МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
КАФЕДРА МНОГОПРОЦЕССОРНЫХ СИСТЕМ И СЕТЕЙ

ПИЩАЛОВА Агата Александровна

Разработка алгоритмов кооперативного планирования расписаний поставок на нефтяные платформы судами снабжения

Магистерская диссертация
Специальность 1-31 80 09 Прикладная математика и информатика

Научный руководитель:
Ковалев Михаил Яковлевич
доктор физико-математических наук
профессор

Допущена к защите

“___” _____ 2021 г

Зав. кафедрой Многопроцессорных систем и сетей
кандидат физико-математических наук, доцент С. В. Марков

Минск 2021

Оглавление

РЕФЕРАТ

Магистерская диссертация, 50 страниц, 3 иллюстрации, 5 таблиц, 24 источника, 2 приложения.

Ключевые слова: Теория игр. Кооперативные игры. Нефтегазовая логистика. Кооперация. Проблема планирования расписаний судов снабжения.

Объект исследования – алгоритмы кооперативного планирования расписаний поставок на нефтяные платформы судами снабжения.

Цель работы – разработка алгоритмов кооперативного планирования расписаний поставок на нефтяные платформы судами снабжения.

В результате – разработан и программно реализован алгоритм кооперативного планирования расписаний поставок на нефтяные платформы судами снабжения, исследованы дележи для данного примера.

РЭФЕРАТ

Магістарская дысертацыя, 50 старонак, 3 ілюстрацыі, 5 табліц, 24 крыніцы, 2 прыкладання.

Ключавыя словы: Тэорыя гульняў. Кааператыўныя гульні. Нафтагазавая лагістыка. Кааперацыя. Праблема планавання раскладаў судоў забеспячэння.

Аб'ект даследавання – алгарытмы кааператыўнага планавання раскладаў паставак на нафтавыя платформы судамі забеспячэння.

Мэта працы – распрацоўка алгарытмаў кааператыўнага планавання раскладаў паставак на нафтавыя платформы судамі забеспячэння.

У выніку – распрацаваны і праграмна рэалізаваны алгарытм кааператыўнага планавання раскладаў паставак на нафтавыя платформы судамі забеспячэння, даследаваны дзяльба для дадзенага прыкладу.

ABSTRACT

The master's thesis, 50 pages, 3 illustrations, 5 tables, 24 sources, 2 appendices.

Key words: Game theory. Cooperative games. Oil and gas logistics. Cooperation. Supply vessel planning problem.

The object of the research: algorithms for cooperative planning of delivery schedules to oil platforms by supply vessels.

The purpose of the work is to develop algorithms for cooperative planning of delivery schedules to oil platforms by supply vessels.

As a result, an algorithm for cooperative planning of schedules for deliveries to oil platforms by supply vessels was developed and implemented in software, the divisions for this example were investigated.

ВВЕДЕНИЕ

В данной работе будет изучено кооперативная теория игр применительно к проблеме планирования расписаний поставок на нефтяные платформы судами снабжения.

Сама проблема является широко известной, и в литературе можно найти много точных и приближенных алгоритмов, посвященных различным формулировкам обозначенной задачи. Однако, кооперативное планирование еще никогда не рассматривалось в рамках данной проблемы.

Данная проблема является крайне важной для компаний, оперирующих судами снабжения, так как содержание судов снабжения крайне дорогостоящие занятие, а при применении кооперативного планирования на реальных примерах задачи получается значительно сократить количество кораблей для содержания. Более того, в процессе этого уменьшаются и транспортные расходы, так как становится возможным более эффективным способом построить маршруты.

Данная задача является сложной, потому что для моделирования данных ситуаций требуется множество ограничений, и формулировка моделей получается сложной, а с добавлением кооперации задача становиться многократно сложнее.

Данная работа будет представлена следующим способом: в первой части, будут рассмотрены теория игр и, в частности, кооперативная теория игр. Будут введены понятия C-Ядра, N-ядра, K-ядра, сильного ε-ядра и вектора Шепли. Будет рассмотрено понятие пустого ядра и приведен способ проверки ядра на пустоту. Также будут рассмотрены выпуклые кооперативные игры и их свойства. Все это будет использовано далее, при проведении численного эксперимента.

В следующей части будет проведен краткий обзор литературы по темам, относящимся к рассматриваемой задаче.

Далее будет приведена формулировка задачи, как она дана в литературе. Будет рассмотрено два способа формулировки задачи – один из способ заключается в широко известном построение транспортной сети, второй способ представляет собой двухфазный метод построения решения, где на первом этапе генерируется возможные маршруты движения кораблей, а на втором этапе решается модель, принимающая эти маршруты как входные данные. Будет разработан алгоритм генерации возможных маршрутов и математическая модель для облегченного варианта задачи.

Будет построена модель с помощью языка математического моделирования AMPL.

В процессе формулировки задачи будет рассмотрено множество дополнительных ограничений, которые встречаются на практике и поэтому должны быть включены в формулировку задачи. Например, будут рассмотрены ограничения, которые распределяют времена отплытия судов снабжения от базы в течение недели, а также ограничения, которые не позволяют двум судам снабжения находиться на определенной нефтяной платформе одновременно.

С помощью модели, будет решена задача для конкретного примера, решение будет проиллюстрировано. Будут приведены результаты работы программы и с помощью вектора Шепли будет получен дележ для коалиций. Будет показано, какие коалиции сформируются, и находится ли полученный дележ в C-ядре. Также докажем, что ядро не пустое.

Будет получен дележ для некоторых других простых методов, рассмотренных в разделе 1, а также с помощью N-ядра.

ГЛАВА 1. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

1.1 Теория игр

Прежде чем перейти к определению теории игр, следует ознакомиться с понятием конфликтная ситуация.

Конфликтная ситуация – это ситуация, в которой находятся два (или более) противоборствующих субъектов, интересы которых сталкиваются. Основная цель сторон конфликта – это достижение наибольшего успеха в конфликтной ситуации, однако, так как стороны являются противодействующими, успех одной стороны означает проигрыш другой. Следовательно, каждый из субъектов ищет допустимое для себя решение в условии неопределенности поведения противоборствующих, так как выигрыши и проигрыши зависят от принятых решений каждой из сторон.

Тем самым, *теория игр* – это теория, исследующая математические модели принятия решения в условии неопределенности поведения других субъектов, причем в процессе игры «игроку» известны лишь:

- Множество ситуаций, в которых он может находиться;
- Множество решений, которые он может принять;
- Количество выигрыша, которое он может получить в конце игры в зависимости от принятого решения, в данной конкретной ситуации;

Что касается применения данной теории, то широкий и разнообразный список явлений может быть описан с ее помощью: биологическая война за существование, различные конфликты (классовые, потребительские, правовые), ценообразование, управление различными динамическими объектами и т. д. В список дисциплин, которые используют теорию игр можно смело занести: прикладную математику, политологию, военное дело, биологию, экономику, социологию и другие (Кремлев, 2016).

1.1.1 Определение игры и ее элементов

В данном пункте будут приведены основные понятия теории игр.

Начнем с определения, что же такое «игра».

Игра – это идеализированная математическая модель поведения нескольких субъектов, имеющих различные интересы, что приводит к некоторым разногласиям.

Отличием игры от конфликтной ситуации является наличие того факта, что игра ведется по строгому набору правил. Данный свод инструкций включает в себя:

- Последовательность принятия решений субъектами
- Количество информации, доступный игрокам, о поведении оппонентов
- Результаты игры, в зависимости от текущей ситуации
- Описание ситуации конца игры (последовательности ходов, после которых следующие ходы делать нельзя)

В теории игр используется определение *игроков* – это субъектов, принимающих участие в игре, которое будет использоваться далее.

В условиях игры игрокам также известно множество всех возможных стратегий оппонента, неизвестно лишь какой именно противник воспользуется, что неприменимо к конфликтам. Еще одним ограничением игры является факт принятия полной идеальности оппонента (противник будет стараться принимать наиболее выигрышное для себя решение, не стоит надеяться на его глупость).

Теория игр предполагает, что составляющей частью игры являются *ходы*, которые могут быть последовательными для игроков, либо же одновременными. *Партией* в игре называется множество ходов, которые были сделаны игроками на протяжении игры.

Одним из основных понятий теории игр является *стратегия. Стратегия* – это множество правил, которые определяют выбор варианта хода при сложившейся ситуации в игре. Стратегия является *оптимальной*, если она обеспечивает игроку максимальный средний выигрыш или минимальный средний проигрыш при множественном повторении игры, независимо от стратегий соперников.

1.1.3 Классификация игр

Классификация игр очень разнообразна, поэтому будут приведены лишь некоторые основные характеристики:

- Число игроков – парные и множественные;
- Характер отношений – бескоалиционные (игроки не могут вступать в коалиции, цель игрока – наибольший индивидуальный выигрыш), коалиционные (цель игроков – максимизация выигрыша коалиции без последующего его разделения между

членами коалиции) и кооперативные (выигрыш коалиции делится между членами по заранее оглашенным договоренностям);
- Количество стратегий – конечные и бесконечные;
- Полнота информации – с полной информацией (имеется все информация о предыдущих ходах и возможных стратегиях противника) и с неполной информацией;
- Характер выигрыша – с нулевой суммой (сумма выигрышей всех игроков равна 0, проигрыш одного игрока – выигрыш другого) и с ненулевой суммой и т. д.

1.2 Кооперативные игры

Кооперативная теория игр — это подраздел теории игр, изучающий игры, в которых игроки могут объединяться в коалиции. В отличии от некооперативных игр, здесь стоит на первом месте проблема коалиции игроков, например, раздел выигрыша между членами коалиции.

Кооперативные игры делятся на две группы: Кооперативные игры с трансферабельной полезностью и с нетрансферабельной полезностью.

Кооперативные игры с трансферабельной полезностью – игры, полезность которых измеряется в общепринятых игроками единицах, которые могут быть переданы от игрока к игроку без потерь и трансформаций.

Кооперативная игра с трансферабельной полезностью, с математической точки зрения, — это пара (I, u), задающаяся I – множеством игроков и u – характеристической функцией.

Коалиции игроков – $\forall S \subseteq I$ – любое подмножество из множества игроков.

Характеристическая функция $u(S),\ u: 2^I \to R$ – это функция, которая для каждой коалиции S сопоставляет максимально возможный выигрыш, который она может получить и в последствии разделить между членами коалиции.

Кооперативные игры выполняют свойство супераддитивности – для двух непересекающихся множеств $X \cap Y \neq \emptyset \Rightarrow u(X + Y) \geq u(X) + u(Y)$. Это означает, что добавление игрока к коалиции не уменьшает ее полезность (наибольший выигрыш).

1.3 *C*-Ядро игры

Прежде чем перейти к понятию С-ядра, требуется ввести дополнительное определение.

Дележ – это описание исхода игры. Это вектор $x = (x_1, x_2, \dots, x_n)$, элементы которого удовлетворяют: $\sum_{i \in N} x_i \leq u(I)$.

С-Ядро – множество дележей, для которых выполняется два свойства:

- $\sum_{i \in I} x_i = u(I)$ – условие коллективной рациональности – возможный суммарный выигрыш делится между членами (если денег больше – в игре лишние деньги нарушение условий или меньше – кооперироваться нет смысла)
- $x_i \geq u(C), \forall C \in I$ – условие групповой рациональности.

Семейство коалиций W в I – сбалансированное – если

$$\forall S \in W \; \exists \lambda_S \geq 1 \rightarrow \sum_{S \in W : i \in S} \lambda_S = 1 \; \forall i \in I.$$

Кооперативная игра является сбалансированной, если

$$\sum_{S \in W} \lambda_S u(S) \leq u(I)$$

Из статьи (МАРАКУЛИН, 2019) можно увидеть, что ядро кооперативной игры не пусто тогда и только тогда, когда игра сбалансирована.

1.4 Сильное ε-Ядро

Поскольку С-ядро может быть пустым, обобщение было введено в (Shapley & Shubik, Quasi-cores in a monetary economy with non-convex preferences, 1966). Сильное ε-ядро для некоторого $\varepsilon \in R$ это множество дележей:

$$C_\varepsilon(u) = \left\{ x \in R^V : \sum_{i \in N} x_i = u(I) \, ; \sum_{i \in S} x_i \geq u(S) - \varepsilon, \forall S \subseteq I \right\}$$

С экономической точки зрения сильное ε-ядро — это набор предварительных дележей, при которых ни одна коалиция не может улучшить свой выигрыш, покинув большую коалицию, если она должна заплатить штраф в размере ε для ухода. Обратите внимание, что ε может

быть отрицательным, и в этом случае он представляет собой бонус за выход из большой коалиции. Очевидно, что независимо от того, пусто ли ядро, сильное ε-ядро будет непустым для достаточно большого значения ε и пустым для достаточно маленького (возможно отрицательного) значения ε.

1.5 Вектор Шепли

Вклад игрока i в коалицию $S(i \notin S)$ – величина $u(S \cup i) - u(S)$.

Вектор Шепли –это вектор $\phi(u) = (\phi_1(u), \dots, \phi_n(u))$, удовлетворяющий аксиомам Шепли. Причем этот вектор всегда существует, и он единственный.

С математической точки зрения, вектор Шепли можно представить как

$$\phi_i(u) = \sum_{i \in S} \frac{(|S| - 1)!\,(n - |S|)!}{n!} [u(S) - S \backslash \{i\}]$$

Далее будут приведены *аксиомы Шепли*:

- *Аксиома эффективности*: $u(I) = \sum_{i \in I} u(i)$ – при разделении выигрыша ничего не достанется не членам коалиции.
- *Аксиома симметрии:* если любых 2 игрока делают одинаковый вклад в игру – они получают одинаковый выигрыш.
- *Аксиома агрегатности*: $\phi(u + v) = \phi(u) + \phi(v)$ – при участии игроков в 2 играх их выигрыши в отдельных играх суммируются.

1.6 N-Ядро

В отличие от вектора Шепли, решения по принципу N-Ядра основаны на минимизации неудовлетворённости выигрышем игроков.

Неудовлетворенность измеряется вектором эксцессов:

$$e(S, x) = u(S) - \sum_{i \in S} x_i = u(S) - x(S)$$

Если эксцесс положителен, он представляет собой неудовлетворение из-за текущего распределения выигрыша. Отрицательное же значение обозначает, дополнительный выигрыш, который коалиция S получила в результате дележа x. Таким образом можно утверждать, что дележ x является ядром тогда и только тогда, когда все его эксцессы не положительны.

N-ядром относительно множества X называется точка, соответствующая минимуму отношения лексикографического порядка на множестве всевозможных векторов $e(S,x), x \in X$. (Schmeidler, 1969)

$$\min_{x \in X} \max_{S \subseteq N \setminus \{\emptyset\}} e(S,x)$$

Если множество X совпадает с множеством всех допустимых распределений выигрышей, соответствующее N-ядро называется пред-N-ядром игры (I,u)

1.7 Другие способы дележа

Существует бесконечное множество способов дележа выигрыша. Приведем лишь некоторые из них.

В одном методе, можно разделить выигрыш поровну между игроками:

$$x_i = u(i) - \frac{1}{|I|}\left[\sum_{j \in N} u(j) - u(I)\right]$$

Существует метод Мориарти, или метод пропорций:

$$x_i = u(i) - \frac{u(i)}{\sum_{j \in I} u(j)}\left[\sum_{k \in I} u(k) - u(I)\right] = \frac{u(i)}{\sum_{j \in I} u(j)} u(I)$$

Метод справедливого разделения без предельных издержек. Назовем предельной издержкой $M(i) = u(I) - u(I \setminus \{i\})$. Тогда выигрыши будут считаться так:

$$x_i = M(i) + \frac{1}{|I|}\left[u(I) - \sum_{j \in I} M(j)\right]$$

Можно смешать два предыдущих метода:

$$x_i = M(i) + \frac{M(i)}{\sum_{j \in I} M(j)}\left[u(I) - \sum_{k \in I} M(k)\right] = \frac{M(i)}{\sum_{j \in I} M(j)} u(I)$$

Также можно рассмотреть пропорциональное N-ядро, в котором вектор эксцессов определяется по формуле:

$$e(S,x) = \frac{u(S) - \sum_{i \in S} x_i}{u(S)}$$

Наконец, в литературе существует понятие disruptiv nucleolus – соотношение между тем, что потеряли бы $N\backslash S$ и S если x было бы отвергнуто (LITTLECHILD, S and VAIDJA, K , 1976):

$$d(S,x) = \frac{u(I\backslash S) - \sum_{i\in I\backslash S} x_i}{u(S) - \sum_{i\in S} x_i}$$

1.8 Выпуклые кооперативные игры

Одним из важных классов кооперативных игр являются выпуклые кооперативные игры. Они были представленные Шепли в (Shapley L. , 1971) и обладают интуитивным свойством «снежного кома», присущим некоторым играм. В частности, игра является выпуклой, если ее характеристическая функция u является супермодульной:

$$u(S \cup T) + u(S \cap T) \geq u(S) + u(T), \forall S, T \subseteq I$$

Можно показать (Driessen, 1988) что супермодульность u эквивалентна

$$u(S \cup \{i\}) - u(S) \leq u(T \cup \{i\}) - u(T), \forall S \subseteq T \subseteq I\backslash\{i\}, \forall i \in I$$

то есть «стимулы для присоединения к коалиции возрастают по мере роста коалиции» (Shapley L. , 1971), что приводит к вышеупомянутому эффекту снежного кома. Для стоимостных игр неравенства меняются местами, поэтому можно сказать, что стоимостная игра является выпуклой, если характеристическая функция супермодульная.

У выпуклых кооперативных игр есть много хороших свойств:

- Супермодельность тривиально влечет супераддитивность.
- Выпуклые игры полностью сбалансированы: ядро выпуклой игры непусто, а поскольку любая под-игра выпуклой игры выпуклая, ядро любой под-игры также непусто.
- У выпуклой игры есть единственное стабильное множество, совпадающее с ее ядром.
- Вектор Шепли выпуклой игры — это центр тяжести ее C-ядра.

1.9 K-Ядро (kernel)

Пусть $u: 2^N \to R$ будет некоторой игрой, и пусть $x \in R^N$ это некоторый эффективный дележ. Максимальный излишек игрока i над игроком j по отношению к x определяется как

$$s_{ij}^{u}(x) = \max\left\{u(S) - \sum_{k\in S} x_k : S \subseteq N\backslash\{j\}, i \in S\right\}$$

максимальная сумма, которую игрок i может получить без сотрудничества с игроком j, выйдя из большой коалиции N в соответствии с вектором дележа x, предполагая, что другие игроки в коалиции с учетом выхода i удовлетворены своими выигрышами в соответствии с x. Максимальный излишек – это способ измерить переговорную силу одного игрока над другим. К-ядро – это набор дележей x, удовлетворяющих

- $\left(s_{ij}^{u}(x) - s_{ji}^{u}(x)\right) \times \left(x_j - u(j)\right) \leq 0$ и
- $\left(s_{ji}^{u}(x) - s_{ij}^{u}(x)\right) \times \left(x_i - u(i)\right) \leq 0$

для каждой пары игроков i и j. Интуитивно понятно, что игрок i имеет большую переговорную силу, чем игрок j, в отношении дележа x, если $s_{ij}^{u}(x) > s_{ji}^{u}(x)$, но игрок j невосприимчив к угрозам игрока i, если $x_j = u(j)$, потому что он может получить эту выплату работая самостоятельно. К-ядро содержит все дележи, в которых ни один игрок не имеет такой переговорной силы над другим. Эта концепция решения была впервые представлена в (Davis & Maschler, 1965).

ГЛАВА 2. ОБЗОР ЛИТЕРАТУРЫ

Кооперативное планирование как часть теории игр исследуется по многих сферах науки, и в том числе, и в логистике. Одними из хороших примеров литературе по этой сфере будут (LEMAIRE), (Tveita, 2018) а также (Cruijssen, Cools, & Dullaert, 2007).

Многие из концептов, которые в настоящее время используется в теории игр, были введены в середине двадцатого века. Например, концепт вектора Шепли был введен в (Shapley L. S., A value for n-person games, 1953), К-Ядро было введено в (Davis & Maschler, 1965), а N-ядро – в (Schmeidler, 1969), более интуитивно раскрыто в (Maschler, Peleg, & Shapley, 1979). Выпуклые кооперативные игры были введены в (Shapley L. , 1971), а их важные свойства были рассмотрены в (Driessen, 1988).

Концепция самого С-Ядра очень старая – первое упоминание может быть найдено в (Edgeworth, 1881). Современное определение С-Ядра было представлено в (Gillies, 1959) и (Shapley & Shubik, 1969).

Задача составления расписаний поставок на нефтяные платформы судами снабжения широко исследована. (Fagerholt & Lindstad, 2002) сформулировали и решили версию данной проблемы с некоторыми ослабленными ограничениями. (Halvorsen-Weare E. E., Fagerholt, Nonås, & Asbjørnslett, 2012) разработал двухфазный алгоритм для решения данной задачи, который на первом этапе формирует все допустимые маршруты, а затем решает задачу о покрытии множества используя данные прегенерированные маршруты.

Также (Halvorsen-Weare & Fagerholt, 2016) разработал новый метод решения данной задачи на основе транспортной сети и рассмотрел среднее время работы новой модели по сравнению с двухфазным методом. Ранее он сформулировал другую похожую модель в (Halvorsen-Weare E. , 2012), однако она была довольно упрощенной.

(Shyshou, Gribkovskaia, Laporte, & Gilbert, 2012) разработали большой локальный поиск (Large Neighbourhood Search, LNS) алгоритм, который может решать примеры рассматриваемой задачи больших размеров. (Borthen, Loennechen, Wang, Fagerholt, & Vidal, 2017) предложили генетический алгоритм, правда для облегченной задачи рассматривая флот одинаковых кораблей и предполагая, что платформы работают все время.

(Shyshou, A multi-base supply vessel planning problem arising in offshore oil and gas operations, 2010) предложил и сформулировал вариант задачи, в

котором судна снабжения, принадлежащие одной базе, могут обслуживать нефтяные платформы, относящиеся к другой базе.

(Kisialiou, Gribkovskaia, & Laporte, 2018) рассматривают вариант проблемы, включающий в себя неопределенность в виде погодных условий, и решают проблему, генерируя несколько разных маршрутов судна в зависимости от погодных условий, используя метаалгоритм с локальным поиском.

(Halvorsen-Weare, Fagerholt, & Rönnqvist, 2013) применяют одновременно оптимизацию и симуляцию решая проблему перевозки жидкого газа с помощью флота кораблей в неопределенных погодных условиях.

ГЛАВА 3. РАССМАТРИВАЕМАЯ ЗАДАЧА

3.1 Формулировка задачи

В задаче, которую мы рассматриваем, флот судов снабжения осуществляет поставки на нефтяные платформы с базы (в некоторых формулировках с нескольких баз), находящейся на берегу. Обычно груз состоит из оборудования и материалов, нужных на платформах. Данную проблему мы расширим, введя заданное количество компаний (K) желающих осуществить данную операцию, и рассмотрим, как эти компании могут создавать кооперации друг с другом для достижения наименьших затрат для себя. Следует отметить, что рассматриваемые операции крайне дорогостоящие, следовательно и затраты компаний должны сократиться значительно.

Рассмотрим формулировку задачи, приводимую, например, в (Kisialiou, Gribkovskaia, & Laporte, 2018). Данную задача обычно называют Проблемой планирования расписаний судов снабжения (Periodic Supply Vessel Planning Problem). В данной формулировке, существует множество баз, каждая из которых имеет множество нефтяных платформ, которые она обслуживает, и флот судов снабжения, которые этим занимаются. В данной формулировке базы действуют независимо друг от друга, поэтому, не меняя общности задачи, можно считать, что мы имеем дело только с одной базой. Однако, в литературе существуют более сложные формулировки рассматриваемой задачи, в одной из которых суда не всегда привязаны к определенной базе и могут обслуживать любые нефтяные платформы независимо от базы которая обслуживает данную платформу (Shyshou, A multi-base supply vessel planning problem arising in offshore oil and gas operations, 2010).

Каждая платформа требует осуществление доставки определенное количество раз в неделю. Также каждая нефтяная платформа имеет следующие параметры: количество груза, который нужно доставить в сумме за все поставки за неделю; коэффициент, определяющий время, требуемое на погрузочные / отгрузочные операции, осуществляемые судами снабжения, и время, в которое платформа работает и сможет принимать корабли (некоторые платформы закрыты на ночь).

Суда снабжения могут быть разными, и каждое судно снабжения описывается следующими параметрами: его вместимость, стоимость его работы за день, его скорость передвижения, сколько топлива он тратит за определенный период времени, и коэффициент корабля, влияющий на время,

требуемое на погрузочные / отгрузочные операции. По итогу, время на осуществление данных операций это линейная функция, зависящая от двух параметров – коэффициента платформы и корабля.

Каждая база также имеет определенные параметры – такие как время работы, максимальное количество отбытий / прибытий кораблей за один день. Также база обычно задает такие ограничения, как максимальная продолжительность одного путешествия корабля, и максимальное количество нефтяных платформ, которые он может посетить за это путешествие.

Главная цель рассматриваемой задачи – составить расписание поставок груза на платформы так чтобы потребности этих платформ были удовлетворены, а стоимость данных поставок (цена топлива и стоимость использования кораблей) была минимизирована. Такое расписание состоит из множества кораблей, которые были использованы для поставок и множества путешествий, которые совершил каждый из кораблей. Каждое путешествие задается временем отправления с базы, множества нефтяных платформ, которые посетит данное судно в определенном порядке, и продолжительности путешествия. Расстояние как между базой и платформами, так и между платформами считается известным. На рисунке ниже представлен пример путешествий, осуществляемых двумя суднами с одной базы в течение недели.

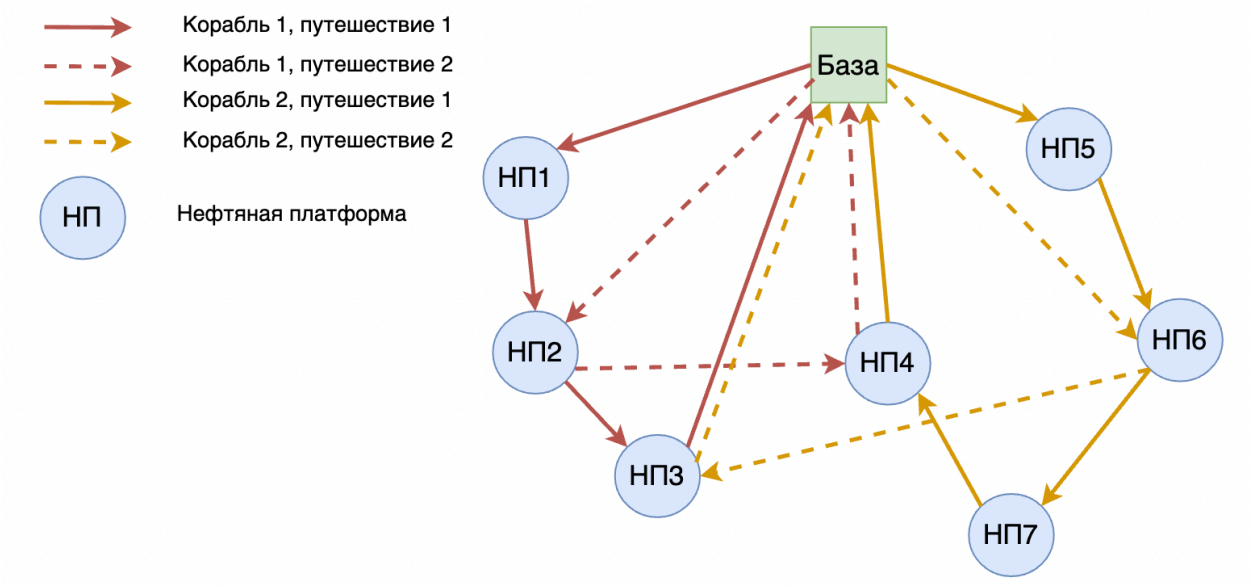

Рисунок 1. Граф путешествий, осуществляемых двумя суднами в течение недели.

Следует заметить, что на практике корабли проводят в одном путешествие в среднем 2-3 дня, соответственно, обычный корабль может совершить 2-3 путешествия за неделю. Пример расписания для двух кораблей, рассмотренных выше, можно посмотреть на рисунке 2.

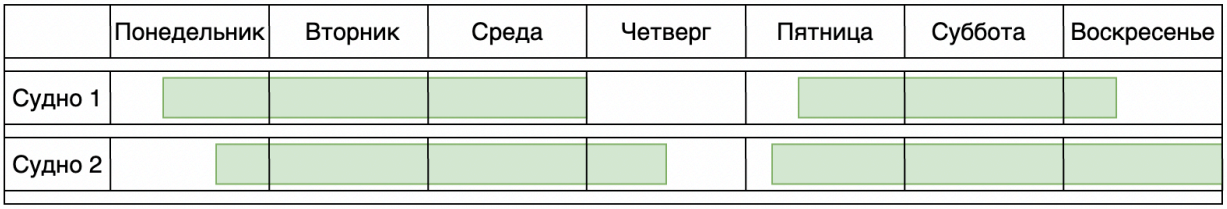

Рисунок 2. Расписание путешествий, осуществляемых двумя суднами в течение недели.

3.2 Модель на основании транспортной сети

Один из способов решения поставленной задачи был представлен в (Halvorsen-Weare & Fagerholt, 2016).

Представляем модель для задачи планирования расписаний поставок на нефтяные платформы судами снабжения. Прежде чем представить общую формулировку модели, мы начнем с введения обозначений.

В базовой транспортной сети для рассматриваемой проблемы узел представляет собой нефтяную платформу или базу. Пусть множество N содержит все узлы. Множество V содержит все доступные суда снабжения. Множество T содержит все периоды времени (дни). Длина горизонта планирования определяется как $|T|$ (семь дней / одна неделя).

C_v^{TC} – это еженедельная стоимость использования для суда снабжения $v, v \in V$. Тогда C_{vij}^{S} – это все затраты связанные с судом снабжения v, который обслуживает нефтяную платформу i, а затем отправляющийся на начало обслуживания нефтяной платформы j,$i, j \in V$. T_{vij} – это время, затраченное судом снабжения v на обслуживание нефтяной платформы i на путешествие от нефтяной платформы i к j, в часах. Индекс o представляет собой базу, $o \in N$. Время обслуживания на базе определяется как минимальное время между путешествиями. S_i – сколько нужно посетить нефтяную платформу i в неделю. T^{MN}и T^{MX}– это минимальная и максимальная продолжительность путешествия в днях, а R^{MN}и R^{MX} представляют собой минимальное и максимальное количество нефтяных платформ, которые можно посетить во время путешествия. D_i – это необходимая вместимость суда снабжения для посещения нефтяной

платформы i, а Q_v – это вместимость суда снабжения v. H – продолжительность периода времени в часах (24), а B_t – максимальное количество судов снабжения, которое может обслуживаться на базе в день $t, t \in T$.

Бинарная переменная δ_v равна 1, если судно снабжения v используется, и 0 в противном случае. Бинарная переменная y_{vijt} равна 1, если судно снабжения v начинает путешествие в день t и путешествует напрямую из i в j в этом путешествии, и 0 в противном случае. Бинарная переменная z_{vit} равна 1, если судно снабжения v посещает узел i в путешествии, начинающемся в день t, и 0 в противном случае. Непрерывная переменная w_{vjt} является переменной для времени ожидания и равна количеству часов, в течение которых судно снабжения v, которое начинает рейс в день t, ожидает до начала обслуживания на нефтяной платформе j по прибытии на нее. Наконец, переменная d_{vt} – это продолжительность путешествия (в целых днях), совершаемого судном снабжения v, которое начинает путешествие в день t.

Тогда целевая функция в данном случае будет выглядеть следующим образом:

$$min \sum_{v \in V} C_v^{TC} \delta_v + \sum_{v \in V} \sum_{i \in N} \sum_{j \in N: i \neq j} \sum_{t \in T} C_{vij}^{S} y_{vijt} \quad (1)$$

А ограничения будут следующими:

$$z_{vit} = \sum_{j \in N: i \neq j} y_{vijt}, v \in V, i \in N, t \in T \quad (2)$$

$$\sum_{v \in V} \sum_{t \in T} z_{vit} \geq S_i, i \in N/\{o\} \quad (3)$$

$$\sum_{j \in N} y_{vjit} - \sum_{j \in N} y_{vijt} = 0, v \in V, i \in N, t \in T \quad (4)$$

$$\sum_{j \in N} y_{vojt} - z_{vit} \geq 0, v \in V, i \in N, t \in T \quad (5)$$

$$\sum_{j \in N} y_{vjot} - z_{vit} \geq 0, v \in V, i \in N, t \in T \quad (6)$$

$$\sum_{j \in N} y_{vijt} \leq \delta_v, v \in V, i \in N, t \in T \quad (7)$$

$$d_{vt} = \left[\left(\sum_{i\in N}\sum_{j\in N: i\neq j}\left(T_{vij}y_{vijt} + \sum_{j\in N} w_{vjt}\right)\right)\frac{1}{H}\right], v \in V, t \in T \ (8)$$

$$T^{MN}z_{vot} \leq d_{vt} \leq T^{MX}z_{vot}, v \in V, t \in T \ (9)$$

$$R^{MN}z_{vot} \leq \sum_{i\in N/\{o\}} z_{vit} \leq R^{MX}z_{vot}, v \in V, t \in T \ (10)$$

$$\sum_{i\in N/\{o\}} D_i z_{vit} \leq Q_v, v \in V, t \in T \ (11)$$

$$\sum_{v\in V} z_{vot} \leq B_t, t \in T \ (12)$$

$$w_{vjt} \geq 0, v \in V, j \in N, t \in T \ (13)$$

$$\delta_v \in \{0,1\}, v \in V \ (14)$$

$$y_{vijt} \in \{0,1\}, v \in V, i, j \in N, t \in T \ (15)$$

$$z_{vit} \in \{0,1\}, v \in V, i \in N, t \in T \ (16)$$

Целевая функция (1) сводит к минимуму затраты на использование судов снабжения и стоимость плавания для путешествий. Ограничения (2) определяют переменные z_{vit} равными 1, если судно снабжения v посещает нефтяную платформу i во время путешествия, начинающегося в день t. Ограничения (3) гарантируют, что нефтяные платформы получат необходимое количество посещений в течение рассматриваемого горизонта планирования (недели). Ограничения (4) – обеспечивают что если судно снабжения посетило нефтяную платформу, то оно должно и отплыть от нее, а ограничения (5) – (6) гарантируют, что путешествия начинаются и заканчиваются на базе. (4) и (5) сами по себе гарантируют, что судно снабжения, который отправляется из базы, также вернуться на базу. Следовательно, ограничения (6) избыточны, но они были сохранены в формулировке модели для того, чтобы модель была последовательной. Ограничения (7) гарантируют, что будет максимум один выезд с базы в заданный день для определенного судна снабжения и максимум одно посещение любой данной нефтяной платформы во время рейса. Ограничения также гарантируют, что, если судно снабжения v отправляется в путешествие, δ_v должно быть равным 1. Ограничения (8) устанавливают значение переменных продолжительности путешествия d_{vt} равными сумме всего времени обслуживания, плавания и ожидания, округленных до

ближайшего целого периода времени (день). Ограничения (9) ограничивают продолжительность путешествий в пределах минимального и максимального количества дней и гарантируют, что d_{vt} равно 0, если судно снабжения v не отправляется в путешествие в день t. Ограничения (10) ограничивают количество посещенных нефтяных платформ во время путешествия в пределах минимума и максимума, ограничения (11) – это ограничения вместимости суден снабжения, а ограничения (12) – ограничения базы, ограничивающие количество суден снабжения, которые могут быть подготовлены к путешествию, начинающегося со дня t. Требования неотрицательности для переменных w_{vjt} устанавливаются ограничениями (13), а ограничения (14) $-$ (16) устанавливают, что переменные δ_v, y_{vijt} и z_{vit} должны быть бинарными.

Чтобы избежать пересечения путешествий судов снабжения друг с другом, необходимо добавить следующие ограничения:

$$t\sum_{j\in N} y_{vojt} + d_{vt} \leq \big((t+p)mod|T|\big)\sum_{j\in N} y_{voj((t+1)mod|T|)} + M_t\left(1-\sum_{j\in N} y_{voj((t+p)mod|T|)}\right), p\in\{1,2\}, v\in V, t\in T \quad (17)$$

Для $p = 1$ ограничения (17) гарантируют, что если судно снабжения v начинает путешествие в день t, оно не может начать новое путешествие в день $t+1$, если продолжительность путешествия составляет более одного дня, и для $p = 2$ они гарантируют, что судно снабжения не сможет начать новые путешествие в день $t+2$, если продолжительность путешествия превышает два дня. Это выражение, умноженное на M_t, гарантирует, что ограничения действительны, если путешествия не начинаются в день $t+1$ или $t+2$. Значение M_t равно $t+T^{MX}$, поскольку левая часть ограничений никогда не может быть больше, чем $t+d_{vt}$ и d_{vt} не может превышать T^{MX}. Модуль используется в данном ограничении, поскольку задача планирования расписаний судов снабжения – это периодическая проблема маршрутизации, при которой расписание повторяется каждые $|T|$ дней. Следовательно, модуль обеспечивает, что, если $t = |T|$, $t+1 = 1$ и так далее.

Чтобы предотвратить циклы в путешествия и ввести временные окна (периоды времени, в которые нефтяные платформы могут обслуживаться),

добавляются временные переменные. Пусть L_{iu}^{E} и L_{iu}^{L}будут самым ранним и самым поздним временем начала обслуживания на нефтяной платформе i, если обслуживание начинается в день u, $u \in T$. Пусть переменная a_{vitu} равна единице, если нефтяная платформа i обслуживается судном снабжения v в день u путешествия, начинающегося в день t, и нулю в противном случае. Эта переменная может быть ненулевой, только если бинарная переменная z_{vit} равна единице. Временная переменная τ_{vit} – это время начала обслуживания на нефтяной платформе i судном снабжения v в путешествии, начинающемся в день t. Тогда можно ввести следующие ограничения:

$$y_{vijt}\left(\tau_{vit} + T_{vij} + w_{vjt} - \tau_{vjt}\right) = 0\,, v \in V, i, j \in N, t \in T \quad (18)$$

$$\sum_{\mu=0}^{2} a_{vit((u+\mu)mod|T|)} L_{i((u+\mu)mod|T|)}^{E} \leq \tau_{vit} \leq \sum_{\mu=0}^{2} a_{vit((u+\mu)mod|T|)} L_{i((u+\mu)mod|T|)}^{L}\,, v \in V, i \in N, t \in T, u = t \quad (19)$$

$$\sum_{\mu=0}^{2} a_{vit((u+\mu)mod|T|)} = z_{vit}\,, v \in V, i \in N, t \in T, u = t \quad (20)$$

Ограничения (18) гарантируют, что, если судно снабжения отправляется из нефтяной платформы i в j во время путешествия, его обслуживание на нефтяной платформе j не может начаться, пока судно снабжения не успеет закончить обслуживание на нефтяной платформе i и отправиться от нефтяной платформы i к j. Это ограничения равенства, и ожидание перед началом обслуживания определяется путем присвоения переменной времени ожидания, w_{vjt}, значения больше нуля. Ограничения нелинейны, но их легко линеаризовать. Линеаризованные ограничения выражаются следующим образом:

$$\left(\tau_{vit} + T_{vij} + w_{vjt} - \tau_{vjt}\right) + M_1 y_{vijt} \geq M_1 \quad (21)$$

$$\left(\tau_{vit} + T_{vij} + w_{vjt} - \tau_{vjt}\right) + M_2 y_{vijt} \leq M_2 \quad (22)$$

Константы M_1 и M_2 это верхняя и нижняя границы выражения $\tau_{vit} + T_{vij} + w_{vjt} - \tau_{vjt}$.

Ограничения (19) – это ограничения временных окон, которые гарантируют, что обслуживание нефтяной платформе начинается в пределах временного окна в день начала обслуживания. Ограничения (20)

гарантируют, что обслуживание на нефтяной платформе может начаться только в один определенный день, если нефтяная платформа была посещена во время путешествия.

Ограничения $(17) - (20)$ достаточны, если максимальная продолжительность путешествий составляет три дня, и дополнительные ограничения должны быть добавлены, если допустимы более длительные путешествия.

Для обеспечения равномерного распределения начала путешествий ("spread" ограничения) введем следующие множества и параметры: множество K содержит потенциальные частоты еженедельных посещений нефтяной платформы. Тогда подмножество $N_k \subseteq N$ содержит нефтяные платформы, требующие k еженедельных посещений. $G_k \in [0, |T|]$ представляет длину подгоризонта для нефтяных установок с частотой посещения k, а $\underline{P_k}$ и $\overline{P_k}$ – нижняя и верхняя границы количества посещений за субгоризонт длины G_k. Тогда определяются следующие ограничения для равномерно распределенных времен отправления:

$$\sum_{v \in V} z_{vit} \leq 1\,, i \in N/\{o\}, t \in T \quad (23)$$

$$\underline{P_k} \leq \sum_{v \in V} \sum_{\mu=0}^{G_k} z_{vi((t+\mu) mod |T|)} \leq \overline{P_k}\,, k \in \{2,3,4,5\}, i \in N_k, t \in T \quad (24)$$

Ограничения (23) гарантируют, что в день t будет проводиться максимум одно путешествие с посещением любой нефтяной платформы i. Они также гарантируют, что на любую нефтяную платформу, которую посещают во время путешествия, будет не более одного посещения, и, следовательно, делают ограничения (7) избыточными, за исключением случая, когда $i = o$. Этих ограничений достаточно для нефтяных платформ, требующих шести или семи посещений в неделю. Ограничения (24) гарантируют, что в течение заданного количества последовательных дней начала путешествия будет минимальное и / или максимальное количество посещений нефтяной платформы с частотой посещений k. $\underline{P_k}$ равно 0 для частоты посещений 2, 1 для частот посещений 3 и 5, и 2 для частоты посещений 4. $\overline{P_k}$ равно 1 для частоты посещений 2 и установлено на большое значение для всех остальных частот посещений. Ограничения $(23) - (24)$ не обеспечивают в строгом смысле равномерного распределения начала

путешествий, но они гарантируют, что начала путешествий в некоторой степени распределены в течение недели.

Ограничения $(23) - (24)$ не гарантируют, что фактические посещения нефтяных платформ распределяются. Следовательно, два посещения конкретной нефтяной установки могут быть очень близкими или даже могут быть запланированы одновременно, если нефтяная установка посещается поздно во время рейса, начинающегося в один день, и рано во время рейса, начинающегося на следующий день. Этого можно избежать, добавив также ограничения, которые будут подробнее рассмотрены в конце следующей части.

3.3 Двухфазный метод с прегенерацией возможных путешествий

Другой из способов решения поставленной задачи был представлен в (Halvorsen-Weare E. E., Fagerholt, Nonås, & Asbjørnslett, 2012). Данный метод состоит их двух этапов: на первом этапе для каждого судна генерируются возможные путешествия, а на втором шаге решается модель, использующая построенные маршруты.

3.3.1 Генерация возможных путешествий

Опишем псевдокод, использующийся для прегенерации возможных путешествий. Первым шагом будет сгенерировать все возможные подмножества нефтяных установок, которые может посетить данное судно снабжения. Размер данных подмножеств будет зависеть как от максимального количества нефтяных платформ, которые могут быть посещены во время одного путешествия, так и от вместимости судна. Затем для каждого такого подмножества нам нужно решить задачу коммивояжера, причем ее усиленный вариант, включающий в себя временные окна. Если подмножество не содержит нефтяных платформ, работающих только в определенное время суток, то решается обычная задача коммивояжера.

Расстояния между всеми точками маршрута считаются известными, поэтому зная скорость судна, можно рассчитать время, которое он будет находиться в пути. Стоимость каждого путешествия вычисляется как потребление топлива, помноженное на стоимость топлива. Считается, что корабль имеет два режима потребление топлива – в дрейфующем состоянии и во время движения. Задачу коммивояжера будем решать, находя маршрут с

наименьшей продолжительностью (по времени), который почти всегда будет и маршрутом с наименьшей стоимостью. Будем решать задачу с помощью метода полного перебора.

Продолжительность возможных путешествий рассчитывается следующим образом: начиная с базы, рассчитаем время плавания до следующей нефтяной платформы основываясь на расстоянии и скорости судна. Далее, если во время прибытия судна нефтяная платформа закрыта, то время ожидания до ее открытия добавляется к общей продолжительности маршрута. Когда нефтяная платформа открывается для обслуживания, к общей продолжительности маршрута добавляется время обслуживания. Если операции по разгрузке / отгрузке не могут быть завершены до закрытия платформы, то к общей продолжительности маршрута добавляем 12 часов (чтобы данный маршрут имел намного меньший приоритет). Далее рассчитываем время перехода к следующей нефтяной платформе и так далее, пока корабль не вернется обратно на базу.

Псевдокод процедуры генерации всех возможных рейсов представлен ниже. Результатом работы процедуры является набор всех возможных рейсов (обозначен как R), которые могут пройти суда.

1: **Процедура генерации путешествий**
2: **Создать** множество кораблей $VesselSet$ с одинаковой скоростью плавания.
3: **Для всех** $VesselSet$:
4: **Найти** корабль в $VesselSet$ с наибольшей вместимостью, $VesselMax$
5: **Перебрать** все множества нефтяных платформ $InstallationSet$ которые удовлетворяют требования по максимальному количеству нефтяных платформ
в путешествии и чьи требования не превышают вместимость $VesselMax$
6: **Для всех** $InstallationSet$:
7: **Найти** маршрут $voyage$ решая задачу коммивояжера с временными окнами
начиная и заканчивая маршрут на базе и посещая каждую нефтяную платформу в $InstallationSet$ ровно один раз
8: **Если** $voyage$ не нарушает верхних ограничений по времени:
9: **Добавить** $voyage$ в $VoyageSet$ для $VesselMax$ ($VoyageSet[VesselMax]$)
10: **Конец Если**
11: **Конец Для всех** $InstallationSet$
12: **Для всех** кораблей v$essel$ в $VesselSet$ **НЕ** $VesselMax$

13: **Для всех** путешествий $voyages$ в $VoyageSet[VesselMax]$
14: **Если** $voyage$ не нарушает верхних ограничений по вместимости v$essel$:
15: **Добавить** $voyage$ в $VoyageSet[\text{v}essel]$
16: **Конец Если**
17: **Конец Для всех** путешествий $voyages$
18: **Конец Для всех** кораблей v$essel$ в $VesselSet$
19: **Конец Для всех** $VesselSet$
20: **Возвратить Все** $VoyageSets$

3.3.2 Второй этап – математическая модель

Математическая модель будет решать текущую проблему выбирая наиболее экономически эффективные суда снабжения и выбирая лучшие из прегенерированных путешествий, которые в комбинации удовлетворяют ограничения.

Пусть V – множество судов снабжения доступных во время рассматриваемого периода времени, N – множество всех нефтяных платформ. Тогда множество R_v состоит из всех прегенерированных путешествий, по которым может плыть судно $v \in V$. Далее, пусть T это множество дней рассматриваемого периода времени (неделя), и L это множество, содержащее все возможные продолжительности путешествий в днях (2 или 3). Тогда подмножество $R_{vl} \subseteq R_v$ содержит все рассматриваемые путешествия продолжительности $l \in L$ по которым может плыть судно $v \in V$.

Обозначим стоимость обслуживания корабля v за неделю как ch_v, а стоимость его путешествия по маршруту r как c_{vr}. Далее, D_{vr} это длительность путешествия r корабля v округленная вверх до ближайшего целого числа, F_v это количество дней в неделе на протяжении которых судно v может использоваться, S_i это количество визитов нужных нефтяной платформе i за неделю, B_t – максимальное количество суден снабжения, которое возможно обслужить в день t на базе, и A_{vir} это бинарный параметр равный 1 тогда и только тогда, когда судно v посещает нефтяную платформу i во время путешествия r.

Для построения модели введем переменную бинарную переменную y_v равную 1 тогда и только тогда, когда судно v использовано, и бинарную переменную x_{vrt} равную 1 тогда и только тогда, когда судно v осуществляет путешествие r в день t.

Тогда целевая функция в данном случае будет выглядеть следующим образом:

$$min \sum_{v \in V} ch_v y_v + \sum_{v \in V} \sum_{r \in R_v} \sum_{t \in T} c_{vr} x_{vrt} \quad (1)$$

А ограничения будут следующими:

$$\sum_{v \in V} \sum_{r \in R_v} \sum_{t \in T} A_{vir} x_{vrt} \geq S_i, i \in N \quad (2)$$

$$\sum_{r \in R_v} \sum_{t \in T} D_{vr} x_{vrt} - F_v y_v \leq 0 \, , v \in V \quad (3)$$

$$\sum_{v \in V} \sum_{r \in R_v} x_{vrt} \leq B_t, t \in T \; (4)$$

$$\sum_{r \in R_{vl}} x_{vrt} + \sum_{r \in R_v} \sum_{k=0}^{l-1} x_{vr,((t+k) mod |T|)} \leq 1, v \in V, t \in T, l \in L \; (5)$$

$$y_v \in \{0,1\}, v \in V \; (6)$$

$$x_{vrt} \in \{0,1\}, v \in V, r \in R_v, t \in T \; (7)$$

Целевая функция (1) минимизирует сумму затрат на обслуживание корабля и на стоимость его путешествий. Так как стоимость обслуживания корабля намного больше, чем стоимость топлива на путешествия, главная задача найти наиболее эффективную структуру флота кораблей. Ограничения (2) обеспечивает получение всеми нефтяными платформами нужного количества визитов за рассматриваемый период времени, а ограничения (3) проверяют что полная продолжительность всех путешествий корабля не превышает количество дней, которые он будет доступен во время рассматриваемого временного промежутка. Также эти ограничения обеспечивают тот факт, что если судно v используется, то соответствующая переменная y_v должна быть равна 1. Далее, ограничения (4) обеспечивают, что в день t на базе находится не более суден, чем она может обслужить. Ограничения (5) гарантируют что судно не начнет новое путешествие до того, как оно вернулось из прошлого.

Также, для данного вида задач обычно добавляются так называемые “spread” ограничения (ограничения распределения). Эти ограничения гарантируют, что начала путешествий суден (то есть день их отплытия с базы) распределены в течение недели. Эти ограничения определяются на основании числа визитов, которые требует конкретная нефтяная платформа в

течение рассматриваемого периода времени (недели). Введем 4 дополнительных множества N_2, N_3, N_4 и N_5, требующие 2, 3, 4 и 5 визитов в неделю соответственно. Тогда ограничения могут быть сформулированы следующим образом:

$$\sum_{v \in V} \sum_{r \in R_v} A_{vir} x_{vrt} \leq 1 , i \in N, t \in T \quad (8)$$

$$\sum_{v \in V} \sum_{r \in R_v} \sum_{k=0}^{2} A_{vir} x_{vr((t+k) mod |T|)} \leq 1 , i \in N_2, t \in T \quad (9)$$

$$\sum_{v \in V} \sum_{r \in R_v} \sum_{k=0}^{2} A_{vir} x_{vr((t+k) mod |T|)} \geq 1 , i \in N_3, t \in T \quad (10)$$

$$\sum_{v \in V} \sum_{r \in R_v} \sum_{k=0}^{3} A_{vir} x_{vr((t+k) mod |T|)} \geq 2 , i \in N_4, t \in T \quad (11)$$

$$\sum_{v \in V} \sum_{r \in R_v} \sum_{k=0}^{1} A_{vir} x_{vr((t+k) mod |T|)} \geq 1 , i \in N_5, t \in T \quad (12)$$

Ограничения (8) обеспечивает что максимум одно путешествие, начатое в день t посетит определенную нефтяную платформу i. Это ограничение достаточно для нефтяных платформ, которые требуют 6 или 7 посещений в течении недели. Ограничения (9) гарантируют, что для нефтяных платформ, требующих двух посещений в неделю, будет не более одного выезда из базы в течение любых трех дней. Далее, ограничения (10) гарантируют, что будет хотя бы один выезд из базы в течение любого трехдневного периода для нефтяных платформ, требующих трех посещений в неделю. Ограничения (11) гарантируют, что нефтяные платформы, требующие четырех посещений в неделю, будут иметь как минимум два выезда из базы в течение любого четырехдневного периода. Наконец, для установок, требующих пяти посещений в неделю, ограничения (12) гарантируют, что будет хотя бы один выезд из базы в течение любого двухдневного периода.

Ограничения (8) - (12) — это всего лишь один из способов формулирования ограничений, обеспечивающих осуществления ограничений распределения, есть другие варианты формулировки. Некоторые из них будут эквивалентны этим ограничениям, в то время как другие предоставят меньшее или большее пространство решений. Например, ограничения (9)

можно сформулировать по-другому, обеспечив минимум одно посещение в течение любого четырехдневного периода. Это приведет к тому же пространству решений. Ограничения (10) можно заменить ограничениями, гарантирующими, что в течение любого двухдневного периода будет максимум одно посещение. Это обеспечило бы более сильную формулировку, поскольку ограничения (10) допускают отправления в течение двух последовательных дней, сокращая пространство решений. В таблице 1 показаны некоторые примеры схем начала путешествий для нефтяных платформ, требующих двух, трех, четырех и пяти посещений в неделю, которые возможны с учетом ограничений (8) - (12).

Количество посещений	Понедельник	Вторник	Среда	Четверг	Пятница	Суббота	Воскресенье
Два	x			x			
Два	x				x		
Три	x		x		x		
Три	x			x		x	
Четыре	x	x			x	x	
Четыре	x		x		x		x
Пять	x	x		x	x		x
Пять	x	x	x		x	x	

Таблица 1. Некоторые паттерны посещений.

Ограничения, представленные выше, гарантируют, что отправления судов к каждой нефтяной платформе распределяются в течение недели. Но эти ограничения не препятствуют одновременному нахождению более чем одного судна на одной нефтяной платформе. Например, определенную нефтяную платформу можно посетить поздно во время рейса, начинающегося во вторник, и рано во время рейса, начинающегося в среду, и это опять же может привести к тому, что два судна, совершающие рейсы, достигают нефтяную платформу примерно в одно и то же время. В большинстве практических ситуаций это будет нежелательно, поскольку два судна не смогут обслуживаться на нефтяной платформе одновременно.

Чтобы предотвратить такие события, могут быть добавлены ограничения, которые принимают времена прибытия и отправления судна на нефтяные платформы во время путешествия в качестве входных данных и предотвращают одновременное нахождение двух судов на установке.

Пусть G_{vir} и H_{vir} будут временем прибытия и отправления с нефтяной платформы i судном v рейса r соответственно. Время прибытия и отправления указывается в часах после отправления с базы и рассчитывается на этапе генерации маршрута. Кроме того, пусть U будет временем выполнения работ на базе, а P представляет собой минимальное количество часов, необходимое между запланированным отправлением и прибытием на нефтяную платформу. Затем можно добавить следующие ограничения:

$$\sum_{v\in V}\sum_{r\in R_v} A_{vir}H_{vir}x_{vrt} \le (48-U)\left(1-\sum_{v\in V}\sum_{r\in R_v} A_{vir}x_{vr((t+1)mod|T|)}\right) + \sum_{v\in V}\sum_{r\in R_v} A_{vir}x_{vr((t+1)mod|T|)}\,(24+G_{vir}-P)\,, i\in N, t\in T \quad (13)$$

$$\sum_{v\in V}\sum_{r\in R_v} A_{vir}H_{vir}x_{vrt} \le (72-U)\left(1-\sum_{v\in V}\sum_{r\in R_v} A_{vir}x_{vr((t+2)mod|T|)}\right) + \sum_{v\in V}\sum_{r\in R_v} A_{vir}x_{vr((t+2)mod|T|)}\,(48+G_{vir}-P)\,, i\in N, t\in T \quad (14)$$

Ограничения (13) гарантируют, что в случае отправления судна на нефтяную платформу i в день $t\ +\ 1$ время прибытия этого судна на нефтяную платформу должно составлять минимум P часов после отправления судна, посещающего ту же нефтяную платформу и отправляющегося с базы в день t. Точно так же ограничения (14) гарантируют, что судно, отправляющееся в день $t\ +\ 2$, прибудет на нефтяную платформу как минимум через P часов после отправления судна, отправляющегося с базы в день t. Этих ограничений достаточно, если максимальная продолжительность рейса составляет три дня.

Поскольку на этапе прегенерации путешествий создается только путешествия с оптимальной последовательностью посещений для данного набора нефтяных платформ, а не все возможные путешествия, добавление этих ограничений может нарушить оптимальность решения и перевести метод решения на основе прегенерации путешествий из точного метода в эвристический. Поэтому алгоритм прегенерации рейсов был изменен и был добавлен модуль прегенерации двойного путешествия. Этот модуль дает возможность создать два путешествия для данного набора нефтяных платформ вместо одного. Второе путешествия создается следующим образом: последняя нефтяная платформа, которая посещалась во время первого путешествия, будет посещена первой во втором путешествии, если

эта нефтяная платформа не имеет часов работы; если имеет, то пусть предпоследняя посещенная платформа будет первой и так далее. Дальнейшая последовательность посещения во время второго путешествия определяется путем нахождения наилучшего способа посещения оставшихся нефтяных платформ. В большинстве случаев это приведет к двум сгенерированным путешествиям, и второе путешествие будет близко к обратному первому путешествию.

3.4 Проведение эксперимента

Рассмотрим упрощенный пример описанной выше задачи, а именно предположим, что:

- Флот кораблей состоит из кораблей одного типа, то есть все их характеристики совпадают.
- Как база, так и нефтяные платформы работают круглосуточно.
- Время обслуживания одинаково на всех нефтяных платформах
- Каждое судно снабжения доступно весь рассматриваемый период времени (неделю).
- База может обслужить любое количество суден за день

В новой формулировке математическая модель будет выглядеть следующим образом:

$$min \sum_{v \in V} ch * y_v + \sum_{v \in V} \sum_{r \in R} \sum_{t \in T} c_r x_{vrt} \quad (1)$$

$$\sum_{v \in V} \sum_{r \in R} \sum_{t \in T} A_{ir} x_{vrt} \geq S_i, i \in N \quad (2)$$

$$\sum_{r \in R} \sum_{t \in T} D_r x_{vrt} - F * y_v \leq 0 \, , v \in V \quad (3)$$

$$\sum_{r \in R_l} x_{vrt} + \sum_{r \in R} \sum_{k=0}^{l-1} x_{vr,((t+k) mod |T|)} \leq 1, v \in V, t \in T, l \in L \; (4)$$

$$y_v \in \{0,1\}, v \in V \; (5)$$

$$x_{vrt} \in \{0,1\}, v \in V, r \in R, t \in T \; (6)$$

Пусть существует 3 различные компании, каждая из которых хочет поставлять грузы на некоторое количество из существующих 10 нефтяных платформ. Приведем в таблице количество визитов нужных каждой нефтяной платформе за неделю от определенной компании.

	НП 1	НП 2	НП 3	НП 4	НП 5	НП 6	НП 7	НП 8	НП 9	НП1 0
Компания 1	3	1	4	2	3	2	–	–	–	1
Компания 2	2	–	5	1	–	–	2	2	–	2
Компания 3	–	–	2	3	–	–	–	2	4	–

Таблица 2. Количество поставок нужных каждой нефтяной платформе за неделю.

Также приведем таблицу расстояний от базы до нефтяных платформ и между самими нефтяными платформами (в километрах):

	База	НП 1	НП 2	НП 3	НП 4	НП 5	НП 6	НП 7	НП 8	НП 9	НП1 0
База	0	538	500	316	522	540	282	608	427	500	492
НП1	538	0	200	360	350	502	500	894	694	860	955
НП2	500	200	0	223	150	304	360	721	531	707	832
НП3	316	360	223	0	206	250	141	538	335	500	610
НП4	522	350	150	206	0	158	304	602	430	610	761
НП5	540	502	304	250	158	0	269	460	316	492	667
НП6	282	500	360	141	304	269	0	412	206	360	471
НП7	608	894	721	538	602	460	412	0	206	141	335
НП8	427	694	531	335	430	316	206	206	0	180	353
НП9	500	860	707	500	610	492	360	141	180	0	206
НП1 0	492	955	832	610	761	667	471	335	353	206	0

Таблица 3. Расстояния, рассматриваемые в задаче.

Будем считать, что скорость каждого судна равно 46 км/ч. Также положим, что стоимость используемого корабля за одну неделю составляет 200.000$, а количество топливо, сжигаемое одним кораблем за 1 час равноценно 500$.

Положим, что время обслуживания на каждой платформе одинаковое и не зависит от судна – составляет 2 часа. Также будем считать, что корабль тратит по 2 часа при отплытии из базы и по прибытию в нее.

3.4.1 Генерация возможных путешествий для данной формулировки задачи.

Код для прегенерации возможных путешествий расположен в Приложении 1. Код разработан с помощью языка программирования Python с использованием библиотеки ortools.

3.4.2 AMPL модель для данной формулировки задачи.

Приведем модель, описанную с помощью языка математического моделирования AMPL.

```
set N;
set V;
set T;
set L;
set R;
set RL{L};

param h;
param F;
param S{N};
param A{R, N};
param D{R};
param c{R};


var y {V} binary;
var x {V, R, T} binary;


minimize Total_Cost: sum {v in V} y[v] * h + sum {v in V, r in R, t in T} c[r] * x[v, r, t];


subject to C1 {i in N}:
         sum {v in V, r in R, t in T} A[r, i] * x[v, r, t] >= S[i];

subject to C2 {v in V}:
         sum {r in R, t in T} D[r] * x[v, r, t] - F * y[v] <= 0;

subject to C3 {v in V, t in T, l in L}:
         sum {r in RL[l]} x[v, r, t] + sum {r in R, k in 1..l} x[v, r, (t + k) mod 7] <= 1;
```

3.4.3 Результаты.

Поочередно запустим AMPL код, приведенный в прошлом разделе для каждой из коалиций, которые можно образовать. Это компании 1, 2 и 3 раздельно, компании 1 и 2 вместе, компании 1 и 3 вместе, компании 2 и 3 вместе, и наконец, компании 1 и 2, и 3 вместе.

Пример результата, полученного с помощью модели (для компании 1):

```
Total_Cost = 492000

x :=
V7 r12453610 0   1
V7 r3        4   1
V9 r3541     4   1
V9 r6531     1   1
;

y [*] :=
 V7  1
 V9  1
;
```

Представим результат графически:

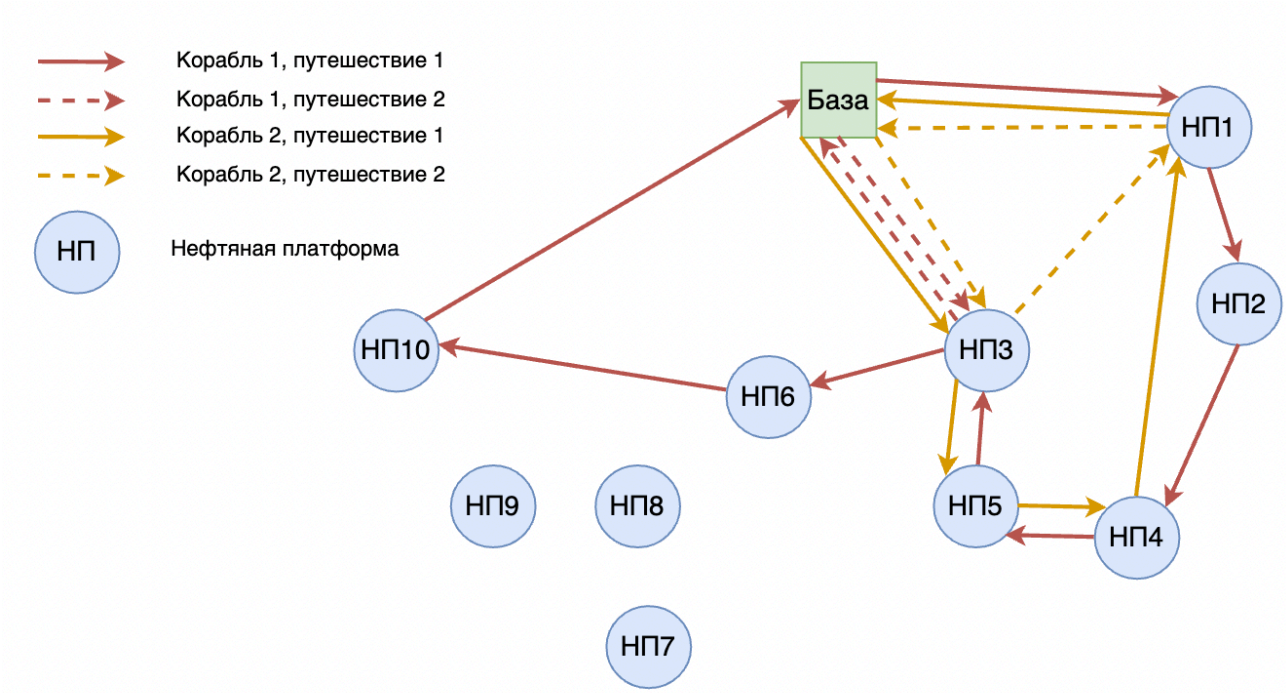

Рисунок 3. Граф путешествий, осуществляемых суднами компании 1 в течение недели.

После выполнения экспериментов, получим следующие результаты:

$$u(1) = 492000; u(2) = 694500; u(3) = 482000; u(1,2) = 1181000;$$

$$u(2,3) = 946500; u(1,3) = 745500; u(1,2,3) = 1424000$$

3.4.3.1 Вектор Шепли

Рассчитаем вектор Шепли для коалиции 1-2-3:

Общая формула:

$$\phi_i(u) = \sum_{i \in S} \frac{(|S|-1)!\,(n-|S|)!}{n!}[u(S) - S\backslash\{i\}]$$

$$\begin{aligned}
\phi_1(u) &= \frac{0!\,2!}{3!}\big(u(1,2,3) - u(2,3)\big) + \frac{1!\,1!}{3!}\big(u(1,2) - u(2)\big) \\
&\quad + \frac{1!\,1!}{3!}\big(u(1,3) - u(3)\big) + \frac{2!\,0!}{3!}\big(u(1) - u(\emptyset)\big) \\
&= \frac{1}{3}(1424000 - 946500) + \frac{1}{6}(1181000 - 694500) \\
&\quad + \frac{1}{6}(745500 - 482000) + \frac{1}{3}(492000) \\
&= \frac{477500}{3} + \frac{243250}{3} + \frac{131750}{3} + \frac{492000}{3} = 448166\frac{2}{3} < u(1)
\end{aligned}$$

$$\begin{aligned}
\phi_2(u) &= \frac{0!\,2!}{3!}\big(u(1,2,3) - u(1,3)\big) + \frac{1!\,1!}{3!}\big(u(1,2) - u(1)\big) \\
&\quad + \frac{1!\,1!}{3!}\big(u(2,3) - u(3)\big) + \frac{2!\,0!}{3!}\big(u(2) - u(\emptyset)\big) \\
&= \frac{1}{3}(1424000 - 745500) + \frac{1}{6}(1181000 - 492000) \\
&\quad + \frac{1}{6}(946500 - 482000) + \frac{1}{3}(694500) \\
&= \frac{678500}{3} + \frac{344500}{3} + \frac{232250}{3} + \frac{694500}{3} = 649916\frac{2}{3} < u(2)
\end{aligned}$$

$$\begin{aligned}
\phi_3(u) &= \frac{0!\,2!}{3!}\big(u(1,2,3) - u(1,2)\big) + \frac{1!\,1!}{3!}\big(u(2,3) - u(2)\big) \\
&\quad + \frac{1!\,1!}{3!}\big(u(1,3) - u(1)\big) + \frac{2!\,0!}{3!}\big(u(3) - u(\emptyset)\big) \\
&= \frac{1}{3}(1424000 - 1181000) + \frac{1}{6}(946500 - 694500) \\
&\quad + \frac{1}{6}(745500 - 492000) + \frac{1}{3}(482000) \\
&= 81000 + 42000 + 42250 + 160666\frac{2}{3} = 325916\frac{2}{3} < u(3)
\end{aligned}$$

Как можно увидеть, стоимость всех компаний уменьшилась. Соответственно, компании 1, 2 и 3 должны объединиться, в результате чего каждый из них сохранит довольно большое количество денег. Первая компания сохранит около 9%, вторая около 6.5%, а третья около 32%. Можно заключить, что компании 3 находится в коалиции наиболее выгодно.

3.4.3.2 С-Ядро

Также докажем, что С- Ядро не пустое для данного примера. Для этого надо доказать, что следующая система уравнений имеет решение:

$$x_1 + x_2 + x_3 = 1424000\ (1)$$
$$x_1 + x_2 \leq 1181000\ (2)$$
$$x_1 + x_3 \leq 745500\ (3)$$
$$x_2 + x_3 \leq 946500\ (4)$$
$$x_1 \leq 492000$$
$$x_2 \leq 694500$$
$$x_3 \leq 482000$$

Как можно легко увидеть, это система уравнений имеет много решений – вычтем из (1) уравнения (2), (3), (4), получим

$$x_1 + x_2 + x_3 = 1424000$$
$$477500 \leq x_1 \leq 492000$$
$$678500 \leq x_2 \leq 694500$$
$$243000 \leq x_3 \leq 482000$$

соответственно ядро не пустое, решений получено бесконечное множество.

Также заметим, что дележ, полученный с помощью вектора Шепли, не принадлежит С-Ядру. Это также означает, что рассматриваемая нами игра не выпуклая – так как для выпуклых игр вектора Шепли должен принадлежать С-ядру.

3.4.3.3 Другие методы подсчета дележа

Рассмотрим другие методы подсчета дележа для данного примера.

Поделим количество сохраненных денег поровну с помощью данной формулы:

$$x_i = u(i) - \frac{1}{|N|}\left[\sum_{j \in N} u(j) - u(N)\right]$$

$$x_1 = 492000 - \frac{1}{3}[1668500 - 1424000] = 492000 - 81500 = 410500$$

$$x_2 = 694500 - \frac{1}{3}[1668500 - 1424000] = 694500 - 81500 = \ 613000$$

$$x_3 = 482000 - \frac{1}{3}[1668500 - 1424000] = 482000 - 81500 = 400500$$

Данный дележ, как можно заметить, также не принадлежит С-Ядру.

Рассмотрим также метод пропорций:

$$x_i = \frac{u(i)}{\sum_{j \in N} u(j)} u(N)$$

$$x_1 = \frac{492000}{1668500} * 1424000 = 419903$$

$$x_2 = \frac{694500}{1668500} * 1424000 = 592729$$

$$x_3 = \frac{482000}{1668500} * 1424000 = 411368$$

Дележ, полученный методом пропорций, также не принадлежит С-Ядру.

3.4.3.4 N-ядро

Для начала рассмотрим вектор эксцессов, который получается, если выбрать дележ, полученный с помощью вектора Шепли. Формула для подсчета Эксцессов:

$$e(S, x) = u(S) - \sum_{i \in S} x_i = u(S) - x(S)$$

Вектор эксцессов:

Коалиция	$u(S)$	$\sum_{i \in S} x_i$	$e(S, x)$
1	492000	$448166\frac{2}{3}$	$43833\frac{1}{3}$
2	694500	$649916\frac{2}{3}$	$44583\frac{1}{3}$
3	482000	$325916\frac{2}{3}$	$156083\frac{1}{3}$
1-2	1181000	$1098083\frac{1}{3}$	$82916\frac{2}{3}$
1-3	745500	$774083\frac{1}{3}$	$-28583\frac{1}{3}$
2-3	946500	$975833\frac{1}{3}$	$-29333\frac{1}{3}$

1-2-3	1424000	1424000	0

Таблица 4. Вектор Эксцессов для дележа, полученного с помощью вектора Шепли.

Как можно заметить, минимальное значение эксцесса равно $-29333\frac{1}{3}$. Оптимальное дележ можно найти с помощью любого математического пакета, например, пакета GameTheoryAllocation для языка R.

Оптимальный дележ: $(492000, 689000, 243000)$

Коалиция	$u(S)$	$\sum_{i \in S} x_i$	$e(S, x)$
1	492000	492000	0
2	694500	689000	5500
3	482000	243000	239000
1-2	1181000	1181000	0
1-3	745500	735000	10500
2-3	946500	932000	14500
1-2-3	1424000	1424000	0

Таблица 5. Вектор Эксцессов оптимального дележа с помощью N-ядра.

Можно заметить, что данный дележ находится в С-ядре, указанном выше, как и должно получаться согласно теории.

ЗАКЛЮЧЕНИЕ

В данной работе была изучена кооперативная теория игр применительно к проблеме планирования расписаний поставок на нефтяные платформы судами снабжения.

Была рассмотрены сама теория игр и кооперативная теория игр. Были введены понятия C-Ядра, N-ядра, K-ядра, сильного ε-ядра и вектора Шепли. Было рассмотрено понятие пустого ядра и приведен способ проверки ядра на пустоту. Были рассмотрены выпуклые кооперативные игры и их свойства.

Был проведен краткий обзор литературы по темам, относящимся к рассматриваемой задаче.

Была приведена формулировка задачи, как она дана в литературе. Было рассмотрено два способа формулировки задачи – на основании построение транспортной сети и двухфазный метод построения решения. Был разработан алгоритм генерации возможных маршрутов и математическая модель для облегченного варианта задачи. Была построена модель с помощью языка математического моделирования AMPL.

Бала решена задача для конкретного примера, приведены результаты работы программы и с помощью вектора Шепли, N-ядра и других методов был получен дележ для коалиций. Было показано, какие коалиции сформируются, и находится ли полученный дележ в C-ядре. Было докажем, что ядро не пустое.

Список использованных источников

МАРАКУЛИН, В. М. (2019). *Элементы теории кооперативных игр.* Институт математики им. С. Л. Соболева СО РАН, Новосибирский государственный университет, Новосибирск.

Cruijssen, F., Cools, M., & Dullaert, W. (2007). Horizontal cooperation in logistics: opportunities and impediments. *Transportation Research Part E: Logistics and Transportation Review 43, 2*, 129–142.

LEMAIRE, J. (б.д.). *AN APPLICATION OF GAME THEORY: COST ALLOCATION.* Umversite Libre de Bruxelles .

Tveita, E. (2018). Methods for Cost Allocation Among Prosumers and Consumers Using Cooperative Game Theory.

Kisialiou, Y., Gribkovskaia, I., & Laporte, G. (2018). Robust supply vessel routing and scheduling.

Shyshou, A. (2010). A multi-base supply vessel planning problem arising in offshore oil and gas operations.

Halvorsen-Weare, E. E., Fagerholt, K., Nonås, L. M., & Asbjørnslett, B. E. (2012). Optimal fleet composition and periodic routing of offshore supply vessels.

Fagerholt, K., & Lindstad, H. (2002). Optimal policies for maintaining a supply service in the Norwegian Sea. *Omega*.

Shyshou, A., Gribkovskaia, I., Laporte, G., & Gilbert, G. (2012). A Large Neighbourhood Search Heuristic for a Periodic Supply Vessel Planning Problem Arising in Offshore Oil and Gas Operations . *nformation Systems and Operational Research*.

Borthen, T., Loennechen, H., Wang, X., Fagerholt, K., & Vidal, T. (2017). A genetic search-based heuristic for a fleet size and periodic routing problem with application to offshore supply planning. *Eur. J. Transport. Logist.*

Halvorsen-Weare, E., Fagerholt, K., & Rönnqvist, M. (2013). Vessel routing and scheduling under uncertainty in the liquefied natural gas business. *Computers & Industrial Engineering*.

Кремлев, А. Г. (2016). *Основные понятия теории игр.* Екатеринбург: Издательство Уральского университета.

Schmeidler, D. (1969). *The nucleolus of a characteristic function game.* SIAM Journal in Applied Mathematics, 17(6):1163–1170.

LITTLECHILD, S and VAIDJA, K . (1976). *The Propensity to Disrupt and the Disruption Nucleolus of a Characteristic Function Game.* Int J Game Theory 151-161.

Halvorsen-Weare, E. E., & Fagerholt, K. (2016). *Optimization in offshore supply vessel planning.*

Shapley, L. (1971). Cores of convex games. *Int J Game Theory*, 11–26.

Driessen, T. (1988). Cooperative Games, Solutions and Applications. *Kluwer Academic Publishers*.

Davis, M., & Maschler, M. (1965). The kernel of a cooperative game. *Naval Research Logistics Quarterly*, 223-259.

Shapley, L. S., & Shubik, M. (1966). Quasi-cores in a monetary economy with non-convex preferences. *Econometrica,34*, 805-827.

Shapley, L. S. (1953). A value for n-person games. *Contributions to the Theory of Games II*, 307–317.

Maschler, M., Peleg, B., & Shapley, L. S. (1979). Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of Operations Research*, 303–338.

Edgeworth, F. Y. (1881). Perfect competition and the core. *Mathematical psychics.*

Gillies, D. B. (1959). Solutions to general non-zero-sum games. *Contributions to the Theory of Games IV.*, 47–85.

Shapley, L. S., & Shubik, M. (1969). On Market Games. *Journal of Economic Theory.*, 9-25.

Halvorsen-Weare, E. (2012). Maritime fleet planning and optimization under uncertainty.

ПРИЛОЖЕНИЕ 1

```
from ortools.constraint_solver import routing_enums_pb2
from ortools.constraint_solver import pywrapcp
import math
from itertools import *


def create_data_model():
    coordinates = [(0, 0), (-200, 500), (0, 500), (100,
300), (150, 500), (300, 450), (200, 200), (600, 100),
(400, 150), (500, 0), (450, -200)]
    data = {}
    data['distance_matrix'] = [[0 for _ in range(11)]
for _ in range(11)]
    for i in range(11):
        for j in range(11):
            x1 = coordinates[i][0]
            y1 = coordinates[i][1]
            x2 = coordinates[j][0]
            y2 = coordinates[j][1]
            data['distance_matrix'][i][j] =
int(math.sqrt((x1-x2)**2 + (y1-y2)**2 ))

    data['num_vehicles'] = 2
    data['depot'] = 0
    return data


def print_solution(manager, routing, solution):
    index = routing.Start(0)
    route_distance = 0
    visited = []
    while not routing.IsEnd(index):
        if manager.IndexToNode(index):
            visited.append(manager.IndexToNode(index))
        previous_index = index
        index = solution.Value(routing.NextVar(index))
        route_distance +=
```

```
routing.GetArcCostForVehicle(previous_index, index, 0)
    return (route_distance, visited)


def main(allowed_nodes):
    data = create_data_model()

    manager =
pywrapcp.RoutingIndexManager(len(data['distance_matrix'
]), data['num_vehicles'], data['depot'])

    routing = pywrapcp.RoutingModel(manager)


    def distance_callback(from_index, to_index):

        from_node = manager.IndexToNode(from_index)
        to_node = manager.IndexToNode(to_index)
        return
data['distance_matrix'][from_node][to_node]

    transit_callback_index =
routing.RegisterTransitCallback(distance_callback)


routing.SetArcCostEvaluatorOfAllVehicles(transit_callba
ck_index)

    for i in range(1, 11):
        if i in allowed_nodes:
            routing.SetAllowedVehiclesForIndex([0], i)
        else:
            routing.SetAllowedVehiclesForIndex([1], i)

    search_parameters =
pywrapcp.DefaultRoutingSearchParameters()
    search_parameters.first_solution_strategy = (

routing_enums_pb2.FirstSolutionStrategy.PATH_CHEAPEST_A
```

```
RC)

    solution =
routing.SolveWithParameters(search_parameters)

    if solution:
        return print_solution(manager, routing,
solution)

u = 0
k1_nodes = [1,2,3,4,5,6,7,8,9,10]
v_number = 10
fw = open('task.dat', 'w')
A_string = 'param A : '
D_string = 'param D := \n'
c_string = 'param c := \n'
R_set = 'set R := '
RL2_set = 'set RL[2] := '
RL3_set = 'set RL[3] := '
for i in range(1, 11):
    A_string += 'I{0} '.format(i)
A_string += ':= \n'

for i in range(1, len(k1_nodes) + 1):
    for j in combinations(k1_nodes, i):
        (route_distance, visited) = main(j)
        name = ''.join([str(i) for i in visited])
        A_string += '\t r{0} '.format(name)
        for k in range(1, 11):
            if k in visited:
                A_string += '1 '
            else:
                A_string += '0 '
        A_string += '\n'
        number_of_hours = int(route_distance / 46) + 4
+ 2 * len(visited)
        number_of_days = math.ceil(number_of_hours /
24.0)
        D_string += '\tr{0} {1}\n'.format(name,
```

```
number_of_days)
        c_string += '\tr{0} {1}\n'.format(name,
number_of_hours * 500)
        R_set += 'r{0} '.format(name)
        if number_of_days == 2:
            RL2_set += 'r{0} '.format(name)
        if number_of_days == 3:
            RL3_set += 'r{0} '.format(name)
        u += 1
A_string += ';\n\n'
D_string += ';\n\n'
c_string += ';\n\n'
R_set += ';\n\n'
RL2_set += ';\n\n'
RL3_set += ';\n\n'

fw.write(A_string)
fw.write(D_string)
fw.write(c_string)
fw.write(R_set)
fw.write(RL2_set)
fw.write(RL3_set)
```

ПРИЛОЖЕНИЕ 2

```
from __future__ import division
from pyomo.environ import *

model = AbstractModel()

model.N = Set()
model.V = Set()
model.T = Set()
model.L = Set()
model.R = Set()
model.RL = Set(model.L)

model.h = Param()
model.F = Param()
model.S = Param(model.N)
model.A = Param(model.R, model.N)
model.D = Param(model.R)
model.c = Param(model.R)

model.y = Var(model.V, domain=Binary)
model.x = Var(model.V, model.R, model.T, domain=Binary)


def total_cost(model):
    return (
            sum(model.y[v] * model.h for v in model.V)
+
            sum(model.c[r] * model.x[v, r, t] for v in
model.V for r in model.R for t in model.T)
    )


def constraint_1(model, i):
    if not any(model.A[r, i] for r in model.R):
        return Constraint.Skip

    return sum(model.A[r, i] * model.x[v, r, t] for v
```

```
in model.V for r in model.R for t in model.T) >=
model.S[i]


def constraint_2(model, v):
    return sum(model.D[r] * model.x[v, r, t] for r in
model.R for t in model.T) <= model.y[v] * model.F


def constraint_3(model, v, t, l):
    return (
            sum(model.x[v, r, t] for r in model.RL[l])
+
            sum(model.x[v, r, (t + k) % 7] for r in
model.R for k in range(1, l))
           ) <= 1


model.OBJ = Objective(rule=total_cost)
model.C1 = Constraint(model.N, rule=constraint_1)
model.C2 = Constraint(model.V, rule=constraint_2)
model.C3 = Constraint(model.V, model.T, model.L,
rule=constraint_3)

instance = model.create_instance(filename='task.dat')
opt = SolverFactory('cplex')
results = opt.solve(instance)
results.write()
instance.solutions.load_from(results)

for v in instance.component_objects(Var, active=True):
    print ("Variable",v)
    varobject = getattr(instance, str(v))
    for index in varobject:
        if varobject[index].value > 0:
            print ("   ",index, varobject[index].value)
```