Санкт-Петербургский государственный университет

Факультет прикладной математики – процессов управления

Кафедра математического моделирования энергетических систем

Широких Вячеслав Андреевич

Выпускная квалификационная работа

Динамическая адаптация эвристических алгоритмов в задачах маршрутизации транспорта

Заведующий кафедрой,
доктор физ.-мат. наук,
профессор Захаров В. В.

Научный руководитель,
доктор физ.-мат. наук,
профессор Захаров В. В.

Рецензент,
кандидат физ.-мат. наук Перцовский А. К.

Санкт-Петербург
2018

Содержание

Введение

Данная работа посвящена математическому исследования задач оптимизации вопросов логистики. Под логистикой обычно понимают *область деятельности предприятия, связанную с планированием, организацией, управлением и контролем движения материальных и информационных потоков в пространстве и во времени от их первичного источника до конечного потребителя, а также науку, изучающую эти процессы* [1].

Одной из основных целей оптимизации логистики является, наравне с максимизацией прибыли, также минимизация расходов, связанных с затратами на логистические операции. Наиболее общими операциями логистики здесь являются *планирование закупок, распределения и сбыта, организация складирования и управление запасами, планирование перевозок, планирование производства и управление производственными процессами* [1].

Фокусом данной работы являются задачи, связанные с планированием маршрутов, поскольку задачи данного класса не только актуальны в практике, но также и являются математически сложными – они принадлежат классу NP вычислительной сложности задач. Это означает, что доказательство оптимальности решения требует полного перебора решений, и в случае даже простейшей задачи маршрутизации n точек этим числом решений будет n!.

С ростом предприятий и рынков сбыта растёт и размерность задач, которые требуется разрешить. Фактически, это означает, что требования к вычислительным ресурсам для решения этой задачи традиционным точным методом растут более чем полиномиально с ростом размерности задачи, и уже сегодня их зачастую не хватает для нахождения оптимального решения этим способом. Таким образом, на практике становятся всё более популярными приближённые (эвристические) методы, а класс задач маршрутизации становится актуальным для теоретического исследования.

В реальности, к тому же, с построением маршрута связаны и многие другие факторы, из-за которых даже отдельный оптимальный маршрут может дать плохое решение для в целом, поскольку в современной практике реальные задачи обычно содержат различные дополнительные условия, ограничения и возможности. В связи с этим в теоретических исследованиях возникает необходимость рассмотрения более сложных задач. Можно выделить четыре основных класса математических моделей маршрутизации, исследуемых в настоящее время: задача коммивояжёра (travelling salesman problem), задача маршрутизации транспорта (vehicle routing problem), задача маршрутизации и управления запасом (inventory routing problem) и задача сбора и доставки (Pickup and Delivery Problems).

Вкладом данной работы являются разработка и формализация общего вида алгоритма Динамической адаптации эвристических алгоритмов и его апробация на различных классах задач маршрутизации. Дополнительно, в ходе работы предложен, сформулирован и исследован новый класс задач маршрутизации – кооперативные игры маршрутизации, являющийся расширением стандартных задач с добавлением возможности кооперации перевозчиков, что позволяет получать более эффективные решения по сравнению со стандартными случаями. Исследованы проблемы, возникающие при решении таких задач и методы их разрешения.

Данная работа включает следующие части. Во второй части представлен обзор литературы по теме актуальных моделей маршрутизации и представлены математические постановки стандартных задач. В третьей части представлена аналогичная работа по теме алгоритмов нахождения решений задач маршрутизации, а также представлена схема с подробным описанием основного алгоритма, использующегося в данной работе – адаптирующегося алгоритма поиска в большой окрестности (Adaptive Large Neighborhood Search, ALNS).

Четвёртая часть содержит описание проделанной работы над концепцией Динамической адаптации: основные идеи, метод применения, общая схема динамически адаптированного алгоритма и полученные результаты при применении к различным задачам. В пятой части предлагается концепция кооперативного расширения стандартных задач маршрутизации, рассматриваются проблемы таких постановок и представляются экспериментальные результаты улучшения качества решений и устойчивости коалиций в такой задаче.

Глава 1. Математические модели маршрутизации

В данной главе проводится наиболее популярных моделей маршрутизации, предоставляются их подробные математические постановки, а также приводится систематический разбор наиболее актуальных модификаций этих моделей в литературе.

1.1. Классическая задача коммивояжёра

Задача маршрутизации впервые предлагается как математическая проблема в первой половине 20 века. Первые известные исследования относятся к 1930 годам, и не имеют однозначного автора [2].

Первые математические публикации в данной области, посвящённые исследованию стандартной модели Задачи Коммивояжёра (Travelling Salesman Problem, TSP), появляются в середине 20 века и представлены, например, работами G.Dantzig, D.R.Fulkerson & S.M.Johnson 1954 года [3], или M.M.Flood 1956 года [4].

Одним из важнейших результатов исследования задачи является доказательство принадлежности задачи к классу NP-трудных в работе R.M.Karp 1972 года [5], что обосновывает сложность нахождения оптимального решения

на практике. Поскольку все дальнейшие развития модели маршрутизации являются либо обобщениями Задачи коммивояжёра, либо содержат её как подзадачу, этот результат так же позволят утверждать NP-трудность остальных моделей маршрутизации, рассматриваемых в данной работе.

В простейшей постановке, Задача Коммивояжёра состоит в нахождение последовательности посещения клиентов с минимальным пройденным путём. Формально, эту задачу можно определить несколькими различными способами.

1. Постановка в форме задачи на графе

Рассмотрим граф $G = (N, A)$, где $N = \{1,2, \dots, n\}$ – множество вершин графа (множества клиентов), $A = \{(i,j) \mid i,j \in N\}$ – множество рёбер графа – путей между клиентами. Для каждого ребра $a = (i,j) \in A$ считается заданной стоимость посещения c_a, что также можно интерпретировать как расстояние или время перемещения между вершинами. Целью задачи ставится нахождение *гамильтонова цикла* $(a_1, a_2, \dots, a_n)$, $a_i \in A$ минимальной стоимости, то есть замкнутого пути, проходящего через каждую вершину графа ровно по одному разу и при этом имеющего минимальную суммарную стоимость рёбер среди всех возможных гамильтоновых циклов:

$$\min \sum_{i=1}^{n} c_{a_i} \tag{1.1}$$

При этом, в задаче коммивояжёра на граф могут налагаться следующие условия:

- Ориентированный или неориентированный граф;
- Полностью связный граф: $\forall i,j \in N$: $(i,j) \in A$ (а также $(j,i) \in A$, в случае ориентированного графа);
- Симметричность матрицы стоимостей: $\forall (i,j), (j,i) \in A$: $c_{ij} = c_{ji}$.

2. *Постановка в форме задачи комбинаторной оптимизации*

Рассмотрим множество клиентов $N = \{1,2,\dots,n\}$. Решением задачи коммивояжёра в комбинаторной форме будет являться *перестановка* $p = (p_1, p_2, \dots, p_n)$ множества клиентов, то есть биекция N на себя: $p: N \to N$: 1) $\forall i,j \in N, i \neq j$: $p_i \neq p_j$ 2) $\forall k \in N\ \exists i \in N: p_i = k$. Для каждой пары клиентов (i,j), $i \in N$, $j \in N$ задана стоимость c_{ij} перемещения из i в j.

Целевой функцией в задаче является функция $f(p) = c_{p_1 p_2} + c_{p_2 p_3} + \cdots + c_{p_{n-1} p_n} + c_{p_n p_1}$. Целью является нахождение *оптимального решения* p^*, доставляющего минимум функции f на множестве P_n всех перестановок N:

$$p^*: f(p^*) = \min_{p \in P_n} f(p) \tag{1.2}$$

Одним из вариантов постановки в данном случае является *евклидова постановка*: каждому клиенту i ставится в соответствие пара координат на плоскости (x_i, y_i), и стоимостью перемещения из i в j считается *евклидово расстояние*: $c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

3. *Постановка в форме задачи целочисленного линейного программирования*

Для формализации задач маршрутизации зачастую используется форма линейного программирования.

Рассмотрим множество клиентов $N = \{1,2,\dots,n\}$ и матрицу стоимостей $C = \{c_{ij} \mid \forall i,j \in N\}$. Введём бинарные переменные $x_{ij} \in \{0,1\}$, которые равны 1, если маршрут проходит через ребро (i,j), и равны 0 иначе. Также введём вспомогательные целочисленные переменные $s_i \in \mathbb{Z}$ для каждого клиента i.

Тогда задача в форме целочисленного линейного программирования будет выглядеть следующим образом:

$$\min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij} : \tag{1.3}$$

$$\sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} x_{ji} = 1, \qquad \forall i \in N \tag{1.4}$$

$$s_i - s_j + n x_{ij} \leq n - 1, \qquad 2 \leq i \neq j \leq n \tag{1.5}$$

Ограничения (1.4) гарантируют, что каждый клиент посещён, причём только единожды. Ограничения (1.5) гарантируют, что все клиенты посещены одним маршрутом. Вспомогательные переменные s_i могут быть заданы как номера клиентов в последовательности посещения. Достаточность условий (1.5) доказывается тем фактом, что в противном случае, при наличии цикла $(i_1, i_2, \dots, i_k)$, не проходящего через клиента 1 группа ограничений вместе приведёт противоречию при суммировании:

$$\left(s_{i_1} - s_{i_2} + n\right) + \left(s_{i_2} - s_{i_3} + n\right) + \cdots + \left(s_{i_k} - s_{i_1} + n\right) = kn \leq k(n-1)$$

1.2. Задача маршрутизации транспорта

Естественным развитием задачи коммивояжёра является Задача маршрутизации транспорта (Vehicle Routing Problem, VRP), которую также иногда называют «задачей k-коммивояжёров». Задача была предложена в конце 50-х годов 20 века, одной из первых работ, рассматривающих эту задачу, является исследование G.Dantzig & J.Ramster 1959 года [6]. В 1964 выходит статья G.Clarke & J.W.Wright [7], описывающая простой и эффективный алгоритм решения задачи, ставший очень популярным в дальнейшем. На сегодняшний день, большинство исследований в области задач маршрутизации посвящено именно задаче маршрутизации транспорта, включая её различные модификации.

Задача маршрутизации транспорта является модификацией и расширением задачи коммивояжёра. Основными особенностями задачи являются 1) отдельная определённая точка местоположения депо и 2) возможность обслуживания клиентов несколькими транспортными средствами – несколькими маршрутами. Зачастую, в задаче определяют так же дополнительные ограничения на отдельный маршрут, таких как ограничение на количество посещённых клиентов, на длину маршрута (или, что аналогично, на время посещения), или же ограничение на вместимость транспортного средства. Далее мы рассмотрим задачу маршрутизации транспорта со вместимостью (Capacitated Vehicle Routing Problem).

Рассмотрим множество клиентов $N = \{1,2, \dots, n\}$ а также точку депо $\{0\}$. Обозначим общее множество точек $V = \{0\} \cup N$. Рассмотрим также матрицу стоимостей $C = \{c_{ij}\}, i \in V, j \in V$. Предположим, что доступны m идентичных транспортных средств, которые имеют ограниченную вместимость Q. Также предположим, что каждому клиенту $i \in N$ соответствует величина спроса d_i. Дополнительно предположим, что $\forall i\colon\ 0 < d_i \leq L$.

Введём переменные $x_{ij} \in \{0,1\}$, равные 1, если какой-то из маршрутов проходит по пути (i,j), и равные 0 иначе. Также, введём вещественные переменные $s_i \in \mathbb{R}$, выражающие суммарное количество доставленного груза в текущем маршруте, включая клиента i. Таким образом, задача в форме линейного программирования будет выглядеть следующим образом:

$$\min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij} : \tag{1.6}$$

$$\sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} x_{ji} = 1, \qquad \forall i \in N \tag{1.7}$$

$$\sum_{j=1}^{n} x_{0j} = \sum_{j=1}^{n} x_{j0} \leq m \tag{1.8}$$

$$s_i - s_j + Qx_{ij} \leq Q - d_j, \qquad \forall i, j \in N, i \neq j \tag{1.9}$$

$$d_i \leq s_i \leq Q, \qquad \forall i \in N \tag{1.10}$$

Целевая функция (1.6) отличается от задачи коммивояжёра наличием дополнительной точки – депо 0. Первая группа ограничений (1.7) аналогична ЗК и гарантирует посещение каждого клиента, и только один раз. Вторая группа неравенств (1.8) ограничивает количество покидающих и возвращающихся в депо транспортных средств. Третья группа ограничений (1.9) задаёт динамику доставки грузов: для участка (i, j) в маршруте $x_{ij} = 1$ и тогда $s_j \geq s_i + d_j$, в противном случае условие $s_i - s_j \leq Q - d_j$ всегда выполняется благодаря последней группе ограничений: $s_i \leq Q$ и $s_j \geq d_j$. Последняя группа ограничений (1.10) также гарантирует, что суммарный спрос, обслуживаемый транспортным средством не превышает его запаса: $s_i \leq Q$. Дополнительно, ограничения (1.9) неявным образом исключают возможность образования замкнутых циклов в маршрутах, без соединения с депо, аналогично неравенствам (1.5) задачи коммивояжёра.

1.3. Задача сбора и доставки

Принципиально новый класс Задач Сбора и Доставки (Pickup and Delivery Problem, PDP, также известная как Dial-a-Ride problem) возникает в 70-х годах 20 века, в таких работах, как N.H.M.Wilson 1971 года [8], или D.M.Stein 1978 года [9].

Отличительной чертой задач сбора и доставки, по сравнению с задачами маршрутизации транспорта, является то, что груз для доставки комплектуется не в депо отправления, а собирается в различных точках карты – задача

рассматривает уже не доставку груза отдельным клиентам, а обслуживание заказов в форме *{объём, точка сбора, точка доставки}*.

Формально, предположим, что вместо множества всех точек-клиентов N изначально задано *множество заказов* $R = \{(p_i, d_i)\}_{i=1}^{r}$. Далее, положим множество всех местоположений клиентов $N = \{p_i\}_{i=1}^{r} \cup \{d_i\}_{i=1}^{r}$, а также $n = 2r = |N|$. Дополнительно, считаем что задана общая точка отправления – депо $\{0\}$.

Аналогично задаче маршрутизации транспорта, считаем доступными m транспортных средств из множества $M = \{1,2, \dots, m\}$.

Специальные условия сбора и доставки, отличающие PDP от других задач маршрутизации, можно формализовать в следующих условиях:

- Обе точки из заказа (p, d) должны быть обслужены одним транспортным средством;
- Точка сбора p из заказа (p, d) должна быть посещена строго до посещения точки доставки d.

Каждое транспортное средство обладает вместимостью запаса Q единиц, которая не может быть превышена по ходу реализации маршрута. Будем считать, что транспортные средства начинают движение из депо с нулевым запасом.

Введём бинарные переменные $x_{ij}^{k} \in \{0,1\}$, равные 1, если маршрут транспортного средства k проходит по пути (i, j), и равные 0 в противном случае. Также, введём бинарные переменные $y_i^k \in \{0,1\}$, равные 1, если маршрут k проходит через точку i. Будем рассматривать вспомогательные переменные $s_i \in \mathbb{R}$, отражающие текущий объём запаса в текущем транспортном средстве после посещения точки i. Рассмотрим вспомогательные переменные $v_i \in \mathbb{R}$, отражающие суммарную стоимость текущего маршрута после посещения точки

i. Дополнительно, введём константу U, ограничивающую сверху максимально возможную стоимость маршрута. Такой константой, например, может являться $U = \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij}$.

В заданных переменных, задача сбора и доставки в форме целочисленного линейного программирования будет иметь вид:

$$\min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} \sum_{k=1}^{m} c_{ij} x_{ij}^{k} : \tag{1.11}$$

$$\sum_{k=1}^{m} y_i^k = 1, \qquad \forall i \in N \tag{1.12}$$

$$\sum_{j=1}^{n} x_{ij}^{k} = \sum_{j=1}^{n} x_{ji}^{k} = y_i^k, \qquad \forall i \in N, k \in M \tag{1.13}$$

$$\sum_{j=1}^{n} x_{0j}^{k} = \sum_{j=1}^{n} x_{j0}^{k} \leq 1, \qquad \forall k \in M \tag{1.14}$$

$$s_i - s_j + Q x_{ij}^{k} \leq Q - d_j, \qquad \forall i, j \in N, i \neq j, k \in M \tag{1.15}$$

$$0 \leq s_i \leq Q, \qquad \forall i \in N \tag{1.16}$$

$$v_i - v_j + U x_{ij}^{k} \leq U - c_{ij}, \qquad \forall i, j \in N, i \neq j, k \in M \tag{1.17}$$

$$v_p \leq v_d, \qquad \forall (p, d) \in R \tag{1.18}$$

$$y_p^k = y_d^k, \qquad \forall (p, d) \in R, k \in M \tag{1.19}$$

$$0 \leq v_i \leq U, \qquad \forall i \in N \tag{1.20}$$

Аналогично, (1.11) задаёт целевую функцию задачи, отражающую суммарную стоимость всех маршрутов. Ограничения (1.12) устанавливают возможность посещения каждой локации только одним транспортным средством. Соблюдение левых частей ограничений (1.13) и (1.14) гарантирует связность маршрутов. Правая часть ограничений (1.13) устанавливает связь между входящими и исходящими путями в каждой локации и фактом посещения этой локации конкретным транспортным средством. Правая часть (1.14) задаёт

ограничения на количество маршрутов: 0, если транспортное средство k не используется. Ограничения (1.15) задают динамику запаса при перевозках, а неравенства (1.16) – ограничения запаса транспортного средства, снизу и сверху. Отметим, что в отличие от VRP, неравенства (1.15) не могут быть использованы для исключения в маршрутах циклов, не проходящих через депо, поскольку в задаче PDP перевозимый запас изменяется немонотонно. Для того, чтобы исключить замкнутые циклы, вводятся дополнительные переменные v_i, выражающие монотонно изменяющийся показатель – накопленную маршрутом стоимость. Ограничения (1.17) задают динамику накопления стоимости маршрутами, а неравенства (1.20) – ограничения накопленной стоимости. Ограничения (1.18)-(1.19) гарантируют соблюдение условий сбора и доставки: (1.18) гарантируют правильный порядок посещения, а ограничения (1.19) – посещение локаций сбора и доставки одним транспортным средством.

1.4. Задача маршрутизации запасов

Альтернативным развитием задачи маршрутизации транспорта является Задачи Маршрутизации Запасов (Inventory Routing Problem, IRP). В различных вариациях задачи маршрутизации транспорта присутствует запас в том или ином виде, например, учёт спроса или ограничение вместимости транспортного средства, но качественным отличием новой задачи является учёт динамики запаса и стоимости хранения в течение длительного времени.

Задача маршрутизации запасов появляется в научной литературе в 80-х годах 20 века в практических исследованиях A.Assad, B.Golden, R.Dahl & M.Dror 1982 года [10], посвящённом разработке системы планирования дистрибьюции пропана, и W.J.Bell, L.M.Dalberto & M.L.Fisher 1983 года [11] посвящённом дистрибьюции промышленных газов.

Задача является комбинацией задачи маршрутизации транспорта и задачи управления запасами, таким образом решением задачи являются как маршруты

посещения клиентов, так и объёмы доставки. Сложность задачи состоит в том, что оба аспекта решения влияют друг на друга: большой объём доставки увеличивает стоимость хранения запасов клиентами, но в тоже время сокращает транспортные издержки благодаря редким посещениям; с другой стороны, сложный маршрут и частые посещения повышают транспортные издержки, но позволяют клиентам снизить объём хранения сверх спроса до минимума, существенно сокращая связанные с этим затраты. Именно поэтому данную задачу невозможно оптимально разбить только на составляющие её части маршрутизации и управления запасами, и только рассматривая совместную Задачу маршрутизации запасов возможно получение наиболее эффективных на практике решений.

В качестве базовой модели IRP будем рассматривать задачу построения маршрутов и планов доставки на T периодов времени из одного депо n потребителям с постоянным спросом, с использованием m транспортных средств. Целью оптимизации является минимизация суммарных затрат на перевозку и хранение.

Аналогично задаче VRP, будем рассматривать множество клиентов $N = \{1,2, \dots, n\}$ и также точку депо $\{0\}$. Обозначим общее множество точек $V = \{0\} \cup N$. Так же, рассмотрим матрицу стоимостей $C = \{c_{ij}\}, i \in V, j \in V$.

Рассмотрим горизонт планирования, состоящий из нескольких периодов времени $t = 1,2, \dots, T$.

Предполагаем, что каждый клиент i имеет возможность хранить товар, затрачивая при этом h_i стоимости за каждую единицу товара (что может выражать единицу товара, единицу объёма и т.п.). Максимальная вместимость склада клиента i считается равной U_i. Спрос на товары для каждого клиента, аналогично ЗМТ, предполагается известным и постоянным, равным d_i.

Дополнительно, вводим новые переменные, связанные с управлением запасами: $q_{it} \geq 0$, выражающие величину доставки клиенту i в начале периода t, а также $I_{it} \geq 0$, выражающие размер запаса клиента i на начало периода t после доставки. Предполагаем также, что величины начального запаса I_{i0} каждого клиента заданы и известны изначально.

Для формулировки задачи маршрутизации запасов необходимо также расширить переменные маршрутов: $x_{ijt} \in \{0,1\}$, равные 1, если какой-то из маршрутов проходит по пути (i,j) в период t, и равные 0 иначе; $s_{it} \in \mathbb{R}$, выражающие суммарное количество доставленного груза в начале периода t в текущем маршруте, включая клиента i.

Таким образом, задача маршрутизации запасов в форме линейного программирования будет выглядеть следующим образом:

$$\min\left(\sum_{t=1}^{T}\sum_{i=0}^{n}\sum_{j\neq i,j=0}^{n} c_{ij}x_{ijt} + \sum_{t=1}^{T}\sum_{i=1}^{n} h_i I_{it}\right): \tag{1.21}$$

$$\sum_{j=1}^{n} x_{ijt} = \sum_{j=1}^{n} x_{jit} \leq 1, \qquad \forall i \in N, \forall t = 1, \dots, T \tag{1.22}$$

$$\sum_{j=1}^{n} x_{0jt} = \sum_{j=1}^{n} x_{j0t} \leq m, \qquad \forall t = 1, \dots, T \tag{1.23}$$

$$I_{it} = I_{i,t-1} - d_i + q_{i,t}, \qquad \forall i \in N, \forall t = 1, \dots, T \tag{1.24}$$

$$0 \leq I_{it} \leq U_i, \qquad \forall i \in N, \forall t = 1, \dots, T \tag{1.25}$$

$$s_{it} - s_{jt} + Qx_{ijt} \leq Q - q_{jt}, \qquad \forall i, j \in N, i \neq j, \forall t = 1, \dots, T \tag{1.26}$$

$$q_{it} \leq s_{it} \leq Q, \qquad \forall i \in N, \forall t = 1, \dots, T \tag{1.27}$$

В целевой функции (1.21) по сравнению с VRP появляется новое слагаемое – затраты на хранение. Первая группа ограничений (1.22) гарантирует связность маршрута, причём посещение клиента в каждый период не гарантируется, если

клиент способен удовлетворить спрос при помощи имеющегося запаса. Вторая группа неравенств (1.23) ограничивает количество покидающих и возвращающихся транспортных средств в депо в каждом периоде. Третья группа ограничений (1.24) задаёт динамику запаса: начальный запас после доставки в каждом периоде составляют начальный запас в начале предыдущего периода, за исключением израсходованного спроса, с добавкой величины запаса, доставленного в начале текущего периода. Следующая группа неравенств (1.25) гарантирует отсутствие дефицита и ограничение вместимости склада каждого клиента. Последние ограничения (1.26) аналогичны постановке ЗМТ, за исключением того, что объемы доставки – q_{it} – переменные величины, а также дополнительного расширения количества ограничений на каждый период времени. Аналогично ЗМТ, можно показать, что ограничения (1.26)-(1.27) не позволяют нарушение вместимости транспортных средств. Для маршрута $(i_1, i_2, \dots, i_k)$: $x_{i_1 i_2 t} = 1$ и неравенства (1.26) принимают вид $q_{i_2 t} \leq s_{i_2 t} - s_{i_1 t}$. Суммируя данные неравенства для $q_{i_2 t}, q_{i_3 t}, \dots, q_{i_k t}$, и используя (1.27) получаем:

$$\sum_{p=2}^{k} q_{i_p t} \leq s_{i_k t} - s_{i_{k-1} t} + \cdots + s_{i_2 t} - s_{i_1 t} = s_{i_k t} - s_{i_1 t} \leq Q - q_{i_1 t}$$

или же:

$$\sum_{p=1}^{k} q_{i_p t} \leq Q$$

1.5. Актуальные модификации задач маршрутизации

Различные модификации, добавляющие реалистичные характеристики в теоретические задачи маршрутизации, являются одним из самых актуальных направлений исследования в научной литературе.

Наиболее популярными такими характеристиками являются временные окна посещения клиентов, учёт пробок, недетерминированные модели спроса, дополнительные цели оптимизации, такие как минимизация числа маршрутов или поддержание экологии. Все упомянутые ограничения являются универсальными и могут быть применены в любой модели маршрутизации.

Временные окна посещения клиентов [12, 13, 14, 15] накладывают ограничения на время посещения локации в форме (a_i, b_i). Введение данных ограничений необходимо для учёта часов работы, поддержания максимального времени доставки и согласования с расписаниями клиентов в общем случае. Для введения временных окон в модель необходимо введение дополнительных переменных, аналогично ограничениям (2.17), и самих ограничивающих неравенств:

$$t_i - t_j + T_{max} x_{ij} \leq T_{max} - \tau_{ij}, \qquad \forall i, j \in N, i \neq j \tag{1.28}$$

$$a_i \leq t_i \leq b_i, \qquad \forall i \in N \tag{1.29}$$

где τ_{ij} – время в пути между i и j, T_{max} – конечное время планирования.

Одной из новых и актуальных модификаций является введение зависимости различных стандартных констант от реального времени посещения [16, 17, 18, 19]. Основным практическим применением данной идеи является учёт пробок на дорогах во время перевозок. В математической модели это означает, что константы c_{ij} становятся функциями, в общем случае непрерывного времени: $c_{ij}(t)$. Важной особенностью является то, что использование подобных функций не позволяет рассматривать постановки задач как задачи целочисленного линейного программирования, а также отдельных базовых предположений, используемых в эвристических алгоритмах, таких как правило треугольника для матрицы времени или линейность сдвига во времени.

В зависимости от используемой модели спроса, задачи можно разделить на три группы. Первая группа – это задачи с *детерминированным* спросом, то есть такие, в которых объём спроса на продукцию каждого клиента известен заранее и не изменятся в периоде планирования. К таким задачам, в том числе, относятся рассмотренные базовые модели CVRP, PDP и IRP. Второй группой задач являются задачи, в которых спрос является *стохастическим* [20, 21, 22]. В таких моделях обычно подразумевается, что для величин спроса нет известного заранее точного значения, но известно вероятностное распределение спроса клиентов или отдельные его параметры. В третью группу задач входят модели с *динамическим* спросом [23, 24, 25, 26] – в такой постановке считается, что как величина спроса, так и сам факт заказа доставки не известен заранее, либо они известны только на небольшое время вперёд.

Одной из часто встречающихся модификаций в различных задачах является введение дополнительной приоритетной цели оптимизации – минимизации числа маршрутов, то есть минимизация числа необходимых транспортных средств [27, 28]. Стандартная целевая функция минимизации транспортных затрат в таком случае считается второстепенной и менее приоритетной. Основания данного подхода лежат в том факте, что на практике логистические затраты на доставку включают в себя не только переменные затраты, зависящие от расстояния или времени, но и постоянные затраты на каждое транспортное средство, такие как стоимость обслуживания, зарплата водителям и т.д. Важно отметить, что при подобном подходе минимизация переменной стоимости остаётся актуальной, т.к. при эффективном сокращении длины маршрутов удаётся компактнее распределять заказы на меньшее число транспортных средств.

Наконец, весьма актуальными в последние годы являются задачи с дополнительной минимизацией влияния транспорта на окружающую среду [29,

30, 31]. Основными методами такой оптимизации являются дополнительный учёт суммарного времени как передвижения, так и простоя, а также добавления специфических ограничений, отражающих экологические нормы.

Для дополнительной наглядности, все работы, связанные с представленными модификациями, представлены ниже в Таблице 1 для каждого класса задач маршрутизации.

Таблица 1. Литература по модификациям Задач Маршрутизации

Модель → / Модификация ↓	TSP	VRP	PDP	IRP
Несколько маршрутов	VRP	+	*	*
Учёт объёмов доставки	–	CVRP	+	+
Несколько периодов	–	Periodic VRP	–	+
Заказы сбора и доставки	–	PDP	+	–
Временные окна	TSPTW[12]	VRPTW[13]	PDPTW[14]	*IRPTW[15]*
Зависимость от времени	TDTSP[16]	TDVRP[17]	*TDPDP[18]*	*TDIRP[19]*
Стохастический спрос	–	SVRP[20]	*SPDP[21]*	SIRP[22]
Динамический спрос	DTSP[23]	DVRP[24]	DPDP[25]	DIRP[26]
Минимизация маршрутов	–	[27]	[28]	–
Оптимизация экологии	–	GVRP[29]	GPDP[30]	GIRP[31]

«–» – задача никогда не формулируется с данной модификацией;
«+» – задача всегда формулируется с данной модификацией;
«*» – задача зачастую формулируется с несколькими транспортными средствами, но такая формулировка не обязательна;
Курсив – задача крайне редко или неполноценно освещена в литературе.

Глава 2. Алгоритмы решения задач маршрутизации

2.1. Обзор и классификация актуальных алгоритмов

Проблема решения задач маршрутизации заключается в том, что задачи данного класса являются вычислительно NP-трудными, т.е. не существует алгоритма нахождения оптимального решения не более чем за полиномиальное время. В связи с этим, не существует единственного наиболее эффективного подхода к решению задач маршрутизации. За годы, в исследованиях было рассмотрено множество различных алгоритмов решения, и новые продолжают рассматриваться до сих пор.

В общем, алгоритмы решения можно разделить на две категории: точные методы, гарантированно приходящие к точному решения, но не ограниченные по времени вычисления, и, наконец, эвристические методы, которые способны находить качественное решение эффективно по времени, но не гарантирующие оптимальности этого решения.

Сама задача маршрутизации представляет собой задачу комбинаторной оптимизации. Не существует аналитического метода решения задачи, поэтому для нахождения точного решения используются различные универсальные методы комбинаторной оптимизации.

Наиболее популярными среди используемых точных методов являются методы ветвей и границ, включая метод ветвей и сечений (branch and cut) [32], метод ветвей, оценок и сечений (branch, price and cut) [33]. Так же популярными методами являются методы релаксации [11, 17].

Поскольку точные методы не имеют ограничений по времени вычисления, то в худшем случае это время будет эквивалентно n! от размерности n входных данных. При достаточно большом n, начиная уже с нескольких десятков, задачу

полного перебора n! решений невозможно решить за приемлемое время, даже на многопроцессорных суперкомпьютерах. На практике, правильно построенный метод может сократить пространство поиска во много раз, но, тем не менее, не понизить вычислительную сложность.

Фактические ограничения на поиск точного решения зависят от исследуемой вариации задачи и от используемого метода. Так, наибольшая по размеру ЗК содержит 85900 узлов и была решена с помощью параллельных вычислений методом Concorde за 1,5 года [34]. Вместе с этим, представлены и более практические ограничения: за 1000 секунд наибольшая задача содержит 2392 точки [35]. С другой стороны, при исследовании более сложной и актуальной задачи маршрутизации запаса, максимальный размер задачи, для которой было найдено точное решение за ограниченное время в 7200 секунд составляет 50 точек с 3 периодами планирования и 30 точек с 6 шестью периодами планирования [33].

Альтернативным методом использования точных методов является использование полученной нижней границы как приближённого решения, что позволяет получать решения на бóльших задачах в пределах 1-5% оптимального [33].

В связи с этими ограничениями, особенно на более сложных задачах, возникает необходимость в более эффективных методах нахождения решения. Такой альтернативой точным методам являются эвристики, которые позволяют быстро находить решение, не гарантированно оптимальное, но как минимум близкое к нему, при правильно построенном алгоритме.

В литературе, эвристические методы пользуются гораздо бóльшей популярностью, нежели точные методы, и за годы исследования было предложено и рассмотрено множество разнообразных алгоритмов. В связи с

существенным преимуществом в вычислительной эффективности по сравнению с точными методами, а также в связи с актуальностью исследования, данная работа использует эвристически для вычисления решений исследуемых задач.

Условно, эвристические алгоритмы можно разделить на алгоритмы построения [13, 17], которые строят решение за одну итерацию, алгоритмы локального поиска [13], находящие лучшее решение в локальной окрестности, и алгоритмы глобального поиска, итеративно приближающие решение к оптимальному. Последние, в свою очередь, можно разделить по используемому подходу, основными из которых являются методы поиска в большой окрестности [15, 28, 36, 38], методы поиска с исключениями [37], а также генетические и популяционные методы [19, 23, 27].

Современные эффективные эвристики имеют тенденцию сочетать три различных подхода в одной схеме алгоритма, как, например, в работах [27, 36, 37, 38]. Так, эвристики построения используются для получения начального решения, причём зачастую качество начального решения имеет сильное влияние на окончательный результат: чем оно ближе к оптимальному, тем быстрее будет глобальный поиск и тем лучше будет результат поиска за ограниченное время. Алгоритмы локального поиска могут быть использованы для обобщения поиска, если глобальный алгоритм использует точечные переходы от одного решения к другому: локальная оптимизация позволяет рассматривать окрестность целиком, а не одну новую точку. При всём этом, алгоритм глобального поиска задаёт «направление» поиска в глобальном пространстве решений.

В данной работе для нахождения решений различных задач используется концепция Адаптирующегося поиска в большой окрестности (Adaptive large neighborhood search, ALNS), которая была адаптирована в различных задачах, включая задачу маршрутизации транспорта [38], задачу маршрутизации запаса [36] и задачу сбора и доставки [28]. Основанием для выбора данного подхода

является значительная гибкость метода для адаптации к различным задачам маршрутизации, по сравнению с популяционными методами, а также экспериментально показанная существенная выгода в скорости вычисления, по сравнению с методами поиска с исключениями.

2.2. Adaptive Large Neighborhood Search

Подробное описание алгоритма можно найти в работах [28, 36, 38].

Универсальная схема алгоритма содержит две основных составляющих: адаптирующийся рандомизированный поиск и моделирование отжига.

Под окрестностью решения понимаются все возможные решения, которые можно получить из текущего решения с помощью заданного набора процедур модификации/изменения. Метод ALNS предполагает наличие большого количества нетривиальных процедур, что создаёт довольно большую окрестность для поиска.

С одной стороны, в ALNS направление поиска, то есть выбор процедуры модификации, осуществляется случайным образом, так как в противном случае такой поиск на каждой итерации занимал бы слишком много времени. С другой стороны, вероятность выбора конкретной процедуры изменяется динамически, в зависимости от эффективности её применения, что позволяет алгоритму «запоминать» выгодные направления в зависимости в каждой конкретной решаемой задаче.

Конкретный набор процедур зависит от типа решаемой задачи и может варьироваться. Обобщённо, они обычно включают процедуры исключения, добавления или перемещения точек в решении различно организованными группами, такими, например, как группы случайно выбранных точек, группы точек с наилучшим улучшением/наименьшим ухудшением, или локальных кластеров точек.

Метод моделируемого отжига применяется для того, что обходить так называемые «локальные минимумы» в пространстве решений – решения, оптимальные лишь в небольшой, локальной окрестности. Идея метода состоит в том, чтобы дать возможность продолжать поиск в некотором направлении, даже если поначалу это направление ведёт к ухудшению решения. Фактически, алгоритм симулирует физические процессы охлаждения и кристаллизации: начальная «температура решения» τ постепенно понижается в ходе вычисления, и вместе с ней понижается вероятность перехода к худшему решению, аналогично вероятности атома занять ту или иную позицию в кристаллической решётке. Эта вероятность также зависит и от удалённости решения от текущего, так что при большом удалении, вероятность перехода будет меньше. Обычно, для вычисления этой вероятность используются формулы следующего вида:

Для текущего решения s и нового решения s', если $f(s) < f(s')$, то есть s' хуже s, вероятность перехода к s' будет равна $p = \exp\left(\frac{f(s)-f(s')}{\tau}\right)$. Как можно заметить, степень экспоненты будет отрицательной, то есть при приближении сверху $f(s')$ к $f(s)$, вероятность будет стремиться к единице, а при удалении – к нулю. Аналогично, при понижении температуры τ, вероятность перехода к решениям на одинаковом заданном расстоянии будет понижаться.

Алгоритм ALNS также подразумевает использование некоторой дополнительной эвристики построения для получения начального решения, а также некоторой процедуры локального поиска, для периодического улучшения новых решений в процессе вычисления.

Формальная схема алгоритма в универсальном виде на естественном языке представлена далее, в форме Алгоритм 1. Процедура $InitialSolution(\cdot)$ – некоторая эвристика построения начального решения. $Update_i(\cdot)$ – процедуры модификации, формирующие большую окрестность решения. Значения $w_i,\ c_i,\ n_i$ – соответствующие каждой процедуре i вес, или доля в вероятностном

распределении, очки улучшения, повышающиеся при нахождении нового решения s или s_{best} и счётчик использований соответственно. $LocalOpt(\cdot)$ – некоторый *быстрый* алгоритм локальной оптимизации.

Алгоритм 1: Adaptive Large Neighborhood Search

$s_{best} = s = InitialSolution(\cdot)$
$w_i = 1,\ c_i = 0,\ n_i = 0,$ для каждой процедуры i; $\tau = \tau_{start}$
Пока $\tau > \tau_{min}$ повторять:
    $s' = s$
    Случайный выбор процедуры k с учётом текущих весов w_i
    $n_k = n_k + 1$
    Применение процедуры k к s': $s' = Update_k(s')$
    Если $f(s') < f(s)$:
        $s = s'$
        Если $f(s) < f(s_{best})$:
            $s_{best} = LocalOpt(s)$
            $c_k = c_k + \sigma_1$
        Иначе :
            $c_n = c_n + \sigma_2$
    Иначе, если $p = random(0; 1) \le exp\left(\frac{f(s) - f(s')}{\tau}\right)$:
        $s = s'$
        $c_n = c_n + \sigma_3$
    $\tau = \varphi * \tau$
    Если число выполненных итераций кратно T :
        Обновить веса: если $n_i > 0,\ w_i = (1 - \theta)w_i + \theta\frac{c_i}{n_i}$
        Обнулить счётчики: $c_i = 0,\ n_i = 0$

Параметрами алгоритма являются начальная и минимальная температуры τ_{start} и τ_{min}, скорость «охлаждения» $0 < \varphi < 1$, очки успеха σ_1, σ_2 и σ_3, коэффициент реакции θ, а также период обновления T. Ориентировочно, эти

параметры могут иметь следующие значения: $\tau_{start} = 30000,\ \tau_{min} = 0.01,\ \varphi = 0{,}9994,\ \sigma_1 = 10,\ \sigma_2 = 5,\ \sigma_3 = 2,\ \theta = 0.3,\ T = 200,$ впрочем возможна и специальная настройка этих значений для эффективного решения однотипных задач некоторой группы.

Глава 3. Динамическая адаптация алгоритмов

3.1. Введение в динамическую адаптацию

Принцип оптимальности сформулирован Беллманом в 1957 году [39]. Согласно его определению, «оптимальное решение обладает таким свойством, что независимо от начальных данных и начальной точки, оставшееся решение будет оптимальным, с учетом реализации первой точки». Таким образом, оптимальное решение сохраняет свойство оптимальности на любых подзадачах.

Эвристические алгоритмы не гарантируют оптимальности решения, и, если полученное решение не оптимально, используя принцип оптимальности, можно заключить, что существует подзадача исходной задача, такая что соответствующее частичное решение также будет не оптимальным.

Данный принцип лежит в основе предлагаемых далее идей устойчивости решений во времени и метода динамической адаптации алгоритмов.

Так, выявление не-оптимальной подзадачи даёт возможность дополнительно улучшить решение с меньшими затратами вычислительных ресурсов и времени, поскольку подзадача по своей сути имеет меньшую размерность, чем исходная полная задача. Далее предлагается метод оптимизации, основанный на выявлении и улучшении частичных решений, названный *Динамической адаптацией* алгоритма.

С другой стороны, исследуя подзадачи, возникающие в реальном времени при реализации решения, можно вывести экспериментальный критерий

возможности их улучшения, названный далее *уровнем состоятельности во времени*.

3.2. Оценка состоятельности во времени

Рассмотрим следующую схему вычислительных экспериментов.

Пусть дано множество тестовых примеров P для некоторой модели задачи маршрутизации. Для каждого примера $p \in P$ генерируем с помощью алгоритма множество $N(p)$ различных решений $s(p) \in N(p)$.

Пусть задан некоторый метод разбиения решения $s(p)$ на последовательность сегментов так, что это соответствует реальному ходу времени. Положим номера этих сегментов $k = 1, \dots, K(s(p))$. Обозначим за $s(k, p)$ оставшуюся последовательность сегментов решения после шага k, то есть совокупность сегментов $\left(k+1, k+2, \dots, K\left(s(p)\right)\right)$. Положим, что на шаге k решение находится в момент времени $t(k)$. На шаге k построим подзадачу $p'(k)$ на интервале $[t(k); T]$, причем сегменты, уже посещённые до момента $t(k)$, будем считать зафиксированными как часть начальных условий задачи $p'(k)$. Далее, пользуясь тем же самым алгоритмом, получаем частичное решение $s\left(p'(k)\right)$ для построенной подзадачи.

Определение. Решение $s(p)$ – *состоятельное во времени*, если для каждого $k = 1, \dots, K(s(p))$ выполняется неравенство $f(s(k, p)) \leq f(s(p'(k)))$, где f – целевая функция текущей задачи.

Очевидно, что оптимальные решения всегда будут состоятельными во времени, согласно принципу оптимальности Беллмана. Более того, если эвристический алгоритм является детерминированным, то в любой подзадаче частичные решения будут совпадать с исходным. Таким образом, данное понятие состоятельности во времени актуально только для рандомизированных эвристик, которые, впрочем, составляют основную массу исследуемых алгоритмов маршрутизации.

Отметим, что в зависимости от особенностей конкретного алгоритма и особенностей конкретной рассматриваемой модели, *несостоятельность во времени* может проявляться в большей или меньшей степени при анализе решений, как и может не проявляться совсем, даже если решения не являются оптимальными.

Продолжая построение вычислительных экспериментов, положим что для каждого решения $s \in N(p)$ проведено $M(s)$ экспериментов. Отдельный эксперимент состоит в проверке решения на состоятельность во времени. Обозначим за $b(s,k)$ число экспериментов, в которых временная состоятельность решения s нарушается на шаге k. Если бы решение s было бы оптимальным, то выполнялось бы равенство $\sum_{k=1}^{K(s)} b(s,k) = 0$. Но так как эвристики не гарантируют оптимальность, то можно утверждать лишь справедливость неравенства: $\sum_{k=1}^{K(s)} b(s,k) \leq M(s)$.

Определение. *Экспериментальным уровнем временно́й состоятельности* ($conL$) будем называть величину, вычисляемую как

$$conL = 1 - \frac{1}{|P|} \times \sum_{p \in P} \frac{1}{|N(p)|} \sum_{s \in N(p)} \frac{1}{M(s)} \sum_{k=1}^{K(s)} b(s,k) \qquad (3.1)$$

Отметим, что $0 \leq conL \leq 1$. Высокий уровень временной состоятельности алгоритма указывает на то, что ожидаемые решения будут «более устойчивы» во времени, по сравнению с ожидаемыми решениями алгоритма с меньшим значением этого критерия.

3.3. Динамическая адаптация алгоритмов

Итеративный метод улучшения решений для задачи маршрутизации транспорта (а точнее, кооперативной игры маршрутизации транспорта) был предложен в [40], и развит для случая задачи маршрутизации запасов в [41].

Используя основную идею метода, мы адаптируем его для общего случая модели маршрутизации и абстрактного алгоритма решения.

Фактически, метод представляет собой моделирование реализации решения во времени. Данная реализация на практике – это последовательное посещение точек маршрутов, последовательно перебирая их соответственно моментам времени. Элементарная единица решения (в рассматриваемой дискретной задаче) есть перемещение в следующую точку маршрута или переход в следующий период планирования, если задача подразумевает долгосрочное планирование В каждый момент времени мы всё ещё потенциально имеем возможность изменить план решения, но только для находящихся в будущем элементов, так как реализованные действия в прошлом становятся константами.

Далее, используем следующие обозначения. Пусть $A(\cdot)$ обозначает функцию адаптируемого алгоритма, s – решение, полученное с помощью алгоритма, S_1 – последовательность посещенных сегментов решения, S_2 – последовательность оставшихся не посещённых сегментов, C – константы задачи, f – целевая функция.

Общая схема Динамической адаптации представлена ниже в Алгоритме 2.

Алгоритм 2. Динамическая адаптация алгоритма

0: Инициализация: начальное решение s_0, $S_1 = \emptyset$, $S_2 = s_0$
1: Пока $S_2 \neq \emptyset$:
2: $s = A\ (\ S_2,\ C\ )$
3: Если $f(s) < f(S_2)$: $S_2 = s$
4: $C \leftarrow S_2[1], \quad S_1 \leftarrow S_2[1], \quad S_2 = S_2 \setminus S_2[1]$

3.4. Неустойчивая задача. Пример IRP.

Для динамической адаптации был выбран алгоритм ALNS. Алгоритм адаптации был реализован на языке C++. Вычислительные эксперименты проводились на HPC-кластере кафедры Моделирования Электромеханических и

Компьютерных Систем, факультета Прикладной Математики – Процессов Управления, СПбГУ. Характеристики кластера: ОС Linux SLES 11 SP1, 12 узлов, каждый с двумя четырёхъядерными процессорами Intel Xeon 5335, 16 ГБ ОЗУ на узел.

Для использования в данной работе была выбрана библиотека тестовых примеров для задачи маршрутизации запасов *Coelho*, так как представляет хорошую вариативность примеров и точно согласуется с рассматриваемой моделью. Примеры находятся в открытом доступе в интернете на ресурсе [42].

Тестовые пример выбранной библиотеки генерировались с помощью случайного механизма. Для проведения экспериментов было взято 20 тестовых примеров из библиотеки [42]. Выбранные примеры различаются по стоимости хранения (h: высокая или низкая), числу периодов в рассматриваемом промежутке времени (T: 3 или 6) и числу клиентов (n: 10/20/30/40/50/100). Эти характеристики также представлены в Таблице 1 в столбце «пример».

Для каждого примера в ходе эксперимента генерировалось 100 решений, среди которых для дальнейшего рассмотрения выбирались $N(p)$ различных. На каждом уникальном решении, как на начальном, Динамический алгоритм был запущен M раз (единица для больших примеров: 6×50 и 6×100, десять для остальных).

В качестве результатов собирались следующие данные: начальное значение целевой функции, конечное предполагаемое значение целевой функции (начальное значение минус сумма всех улучшений), конечное пересчитанное значение целевой функции (на собранном конечном решении), номера шагов, на которых получено улучшение, и величины этих улучшений. Из-за особенностей реализации, учитываются два значения конечной целевой функции, так как эти значения могут различаться: как предполагаемое значение может быть меньше, чем реальное, при ошибочном улучшении, так и реальное может быть меньше,

чем предполагаемое, из-за возможной дополнительной выгоды, при пересчёте решения.

Полученные данные обрабатывались в среде *Wolfram Mathematica*.

По результатам экспериментов, был подсчитан экспериментальный уровень временной состоятельности для задачи маршрутизации запасов с использованием алгоритма ALNS:

$$conL = 0{,}545424$$

Далее используются следующие дополнительные обозначения. $N1$ – общее число нарушений временной состоятельности среди $N(p) \times M$ тестов. $N2$ – число несостоятельных во времени решений, т. е. несостоятельных в каждом из M тестов. Аналогично, $N3$ – это число полностью состоятельных во времени решений. Отметим, что $N2 + N3 \leq N(p)$. Указанные значения приведены в Таблице 2.

В двух последних столбцах Таблицы 2 (ALNS и DALNS) приведено сравнение результатов обычного алгоритма ALNS и его динамической адаптации. Представлены средние значения и стандартные отклонения (σ) для выборки значений целевой функции в каждом случае.

Как можно видеть, в большей части случаев применения динамический алгоритм показал улучшение обычного решения, за исключением разве что примеров малой размерности, в которых получаемые решения уже являются оптимальными или очень близкими к оптимальному. Суммарно, число экспериментов, в которых было получено улучшение, равно 5236, что составляет 46% от общего их числа (11360).

На примерах большей размерности динамический алгоритм показывает растущую эффективность, так как при увеличении размерности эффективность базового алгоритма понижается и решения дальше отдаляются от оптимума, что оставляет больше возможностей для дополнительной оптимизации. В части экспериментов, динамический алгоритм показывает существенное улучшение,

до 22% от начального решения. В среднем же, улучшение начального решения находиться в пределах 0–6,8%, в зависимости от рассматриваемого примера.

Таблица 2. Результаты реализации динамической адаптации

пример			N(p)	M	N1	N2	N3	ALNS		DALNS	
h	T	n						Сред. f	σ	Сред. f	σ
низкая	3	10	7	10	0	0	7	1625,04	56,5964	1625,04	56,5964
		20	26	10	0	0	26	2695,07	351,649	2695,07	351,649
		30	72	10	18	1	67	3770,14	406,636	3770,07	406,12
		40	74	10	73	5	61	4284,57	483,59	4267,39	448,406
		50	92	10	38	2	85	4647,59	375,56	4638,27	347,246
	6	10	51	10	323	25	10	3787,11	172,105	3716,21	152,237
		20	100	10	783	68	13	6143,56	459,988	5922,44	384,808
		30	100	10	827	61	4	8113,19	597,666	7807,11	471,044
		50	100	1	90	90	10	10526,8	624,891	10209,2	709,574
		100	100	1	75	75	25	16873	938,445	16507,2	912,191
высокая	3	10	7	10	0	0	7	3663,97	55,3075	3663,97	55,3075
		20	30	10	17	0	24	6040,8	266,576	6039,99	265,39
		30	87	10	125	4	64	10140,3	361,031	10129,3	330,997
		40	87	10	180	14	62	11398,7	445,25	11377	392,563
		50	91	10	197	13	67	12360,4	305,425	12348,1	290,239
	6	10	72	10	585	48	6	7574,81	191,467	7484,25	193,026
		20	100	10	833	70	8	13045,5	361,751	12900,8	333,288
		30	100	10	884	81	5	20357,3	527,622	20165,1	422,808
		50	100	1	94	94	6	27358,4	656,647	27109,8	635,106
		100	100	1	94	94	6	52188,2	780,861	51821,9	698,469

3.5. Устойчивая задача. Пример PDP.

Аналогично с предыдущей задачей, для решения Задачи сбора и доставки был адаптирован алгоритм ALNS, вместе с методом динамической адаптации. Все алгоритмы были также реализованы на языке C++. Для проведения экспериментов, в отличие от предыдущего случая, был использован персональный компьютер – на ОС Windows 10, с двухъядерным процессором Intel Core i5 и 4Гб оперативной памяти.

Для экспериментов была использована библиотека тестовых примеров *Li & Lim* [43], содержащая постановки задач сбора и доставки с временными окнами

(PDPTW). Со всеми 344 доступными примерами была проведена проверка состоятельности во времени. Примеры, в основном, различаются по количеству заказов: 100, 200, 400, 600, 800 и 1000. В отличие от предыдущего случая, для каждого отдельного примера был проведён 1 запуск алгоритма ALNS и 1 запуск динамической адаптации, с проверкой временной состоятельности.

Во время экспериментов собирались аналогичные предыдущей группе экспериментов данные, плюс данные по времени работы алгоритмов, которые так же обрабатывались в среде *Wolfram Mathematica*.

По результатам экспериментов, был подсчитан экспериментальный уровень временной состоятельности для задачи сбора и доставки с временными окнами с использованием алгоритма ALNS:

$$conL = 0{,}395349$$

Агрегированные результаты экспериментов представлены ниже в Таблице 3. Следующие обозначения были использованы: n – примерное количество пар заказов в группе примеров; M – количество примеров в группе; $N1$ – число нарушений временной состоятельности в группе примеров. Улучшение DALNS вычислялось в каждом отдельном эксперименте как отношение суммарных улучшений на каждом шаге работы DALNS и начального решения, полученного ALNS. Статистические значения минимума, максимума и среднего значения улучшения по каждой группе также приведены в таблице ниже. В дополнение, в последнем столбце представлены среднего значения дополнительного времени, затраченного на выполнение динамически адаптированного алгоритма, по сравнению с исходным временем ALNS.

Таблица 3. Результаты динамической адаптации на PDPTW

Примеры		$N1$	Улучшение DALNS, %			Доп. время DALNS, %
n	M		Min	Max	$Average$	$Average$
100	56	3	0,00	4,10497	0,085323	362,703
200	60	23	0,00	8,4197	0,289145	271,552
400	60	39	0,00	2,5684	0,318022	130,929
600	58	47	0,00	2,46471	0,407474	100,946
800	57	50	0,00	2,23131	0,43562	76,042
1000	53	46	0,00	3,24252	0,562285	56,5242

Аналогично экспериментам с задачей IRP, результаты показывают растущую временную неустойчивость с ростом размерности задач. Таким же образом, можно заключить, что с ростом размерности исходное решение становиться всё более удалённым от оптимального, что оставляет больше возможностей для дополнительного улучшения. Суммарно, количество случаев, в которых было найдено какое-либо улучшение составляет 60,5%, что больше, чем в случае задачи IRP.

Однако, с другой стороны, величина улучшения в среднем гораздо меньше: 0-0,56% в среднем и до 8,45% максимально. Если предположить улучшения меньше 1% несущественными по сравнению с затраченным временем, доля неустойчивых решений резко снижается до 10,75%, что делает задачу сильно устойчивой в этом понимании: условный экспериментальный уровень состоятельности во времени $conL^*$ будет равен:

$$conL^* = 0{,}8925$$

Глава 4. Кооперативные задачи маршрутизации

4.1. Введение

В ситуации, когда несколько перевозчиков организуют доставку и обслуживание разных групп клиентов на одной территории, существует возможность их горизонтальной кооперации с целью выявления общего более эффективного плана посещений и доставки. Подобное сотрудничество может отрывать перспективы для существенной экономии, при правильной организации.

В научной литературе, впрочем, этот вопрос рассматривается лишь отчасти. По большей части, работы в данной области посвящены либо разбору путей кооперации, что охватывает первоначальный процесс принятия решений и игнорирует сложности нахождения решений для задач маршрутизации [44], либо вопросам дележа выигрыша коалиций в кооперативной модели [45]. Существуют небольшое число работ, которые рассматривают задачи маршрутизации в совокупности с моделью кооперативной теории игр. Однако, в данных работах не принимается во внимание характер эвристических решений [46], либо же анализ ведётся непосредственно оптимальных решений [47]. На практике же, при использовании эвристических решений в общем случае не соблюдаются отдельные базовые предположения кооперативной теории, что иногда может приводить к неустойчивой структуре коалиций. На эту проблему было впервые обращено внимание в работе [40], и в данной работе проводится дальнейшее её исследование.

В кооперативной теории игр существует множество различных методов дележа выигрыша внутри коалиции, однако их применение требует явного задания характеристической функции, обладающей свойством супераддивности (для задач максимизации выгоды) или субаддитивности (для задач минимизации

затрат). Наши исследования, в свою очередь, показывают, что эти условия нарушаются в общем случае, если для построения характеристической функции используются потенциально суб-оптимальные эвристические решения.

4.2. Построение кооперативной модели IRP

Рассмотрим задачу маршрутизации запасов, и на её основе построим обобщение задачи на кооперативный случай, *Кооперативную игру маршрутизации запаса*.

Предположим, что имеется направленный граф $G = (\{M, N\}, A)$, в котором $M = \{1, \dots, m\}$ является множеством перевозчиков, представленных своими депо, $N = \{1, \dots, n\}$ – общее множество клиентов в задаче, и A – набор всех путей между точками $M \cup N$ вида (i, j).

Каждому перевозчику $i \in M$ соответствует подмножество его собственных клиентов $N_i \subseteq N$. Индивидуально, перед каждым перевозчиком стоит проблема нахождения решения в стандартной задаче маршрутизации запаса на частичном графе $G = (\{i\} \cup N_i, A_i)$. Мы предполагаем также, что множества клиентов не пересекаются: $\forall i_1, i_2 \in M, i_1 \neq i_2 \Rightarrow N_{i_1} \cap N_{i_2} = \emptyset$.

Стоимости перемещения по путям $(i, j) \in A$ считаются известными и заданными величинами $c_{ij} > 0$.

Горизонт планирования будем считать общим и равным T. В начале каждого периода в каждом депо i производится r_i единиц продукции, а каждый клиент j потребляет d_j единиц продукции из своего запаса, дефицит не допустим.

Будем рассматривать задачу, в которой в распоряжении каждого перевозчика находится одно транспортное средство. В течение одного периода каждое транспортное средство может совершить один маршрут посещения

клиентов. При этом, максимальная вместимость запаса каждого ограничена Q единицами.

Каждый клиент может хранить запас продукции для дальнейшего потребления. Объём хранимого клиентом j запаса на конец периода t выражается переменными I_{it}. Этот объём не может опускаться ниже нуля или превышать максимальную вместимость склада U_j. Хранение запаса между периодами планирования стоит h_j за единицу продукции.

Переменными в задаче являются маршруты посещения и объёмы доставки. Целью оптимизации является минимизация суммарных затрат на перевозку и хранение в течение всего горизонта планирования.

Предположим, что перевозчики из M могут объединяться в коалиции $S \subseteq M$. Задачей каждой такой коалиции будет обслужить объединённое множество клиентов $N_S = \bigcup_{i \in S} N_i$ с помощью доступных перевозчикам из S транспортных средств. Отметим также, что при условии $S_1 \cap S_2 = \emptyset$ множества клиентов также не будут пересекаться: $N_{S_1} \cap N_{S_2} = \emptyset$. Задачей коалиции, таким образом, будет решение *специального случая* задачи маршрутизации запаса с несколькими депо и несколькими транспортными средствами на графе $G = (S \cup N_S, A_S)$.

Построим специальный случай IRP в форме линейного целочисленного программирования. Для этого, введём переменные x_{ijt}, принимающие значение 1, если маршрут проходит по пути (i, j), и значение 0 иначе; переменные q_{jt}, равные объёму доставки клиенту j в период t; и, также, вспомогательные переменные s_{it}, отражающие накопленный в текущем маршруте спрос на момент посещения клиента i в периоде t, и необходимые для исключения замкнутых петель в маршрутах. Уровень запаса депо и складов в начале первого периода считается известным и заданным константами $I_{i0} = I_i^0 = const, \forall i \in M \cup N$.

Таким образом, специальная задача IRP для коалиции S может быть представлена в следующем виде:

$$f(S) = \sum_{t=1}^{T} \sum_{i \in S \cup N_S} \sum_{j \in S \cup N_S} c_{ij} x_{ijt} + \sum_{t=1}^{T} \sum_{i \in S \cup N_S} h_i I_{it} \to min \tag{4.1}$$

При ограничениях:

$$\sum_{i \in N_S} x_{ijt} = \sum_{i \in N_S} x_{jit} \leq 1, \qquad j \in S \cup N_S, \;\; t = 1, \dots, T \tag{4.2}$$

$$I_{it} = I_{i,t-1} + r_i - \sum_{j \in N_S} s_{jt} x_{jit}, \qquad i \in S, \;\; t = 1, \;\; \dots, T \tag{4.3}$$

$$I_{jt} = I_{j,t-1} - d_j + q_{jt}, \qquad j \in N_S, \;\; t = 1, \dots, T \tag{4.4}$$

$$0 \leq I_{it} \leq U_i, \qquad i \in S \cup N_S, \;\; t = 1, \;\; \dots, T \tag{4.5}$$

$$s_{it} - s_{jt} + Q x_{ijt} \leq Q - q_{jt}, \qquad i, j \in N_S, i \neq j, \;\; t = 1, \dots, T \tag{4.6}$$

$$q_{it} \leq s_{it} \leq Q, \qquad i \in N_S, \;\; t = 1, \dots, T \tag{4.7}$$

Целевая функция (4.1) представляет собой сумму общих затрат на перевозку и общих затрат на хранение. Равенства левых частей (4.2) гарантируют непрерывность маршрута в рассматриваемых переменных, а неравенства в их левой части представляют собой ограничения на количество посещений и количество маршрутов из каждого депо. Ограничения (4.3) и (4.4) задают динамику запаса в депо и на складах клиентов соответственно. Ограничения на вместимость складов представлены неравенствами (4.5). Неравенства (4.6) выполняют две функции. Во-первых, они задают динамику увеличения спроса по ходу маршрута: при $x_{ijt} = 1$ неравенства принимают вид $s_{jt} \geq s_{it} + q_{jt}$. Во-вторых, они неявным образом исключают возможность образования замкнутых маршрутов, без соединения с депо. Последняя группа неравенств (4.7) задаёт, с одной стороны, отправное значение для спроса в маршруте, а с другой – ограничения на вместимость транспортных средств.

4.3. Характеристическая функция

Формулировка кооперативной игры требует задания *характеристической функции*.

Определение. *Характеристической функцией* называется функция $c\colon 2^M \to \mathbb{R}$, заданная на множестве всех возможных коалиций 2^M, и удовлетворяющая условию $c(\emptyset) = 0$.

Основным условием формирования рациональной коалиции, т.е. коалиции, при объединении в которую каждый участник уменьшает свои затраты, является условие *субаддитивности* характеристической функции:

$$\forall S, T \subset M\colon S \cap T = \emptyset, \quad c(S \cup T) \leq c(S) + c(T) \tag{4.8}$$

Очевидно, что для двух непересекающихся коалиций S, T оптимальные значения f_{opt} целевой функции (4.1) будут удовлетворять условию субаддитивности:

$$f_{opt}(S \cup T) \leq f_{opt}(S) + f_{opt}(T) \tag{4.9}$$

На практике же, получить оптимальное решение и точное минимальное значение целевой функции может быть проблематично, особенно для задач большого размера, что приводит к необходимости использования эвристических методов. При таком подходе, решая задачу (4.1)-(4.7) для коалиции S можно получить эвристическое значение целевой функции $f^h(S)$. Поскольку эвристические алгоритмы не гарантируют оптимальности решения, будет иметь место лишь следующее неравенство:

$$f_{opt}(S) \leq f^h(S) \tag{4.10}$$

Сочетая (4.9) и (4.10), можно получить неравенства:

$$f_{opt}(S \cup T) \leq f^h(S \cup T), \qquad f_{opt}(S \cup T) \leq f^h(S) + f^h(T) \tag{4.11}$$

Поскольку невозможно предсказать заранее разницу между оптимальным и эвристическим решением, в некоторых случаях будет возникать ситуация:

$$f^h(S \cup T) > f^h(S) + f^h(T) \tag{4.12}$$

Таким образом, можно заключить, что в общем случае функция эвристических значений f^h не обладает свойством субаддитивности и не может быть использована как характеристическая функция для формирования устойчивых коалиций.

Одним методом решения данной проблемы является использование специального алгоритма построения характеристической функции, берущего в основу эвристические значения – алгоритма *прямой индукции коалиций (Direct Coalition Induction, DCI)*:

$$c(S) \stackrel{\text{def}}{=} \begin{cases} f^h(S), & |S| = 1 \\ \min\left\{f^h(S), \ \min\limits_{L \subset S}\big(c(S \setminus L) + c(L)\big)\right\}, & |S| > 1 \end{cases} \tag{4.13}$$

Теорема. *(Захаров, Щегряев)* Характеристическая функция, построенная на основе эвристических значений с помощью алгоритма DCI будет обладать свойством субаддитивности (4.8).

4.4. Пример построения характеристической функции

Возьмём за основу уже использованные ранее тестовые примеры из библиотеки [42]. Из трёх примеров *abs1n10*, *abs2n10* and *abs3n10* составим постановку кооперативной игры маршрутизации запаса так, как это показано на

рисунке 1. Депо и соответствующие ему клиенты каждого отдельного примера представлены отдельными цветами.

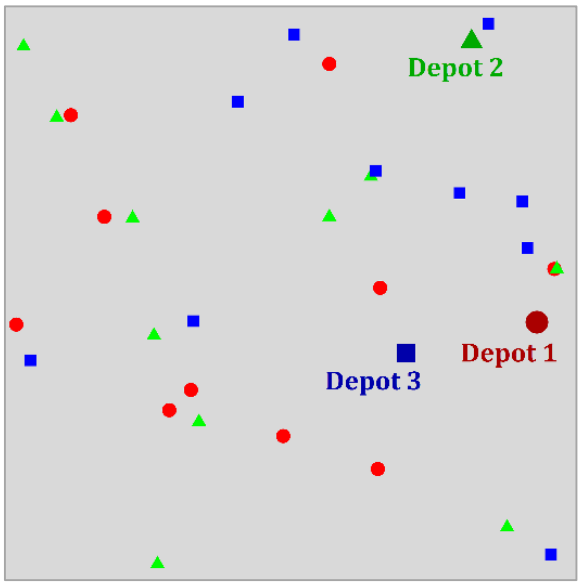


Рис. 1. Кооперативная игра маршрутизации запаса

В полученном примере, возможны семь различных коалиций игроков, и для каждой из них было найдено эвристическое значение f^h. Результаты вычислений, для двух алгоритмов – ALNS и DALNS – представлены в таблице 4 ниже.

Таблица 4. Эвристические значения $f^h(S)$, полученные с ALNS и DALNS

Коалиция	{1}	{2}	{3}	{1,2}	{1,3}	{2,3}	{1,2,3}
Решение ALNS	10988.3	11443.5	9866.42	21135.3	22062.7	21567.3	30335.8
Решение DALNS	7885.53	8407.1	7588.8	15839.9	15702.4	15316.7	22735.3

Полученные решения проиллюстрированы на рисунках 2 и 3. В верхней части – комбинация индивидуальных решений игроков, а в нижней – кооперативное решение для полной коалиции {1,2,3}. Два рисунка отражают, соответственно, решения, полученные с помощью ALNS, на рисунке 2, и решения, полученные с помощью DALNS, на рисунке 3.

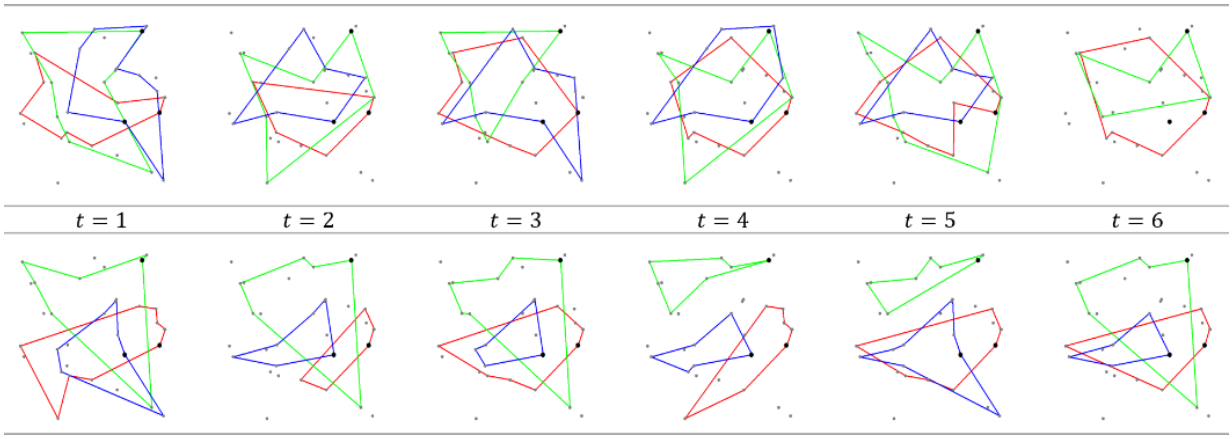


Рисунок 2. Решения ALNS

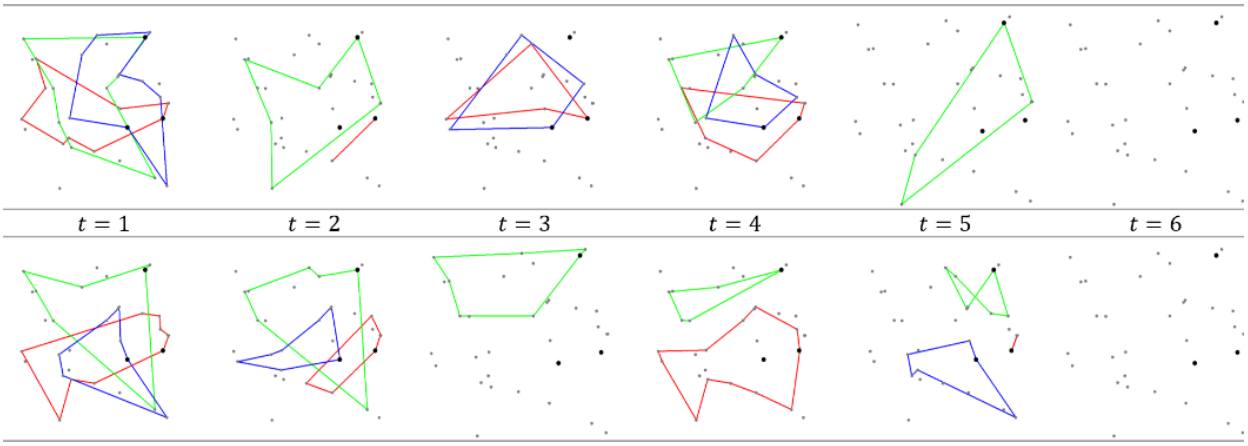


Рисунок 3. Решения DALNS

Основываясь на значениях таблицы 4 и используя алгоритм прямой индукции коалиций (4.13) построим характеристические функции для решений ALNS и DALNS.

Согласно алгоритму прямой индукции коалиций, установим начальные значения по умолчанию: $c(\{1\}) \stackrel{\text{def}}{=} f^h(\{1\}),\ c(\{2\}) \stackrel{\text{def}}{=} f^h(\{2\}),\ c(\{3\}) \stackrel{\text{def}}{=} f^h(\{3\})$.

Для коалиции $\{1,2\}$ имеем следующее неравенство: $f^h(\{1,2\}) = 21135.3 < c(\{1\}) + c(\{2\}) = 22431.8$. Поэтому, устанавливаем $c(\{1,2\}) = f^h(\{1,2\}) = 21135.3$.

$$f^h(\{1,3\}) = 22062.7 > c(\{1\}) + c(\{3\}) = 20854.72 \Rightarrow c(\{1,3\}) = 20854.72$$

$$f^h(\{2,3\}) = 21567.3 > c(\{2\}) + c(\{3\}) = 21309.92 \Rightarrow c(\{2,3\}) = 21309.92$$

Наконец, для полной коалиции $\{1,2,3\}$ имеем следующие неравенства:

$$\begin{cases} f^h(\{1,2,3\}) = 30335.8 < c(\{1,2\}) + c(\{3\}) = 31001.72 \\ f^h(\{1,2,3\}) = 30335.8 < c(\{1\}) + c(\{2,3\}) = c(\{1,3\}) + c(\{2\}) = 32298.22 \end{cases}$$

$$\Rightarrow c(\{1,2,3\}) = f^h(\{1,2,3\}) = 30335.8$$

Аналогичные выкладки проводятся и для решений DALNS:

$$f^h(\{1,2\}) < c(\{1\}) + c(\{2\}) = 16292.69 \Rightarrow c(\{1,2\}) = f^h(\{1,2\}) = 15839.9$$

$$f^h(\{2,3\}) < c(\{2\}) + c(\{3\}) = 15995.96 \Rightarrow c(\{2,3\}) = f^h(\{2,3\}) = 15316.7$$

$$f^h(\{1,3\}) > c(\{1\}) + c(\{3\}) \Rightarrow c(\{1,3\}) = c(\{1\}) + c(\{3\}) = 15474.33$$

$$\begin{cases} f^h(\{1,2,3\}) = 22735.3 < c(\{1\}) + c(\{2,3\}) = 23202.23 \\ f^h(\{1,2,3\}) = 22735.3 < c(\{1,2\}) + c(\{3\}) = 23428.7 \\ f^h(\{1,2,3\}) = 22735.3 < c(\{1,3\}) + c(\{2\}) = 23881.49 \end{cases}$$

$$\Rightarrow c(\{1,2,3\}) = f^h(\{1,2,3\}) = 22735.3$$

Таким образом, для каждого отдельного алгоритма была построена субаддитивная характеристическая функция, и, по определению, задана постановка кооперативной игры.

4.5. Вычислительные эксперименты

Для вычислительных экспериментов были использованы алгоритмы, реализованные для экспериментов главы 4, с отдельными модификациями под специальный случай задачи маршрутизации и хранения с несколькими депо и несколькими транспортными средствами. Вычисления также были проведены на HPC-кластере кафедры Моделирования Электромеханических и Компьютерных Систем, факультета Прикладной Математики – Процессов Управления, СПбГУ.

В качестве основы для построения тестовых примерах была взята использованная ранее библиотека примеров IRP [42]. Каждый пример кооперативной игры маршрутизации запасов был построен путём комбинирования трёх примеров стандартной IRP одного размера, аналогично предыдущему разделу.

В итоге, было создано 18 тестовых примеров кооперативной задачи: для случая 3 периодов планирования – содержащие в сумме 90, 120 и 150 клиентов, каждый с вариацией с высокой и с низкой стоимостью хранения, для случая 6 периодов планирования – содержащие в сумме 30, 60, 90, 150, 300 и 600 клиентов, каждый с вариацией с высокой и с низкой стоимостью хранения.

Для каждого примера и каждой подзадачи для каждой коалиции в примере было проведено от 100 до 2000 запусков DALNS (что одновременно даёт и статистику по решениям ALNS), в зависимости от размера примера и времени вычисления. Собранная статистика была агрегирована до минимального, максимального и среднего значений по выборке для каждой отдельной коалиции. Далее, для адекватного сравнения, полученные данные были просуммированы для отражения коалиционной ситуации с учётом всех участников, т.е., например, при отсутствии кооперации ситуация будет иметь вид $(1,2,3)$, и значениями для сравнения будут суммы индивидуальных значений каждого игрока: $f^h(\{1\}) + f^h(\{2\}) + f^h(\{3\})$, а при кооперации только игроков 1 и 3 ситуация будет иметь вид $(2,13)$, и значением для сравнения будет $f^h(\{2\}) + f^h(\{1,3\})$.

Пример агрегированных данных по одному примеру представлен ниже в Таблице 5, а полную статистику по всем полученным данным можно найти в таблице в Приложении 1.

Таблица 5. Пример агрегированных данных вычислительных экспериментов

Размерность примера Примеры IRP в основе	Алгоритм	Минимум/Среднее/Максимум для ситуации:				
		1,2,3	1,23	2,13	3,12	123
$(3,90, low)$ *abs1n30* *abs2n30* *abs3n30*	ALNS	9223.96 10308.5 13076.3	8203.18 9898.38 12090.	8117.86 10335.6 12763.	8030.97 9937.85 12102.6	7171.8 9284.36 10999.3
	DALNS	9223.96 10305.5 12989.8	8203.18 9855.08 12053.7	8117.86 10290.5 12044.8	8030.97 9864.3 11992.6	7058.08 9239.98 10929.

Для визуализации полученных данных, была дополнительно разработана специальная форма графиков, представленная на рисунках 4-9. Данные по каждой коалиционной ситуации представлены в виде отрезка от минимального до максимального значения, причём тёмная часть отрезка соответствует значениям от среднего до максимального значения, а светлая – от минимального до среднего. В дополнение, на графиках для каждой ситуации представлены два отрезка: красный отражает выборку решений ALNS, а синий – решений DALNS.

Рисунок 4. $(3,90, low)$

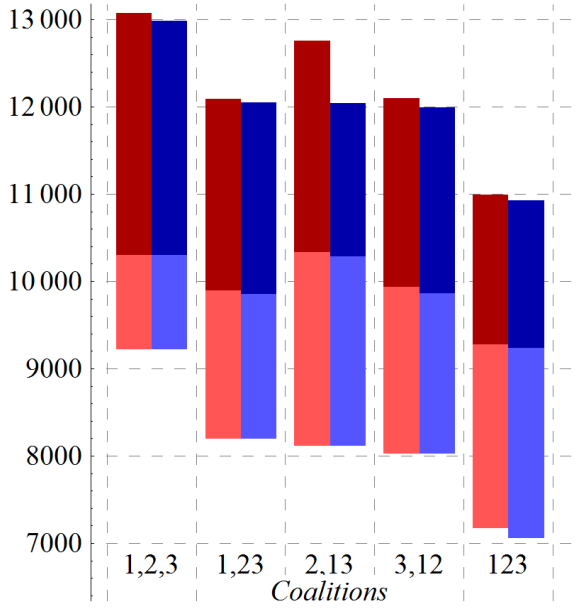


Рисунок 5. $(3,120, high)$

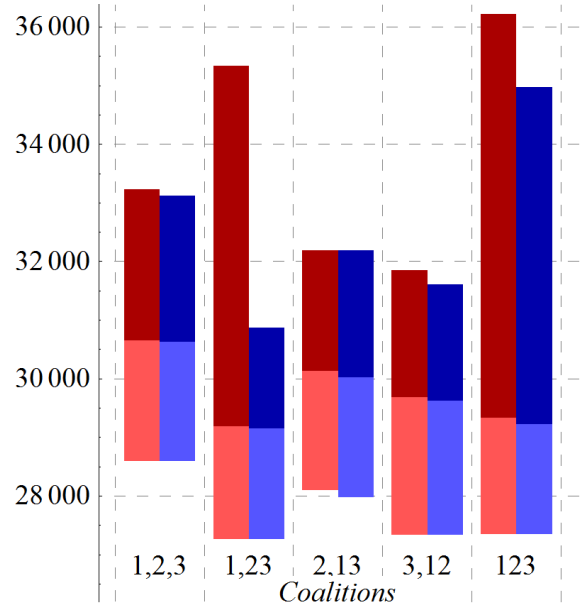


Рисунок 6. $(6{,}90, low)$

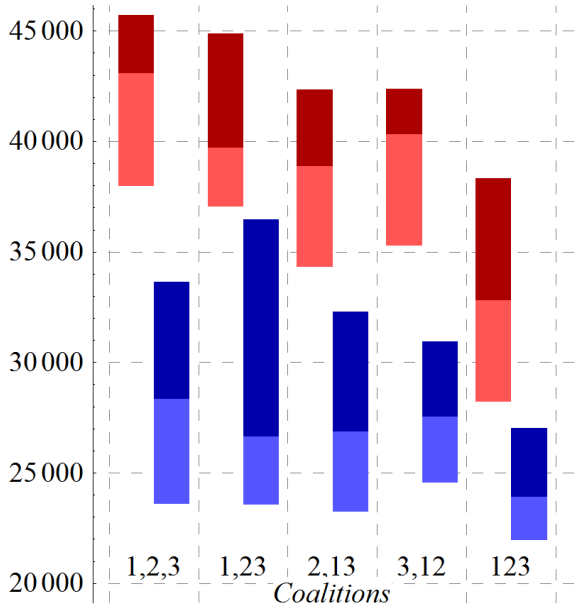


Рисунок 7. $(6{,}30, high)$

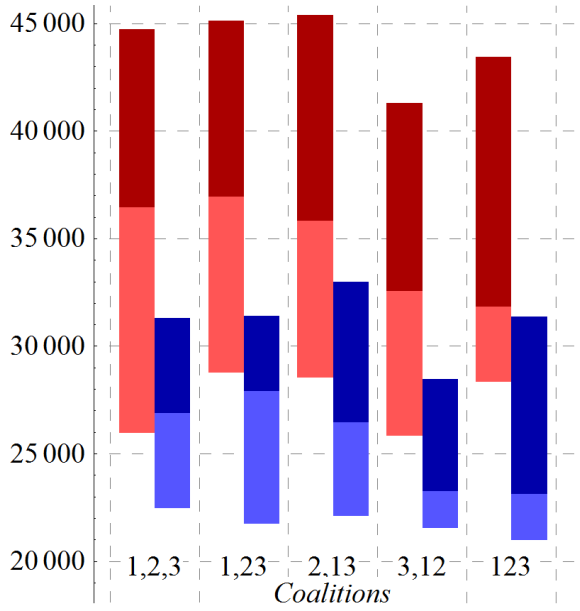


Рисунок 8. $(6{,}300, low)$

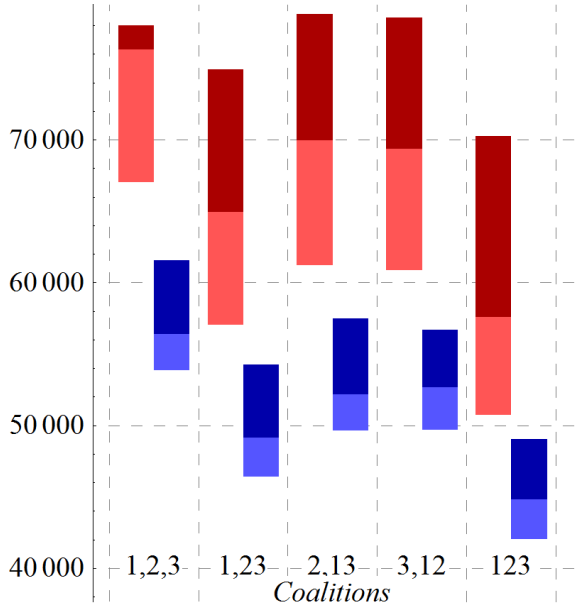
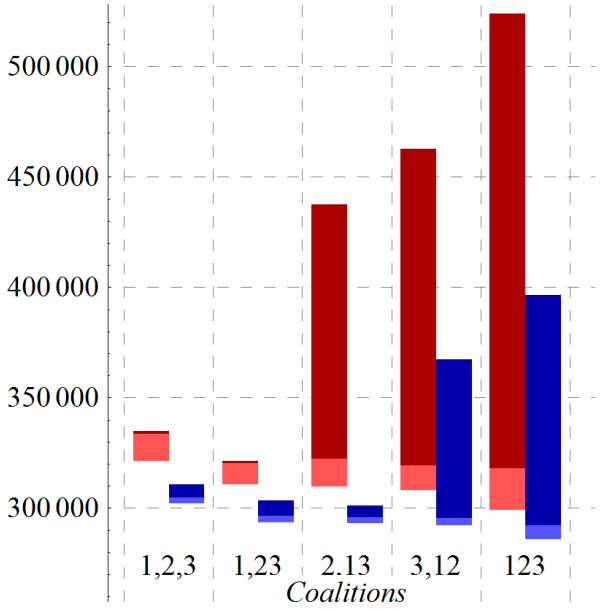


Рисунок 9. $(6{,}600, high)$



Из полученных данных можно вывести следующие общие выводы.

Во-первых, результаты показывают существенную выгоду при кооперации перевозчиков. Сравнивая ситуации индивидуальной работы и полной кооперации, можно видеть, что во всех случаях среднее статистическое значение расходов полной коалиции всегда строго меньше суммы средних значений индивидуальных значений. Расчёты показывают, что это улучшение лежит в

интервале от 1,1% до 25,1%, со средним значением равным 12,24%, для решений ALNS, и в интервале [2,18%, 20,65%], со средним 9,77%, для решений DALNS.

Однако, рассматривая аналогичным образом лучшие (минимальные по стоимости) решения, значения улучшения уже будут лежать в интервале [-17,88%, 27,78%], со средним значением равным 11,64%, при использовании ALNS, или в интервале [-3,7%, 24,13%], со средним значением 9,52%, при использовании DALNS. Отрицательные значения улучшения показывают, что полное кооперативное решение хуже совокупности индивидуальных решений, причём, в проведённых нами экспериментах, такое ухудшение может достигать 17,88%.

Во-вторых, из полученных данных можно вывести меру необходимости использования алгоритма прямой индукции коалиций для последующего расчёта характеристической функции. Такой мерой может служить относительная величина пересечения интервалов решений в ситуации индивидуальной работы (1,2,3) и полной кооперации (123). Минимальное значение этой величины, 0%, означает, что любое кооперативное решение строго лучше совокупности любых индивидуальных решений, в то время как максимальное значение, 100%, означает, что один из интервалов лежит внутри другого. Последнее, в свою очередь может означать, либо что для любых индивидуальных решений существует ненулевая вероятность получить кооперативное решение хуже их совокупности, либо что существует возможность получения таких индивидуальных решений, что любое найденное кооперативное решение будет хуже. В общем случае, чем выше величина пересечения интервалов, тем выше вероятность того, что случайно выбранное кооперативное решение будет хуже, чем совокупность так же случайно выбранных индивидуальных решений.

Согласно полученным данных, в зависимости от рассматриваемого примера, величина относительно пересечения интервалов может достигать как 0%, так и 100%. Единственные два случая, в которых эта величина равна 0% – это примеры $(6{,}150, low)$, при использовании как ALNS, так и DALNS, и $(6{,}300, low)$, при использовании DALNS (рис.8). В среднем, среди рассмотренных примеров эта величина имеет среднее значение 71,06%, при использовании ALNS, и среднее значение 65,4%, при использовании DALNS, что указывает на существенную вероятность нарушения субаддитивности, и, соответственно, на необходимость использования алгоритма DCI для построения характеристической функции.

Наконец, результаты экспериментов иллюстрируют выгоду использования метода динамической адаптации, в дополнение к экспериментам частей 4.3 и 4.4. Интервал улучшения исходных решений ALNS в данном случае составляет [0,014%, 43,95%], при среднем статистическом улучшении равном 15,25%.

Для наглядности, все проанализированные характеристики экспериментальных данных также представлены в Таблице 6 ниже.

Таблица 6. Различные показатели результатов экспериментов.

Показатель		ALNS			DALNS		
		Интервал		Среднее	Интервал		Среднее
Улучшении при кооперации:	Среднее	1,1%	25,1%	12,24%	2,18%	20,65%	9,77%
	Минимум	-17,88%	27,78%	11,64%	-3,7%	24,13%	9,52%
Пересечение интервалов		0%	100%	71,06%	0%	100%	65,4%
Улучшение DALNS		-	-	-	0,014%	43,95%	15,25%

4.6. Анализ устойчивости коалиций

Для дальнейшего анализа, введём в рассмотрение некоторые понятия из кооперативной теории игр.

Определение. *Дележом* выигрыша коалиции S в кооперативной игре называется вектор $x^S = (x_1, x_2, \dots, x_{|S|})$, удовлетворяющий условиям:

$$x_i \le c(\{i\}), \quad \forall i \in S \qquad (\textit{индивидуальная рациональность})$$

$$\sum_{i \in S} x_i = c(S) \qquad (\textit{коллективная рациональность})$$

Определение. Будем говорить, что коалиция S *устойчива относительно дележа* x, если для x выполнены условия дележа, а также выполняется условие:

$$x_i^S < x_i^T, \qquad \forall i \in S, \ \forall T \ne S, i \in T \qquad (\textit{условие стабильности})$$

Целью дальнейшего анализа является исследование свойств устойчивости *полной коалиции*: $S = M$.

Задача построения не имеет единственного «правильного» решения, поскольку, используя различные принципы «справедливости» разделения выигрыша, можно получить различные векторы дележей. Поэтому, для более объективного анализа, будем рассматривать четыре различных метода построения: вектор Шепли, MSC-вектор, вектор метода разницы цен и вектор метода равных доходов.

Вектор Шепли [48] был предложен L.S.Shapley в 1953 году. Идея метода состоит в том, чтобы разделить выигрыш согласно математическому ожиданию вклада каждого игрока в формирование полной коалиции. Элемент вектора Шепли вычисляется следующим образом:

$$Sh_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \big(c(S \cup \{i\}) - c(S)\big)$$

MSC-вектор, или Marginal Subcore element [49] был предложен В.В.Захаровым в 2008 году. Этот метод позволяет выбирать вектор из С-ядра игры согласно относительному (marginal) вкладу игрока в выигрыш коалиции. Элемент MSC-вектора вычисляется с помощью следующих формул:

$$MSC_i = \xi_i^0 + \alpha_i^{msc} \left(c(N) - \sum\nolimits_{j \in N} \xi_j^0 \right),$$

$$\alpha_i^{msc} = \frac{\sum_{S \subseteq N \setminus \{i\}} \big(c(S \cup \{i\}) - c(S)\big)}{\sum_{j \in N} \sum_{S \subseteq N \setminus \{j\}} \big(c(S \cup \{j\}) - c(S)\big)},$$

$$\xi_i^0 : \max\left(\sum_{i=1}^{n} \xi_i\right),\ sbj.\ to : \sum_{i \in S} \xi_i \le c(S), \forall S \ne N.$$

Метод разницы цен, или Cost Gap Method [50] был предложен S.H.Tijs & T.S.H.Driessen в 1986 году. Идея метода состоит в том, чтобы распределять неразделимые затраты игроков согласно их функциям ценовой разницы, отражающим их индивидуальный вклад. Элементы вектора вычисляются по следующим правилам:

$$CGM_i = m_i + w_i g(N),$$

$$m_i = c(N) - C(N \setminus \{j\}),\ \ g(S) = c(S) - \sum_{j \in S} m_j,$$

$$w_i = \frac{W_i}{\sum_{j \in N} W_j},\ \ W_i = \arg\min_{S : i \in S} g(S).$$

Метод равных доходов, или Equal Profit Method [51] был предложен M. Frisk в 2010 году. Идеей данного подхода является попытка равного разделения относительных доходов игроков (относительно их индивидуальных выигрышей).

Вектор дележа в данном методе получается при решении следующей задачи минимизации:

$$EPM_i = x_i:$$

$$\min f \text{, при ограничениях: } f \geq \frac{x_i}{c(\{i\})} - \frac{x_j}{c(\{j\})}, \ \forall (i,j),$$

$$\sum_{i \in N} x_i = c(N), \quad \sum_{i \in S} x_i \leq c(S), \ \forall S \neq N.$$

Для анализа были использованы данные вычислительных экспериментов п.5.4. Данные экспериментов рассматривались в двух измерениях: по методу выбора решения из статистической выборки – рассматривались средние и минимальные значения, и по алгоритму решения – ALNS и DALNS. В качестве первого и исходного набора данных рассматривался случай, в котором эвристические значения f^h были приняты в качестве характеристической функции. Для второго набора данных на основе значений f^h для построения характеристической функции был использован метод прямой индукции коалиций (DCI). Таким образом, для каждого из 18 тестовых примеров было рассмотрено 8 возможных ситуаций: согласно методу нахождения решения (ALNS или DALNS), методу выбора решения из выборки (среднее или минимальное) и методу построения характеристической функции (с $= f^h$ или $c = DCI(f^h)$).

В таблице 7 представлены частоты строгой устойчивости и строгой неустойчивости полной коалиции в разрезах разных измерений и в целом по набору данных при использовании четырёх различных методов построения дележа. Таким образом, по представленным результатам можно сравнить устойчивость решений по различным показателям. Отметим, что сумма одинаковых показателей в левой и в правой части таблицы не всегда соответствует общему числу примеров, поскольку существуют также ситуации,

в которых игрокам безразлично, участвовать в коалиции или работать индивидуально, так как их индивидуальные затраты при этом не изменяются – это ситуации, в которых $\exists T \neq S: x_i^S = x_i^T$.

Таблица 7. Результаты анализа устойчивости

	Частоты устойчивости				Частоты неустойчивости			
	Shapley	MSC	CGM	EPM	Shapley	MSC	CGM	EPM
f^h	41	28	39	41	31	44	33	25
DCI	45	28	42	41	23	42	26	21
ALNS	46	30	42	42	25	41	29	21
DALNS	40	26	39	40	29	45	30	25
Среднее	47	32	47	42	25	40	25	22
Минимум	39	24	34	40	29	46	34	24
В общем	86	56	81	82	54	86	59	46
Всего %	59.7%	39%	56.3%	57%	37.5%	59.7%	41%	32%

Полученные результаты показывают значительную выгоду использования метода DCI для построения субаддитивной характеристической функции: построенные с помощью этого метода игры обладают большей степенью устойчивости.

По результатам анализа, можно наблюдать, что эффективность алгоритма также вносит существенный вклад в обнаружение нестабильности. Так, более эффективный DALNS показывает меньшую степень устойчивости среди исследуемых ситуации. Однако, учитывая, что решения DALNS ближе к оптимальному, это более правдоподобно отражает реальную картину в кооперативной игре.

Наконец, можно видеть, что устойчивость коалиций существенно зависит от использованного для дележа метода. Среди использованных методов, наиболее устойчивым оказались дележи при помощи вектора Шепли: в 59,7% случаев полная коалиция была устойчива. Наименее устойчивым оказался MSC-

вектор: при его использовании, только в 39% ситуаций наблюдалась устойчивость полной коалиции.

Заключение

Таким образом, была проведена работа по обширному исследованию различных математических моделей задач маршрутизации и был проведён сравнительный анализ экспериментальных показателей в разных задачах.

В ходе работы была разработана эффективная реализация на языке C++ современного алгоритма эвристической оптимизации ALNS, и была проведена адаптация этого алгоритма на широкий круг различных задач.

На базе созданной реализации была собрана обширная база экспериментальных данных по нескольким актуальным задачам маршрутизации. Был проведён анализ всех полученных данных, по результатам которого были сделаны выводы:

- об эффективности нового метода *Динамической адаптации* и актуальности его использования в тех или иных задачах, основываясь на новом методе экспериментальной оценки *Состоятельности во времени*;
- о существенных перспективах исследования и выгоде практического применения новой модели *Кооперативной игры маршрутизации запасов*;
- об устойчивости различных подходов к кооперации, основанных на различных методах дележа и различных методах построения характеристической функции игры;
- о важности вычисления характеристической функции по новому методу *Прямой индукции коалиций* для повышения устойчивости кооперации в задачах маршрутизации.

По результатам проведённой в совокупности научной работы были опубликованы работы «*Dynamic adaptive large neighborhood search for inventory routing problem*» в журнале *Modelling, Computation and Optimization in Information Systems and Management Sciences* издательства Springer, «*Heuristic evaluation of the characteristic function in the Cooperative Inventory Routing Game*» в *Journal on Vehicle Routing Algorithms* издательства Springer и подготовлена статья «*Проблема формирования устойчивых коалиций в кооперативной игре маршрутизации и управления запасами*» в журнал *Математическая теория игр и её приложения*.

Список литературы

1. Модели и методы теории логистики: Учебное пособие / Под ред. В. С. Лукинского. СПб.: Питер, 2007. 448 с.
2. Задача коммивояжёра. // https://ru.wikipedia.org/wiki/Задача коммивояжёра
3. Dantzig G., Fulkerson R., Johnson S. Solution of a large-scale traveling-salesman problem // Journal of the operations research society of America. 1954. Vol. 2, No 4. P. 393-410.
4. Flood M. M. The traveling-salesman problem // Operations Research. 1956. Vol. 4, No 1. P. 61-75.
5. Karp R. M. Reducibility among combinatorial problems // Complexity of computer computations. – Springer, Boston, MA, 1972. – С. 85-103.
6. Dantzig G. B., Ramser J. H. The truck dispatching problem //Management science. – 1959. – Т. 6. – №. 1. – С. 80-91.
7. Clarke G., Wright J. W. Scheduling of vehicles from a central depot to a number of delivery points // Operations research. 1964. Vol. 12, No 4. P. 568-581.
8. Wilson N. H. M. et al. Scheduling algorithms for a dial-a-ride system. – Massachusetts Institute of Technology. Urban Systems Laboratory, 1971.

9. Stein D. M. Scheduling dial-a-ride transportation systems //Transportation Science. – 1978. – Т. 12. – №. 3. – С. 232-249.
10. Assad A., Golden B., Dahl R., Dror M. Design of an inventory routing system for a large propane-distribution firm // Proc. of Southeast TIMS Conference, 1982. P. 315-320.
11. Bell W. J., Dalberto L. M., Fisher M. L., Greenfield A. J., Jaikumar R., Kedia P., Mack R. G., Prutzman P. J. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer // Interfaces. 1983. Vol. 13, No 6. P. 4-23.
12. Dumas Y. et al. An optimal algorithm for the traveling salesman problem with time windows //Operations research. – 1995. – Т. 43. – №. 2. – С. 367-371.
13. Bräysy O., Gendreau M. Vehicle routing problem with time windows, Part I: Route construction and local search algorithms //Transportation science. – 2005. – Т. 39. – №. 1. – С. 104-118.
14. Dumas Y., Desrosiers J., Soumis F. The pickup and delivery problem with time windows //European journal of operational research. – 1991. – Т. 54. – №. 1. – С. 7-22.
15. Liu S. C., Lee W. T. A heuristic method for the inventory routing problem with time windows //Expert Systems with Applications. – 2011. – Т. 38. – №. 10. – С. 13223-13231.
16. Picard J. C., Queyranne M. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling //Operations Research. – 1978. – Т. 26. – №. 1. – С. 86-110.
17. Malandraki C., Daskin M. S. Time dependent vehicle routing problems: formulations, properties and heuristic algorithms //Transportation science. – 1992. – Т. 26. – №. 3. – С. 185-200.

18. Xiang Z., Chu C., Chen H. The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments //European Journal of Operational Research. – 2008. – Т. 185. – №. 2. – С. 534-551.
19. Cho D. W. et al. An adaptive genetic algorithm for the time dependent inventory routing problem //Journal of Intelligent Manufacturing. – 2014. – Т. 25. – №. 5. – С. 1025-1042.
20. Gendreau M., Laporte G., Séguin R. Stochastic vehicle routing //European Journal of Operational Research. – 1996. – Т. 88. – №. 1. – С. 3-12.
21. Zhu L., Sheu J. B. Failure-specific cooperative recourse strategy for simultaneous pickup and delivery problem with stochastic demands //European Journal of Operational Research. – 2018.
22. Bertazzi L. et al. A stochastic inventory routing problem with stock-out //Transportation Research Part C: Emerging Technologies. – 2013. – Т. 27. – С. 89-107.
23. Mavrovouniotis M., Yang S. Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors //Applied Soft Computing. – 2013. – Т. 13. – №. 10. – С. 4023-4037.
24. Pillac V. et al. A review of dynamic vehicle routing problems //European Journal of Operational Research. – 2013. – Т. 225. – №. 1. – С. 1-11.
25. Berbeglia G., Cordeau J. F., Laporte G. Dynamic pickup and delivery problems //European journal of operational research. – 2010. – Т. 202. – №. 1. – С. 8-15.
26. Coelho L. C., Cordeau J. F., Laporte G. Heuristics for dynamic and stochastic inventory-routing // Computers & Operations Research. 2014. Vol. 52. P. 55-67.
27. Nagata Y., Bräysy O., Dullaert W. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows //Computers & operations research. – 2010. – Т. 37. – №. 4. – С. 724-737.

28. Ropke S., Pisinger D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows //Transportation science. – 2006. – Т. 40. – №. 4. – С. 455-472.
29. Erdoğan S., Miller-Hooks E. A green vehicle routing problem //Transportation Research Part E: Logistics and Transportation Review. – 2012. – Т. 48. – №. 1. – С. 100-114.
30. Soysal M., Çimen M., Demir E. On the mathematical modeling of green one-to-one pickup and delivery problem with road segmentation //Journal of Cleaner Production. – 2018. – Т. 174. – С. 1664-1678.
31. Soysal M. et al. Modeling a green inventory routing problem for perishable products with horizontal collaboration //Computers & Operations Research. – 2018. – Т. 89. – С. 168-182.
32. Archetti C., Bertazzi L., Laporte G., Speranza M. G. A branch-and-cut algorithm for a vendor-managed inventory-routing problem // Transportation Science. 2007. Vol. 41, No 3. P. 382-391.
33. Desaulniers G., Rakke J. G., Coelho L. C. A branch-price-and-cut algorithm for the inventory-routing problem // Les Cahiers du GERAD G-2014-19, GERAD, Montréal, Canada. 2014.
34. Solving the pla85900 TSP // http://www.math.uwaterloo.ca/tsp/pla85900/compute/compute.htm
35. Concorde Home. Benchmark information // http://www.math.uwaterloo.ca/tsp/concorde/benchmarks/bench.html
36. Coelho L. C., Cordeau J. F., Laporte G. The inventory-routing problem with transshipment //Computers & Operations Research. – 2012. – Т. 39. – №. 11. – С. 2537-2548.
37. Archetti C., Bertazzi L., Hertz A., Speranza M. G. A hybrid heuristic for an inventory routing problem // INFORMS Journal on Computing. 2012. Vol. 24, No 1. P. 101-116.

38. Hemmelmayr V. C., Cordeau J. F., Crainic T. G. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics //Computers & operations research. – 2012. – Т. 39. – №. 12. – С. 3215-3228.
39. Беллман Р. Динамическое программирование. М.: Изд-во иностранной литературы, 1960. 400 с.
40. Zakharov V. V., Schegryaev A. N. Multi-period cooperative vehicle routing games // Contributions to Game Theory and Management, 2014. Vol. 7. P. 349-359.
41. Shirokikh V. A., Zakharov V. V. Dynamic adaptive large neighborhood search for inventory routing problem //Modelling, Computation and Optimization in Information Systems and Management Sciences. – Springer, Cham, 2015. – С. 231-241.
42. Leandro C. Coelho. Problem Instances: Inventory-Routing. // http://www.leandro-coelho.com/instances/inventory-routing
43. PDPTW. Li & Lim benchmark. // https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark
44. Verdonck L. et al. Collaborative logistics from the perspective of road transportation companies //Transport Reviews. – 2013. – Т. 33. – №. 6. – С. 700-719.
45. Guajardo M., Rönnqvist M. A review on cost allocation methods in collaborative transportation //International transactions in operational research. – 2016. – Т. 23. – №. 3. – С. 371-392.
46. Krajewska M. A. et al. Horizontal cooperation among freight carriers: request allocation and profit sharing //Journal of the Operational Research Society. – 2008. – Т. 59. – №. 11. – С. 1483-1491.
47. Kimms A., Kozeletskyi I. Core-based cost allocation in the cooperative traveling salesman problem //European Journal of Operational Research. – 2016. – Т. 248. – №. 3. – С. 910-916.

48. Shapley L. S. A value for n-person games //Contributions to the Theory of Games. – 1953. – Т. 2. – №. 28. – С. 307-317.
49. Zakharov V. et al. Comparing solutions in joint implementation projects //International Game Theory Review. – 2008. – Т. 10. – №. 01. – С. 119-128.
50. Tijs S. H., Driessen T. S. H. Game theory and cost allocation problems //Management Science. – 1986. – Т. 32. – №. 8. – С. 1015-1028.
51. Frisk M. et al. Cost allocation in collaborative forest transportation //European Journal of Operational Research. – 2010. – Т. 205. – №. 2. – С. 448-458.

Приложения

Приложение 1: Агрегированные данные выч. экспериментов п.4.5.

Размерность примера Примеры IRP в основе	Алгоритм	Минимум/Среднее/Максимум для ситуации:				
		1,2,3	1,23	2,13	3,12	123
$(3,90,low)$ *abs1n30* *abs2n30* *abs3n30*	ALNS	9223.96 10308.5 13076.3	8203.18 9898.38 12090.	8117.86 10335.6 12763.	8030.97 9937.85 12102.6	7171.8 9284.36 10999.3
	DALNS	9223.96 10305.5 12989.8	8203.18 9855.08 12053.7	8117.86 10290.5 12044.8	8030.97 9864.3 11992.6	7058.08 9239.98 10929.
$(3,90,high)$ *abs1n30* *abs2n30* *abs3n30*	ALNS	10209.9 11805.5 15163.9	8692.72 10660.4 12805.8	8993.28 11699.5 13930.2	8656.04 10904.9 16275.1	7842.1 10993.7 13130.3
	DALNS	10209.9 11795.7 15134.3	8692.72 10632.7 12741.3	8993.28 11627.8 13882.4	8656.04 10853.1 13330.3	7746.28 10907.1 12838.7
$(3,120,low)$ *abs1n40* *abs2n40* *abs3n40*	ALNS	11913.1 14200.8 17034.6	10998.9 14176.4 20252.4	10996.2 14135.9 16952.1	10928.7 13505.5 17331.4	10727.5 14045.3 21875.1
	DALNS	11913.1 14186.4 16686.2	10998.9 14102.4 16220.9	10996.2 14089.2 16476.6	10928.7 13415.4 16143.	10727.5 13877.5 15280.4

$(3,120,high)$ *abs1n40* *abs2n40* *abs3n40*	ALNS	17458.7 25935. 29147.4	19787.7 23646.9 26830.2	21020.6 24995.3 28368.7	17414.2 23897.7 26409.2	20581.1 21454.6 25334.4
	DALNS	12544.5 16247.4 20040.2	12247. 16433.8 19880.3	12346.8 16079.1 20343.8	11515.6 13874.5 18333.6	11308.7 13304. 18086.1
$(3,150,low)$ *abs1n50* *abs2n50* *abs3n50*	ALNS	29115.8 33974.3 35699.1	26098.1 30188.6 33404.	27044.8 31819.8 35071.2	27752. 32245. 33588.3	23985 27873. 34641.5
	DALNS	17623.6 22149.1 25974.	17795.3 20944.7 24771.9	16356.7 21650. 25391.	18682.1 22044.3 27966.6	18276.4 20126.4 25998.7
$(3,150,high)$ *abs1n50* *abs2n50* *abs3n50*	ALNS	37991.5 43100.8 45742.3	37067. 39726.9 44890.7	34316.3 38877.5 42368.3	35293.1 40326.2 42392.8	28212.4 32815.5 38349.9
	DALNS	23586.7 28348.5 33648.3	23563. 26669.1 36489.	23234.7 26870.3 32300.4	24570.1 27561.5 30948.8	21964.9 23927.4 27049.7
$(6,30,low)$ *abs1n10* *abs2n10* *abs3n10*	ALNS	52356.1 57751.3 64649.9	46637.9 53655.4 62578.2	45819. 50139.9 60648.8	46680.6 50882.2 57944.	37809.5 43251.3 50478.3
	DALNS	38163.4 40674.9 47721.5	36037.6 38637.3 44102.	33598.4 36393.3 42441.8	34468.7 37040.3 43082.1	30070.7 32275.3 35747.3
$(6,30,high)$ *abs1n10* *abs2n10* *abs3n10*	ALNS	67048.2 76361.4 78040.	57050. 64980.3 74936.6	61229.2 69978.3 78847.8	60884.6 69364.8 78596.7	50740.2 57600.3 70280.
	DALNS	53875.4 56398.6 61592.6	46428.9 49155.2 54283.2	49663.6 52182.5 57489.8	49687.2 52666. 56709.5	42052.2 44828.7 49074.
$(6,60,low)$ *abs1n20* *abs2n20* *abs3n20*	ALNS	103199. 117007. 133409.	92430.3 104519. 128691.	91657.9 103679. 128210.	90190.9 101872. 118553.	81258 92119.2 113749.
	DALNS	85704.4 90282.8 97141.9	77804.4 82435.5 88479.	77380.5 82266.2 93066.8	76033.5 81196.5 88999.9	69615.8 75262. 86551.3

$(6,60,high)$ *abs1n20* *abs2n20* *abs3n20*	ALNS	25752.2 27197.7 29464.7	24998.1 26745. 28832.	24912.4 27041.2 28643.9	24924.4 26900. 29476.3	23949.1 25995.9 31863.4
	DALNS	25752.2 27191. 29464.7	24965.3 26678.6 28703.8	24912.4 26990.5 28559.	24924.4 26814.8 28919.2	23927.6 25923. 30642.6
$(6,90,low)$ *abs1n30* *abs2n30* *abs3n30*	ALNS	28600.9 30658.1 33228.7	27266.3 29192.1 35340.5	28104.1 30140.8 32197.1	27342.6 29682.3 31856.9	27350.8 29337. 36227.7
	DALNS	28600.9 30633.9 33126.9	27266.3 29154.5 30870.1	27974.7 30031.5 32197.1	27341.6 29621.2 31616.3	27350.8 29221.4 34978.8
$(6,90,high)$ *abs1n30* *abs2n30* *abs3n30*	ALNS	36666. 39286.6 40849.2	35954.6 38379.4 39685.1	36451.3 38882.7 45284.5	35890.7 38110.4 43486.9	36023.2 37993.2 45268.3
	DALNS	36666. 39255.1 40838.6	35933.7 38334.5 39461.8	36309. 38788.5 40266.	35890.7 38004. 39120.5	36023.2 37875.8 39159.7
$(6,150,low)$ *absH6low1n50* *absH6low2n50* *absH6low3n50*	ALNS	25962.1 36448.3 44744.3	28779.8 36960.9 45144.5	28552.5 35827.9 45415.7	25857.4 32561.2 41322.5	28339.8 31846.9 43465.7
	DALNS	22469.1 26889. 31303.9	21765.7 27920.4 31401.8	22115.1 26458.3 33009.9	21537.9 23280.1 28468.5	20992.4 23127.3 31379.4
$(6,150,high)$ *absH6high1n50* *absH6high2n50* *absH6high3n50*	ALNS	49787.4 54546. 70209.9	46569.7 51744.8 73580.5	47888. 52748.3 66260.2	48304.6 53005.4 75722.	44917.3 50041.6 73574.4
	DALNS	37340.4 41539.8 58720.5	36541. 41175.7 60180.5	36612.8 41413.9 44340.	38328.7 41761.3 58666.3	38658.9 40609. 52971.5
$(6,300,low)$ *absH6low1n100* *absH6low2n100* *absH6low3n100*	ALNS	69260.7 74834. 92998.1	65641.6 71764. 101238.	65724.3 71473.5 91611.9	67054.5 72447.7 103096.	59898.6 65388.1 97058.5
	DALNS	54825.5 59510.5 79513.5	55727.5 58325.5 88355.9	54118.1 58832.3 75802.4	55506. 58505.1 78337.2	54366.1 56277.4 72947.9

$(6,300,high)$ *absH6high1n100* *absH6high2n100* *absH6high3n100*	ALNS	99413.7 107183. 108205.	96041.6 104230. 132674.	92636.5 100114. 125330.	93850. 100851. 125378.	87659 92875.3 140804.
	DALNS	86117.2 88767.9 93008.2	84904.9 86749.4 89682.8	81301.5 84884.7 87496.5	83781.9 84637.9 98287.3	78506.9 80487.2 103192.
$(6,600,low)$ *absH6low1n200* *absH6low2n200* *absH6low3n200*	ALNS	167880. 177231. 215443.	158123. 165822. 209799.	162033 170158. 217931.	161903. 171279. 249574.	152409 161753. 260136.
	DALNS	153640. 155378. 160601.	146119. 147833. 151489.	149380. 151657. 155751.	149357. 151152. 156217.	142069. 144144. 185334.
$(6,600,high)$ *absH6high1n200* *absH6high2n200* *absH6high3n200*	ALNS	321648. 333786. 335075.	310770. 320367. 321417.	309817. 322608. 437723.	308404. 319666. 462928.	299216 318191. 523977.
	DALNS	302328. 305003. 310903.	293697. 296516. 303501.	293497. 296030. 301189.	292434. 295751. 367432.	286056. 292270. 396690.