвристики лежит последовательное добавление магазинов в маршрут жадным образом с последующим улучшением решения. Для вычислительных экспериментов использованы реальные данные.

1. Введение

Задачи транспортной логистики, возникающие в реальной практике, существенно отличаются от классических задач маршрутизации транспорта. В данной работе мы рассматриваем одну из таких реальных задач, возникающую при доставке товаров со склада в розничные магазины крупной ритейлинговой компании. Это задача маршрутизации тягачей с прицепами различного формата с ограничениями магазинов, мягкими и жесткими временными окнами и разбиением доставок. Задача состоит в минимизации общей стоимости доставки заказов всех магазинов.

В нашей задаче допустимое транспортное средство и время обслуживания зависят от магазина, которому требуется доставить товары. Некоторые магазины могут быть обслужены только малогабаритными автомобилями, некоторые – только тягачами без прицепа, некоторые – автомобилями с холодильным оборудованием и т. д. Кроме того, заказ магазина может состоять из двух частей: одной части, требующей холодильное оборудование, и другой – не требующей. Соответственно, парк автотранспорта разного формата, включает различные типы транспортных средств, а стоимость перевозки на каждом участке маршрута зависит от автомобиля и его состояния (проезжает ли автомобиль этот участок вместе с прицепом или без него). Каждый автомобиль также имеет фиксированную стоимость использования, учитываемую, если он используется для доставки. Каждый используемый автомобиль делает только один маршрут.

Магазины, к которым разрешен подъезд только тягачом без прицепа, мы будем называть магазины типа «одиночка», а магазины, к которым можно подъехать прицепом, – магазины типа «сцепка». Чтобы обслужить такой магазин автопоезд может оставить прицеп либо на специальной перегрузочной площадке, либо на разгрузку у магазина типа «сцепка». При этом тягач без прицепа может обслужить несколько магазинов, включая также магазины типа «сцепка», но затем он должен вернуться за своим прицепом. Такие круговые маршруты, совершаемые тягачом, мы будет называть маршрутами тягача. Если прицеп остается на разгрузке у магазина типа «сцепка», этот магазин выгружает свой заказ из прицепа, а тягач в это время обслуживает другие магазины. В этом случае может потребоваться либо перегруз товаров из прицепа в тягач, если в тягаче отсутствуют заказы магазинов, которые тягач должен объехать без прицепа, либо разгрузка товаров из тягача, если не все заказы магазина, у которого остается прицеп, есть в прицепе.

Жесткое временное окно определяется временем открытия и закрытия магазина. Жесткие временные окна не могут быть нарушены. Каждый магазин также имеет мягкое временное окно, определяющее предпочтительное время доставки. Мягкие временные окна

могут быть нарушены, но общее число таких нарушений ограничено. Для перегрузочных площадок задаются только жесткие временные окна. Если заказ магазина превышает объем самого крупного допустимого транспортного средства, то он разбивается на минимально возможное число транспортных средств (вынужденное разбиение). Меньший заказ может быть разделен между двумя транспортными средствами, но число таких «невынужденных» разбиений ограничено. При разбиении заказа только одна из доставок должна быть начата в пределах мягкого временного окна магазина, а все остальные должны удовлетворять только жесткому временному окну.

Одна из первых работ, посвященных задаче маршрутизации тягачей с прицепами, принадлежит Семету и Тайлларду (1993). Эти авторы предложили алгоритм табу-поиска для возникающей в реальной практике задачи маршрутизации тягачей с прицепами различного формата с ограничениями магазинов и с жесткими временными окнами. При этом в их задаче разбиение доставок не допускается, а прицеп может быть оставлен только на разгрузке у магазина типа «сцепка». Для похожей задачи, но без временных окон Семет (1995) разработал двухэтапную эвристику, которая разбивает все магазины на кластеры на первом этапе и строит оптимальный циклический маршрут в каждом кластере на втором этапе. Автор также представил модель целочисленного программирования для этой задачи. В работах Семета и Тайлларда (1993) и Дрэксла (2011) задача маршрутизации тягачей с прицепами рассматривается в формулировке очень близкой к нашей задаче. В частности, в их формулировках тягачи и прицепы могут иметь разную вместимость, стоимость доставки, время доставки, а прицепы строго привязаны к тягачам, так что один тягач не может ехать с прицепом от другого тягача. Задачи в такой формулировке могут быть названы как: Задача Маршрутизации Тягачей с Прицепами Различного Формата (ЗМТПРФ) и ЗМТПРФ с Временными Окнами (ЗМТПРФВО).

В ряде работ рассматривается задача маршрутизации тягачей с прицепами в более простой формулировке с однородным автопарком автомобилей, в котором все тягачи и прицепы идентичны. Задачи в такой формулировке обычно называются в литературе как: Задача Маршрутизации Тягачей с Прицепами (ЗМТП). Работы по ЗМТП принадлежат авторам Чао (2002), Шьюрер (2006), Лин и др. (2009, 2010, 2011), Вильегас и др. (2011а,б). Чао (2002) предложил алгоритм табу-поиска для ЗМТП. Шьюрер (2006) разработал две конструктивных эвристики и алгоритм табу-поиска для ЗМТП. Лин и др. (2009, 2010) предложили эффективный алгоритм имитации отжига для ЗМТП, а позднее Лин и др. (2011) применили аналогичную эвристику для ЗМТП с жесткими Временными Окнами (ЗМТПВО). Вильегас и др. (2011а) предложили гибридную эвристику GRASP/VNS для ЗМТП, а позднее Вильегас и др. (2011б) скомбинировали эту эвристику с математической формулировкой ЗМТП, основанной на разбиениях множеств.

Хофф (2006), Хофф и Локкетанген (2007) разработали алгоритм табу-поиска для задачи ЗМТПРФ с несколькими депо, возникающей при транспортировке молока. Карамиа и Гуэрриэро (2010а,б) предложили итеративную трехэтапную эвристику для аналогичной задачи ЗМТПРФ транспортировки молока, которая на первом этапе разбивает магазины на кластеры, на втором этапе строит оптимальные циклические маршруты в каждом кластере, и на третьем этапе применяет алгоритм локального поиска. Дрексл (2011) разработал модель целочисленного программирования и алгоритм ветвей и цен для задачи ЗМТПРФВО.

В данной статье мы предлагаем итеративную жадную эвристику для задачи ЗМТПРФ с ограничениями магазинов, с мягкими и жесткими временными окнами и с разбиением доставок. Насколько нам известно, в такой общей постановке данная задача не рассматривалась в литературе. Наша эвристика основана на жадном алгоритме добавления магазинов в маршрут, аналогичном жадным алгоритмам, предложенным Соломоном (1987). Дополнительно мы применяем простую процедуру улучшения решения, которая пытается сдвигать все доставки в текущем маршруте на более раннее время, чтобы избежать опозданий.

Рассматриваемая задача является обобщением NP-трудной Задачи Маршрутизации Транспорта с Временными Окнами (ЗМТВО). Доказательство NP-трудности ЗМТВО можно найти, например, в книге Тос и Виго (2002). Задача ЗМТВО получается из рассматриваемой нами задачи при следующих условиях:

1. Отсутствуют заказы, требующие холодильное оборудование.
2. Отсутствуют автомобили с прицепами и машины с холодильным оборудованием.
3. Все автомобили одинаковые.
4. Отсутствуют мягкие временные окна.
5. Время обслуживание магазина одинаково для всех автомобилей.
6. Стоимость доставки между пунктами одинакова для всех автомобилей.
7. Для всех магазинов запрещается разбивать доставку на два автомобиля.
8. Отсутствуют заказы, превышающие вместимость одного автомобиля.

Таким образом, рассматриваемая задача является NP-трудной, как обобщение NP-трудной задачи ЗТМВО.

Предложенная нами модель математического программирования для этой задачи приведена в приложении. Ее подробное описание не приводится в работе ввиду ограничений по объему.

В следующем разделе представлено описание алгоритма, а также его псевдокод. Вычислительные эксперименты на реальных данных, содержащих до 400 заказов и 100 автомобилей, приведены в разделе 3.

2. Описание алгоритма

В этом разделе дано описание предложенной итеративной жадной эвристики. В приведенном далее псев-

докоде используются следующие обозначения: n – число магазинов; f – целевая функция, или общая стоимость доставки; V – множество всех магазинов; K – множество всех автомобилей; R – текущий маршрут для одного автомобиля; S – текущее решение, заданное множеством маршрутов; q_i – объем заказов магазина i; f_k – фиксированная стоимость автомобиля k при его использовании; e_i – начало мягкого временного окна; l_i – конец мягкого временного окна; b_i^k – время начала разгрузки автомобиля k у магазина i; t_{ij}^{kl} – время поездки из пункта i в пункт j автомобилем k с прицепом (l = 1) или без него (l = 2); c_{ij}^{kl} – стоимость поездки из пункта i в пункт j автомобилем k с прицепом (l = 1) или без него (l = 2); 0 – идентификатор склада.

Основная процедура алгоритма достаточно проста. Мы строим множество решений жадным образом и выбираем лучшее из них по значению целевой функции f, т.е. решение с наименьшей общей стоимостью доставки. Сложность этой процедуры в N раз больше сложности процедуры BuildGreedySolution(), где N = *iterationsNumber* – это число итераций.

Алгоритм 1 Основная процедура

```
function Main(iterationsNumber)
output: решение S*
    f* = ∞
    for i=1 to iterationsNumber do
            S = BuildGreedySolution()
            if f(S) < f* then
               S* = S
               f* = f(S)
          endif
    endfor
    return S*
end function
```

Мы получаем новое решение путем последовательного построения маршрутов. В свою очередь каждый маршрут также получается последовательным добавлением магазинов. Первый магазин в маршруте выбирается случайным образом из μ самых «дорогих» магазинов, которые имеют не доставленные заказы. «Стоимость» магазина j измеряется как стоимость c_{0j}^{k1} прямой доставки непосредственно от склада. Значение μ в наших вычислительных экспериментах на реальных данных было выбрано эмпирически равным 5. Ограничения первого магазина определяют тип выбранного транспортного средства. Мы выбираем доступный автомобиль наибольшей вместимости, который может обслужить данный магазин. Например, если магазину необходимо доставить товары, требующие холодильное оборудование, мы выбираем самое большое свободное транспортное средство с холодильным оборудованием. Сложность этого алгоритма 2 определяется двумя вложенными циклами, которые добавляют магазины в маршруты до тех пор, пока все магазины не будут обслужены. Поскольку объем заказа любого магазина ограничен (исходя из реальных данных, заказ всегда умещается в три сцепки), то сложность алгоритма 2 равна $O(n)$ умножить на сложность процедуры InsertBestCustomer(), где n – это число магазинов.

Алгоритм 2 Построение жадного решения

```
function BuildGreedySolution()
output: решение S
    S = Ø
    ▷ U – множество магазинов, имеющих не обслуженные заказы
    U = V
    while U ≠ Ø do
        ▷ выбираем магазин случайно из μ наиболее «дорогих» не
        обслуженных магазинов
        customer = ChooseRandomly(U, μ)
        ▷ создаем новый маршрут с одним магазином
        R = (customer)
        k = getProperVehicle(customer)
        success = true
        while success do
            success = InsertBestCustomer(R)
        end while
        S = S ∪ R
    end while
    return S
end function
```

Функция InsertBestCustomer итеративно находит и вставляет в текущий маршрут R магазин с самым большим значением величины *totalCustomerGoodness*. Величина *totalCustomerGoodness* показывает, насколько выгодно добавление данного магазина в текущий маршрут. Построение нового маршрута останавливается, когда функция *InsertBestCustomer* возвращает значение *false*. Это происходит, когда либо ни один магазин не может быть добавлен в маршрут, либо добавление нового магазина становится невыгодно, потому что для любого из оставшихся магазинов оказывается дешевле доставить заказы отдельным автомобилем непосредственно от склада, чем «делать крюк» к нему в текущем маршруте. Так как в списке L магазинов, которые могут быть добавлены в текущий маршрут, могут быть все n магазинов, то сложность алгоритма 3 в $O(n)$ раз больше сложности процедуры calculateMinInsertionBadness().

Алгоритм 3 Вставка магазина с наибольшим значением *totalCustomerGoodness*

```
function InsertBestCustomer(R, k)
output: true, если добавлен новый магазин, false – иначе
    L ← список оставшихся магазинов, которые могут быть обслужены автомобилем k, выбранным для текущего маршрута R
    maxTotalCustomerGoodness = −∞
    foreach i ∈ L do
      goodness = λ·c_0i^k1
      if not setDemandToDeliver(R, i, goodness) then
```

```
    continue
  endif
  minInsertionBadness, position ←
    calculateMinInsertionBadness(R, i)
  ▷ Оценка прямого обслуживания из склада
  directCost = c_{0i}^{k1} + f_k / |R|
  if (directCost < minInsertionBadness) then
    continue ▷ этот магазин дешевле обслужить отдельным автомобилем
  endif
  totalCustomerGoodness ← goodness –
    minInsertionBadness
  if totalCustomerGoodness > maxTotalCustomerGoodness
    then
    maxTotalCustomerGoodness ← totalCustomerGoodness
    bestCustomer ← customer
    bestPosition ← position
  endif
 end foreach
 if (maxTotalCustomerGoodness = −∞ ) then
  return false
 endif
 вставить магазин bestCustomer на место bestPosition в маршруте R
end function
```

Как было описано выше, функция *InsertBestCustomer* выбирает, какой магазин лучше добавить в маршрут *R*. Мы храним список *L* магазинов с не доставленными заказами, которые могут быть обслужены автомобилем *k*, выбранным для текущего маршрута *R*. Для каждого магазина *i* из списка *L* мы решаем, какая часть от всего объема q_i его заказов должна быть доставлена в текущем маршруте. Это делает функция *setDemandToDeliver*. Здесь мы снова используем жадный подход. В случае, когда можно доставить весь объем заказов магазина, разумно это сделать. Если же в текущем маршруте транспортное средство может обслужить только часть этого объема, мы случайным образом решаем, выполнить частичную доставку с вероятностью π или не выполнять ее. Поскольку у нас есть ограничение на общее количество разбиений доставки, вероятность π зависит от оставшегося числа разрешенных разбиений и от ожидаемого числа разбиений. Если выбрано решение не разбивать доставку, функция *setDemandToDeliver* возвращает *false*.

Значение *totalCustomerGoodness* рассчитывается для каждого магазина, имеющего не доставленные заказы, которые могут быть полностью или частично доставлены в текущем маршруте. Величина *totalCustomerGoodness* рассчитывается как разность между фиксированным значением величины *goodness* и значением *minInsertionBadness*, которое динамически вычисляется для текущего магазина и маршрута. Величина *goodness* для магазина *j* прямо пропорциональна стоимости c_{0j}^{k1} прямой доставки непосредственно из склада.

Алгоритм 4 Выбор, какую часть объема заказов обслужить

```
function setDemandToDeliver(R, i, k)
  freeCapacity ← свободный объем в автомобиле k на маршруте R
  if объем товаров, который может быть доставлен автомобилем k в
  магазин i > объем не доставленных товаров магазина i then
    deliverTotalDemand()
  else
    doSplit ← true с вероятностью π = remainingSplitsNumber / expectedSplitsNumber
    if (doSplit) then
      доставить весь доступный объем товаров
    else
      return false;
    endif
  endif
  return true
end function
```

Алгоритм 5 описывает, как вычисляется значение *minInsertionBadness*. Величина *minInsertionBadness* показывает, как дорого обойдется добавление магазина в текущий маршрут. Мы разрешаем добавление магазина в любое место маршрута. Поэтому мы выбираем минимальное из всех возможных значений, которые соответствуют различным позициям в маршруте и различным вариантам добавления. В алгоритме 5 описывается самый простой вариант вставки магазина в маршрут. Другие, более сложные, варианты можно видеть на рисунках 1 и 2.

В случае, когда доставка началась после мягкого временного окна, мы измеряем опоздание *delay* = $\min_{k \in K} \max(b_j^k - l_j, 0)$ – разность между временем начала доставки и самым поздним предпочтительным временем доставки (концом мягкого временного окна). Поскольку для случая разбиения доставки только один автомобиль должен приехать в пределах мягкого временного окна, мы берем минимальную задержку среди всех автомобилей. Для всех возможных позиций в маршруте *R* мы пытаемся вставить магазин *i* и минимизировать сумму опозданий за счет сдвига (shiftEarlier) всех доставок в маршруте на более раннее время. Мы делаем этот сдвиг, так чтобы опоздания стали минимальными, но и сдвиг был как можно меньше, чтобы не увеличивать число нарушений мягких временных окон слишком сильно из-за ранних времен доставки. Нарушение времени начала мягкого временного окна (*earlierViolation*) измеряется как $\min_{k \in K} \max(e_j - b_j^k, 0)$.

Сложность процедуры shiftEarlier() равна $O(n)$, поскольку в худшем случае она сдвигает все доставки в маршруте, а маршрут потенциально может включать все *n* магазинов. Тогда сложность алгоритма 5 равна $O(n^3)$ за счет двух вложенных циклов, внутри которых вызывается процедура shiftEarlier(). Следовательно, сложность алгоритма 3 равна $O(n^4)$, алгоритма 2 – $O(n^5)$, а сложность предложенной эвристики – $O(Nn^5)$.

Мы вычисляем значение *insertionBadness* как сумму *costDelta* и *penaltyDelta*. Функция costDelta вычисляет

разность между стоимостью маршрута без магазина *i* и стоимостью маршрута с магазином *i*, добавленным на позицию *p*. Переменная *penaltyDelta* отражает величину нарушений мягких временных окон. Мы используем входной параметр *penalty*, чтобы задать, насколько важны нарушения мягких временных окон. Когда его значение равно бесконечности, никакие нарушения не возможны, а когда нулю – мягкие временные окна не будут учитываться. Значение штрафа выбирается эмпирически так, чтобы число нарушений в полученных жадным образом решениях не превышало заданное ограничение слишком часто.

Алгоритм 5 Вычисление минимального значения *badness* для всех позиций магазина в маршруте

function calculateMinInsertionBadness(*R, i*)
output: минимальное значение величины *badness*, наилучшая позиция для вставки магазина в маршрут
minInsertionBadness ← ∞
bestPosition ← 0
▷ текущий маршрут $R = (i_1,...,i_m)$
foreach $p \in \{2,3,...,m\}$ **do**
▷ вставить магазин *i* на позицию *p* в маршруте *R*
$R' \leftarrow (i_1,...,i_{p-1},i,i_p,...,i_m)$
laterViolationsDelta ← 0
foreach $j \in \{i,i_p,...,i_m\}$ **do**
if (delay(j, R') > 0)
earlierViolationsDelta ← shiftEarlier(j, R')
laterViolationsDelta ← *laterViolationsDelta* + delay(j, R');
endif
end foreach
penaltyDelta ← *penalty* * (*earlierViolationsDelta* + *laterViolationsDelta*)
insertionBadness ← costDelta(*i, p, R'*) + *penaltyDelta*
if (*insertionBadness* < *minInsertionBadness*) **then**
minInsertionBadness ← *insertionBadness*
bestPosition ← *p*
endif
end foreach
return *minInsertionBadness, bestPosition*
end function

Так как мы рассматриваем задачу маршрутизации транспорта с тягачами и прицепами, значение costDelta вычисляется не так легко. Добавление магазина в сложный маршрут с маршрутами тягача может быть иметь несколько вариантов. Есть два варианта вставки магазина (u) в маршрут тягача с прицепом между двумя магазинами типа «сцепка». Первый случай тривиален (рис. 1а-1): тягач с прицепом едет от магазина (i) к магазину (u) и затем к магазину (j). В этом случае costDelta рассчитывается как $costDelta = c_{iu}^{k1} + c_{uj}^{k1} - c_{ij}^{k1}$.

Во втором случае (рис. 1а-2) тягач оставляет прицеп на разгрузку у магазина (i), едет к магазину (u), и затем возвращается в магазин (i), чтобы забрать прицеп. После этого тягач с прицепом едет к магазину (j). В этом случае $costDelta = c_{iu}^{k2} + c_{ui}^{k2}$. Понятно, что первый случай лучше с точки зрения стоимости, но второй случай может оказаться лучше с точки зрения времени. Это потому что разгрузка у магазина (i) может занять длительное время, а во втором случае она выполняется параллельно с доставкой и разгрузкой товаров в магазине (u).

Магазин типа «одиночка» может быть вставлен в маршрут тягача с прицепом двумя способами (рис. 1б) . В первом случае автомобиль оставляет прицеп на разгрузку у магазина (i), едет без прицепа и обслуживает магазин (u), возвращается назад за прицепом в магазин (i) и после этого отправляется с прицепом в магазин (j) . В этом случае $costDelta = c_{iu}^{k2} + c_{ui}^{k2}$. Во втором случае автомобиль оставляет прицеп на доступной перегрузочной площадке (i') наиболее близкой к магазину (u) и отправляется в магазин типа «одиночка» (u), чтобы обслужить его. После разгрузки у магазина (u) тягач возвращается за прицепом на перегрузочную площадку (i') и едет с прицепом к магазину типа «сцепка» (j). Поэтому $costDelta = c_{ii'}^{k1} + c_{i'u}^{k2} + c_{ui'}^{k2} + c_{i'j}^{k1} - c_{ij}^{k1}$.

На рисунке 1г представлен случай, когда маршрут тягача «разрывается». В этом случае магазин типа «сцепка» (u) вставляется в маршрут тягача между магазинами (i) и (j). В первом случае (рис. 1г-1) автомобиль возвращается к магазину (i'), чтобы забрать прицеп после обслуживания магазина (i), затем едет с прицепом к магазину (u) и оставляет прицеп на разгрузке у него, едет без прицепа к магазинам от (j) до (j'), обслуживая их из тягача, и после магазина (j') возвращается к магазину (u), чтобы забрать прицеп. В этом случае:
$costDelta = c_{ii'}^{k2} + c_{i'u}^{k1} + c_{uj}^{k2} + c_{j'u}^{k2} + c_{uj''}^{k1} - c_{ij}^{k2} - c_{j'i'}^{k1} - c_{i'j''}^{k1}$.

Во втором случае (рис. 1г-2) магазин типа «сцепка» (u) вставляется непосредственно в маршрут тягача, и тягач обслуживает его без прицепа. В этом случае $costDelta = c_{iu}^{k2} + c_{uj}^{k2} - c_{ij}^{k2}$.

Есть два случая изменения пункта оставления прицепа (рис. 2). В первом случае (рис. 2а) магазин типа «сцепка» (u) вставляется после магазина типа «сцепка» (i), у которого был оставлен прицеп. Таким образом, мы можем поменять пункт оставления прицепа с (i) на (u). В этом случае $costDelta = c_{iu}^{k1} + c_{ui'}^{k2} + c_{ju}^{k2} + c_{uj'}^{k1} - c_{ii'}^{k2} - c_{ji}^{k2} - c_{ij'}^{k1}$. Во втором случае (рис. 2б) прицеп оставляется на разгрузке у добавленного магазина (u) вместо перегрузочной площадки (i). В этом случае $costDelta = c_{i''u}^{k1} + c_{ui'}^{k2} + c_{ju}^{k2} + c_{uj'}^{k1} - c_{i''i}^{k1} - c_{ii'}^{k2} - c_{ji}^{k2} - c_{ij'}^{k1}$.

3. Вычислительные эксперименты

Вычислительные эксперименты проводились для двух задач с разным количеством магазинов. В первой задаче небольшое число магазинов – 11. Список магазинов и их параметров представлен в таблице 2. Матрицы времен и затрат на доставку для автомобилей без

холодильного оборудования представлены в таблице 3 и таблице 4 соответственно.

Вторая задача основана на реальных данных заказов от 292 магазинов. В этой задаче общий объем заказов, требующих холодильное оборудование, составляет 490,5 поддономест, а общий объем заказов, не требующих холодильное оборудование, составляет 2293,5 поддономест. В обеих задачах используются два типа транспортных средств. Автомобили первого типа имеют холодильное оборудование и вместимость 19,5 поддономест в тягаче и 19 поддономест в прицепе. Автомобили второго типа – без холодильного оборудования и имеют вместимость 19 поддономест в тягаче и 20 поддономест в прицепе. Фиксированная стоимость использования автомобиля без холодильного оборудования составляет 2000, а фиксированная стоимость использования автомобиля с холодильным оборудованием – 2600. Автомобили с холодильным оборудованием имеют в 1,5 раза большие затраты на доставку, чем автомобили без него. В обеих задачах мы допускаем 20% нарушений мягких временных окон и 15% разбиений доставки. Результаты вычислительных экспериментов представлены в таблице 1.

Во второй строке таблицы представлены характеристики точного решения для первой задачи. Простая жадная эвристика не может найти точное решение даже для такой небольшой задачи. Решение, найденное этой эвристикой, приведено в третьей строке. Наш подход позволяет найти точное решение менее чем за 100 итераций (см. строку 4). Строки 5-9 показывают результаты, полученные для второй задачи, возникающей в реальной практике. Предложенный в работе алгоритм находит решение со стоимостью на 20% ниже, чем решение, найденное простой жадной эвристикой. Алгоритму требуется около одного часа вычислений, чтобы сделать 100 тысяч итераций и получить такое решение. Выполнение алгоритма на более чем 100 тысяч итераций не имеет большого смысла, потому что даже при 1 миллионе итераций решение улучшается незначительно по сравнению со 100 тысячами итераций.

Работа выполнена при поддержке Лаборатории алгоритмов и технологий анализа сетевых структур НИУ ВШЭ, грант правительства РФ дог. 11.G34.31.0057.

Список литературы

[1] Caramia M., Guerriero F., 2010a. A heuristic approach for the truck and trailer routing problem. Journal of the Operational Research Society 61, p. 1168–1180.

[2] Caramia M., Guerriero F., 2010b. A milk collection problem with incompatibility constraints. Interfaces 40(2), p. 130–143.

[3] Chao, I. M., 2002. A tabu search method for the truck and trailer routing problem. Computers & Operations Research 29(1), p. 33–51.

[4] Drexl, M., 2011. Branch-and-Price and Heuristic Column Generation for the Generalized Truck-and-Trailer Routing Problem. Journal of Quantitative Methods for Economics and Business Administration 12(1), p. 5-38.

[5] Hoff, A., 2006. Heuristics for rich vehicle routing problems. Ph.D. Thesis. Molde University College.

[6] Hoff, A., Lokketangen A., 2007. A tabu search approach for milk collection in western Norway using trucks and trailers. The sixth triennial symposium on transportation analysis TRISTAN VI. Phuket, Tailand.

[7] Lin, S.-W., Yu, V. F., Chou, S.-Y., 2009. Solving the truck and trailer routing problem based on a simulated annealing heuristic. Computers & Operations Research 36(5), p. 1683–1492.

[8] Lin, S.-W., Yu, V. F., Chou, S.-Y., 2010. A note on the truck and trailer routing problem. Expert Systems with Applications 37(1), p. 899–903.

[9] Lin, S.-W., Yu, V. F., Lu, C.-C., 2011. A simulated annealing heuristic for the truck and trailer routing problem with time windows. Expert Systems with Applications 38, p. 15244-15252.

[10] Scheuerer, S., 2006. A tabu search heuristic for the truck and trailer routing problem. Computers & Operations Research 33, p. 894–909.

[11] Semet, F., Taillard, E., 1993. Solving real-life vehicle routing problems efficiently using tabu search. Annals of Operations Research 41, p. 469–488.

[12] Semet, F., 1995. A two-phase algorithm for the partial accessibility constrained vehicle routing problem. Annals of Operations Research 61, p. 45–65.

[13] Solomon, M. M., 1987. Algorithms for the vehicle routing and scheduling problem with time window constraints. Operations Research 35, p. 254–265.

[14] P. Toth, D. Vigo. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, 2002, 367 p.

[15] Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., Velasco, N., 2011a. A GRASP with evolutionary path relinking for the truck and trailer routing problem. Computers & Operations Research 38(9), p. 1319-1334.

[16] Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., Velasco, N., 2011b. Heuristic column generation for the truck and trailer routing problem. International Conference on Industrial Engineering and Systems Management IESM'2011. Metz, France.

Приложения

Модель смешанного целочисленного линейного программирования

Параметры:

n, n_1, n_2, n_3, – число магазинов, магазинов типа «сцепка», типа «одиночка» и перегрузочных площадок

0 – депо (склад)

$V_1 = \{1,...,n_1\}$ – множество магазинов типа «сцепка»,

$V_2 = \{n_1+1,...,n_1+n_2\}$ – множество магазинов типа «одиночка»,

$V_3 = \{n+1,...,n+n_3\}$ – множество перегрузочных площадок,

$V = \{0\} \cup V_1 \cup V_2 \cup V_3$ – множество все пунктов,

K, K_1, K_2, K^* – множество всех автомобилей, с прицепом, без прицепа, с холодильным оборудованием
f_k – фиксированная стоимость использования автомобиля k,
Q_k, Q_k^1, Q_k^2 – полная вместимость автомобиля k, вместимость прицепа, вместимость тягача
q_i, q_i^* – весь заказ магазина i, его заказ, требующий холодильное оборудование
s_i^k – время выгрузки всего заказа магазина i для автомобиля k,
r_i^k – время на оставление прицепа и перегруз в пункте i для автомобиля k,
op_i, cl_i – время открытия и закрытия пункта i (жесткое временное окно),
er_i, lt_i – предпочтительный интервал доставки для магазина i (мягкое временное окно),
c_{ij}^{kl}, t_{ij}^{kl} – стоимость и время пути (i, j) для автомобиля k с прицепом ($l = 1$) или без ($l = 2$),
υ – максимальное число нарушений мягких временных окон,
σ – максимальное число разбиений доставки, исключая «вынужденные» разбиения

Переменные:

$x_{ij}^{kl} \in \{0,1\}$ – равно 1, если автомобиль k проходит ребро (i, j) с прицепом ($l = 1$) или без ($l = 2$),
$y_i^k \in [0,1]$ – часть заказа магазина i, доставленная автомобилем k,
$z_i^k \in \{0,1\}$ – равно 1, если автомобиль k доставляет часть заказа магазину i в своем маршруте,
$z_k \in \{0,1\}$ – равно 1, если автомобиль k используется в одном из маршрутов,
$u_i^k \in \{0,1\}$ – равно 1, если автомобиль k оставляет прицеп в пункте i,
$b_i^k \in R_+$ – время начала разгрузки у магазина i автомобилем k,
$e_i^k \in R_+$ – время конца разгрузки у магазина i автомобилем k,
$a_i^k \in R_+$ – время прибытия автомобиля k к магазину i,
$d_i^k \in R_+$ – объем заказа, доставленный тягачом автомобиля k в маршруте тягача, считая от первого магазина в этом маршруте и до магазина i, например, для маршрута $(j_0, j_1, ..., j_p, i, ..., j_0)$, в котором j_1 это первый магазин, обслуженный тягачом после магазина j_0, где оставлен прицеп, это объем будет равен: $d_i^k = \sum_{j=j_1,...,j_p,i} q_j y_j^k$
$v_i^k \in \{0,1\}$ – равно 1, если автомобиль k не был использован для доставки магазину i либо был, но начал разгрузку в i с нарушением его мягкого временного окна,
$v_i \in \{0,1\}$ – равно 1, если мягкое временное окно магазина i было нарушено,
$\sigma_i \in \{0,1\}$ – равно 1, если доставка магазину i разбита на два автомобиля, хотя весь его заказ может быть доставлен одним автомобилем,
$w_{k_1 k_2} \in \{0,1\}$ – равно 1, если доставка автомобилем k_2 части заказа магазину i завершилась до начала доставки автомобилем k_1 части заказа этому магазину.

Целевая функция:

$$f = \sum_{l=1}^{2} \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij}^{kl} x_{ij}^{kl} + \sum_{k \in K} f_k z_k \to \min \tag{1}$$

Ограничения по объему заказов и вместимости автомобилей:

$$\forall i \in V_1 \cup V_2 \quad \sum_{k \in K} y_i^k = 1 \tag{2}$$

$$\forall k \in K, \forall i \in V_1 \cup V_2 \quad z_i^k \geq y_i^k, z_k \geq y_i^k \tag{3}$$

$$\forall k \in K \quad \sum_{i \in V_1 \cup V_2} q_i y_i^k \leq Q_k \tag{4}$$

Ограничения сохранения потока:

$$\forall k \in K, \forall i \in V_2, \forall j \in V \quad x_{ij}^{k1} = x_{ji}^{k1} = 0 \tag{5}$$

$$\forall l \in \{1,2\}, \forall k \in K_l \quad \sum_{i \in V \setminus \{0\}} x_{0i}^{kl} \geq z_k \tag{6}$$

$$\forall k \in K, \forall i \in V \setminus \{0\} \quad \sum_{l=1}^{2} \sum_{j \in V} x_{ij}^{kl} \geq z_i^k \tag{7}$$

$$\forall l \in \{1,2\}, \forall k \in K, \forall i \in V \quad \sum_{j \in V} x_{ij}^{kl} = \sum_{j \in V} x_{ji}^{kl} \tag{8}$$

Ограничения жестких временных окон:

$$\forall k \in K \quad a_0^k = b_0^k = e_0^k = 0 \tag{9}$$

$$\forall k \in K, \forall i \in V \setminus \{0\} \quad b_i^k \geq op_i,\ a_i^k \geq op_i,\ e_i^k \leq cl_i,\ e_i^k \geq b_i^k + s_i^k y_i^k \tag{10}$$

$$\forall k \in K, \forall i \in V_1 \cup V_3 \cup \{0\}, \forall j \in V_1 \cup V_3 \quad b_j^k \geq e_i^k + t_{ij}^{k1} - M\left(1 - x_{ij}^{k1}\right) \tag{11}$$

$$\forall k \in K, \forall i \in V_1 \cup V_2, \forall j \in V_1 \cup V_2 \quad b_j^k \geq e_i^k + t_{ij}^{k2} - M\left(u_i^k + u_j^k + 1 - x_{ij}^{k2}\right) \tag{12}$$

$$\forall k \in K, \forall i \in V_1 \cup V_3 \cup \{0\}, \forall j \in V_1 \cup V_3 \quad a_j^k \geq e_i^k + t_{ij}^{k1} - M\left(u_i^k + 1 - u_j^k + 1 - x_{ij}^{k1}\right) \tag{13}$$

$$\forall k \in K, \forall i \in V_1 \cup V_2, \forall j \in V_1 \cup V_3 \quad e_j^k \geq e_i^k + t_{ij}^{k2} - M\left(u_i^k + 1 - u_j^k + 1 - x_{ij}^{k2}\right) \tag{14}$$

$$\forall k \in K, \forall i \in V_1 \cup V_3, \forall j \in V_1 \cup V_2 \quad b_j^k \geq a_i^k + r_i^k + t_{ij}^{k2} - M\left(1 - u_i^k + 1 - x_{ij}^{k2}\right) \tag{15}$$

Ограничения мягких временных окон:

$$\forall k \in K, \forall i \in V \setminus \{0\} \quad v_i^k \geq 1 - z_i^k, v_i^k \geq (er_i - b_i^k)/M, v_i^k \geq (b_i^k - lt_i)/M \tag{16}$$

$$\forall i \in V \setminus \{0\} \quad v_i \geq \sum_{k \in K} v_i^k + 1 - |K| \tag{17}$$

$$\sum_{i \in V \setminus \{0\}} v_i \leq \upsilon \tag{18}$$

Ограничения разбиения доставок по числу:

$$\forall i \in V_1 \cup V_2 \quad \left\lceil q_i / \max_{k \in K} Q_k \right\rceil \geq 2 \quad \Rightarrow \quad \sum_{k \in K} z_i^k = \left\lceil q_i / \max_{k \in K} Q_k \right\rceil \tag{19}$$

$$\forall i \in V_1 \cup V_2 \quad \left\lceil q_i / \max_{k \in K} Q_k \right\rceil = 1 \quad \Rightarrow \quad \sum_{k \in K} z_i^k \leq 2 \tag{20}$$

$$\forall i \in V_1 \cup V_2 \quad \sigma_i = \sum_{k \in K} z_i^k - \left\lceil q_i / \max_{k \in K} Q_k \right\rceil \tag{21}$$

$$\sum_{i \in V_1 \cup V_2} \sigma_i \leq \sigma \tag{22}$$

Ограничения разбиения доставок по времени:

$$\forall i \in V \setminus \{0\}, \forall k_1, k_2 \in K, k_2 > k_1 \quad w_{k_1 k_2} > (b_i^{k_1} - e_i^{k_2})/M - M(1 - z_i^{k_1} + 1 - z_i^{k_2}) \tag{23}$$

$$\forall i \in V \setminus \{0\}, \forall k_1, k_2 \in K, k_2 > k_1 \quad w_{k_1 k_2} \leq 1 + (b_i^{k_1} - e_i^{k_2})/M + M(1 - z_i^{k_1} + 1 - z_i^{k_2}) \tag{24}$$

$$\forall i \in V \setminus \{0\}, \forall k_1, k_2 \in K, k_2 > k_1 \quad e_i^{k_1} \leq b_i^{k_2} + M w_{k_1 k_2} + M(1 - z_i^{k_1} + 1 - z_i^{k_2}) \tag{25}$$

Ограничения вместимости для маршрутов тягача:

$$\forall k \in K, \forall i \in V \setminus \{0\}, \forall j \in V \setminus \{0\} \quad d_j^k \geq d_i^k + q_j y_j^k - M(u_j^k + 1 - x_{ij}^{k2}) \tag{26}$$

$$\forall k \in K, \forall j \in V_1 \cup V_3 \quad d_j^k \le Q_k^2 \tag{27}$$

Ограничения на заказы, требующие холодильное оборудование:

$$\forall i \in V_1 \cup V_2 \quad \sum_{k \in K^*} y_i^k \ge q_i^* / q_i \tag{28}$$

Вставка магазина в маршрут

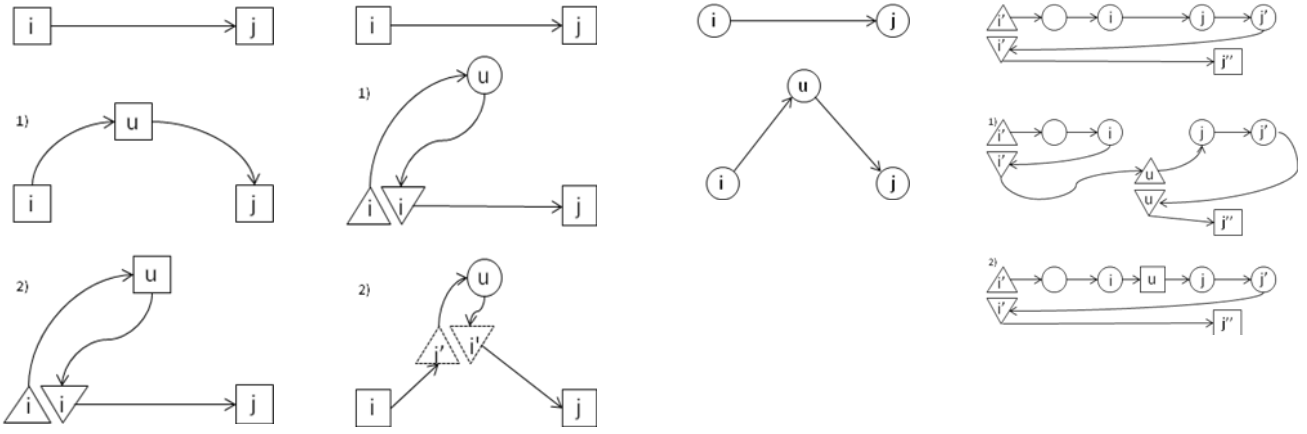

Рис. 1 (а) Вставка магазина типа «сцепка»

(б) Вставка магазина типа «одиночка» в маршрут тягача с прицепом

(в) Вставка магазина типа «одиночка» в маршрут тягача

(г) Вставка магазина типа «сцепка» в маршрут тягача

магазин типа «сцепка» магазин типа «одиночка» оставление прицепа на разгрузку у магазина типа «сцепка»

прицепление прицепа после разгрузки у магазина типа «сцепка» оставление прицепа на перегрузочной площадке

прицепление прицепа на перегрузочной площадке

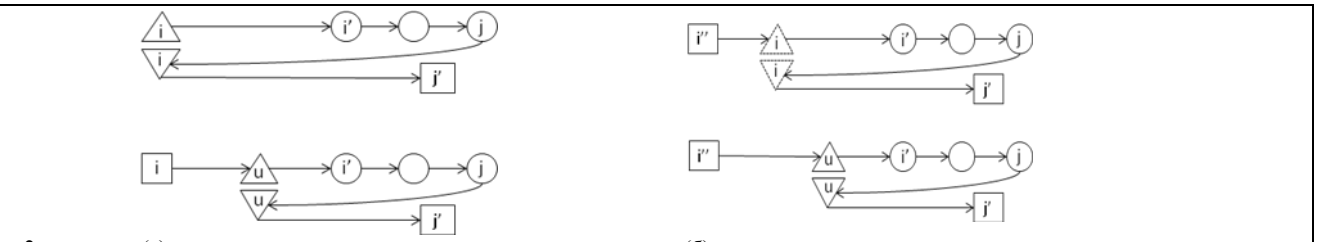

Рис. 2 (а) (б)

Таблица 1. Вычислительные эксперименты.

Задача	Общая стоимость, руб	Расстояние, км	км/подд ономе-ста	Число задействованных автомобилей без холодильного оборудования	Число задействованных автомобилей с холодильным оборудованием	Процент нарушений временных окон	Процент разбиений доставок
Задача 1 – точное решение	70035	3720	35.77	2	4	18	0
Задача 1 – решение простой жадной эвристики	102464	4342	51.75	4	5	18	0
Задача 1 – наш алгоритм, 100 итераций	70035	3720	35.77	2	4	18	0
Задача 2 – решение простой жадной эвристики	1217452	64907	24.72	61	51	20	11
Задача 2 – наш алгоритм, 1000 итераций	984599	52780	18.96	50	39	17	11
Задача 2 – наш алгоритм, 10000 итераций	974712	52325	18.79	49	39	18	8

Задача 2 – наш алгоритм, 100000 итераций	966671	51718	18.58	49	39	18	9
Задача 2 – наш алгоритм, 1000000 итераций	963242	51324	18.32	48	39	19	12

Таблица 2. Магазины.

Магазин	Объем заказа, требующего холодильное оборудование	Объем заказа, не требующего холодильное оборудование	Начало мягкого временного окна	Конец мягкого временного окна	Время открытия (жесткое временное окно)	Время закрытия (жесткое временное окно)	Магазин типа «одиночка»
1	0	10.5	15:00	16:00	10:00	21:00	да
2	0	10	11:30	12:30	10:00	22:30	да
3	1	4.5	15:00	16:00	10:00	21:30	да
4	0	9.5	10:00	11:00	9:30	21:00	нет
5	2	7	10:00	11:00	10:00	22:00	нет
6	1.5	7	10:00	11:00	10:00	21:30	нет
7	0	9	16:00	17:00	10:00	21:00	нет
8	6.5	9.5	11:00	12:00	10:00	22:00	да
9	2	7	12:30	13:30	10:00	22:00	да
10	0	7.5	17:30	18:30	10:00	22:00	да
11	2.5	7	18:00	19:00	9:30	20:00	нет

Таблица 3. Время доставки.

#	0	1	2	3	4	5	6	7	8	9	10	11
0	0	48	310	588	357	314	311	136	183	453	448	606
1	48	0	290	568	374	294	291	153	199	470	464	586
2	310	290	0	290	509	5	2	402	449	482	680	509
3	588	568	290	0	787	294	291	680	727	351	548	272
4	357	374	509	787	0	513	510	230	261	590	585	794
5	314	294	5	294	513	0	4	406	453	486	684	513
6	311	291	2	291	510	4	0	403	450	483	680	509
7	136	153	402	680	230	406	403	0	155	463	458	666
8	183	199	449	727	261	453	450	155	0	416	410	619
9	453	470	482	351	590	486	483	463	416	0	204	210
10	448	464	680	548	585	684	680	458	410	204	0	375
11	606	586	509	272	794	513	509	666	619	210	375	0

Таблица 4. Стоимость доставки.

#	0	1	2	3	4	5	6	7	8	9	10	11
0	0,00	188,90	3023,37	6218,58	3693,33	3038,81	3026,15	1225,67	1521,02	4604,38	4716,52	6389,66
1	188,90	0,00	2805,19	6000,40	3844,78	2820,62	2807,96	1377,13	1672,48	4755,84	4867,97	6171,48
2	3023,37	2805,19	0,00	3260,43	5540,30	20,46	7,80	4030,56	4325,91	5086,74	7100,10	5511,83
3	6218,58	6000,40	3260,43	0,00	8735,51	3275,86	3263,20	7225,77	7521,12	3568,39	5581,75	2593,15
4	3693,33	3844,78	5540,30	8735,51	0,00	5555,73	5543,07	2505,31	2533,46	6336,44	6448,57	8582,89
5	3038,81	2820,62	20,46	3275,86	5555,73	0,00	15,77	4045,99	4341,34	5102,17	7115,53	5527,27
6	3026,15	2807,96	7,80	3263,20	5543,07	15,77	0,00	4033,33	4328,68	5089,51	7102,87	5514,61
7	1225,67	1377,13	4030,56	7225,77	2505,31	4045,99	4033,33	0,00	1262,44	4789,63	4901,76	7036,08
8	1521,02	1672,48	4325,91	7521,12	2533,46	4341,34	4328,68	1262,44	0,00	3946,21	4058,34	6192,66
9	4604,38	4755,84	5086,74	3568,39	6336,44	5102,17	5089,51	4789,63	3946,21	0,00	2038,39	2271,47
10	4716,52	4867,97	7100,10	5581,75	6448,57	7115,53	7102,87	4901,76	4058,34	2038,39	0,00	3944,61
11	6389,66	6171,48	5511,83	2593,15	8582,89	5527,27	5514,61	7036,08	6192,66	2271,47	3944,61	0,00