МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования   
«САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой № 43

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Профессор, д.т.н., профессор |  | 31.05.2023 |  | М.Ю. Охтилев |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

|  |  |
| --- | --- |
| на тему | Оптимизация цепей поставок древовидной структуры |
|  | |
|  | |

|  |  |
| --- | --- |
| выполнена | Чепусовым Владиславом Константиновичем |
| фамилия, имя, отчество студента в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 09.04.04 |  | Программная инженерия |
|  | код |  | наименование направления |
|  | | | |
| наименование направления | | | |
| направленности | 09.04.04.02 |  | Проектирование интеллектуальных |
|  | код |  | наименование направленности |
| программных систем | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент группы № | 4130М |  | 31.05.2023 |  | В.К.Чепусов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Руководитель

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Доцент, к.т.н., доцент |  | 31.05.2023 |  | А.А.Ключарёв |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования   
«САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

УТВЕРЖДАЮ

Заведующий кафедрой № 43

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Профессор, д.т.н., профессор |  | 24.03.2023 |  | М.Ю.Охтилев |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

|  |  |  |  |
| --- | --- | --- | --- |
| студенту группы № | 4130М |  | Чепусов Владислав Константинович |
|  |  |  | (фамилия, имя, отчество) |

|  |  |
| --- | --- |
| на тему | Оптимизация цепей поставок древовидной структуры |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| утвержденную приказом ГУАП от | 23.03.2023 | № | 11-312/23 |

|  |  |
| --- | --- |
| Цель исследования: | разработка и реализация алгоритма оптимизации цепей поставок |
|  | |

|  |  |
| --- | --- |
| Задачи исследования: | 1) Построение теоретико-игровой модели оптимизации |
| цепи поставок древовидной структуры | |
| 2) Выбор критерия оптимизации и разработка алгоритма построения оптимального | |
| решения | |
| 3) Программная реализация разработанного алгоритма | |

|  |  |
| --- | --- |
| Содержание диссертации (основные разделы): | Введение; Анализ особенностей цепей |
| поставок; Разработка алгоритма решения; Разработка и проектирование программного | |
| обеспечения; | |
|  | |
|  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Срок сдачи диссертации « | 31 | » | мая | 20 | 23 |

Руководитель

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Доцент, к.т.н., доцент |  | 24.03.2023 |  | А.А.Ключарёв |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

Задание принял к исполнению

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| студент группы № | 4130М |  | 24.03.2023 |  | В.К.Чепусов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

**РЕФЕРАТ**

Магистерская диссертация 107 с., 19 рис., 11 табл., 31 источников, 1 прил.

Ключевые слова: цепь поставок, древовидный граф, модель Курно, равновесие по Нэшу, теория игр, критерий оптимальности, оптимизация.

Объектом исследования являются цепи поставок древовидной структуры.

Целью данной выпускной квалификационной работы является разработка и программная реализация алгоритма оптимизации цепи поставок древовидной структуры по критерию прибыли.

В результате выполнения работы был произведен анализ и обзор литературных источников, касающихся методов координации, теоретических аспектов взаимодействия и конкуренции в цепях поставок. Была создана математическая модель для древовидной цепи поставок, а также был разработан способ поиска оптимального решения относительно прибыли участников, конкурирующих по экономической модели Курно и реализовано программное обеспечение по данному алгоритму.

Область применения: экономические сферы, связанные с цепями поставок, такие как промышленность, производство, розничная торговля.

**ABSTRACT**

Master's thesis 107 p., 19 fig., 11 table, 31 sources, 1 adj.

Keywords: supply chain, tree graph, Cournot model, Nash equilibrium, game theory, optimality criterion, optimization.

The object of the study is the supply chain of a tree structure.

The purpose of this final qualifying work is the development and software implementation of an algorithm for optimizing the supply chain of a tree structure according to the profit criterion.

As a result of the work, an analysis and review of literature sources concerning coordination methods, theoretical aspects of interaction and competition in supply chains was carried out. A mathematical model for a tree-like supply chain was created, and a method for finding the optimal solution regarding the profits of participants competing according to the Cournot economic model was developed, and software for this algorithm was implemented.

Scope of application: economic spheres related to supply chains, such as industry, manufacturing, retail trade.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc135863326)

[1. Анализ особенностей цепей поставок 9](#_Toc135863327)

[1.1. Анализ задачи оптимизации цепи поставок 9](#_Toc135863328)

[1.2. Обзор литературы 15](#_Toc135863329)

[1.3. Математическая модель цепи поставок 28](#_Toc135863330)

[1.4. Анализ существующих средств решения задачи 35](#_Toc135863331)

[Выводы к разделу 39](#_Toc135863332)

[2. Разработка алгоритма решения 40](#_Toc135863333)

[2.1. Пример решения задачи 42](#_Toc135863334)

[2.2. Поведение при отрицательных объемах 47](#_Toc135863335)

[2.3. Пример решения задачи с отрицательными объемами 50](#_Toc135863336)

[Выводы к разделу 52](#_Toc135863337)

[3. Разработка и проектирование программного обеспечения 53](#_Toc135863338)

[3.1. Функциональные требования к разрабатываемой системе 53](#_Toc135863339)

[3.2. Нефункциональные требования к разрабатываемой системе 54](#_Toc135863340)

[3.3. Требования к организации входных и выходных данных 55](#_Toc135863341)

[3.4. Варианты использования системы 59](#_Toc135863342)

[3.5. Выбор и обоснование используемых инструментов 72](#_Toc135863343)

[3.6. Инструкция к запуску программы 79](#_Toc135863344)

[3.7. Пример работы ПО со скриншотами интерфейса 80](#_Toc135863345)

[Выводы к разделу 89](#_Toc135863346)

[ЗАКЛЮЧЕНИЕ 90](#_Toc135863347)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 91](#_Toc135863348)

[ПРИЛОЖЕНИЕ А 95](#_Toc135863349)

# ВВЕДЕНИЕ

Цепи поставок — неотъемлемая часть мировой экономики и бизнеса, которая тесно связана с торговлей и производством. Компаниям, находящимся на рынке, необходимо налаживать операционную деятельность для реализации собственных продуктов путем создания системы товарных потоков и торговых связей в рамках цепи поставок. Последние исследования в области управления цепочками поставок показали, что в структуре, состоящей из независимых фирм, принимаемые решения оказывают влияние не только на прибыль самих компаний, но и других участников цепи. Это в свою очередь воздействует на всю цепь поставок. Именно данный момент послужил причиной к изучению стратегических решений, принимаемых участниками цепи, и их влиянию на всю цепь поставок.

С течением времени численность и сложность подобных систем увеличиваются под воздействием технологического развития и процесса мировой интеграции. А также налаживание и оптимизация уже существующих цепей поставок становятся еще более трудоемкой и комплексной задачей. Однако, важность ее решения часто недооценивается, что приводит к финансовым потерям и упущенной выгоде. Поэтому необходимо оптимизировать цепь поставок по критерию прибыли и координировать действия участников цепи. Данная задача, без сомнения, становится особенно актуальной в связи с широкой распространенностью цепей поставок и их важностью в современном бизнесе [6].

Для анализа ситуаций, где результат зависит от принимаемых решений каждой из участвующих сторон, наиболее подходящим инструментом является теория игр. В настоящее время исследователи управления цепями поставок все чаще используют теорию игр для изучения, прогнозирования и помощи управленцам в принятии стратегических решений в сложных цепях поставок. Теория игр позволяет учестьвоздействие каждого участника цепи на общую прибыль (или другую меру эффективности работы цепи поставок), а также способствует прогнозированию результатов применения участникамиразличных стратегий.

Целью данной работы является разработка и программная реализация алгоритма оптимизации цепи поставок древовидной структуры по критерию прибыли.

Для более эффективного изучения цепей поставок важно установить их структуру и взаимосвязи между элементами. Один из практических подходов формализации заключается в представлении цепей в виде древовидного графа. Это позволяет лучше понять и управлять процессом поставок.

Нами будет рассмотрен древовидный граф. Внутри каждого из его узлов находятся несколько фирм, между которыми происходит конкуренция по модели Курно [22]. В корневом узле или, говоря иначе, узле-дистрибьюторе, распределяется товар между дочерними узлами. В последующих узлах, если они не являются терминальными, происходит модернизация исходного продукта, после чего осуществляется продажа своим дочерним узлам. Терминальные узлы реализуют товар на рынках.

Ключевым задачами для достижения цели исследования, являются:

* произвести анализ и обзор литературных источников, касающихся методов координации, теоретических аспектов взаимодействия и конкуренции в цепях поставок;
* построить теоретико-игровую модель для оптимизации цепи поставок по критерию прибыли;
* разработать алгоритм, который позволит построить оптимальное решение для данной цепи;
* создать реализацию разработанного алгоритма в программном виде;

Работа содержит 3 раздела. Основная часть работы изложена на 94 страницах, содержит в себе 19 рисунков и 11 таблиц.

В первом разделе представлены обзор литературы, постановка задачи, математическая модель, проанализированы аналоги программного обеспечения (ПО) на рынке.

Во втором разделе разработан алгоритм решения задачи, а также разобран аналитический пример решения.

В третьем разделе представлены функциональные и нефункциональные требования к разрабатываемой системе, а также выбор и обоснование программно-технических средств, инструкция по запуску ПО, результаты и скриншоты работы программного продукта.

В заключение приведены выводы по результатам работы, список использованной литературы и программный листинг (приложение А).

# 

# Анализ особенностей цепей поставок

# Анализ задачи оптимизации цепи поставок

Цепи поставок являются интегральной частью функционирования современных компаний и бизнеса в целом. В расширяющемся и все более глобализированном мире, преодоление коммуникационных, производственных и логистических барьеров, а также стремление к оптимизации и упрощению реализации товаров и услуг становятся ключевыми задачами всех участников рынка. Цепь поставок представляет собой совокупность всех этапов, связанных с передачей товаров и услуг от производителя к конечному потребителю. Включая поставщиков сырья, производство, хранение, транспортировку, дистрибуцию, а также более широкий круг внутренних и внешних связей, необходимых для функционирования и развития предприятий. Цепи поставок, как правило, включают в себя несколько международных, национальных и/или региональных организаций, а также инфраструктурных элементов, которые обеспечивают непрерывность в создании, производстве и продаже продукции и услуг.

Цепь поставок является сложной сетью взаимосвязанных фирм, где каждая из них стремится максимизировать свою прибыль, принимая во внимание действия и решения конкурентов. В задаче оптимизации цепи поставок необходимо найти наилучшую стратегию для каждой фирмы с целью достижения общего экономического выгодного состояния.

В контексте нашей задачи оптимизации цепи поставок рассмотрим несколько моделей конкуренции, которые часто применяются в экономической теории:

1. Модель Бертрана: В этой модели предполагается, что фирмы конкурируют по цене, а не по количеству производимого товара. При наличии нескольких фирм, каждая из них ставит цену ниже своих конкурентов, чтобы привлечь больше покупателей. Модель Бертрана применяется, когда товары являются однородными, а стоимость производства не зависит от объема.
2. Модель Курно: В этой модели предполагается, что фирмы конкурируют по количеству производимого товара, а не по цене. Фирмы выбирают свои производственные объемы с учетом ожидаемого спроса и реакции конкурентов.
3. Модель Штакельберга: Эта модель подразумевает наличие монопольного производителя, который является первопроходцем на рынке, и других производителей, которые присоединяются к нему. В этой модели монопольный производитель может устанавливать цены, тогда как другие производители должны приспосабливаться к этим ценам.

Каждая из этих моделей имеет свои особенности и применяется в зависимости от конкретных условий рынка и типа конкуренции, которая на нем происходит. Например, модель Бертрана может быть применима в случаях, когда производители имеют одинаковые затраты на производство и конкурируют на ценах. Модель Курно часто используется для моделирования рынков, где производители имеют непостоянные затраты на производство, а модель Штакельберга для моделирования монопольных рынков, где производитель устанавливает цены и определяет уровень производства.

При анализе рыночных моделей конкуренции существуют различные подходы, которые описывают различные виды взаимодействия между фирмами. Одной из наиболее распространенных моделей является модель Курно. Она основывается на предположении, что фирмы в цепи поставок выбирают свой производственный объем независимо друг от друга, принимая во внимание решения своих конкурентов. Это означает, что каждая фирма стремится максимизировать свою прибыль, учитывая предполагаемый отклик конкурентов на свои действия.

Выбор модели Курно в нашей задаче оптимизации цепи поставок обусловлен несколькими факторами. Во-первых, модель Курно широко используется в исследованиях и приложениях, связанных с конкуренцией на рынке. Она является стандартной моделью, которая позволяет анализировать взаимодействие фирм в условиях несовершенной конкуренции.

Во-вторых, она предоставляет возможность изучить стратегии производственного объема и ценообразования фирм, учитывая их взаимодействие и реакцию конкурентов. Это позволяет оценить эффекты таких факторов, как снижение цены, увеличение объема производства и изменение доли рынка.

Таким образом, модель Курно представляет собой релевантный инструмент для решения задачи оптимизации цепи поставок, где фирмы конкурируют за рыночную долю и стремятся максимизировать свою прибыль, принимая во внимание решения конкурентов.

Экономическая модель Курно является одной из основных моделей теории игр и используется для описания рыночных ситуаций, где фирмы конкурируют друг с другом, устанавливая цены на товары.

В модели Курно предполагается, что на рынке присутствуют несколько фирм, каждая из которых может устанавливать свою цену на продукт. Однако, в отличие от идеальной конкуренции, цены не являются равными стоимости производства товара. Вместо этого каждая фирма принимает во внимание реакцию других фирм на свою цену [19].

В модели Курно фирмы максимизируют свою выручку, выбирая цену, которая дает наибольшую прибыль при условии, что остальные фирмы удержат свои цены неизменными. В результате каждая фирма устанавливает цену, которая ниже оптимальной в условиях идеальной конкуренции.

Кроме того, в модели Курно предполагается, что фирмы действуют независимо друг от друга и не сотрудничают между собой, что может привести к неэффективности рынка и снижению общественного благосостояния.

Особенности модели Курно состоят в следующем [20]:

1. Несколько фирм на рынке: модель Курно предполагает, что на рынке действуют несколько фирм, каждая из которых конкурирует с другими.
2. Фирмы устанавливают цены: в отличие от модели идеальной конкуренции, где цены устанавливаются на основе стоимости производства, в модели Курно каждая фирма устанавливает свою цену на продукт.
3. Фирмы максимизируют прибыль: цель каждой фирмы – максимизировать свою выручку, а не свою долю рынка, посредством выбора цены, которая дает наибольшую прибыль при условии, что остальные фирмы удержат свои цены неизменными.
4. Каждое предприятие знает, что другие предприятия будут реагировать на его решения, и, следовательно, на различные стратегии, которые оно может выбрать.
5. Предприятия принимают решение о выпуске продукции до того, как они узнают стратегию других участников рынка.
6. Каждому предприятию выгодно действовать, учитывая реакции других участников рынка, то есть использовать игровую теорию для принятия решений.
7. Неэффективность рынка: в модели Курно фирмы действуют независимо друг от друга и не координируют свои действия, что может привести к неэффективности рынка и снижению общественного благосостояния.
8. Необходимость анализа равновесия Нэша: в модели Курно необходимо анализировать равновесие Нэша, которое возникает при определенных стратегиях фирм и описывает ситуацию, при которой ни одна фирма не может увеличить свою прибыль, изменив свою стратегию.

Для изучения и оптимизации цепей поставок важно формализовать их структуру. Одним из возможных подходов к такой формализации может быть представление их в виде древовидного графа или дерева решений. В структуре древовидного графа, узлы представляют собой отдельные участников (т.е., производители, поставщики, другие фирмы и т.д.), а рёбра соединяют этих участников и символизируют потоки товаров, сырья и услуг между ними.

Рассмотрим пример цепи поставок лесной промышленности (см. рисунок 1).

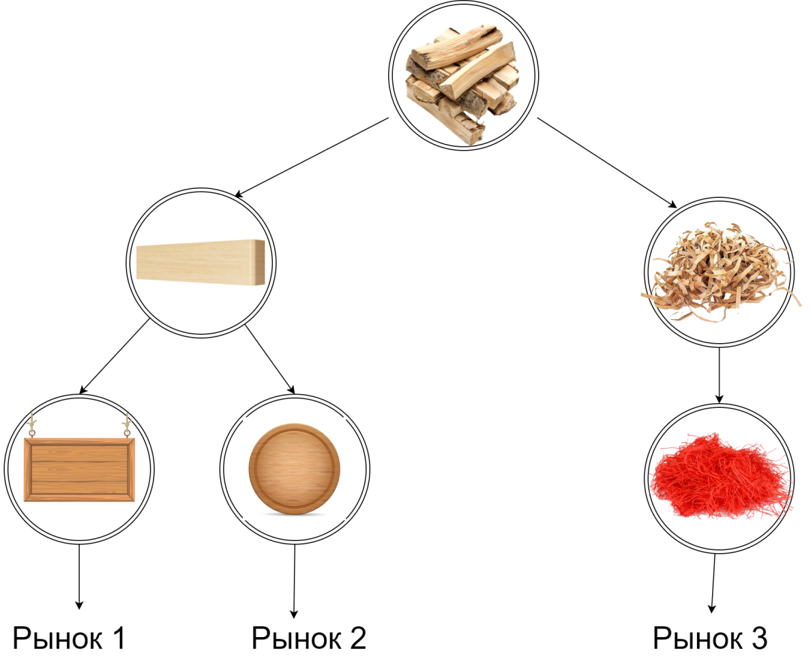


Рисунок 1 – цепь поставок лесной промышленности

У нас есть оптовые дистрибьюторы, которые поставляют необработанную древесину розничным дистрибьюторам. Они, в свою очередь, модернизируют полученный товар для продажи ритейлерам. Последние также вносят изменения в продукцию и реализуют товар на рынках.

Формализуем эту ситуацию, используя древовидный граф. Внутри каждого из его узлов находятся несколько фирм, между которыми происходит конкуренция по модели Курно.

Рассмотрим древовидный граф на рисунке 2.

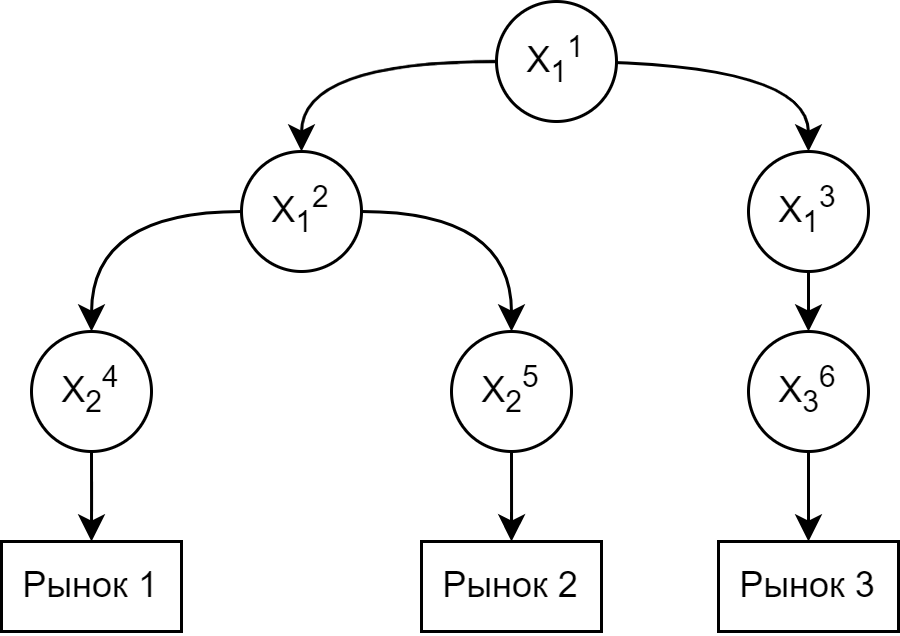


Рисунок 2 – Пример цепи поставок с древовидной структурой.

Допустим, что каждый узел в цепи поставок представляет группу компаний, которые соперничают друг с другом, производя одинаковый продукт с использованием одинаковых ресурсов. Хотя переменные издержек у фирм могут отличаться, они являются общедоступными параметрами в сети. Количество компаний в каждом узле также известно. Предполагается, что компании, находящиеся в одном узле, продают продукт по одной и той же цене.

В корневом узле или, говоря иначе, узле-дистрибьюторе (), распределяется товар между дочерними узлами ( и ). В последующих узлах, если они не являются терминальными, происходит модернизация исходного продукта, после чего осуществляется продажа уже своим дочерним узлам. Терминальные узлы реализуют товар на рынках. Предполагается, что данные рынки, не конкурируют между собой и работают в соответствии с моделью Курно, где используется линейная функция спроса и фиксированные издержки.

Все фирмы стремятся к максимизации собственной выручку, однако на значение прибыли отдельно взятого участника цепочки поставок влияют не только его собственные решения, но и действия других участников. Исходя из этого было решено рассматривать данную задачу в контексте теоретико-игрового подхода.

# Обзор литературы

Цепи поставок встречаются во множестве отраслей торговли. Такая распространенность делает их темой изучения различных рыночных, логистических и научных работ. Доктор экономических наук Крылатков Петр Петрович в соавторстве со своей коллегой Прилуцкой Марией Андреевной дает следующее определение «Цепь поставок — это три или более экономических единиц (организации или лица), напрямую участвующих во внешних и внутренних потоках продукции, услуг, финансов и/или информации от источника до потребителя» [4]. Иначе говоря, сети поставок ­— это цепь взаимозависимых и тесно связанных организаций, взаимодействующих сообща и на базе взаимного сотрудничества в целях контроля за управлением и развитием потока информации и ресурсов от поставщиков к конечным потребителям. Управление цепями поставок характеризуется как управление восходящими и нисходящими рыночными отношениями с поставщиками и потребителями для обеспечения наилучшей потребительской ценности по наименьшим издержкам для цепочки поставок в целом [5].

Устоявшегося мнения насчет того, кто и когда первым ввел понятие управления цепей поставок, в научной среде нет, но ориентировочно все теории сходятся на том, что это было в 80-х годах прошлого века «В специализированной литературе встречаются различные точки зрения относительно авторства и времени опубликования в печати пионерских работ по Supply Chain Management. Некоторые специалисты первое употребление термина “управление цепями поставок” относят к статье консультантов Р. Оливера и М. Вебера “Supply chain management: logistics catches up with strategy”, которые в 1982 году предложили рассматривать материальные потоки от производителя исходного сырья до конечного потребителя в рамках интегрированной стратегии SCM. Другие авторы утверждают, что термин “Supply chain management” был впервые предложен известным системным интегратором “i2 Technologies” и американскими консультантами (в частности компанией “Артур Андерсен”) в начале 1980-х годов и впоследствии стал широко использоваться в науке и бизнесе» – пишет в своей статье Парфёнов М. А. [7]. Но, несмотря на вышесказанное, данная концепция была сразу принята и получила значительное распространение и развитие.

В учебном пособии «Управление цепью поставок (SCM)» и книге «Logistics and Supply Chain Management: Strategies for Reducing Cost and Improving Service» особое внимание уделяется тому, что управление цепочками поставок становится качественным инструментом повышения эффективности деятельности компаний. Быстрое развитие рынка, усиление конкуренции и потребность в повышении качества обслуживания клиентов, устанавливают перед фирмами новые задачи. Для того чтобы оставаться конкурентоспособным и укреплять свои преимущества, современному предприятию необходимо оптимизировать все без исключения процессы формирования цены — от поставки сырья до обслуживания конечного потребителя. Для решения данных проблем руководство фирм обращается к SСМ-решениям. Это часть бизнеса, сопряженная с оптимизацией движения продукта с момента его введения до момента его продажи или реализации. В обеих книгах дается комплексное представление о задачах, а также функциях логистических систем, концептуальных принципах и методах эффективного управления вещественными потоками в области производства, поставок и реализации товара. Рассказывается о вариациях методов конструирования собственных цепей поставок и классификации последних.

«Эволюция управления цепями поставок: от интегрированной логистики к сетевым структурам» – это научная статья, в которой рассматривается развитие логистической концепции управления цепями поставок. Автор, в доступной форме, описывает настоящие подходы к сравнительной характеристике экономических категорий: «логистика» и «управление цепочками поставок». Мы, как читатели, получаем историческое обозрение не только зарождения понятия «SCM», но и динамическое развитие этого молодого направления за последние сорок лет. Парфёнов М.А. определяет отличительные черты использования сетевого подхода к организации взаимодействия в цепях поставок, производит сравнительный анализ сильных и слабых сторон стабильных и динамических сетевых структур сетей поставок.

Существуют различные структуры цепей поставок, однако в общем случае ее участники подразделяются на изготовителей, поставщиков, дистрибьюторов, ритейлеров и потребителей [4]. Вид сети зависит от фокусной компании. Зачастую в ее роли может быть производитель по причине того, что он является ключевой фигурой всей цепи поставок. Но в иных случаях роль фокусной компании может занять дистрибьютор, формируя таким образом дистрибутивную сеть [8].

«Стратегическая цель конфигурирования цепи заключается в том, чтобы добиться конкурентоспособности и приемлемой рентабельности фокусной компании, а также всей цепи поставок за счет повышения общей эффективности и производительности ее участников. При этом каждая компания может выстраивать свою цепь поставок, поскольку ее руководство видит именно свою компанию в качестве центральной и рассматривает потенциальных участников сетевой структуры исходя из интересов своей компании…» [4]. Остальные участники сети поставок рассматривают сеть как произвольную, поскольку каждый из них может создать свою собственную сеть, в центре которой фокусной фирмой будет являться именно данная компания. Следовательно, каждая организация может быть как фокусной компанией своей собственной сети поставок, так и участником другой сети поставок.

Фирмы, которые взаимодействуют напрямую или косвенно через иных участников цепи различных уровней с фокусной компанией составляют сеть поставок. Их группируют по воздействию на ценность, предоставляемую конечным потребителям на «ключевых» и «вспомогательных» участников. Доктор экономических наук Щербаков Владимир Васильевич дает этим понятиям следующие определения: «Ключевые участники цепи поставок – это независимые компании или структурные подразделения фокусной компании (ее филиалы или дочерние предприятия), оказывающие непосредственное влияние на интеграцию и управление логистическими процессами, протекающими в цепи поставок. Вспомогательные участники цепи поставок – это компании, которые не оказывают существенного влияния на интеграцию и управление логистическими процессами в цепи поставок, но предоставляют часть своих ресурсов ключевым участникам для выполнения ими своих операций» [8]. К последним можно отнести банки, страховые или консалтинговые компании и прочие государственные и негосударственные фирмы.

Учебное пособие Щербакова В.В. «Логистика и управление цепями поставок» передает авторский взгляд на логистику как на область предпринимательской деятельности по управлению материальными, информационными, финансовыми и другими потоками. Автор объясняет главные концепции логистики, аргументирует функциональное единство и результативность логистических решений. Книга нацелена на самостоятельное обучение читателя применять инновационное методы и формы реализации фундаментальных принципов – законов логистики, содержание которых непрерывно пополняется связи с ее научной ориентацией на обслуживание хозяйственных связей и управление цепями поставок клиентов.

Самые ранние работы в сфере управления цепями поставок были нацелены на подробное изучение децентрализованных моделей. Децентрализованная сеть поставок отличается тем, что компании, входящие в ее состав, являются независимыми единицами, которые стремятся повысить собственную прибыль и оптимизировать свои индивидуальные цели, и что действия, совершаемые компанией, также сказываются на показателях и работе других участников сети поставок. Авангардными работами можно считать труды Тьяги Р. К. [9] и Зисса С [10].

В работе «On the effect of downstream entry read» за авторством Тьяги Р. рассматриваются последствия входа на рынок новой фирмы-дистрибьютора, где фирмы покупают производственные ресурсы у поставщика, расположенного выше по уровню, и продают свою продукцию потребителям. В качестве примера можно привести фирмы (например, Compaq и IBM, CVS и Safeway) покупающие исходные ресурсы (например, микропроцессоры) у поставщиков, расположенного выше по уровню (например, Intel, Procter & Gamble), и продают свою уже продукцию потребителям. В работе показаны условия спроса, при которых, вопреки общепринятому мнению, вход на рынок новой фирмы снижает объем производства на рынке нисходящего потока и повышает рыночную цену. Таким образом, потребителям может быть выгоднее, если на таких рынках будет меньше продавцов. В статье также показано, что такой рынок может привести к тому, что прибыль каждой действующей фирмы, занимающейся переработкой и сбытом может кардинально меняться. Кроме того, для класса широко используемых условий спроса показано, что оптимальная цена поставщика неизменна в зависимости от входа/выхода его фирм-покупателей. В конце статьи приводится классификация всех возможных последствий выхода на рынок с точки зрения фундаментальных условий рыночного спроса.

Автор в своей статье «Vertical separation and horizontal mergers» рассматривает дуополию, состоящую из двух уровней, производитель и розничный продавец, в которой соглашение между парой производитель –розничный продавец определяет тариф, состоящий из двух частей. В отсутствие внутренней конкуренции автор показывает, что слияние вверх по уровням является антиконкурентным при общих условиях спроса и затрат. Слияние в нисходящем потоке анализируется с использованием линейного спроса и постоянных предельных издержек, и показано, что оно не является антиконкурентным как при наличии внутригрупповой конкуренции, так и без нее, а также при ценовой и производственной конкуренции между розничными торговцами в режиме, предшествующем слиянию. «On the effect of downstream entry» и «Vertical separation and horizontal mergers»одни из первых работ, по децентрализованным сетям, чьи недочеты и недоработки были устранены в начале 21 века, такими людьми, как Шабан, Жоу, Кармаркара и прочими.

Каждый год под действием прогресса и глобализации растет не только количество систем торговых связей, но и их сложность, а именно структура и масштабность. Также возникают задачи оптимизации уже организованных цепей поставок, однако важность решения таких проблем иногда бывает недооцененной. В результате плохо организованная операционная деятельность приводит к убыткам или нереализованной прибыли [6].

Статья Жоу, Кармаркара и Джанга рассматривает случай конкуренции в многоуровневых цепочках поставок с дистрибутивной структурой используя модель Курно для олигополии с несколькими уровнями. Данная статья является логическим обобщением и дополнением ранних работ Кармаркара [15-16]. Но, несмотря на то, что в этой статье используется та же последовательная модель Курно для анализа дистрибутивной структуры, что и в статье Корберта и Кармаркара [15] для последовательного случая, а также статье Карра и Кармаркара [16] для случая сборки, равновесное решение дает представление, отличное от этих работ. В дистрибутивной сети, вопреки общепринятому мнению, при изменении концентрации секторов цена ресурса может увеличиваться, уменьшаться или оставаться неизменной в зависимости от относительных параметров всех распределительных секторов. Таким образом, прибыль и объем производства действующих компаний могут измениться в любом направлении под совместным воздействием цен на ресурсы и конкуренции. Это уникальное поведение структуры не наблюдается в серийных и сборочных конструкциях, проанализированных в предыдущих статьях Кармаркара. В статье были разработаны явные выражения для равновесных цен и объёмов, как решение системы линейных уравнений, которые можно вывести из структуры сети. Данные идеи будут развиваться в будущем Зенкевичем Н.А. и Лонягиной Ю.Е.

Цепь поставок называется централизованной, если ее участники действуют сообща в целях максимизации различного рода выгоды для сети в целом. Именно данная тема следующей привлекла ученых. Воплощение в жизнь способов прогнозирования и планирования в таких сетях требует высокого уровня взаимодействия между фирмами-участниками. В централизованном управлении можно обнаружить случаи, когда сеть поставок состоит из организационных единиц, входящих в одну родительскую компанию. Данные фирмы формируют альянсы и преследуют общую для всей сети цель. Профессор пенсильванского университета Кахон Г.П. рассматривал вопрос о воздействии координирующих контрактов [11]. В то время Озер О. и Кайя М. изучили вопрос контактирования цепи для разделения по участникам прибыли, информации, издержек и рисков [12].

В своей работе Кахон Г.П «Supply Chain Coordination with Contracts» рассматривает координацию цепей поставок с помощью контрактов для различных моделей цепей поставок. В каждой модели определяются оптимальные действия цепи поставок, а сама координация является сложной задачей, поскольку фирмы не могут заключать контракты на основе выбранных стимулов. В статье также рассматривается стохастическая модель спроса с бесконечным горизонтом, в которой розничный продавец получает пополнение от поставщика через постоянные промежутки времени. Но в исследовании не хватает эмпирически проверяемых гипотез и доказательств. Сам автор говорит о том, что необходимы дополнительные исследования того, как множественные поставщики конкурируют за привлечение нескольких розничных торговцев, т.е. необходимо уделить дополнительное внимание структурам цепей поставок "многие к одному" или "многие ко многим".

Статья «Pricing in business-to-business contracts: sharing risk, profit and information» за авторством Озера О. и Кайя М. посвящена обсуждению того, каким образом условия ценообразования в договорах на рынке B2B [30] могут быть использованы для достижения надлежащего распределения рисков и информации в цепочке поставок между фирмами-союзниками. Изначально авторы показывают зависимость между избытками и недостатками запасов, рассматривая решение на примере работы одного производителя. Далее исследуются риски товарных запасов в децентрализованной цепи поставок между производителем и розничным продавцом. Авторы показывают, что договор об оптовой цене не позволяет согласовать стимулы розничного торговца и цепочки поставок, заставляя розничного торговца заказывать и складировать меньше, чем оптимальное количество для цепочки поставок. Сосредоточив внимание на издержках, связанных с нехваткой и избытком товара, иллюстрируется первопричина этой неэффективности, как неправильное распределение рисков, связанных с запасами, между производителем и розничным продавцом. Далее мы рассматриваем ряд B2B-контрактов, которые могут координировать цепочку поставок путем согласования стимулов розничного продавца со стимулами цепочки поставок. Для каждого типа договора мы определяем параметры координации и обсуждаем, как договор достигает произвольной выгоды и распределения рисков между компаниями. Наконец, мы изучаем эффективность контрактов B2B, когда розничный продавец определяет цену продажи продукта на потребительском рынке в дополнение к количеству товара на складе.

В то же время продолжало расширяться и углубляться направление изучения конкуренции в децентрализованной модели цепи поставок. Так, ДеМигель В. в соавторстве с Адида Е. разбирали проблему конкуренции с уклоном в исследование влияния дифференциации товаров и потребителей, а в 2011 году изучили тему конкуренции между несколькими производителями и ритейлерами в цепях поставок [13]. В статье «Supply Chain Competition with Multiple Manufacturers and Retailers» изучается конкуренция в цепочке поставок, где несколько производителей конкурируют между собой (на своем уровне), чтобы поставлять набор продуктов нескольким, не склонным к риску розничным торговцам, которые конкурируют между собой (на уровне ритейлеров), чтобы удовлетворить неопределенный потребительский спрос. Авторы заявляют, что при наличии достаточно большого числа производителей и розничных торговцев эффективность цепочки поставок может быть повышена путем стимулирования правильной степени дифференциации розничной торговли.

Говоря о моделировании многоуровневых цепей поставок, стоит отметить, что их изучение и освоение начались относительно недавно. Одной из первых научных работ на данную тему стала статья Хау Ли «Децентрализованные многоуровневые цепочки поставок: стимулы и информация» 1999 года. Он акцентировал внимание на том, что основополагающая причина децентрализации заключается в следующем: «местные фирмы владеют локальной информацией, недоступной для штаб-квартиры, и поэтому на них полагаются при принятии некоторых решений. В условиях такой информационной асимметрии децентрализованная политика с соответствующим механизмом стимулирования для координации действий местных менеджеров может жестко доминировать над централизованным принятием решений без доступа к местной информации» [14]. Однако о самой стратегии с асимметричной информацией автор не говорит. Следующая значимая научная работа в сфере литературы о моделировании многоуровневых сетей была статья Корбетта Ч. и Кармакара У. от 2001 года. В ней авторы подробно описали модель конкуренции в многоуровневых сетях поставок с заданным спросом [15]. Эта модель являлась одним их ключевых шагов к созданию строгой основы для анализа конкурентной стратегии в цепях поставок. Также авторами было исследовано влияние постоянных и переменных затрат на структуру и конкурентоспособность цепи поставок и чувствительность цены линейным спросом, т.е. по модели Курно.

Следующей работой в 2005 году Кармакара в соавторстве со Скоттом Карром была статья [16], где была изучена модель конкуренции в многоуровневой цепи поставок со сборочной структурой. Фирмы в цепи поставок группируются в однородные сектора (узлы), которые содержат идентичные фирмы с одинаковыми производственными возможностями, производящие ровно один недифференцированный продукт. Каждый сектор может использовать несколько ресурсов для производства своего продукта, и эти ресурсы поставляются разными секторами. Производственный процесс в любом секторе принимается как чистая сборка в фиксированных пропорциях. Количество фирм в каждом секторе известно. Кривая спроса на конечный продукт предполагается линейной, как и производственные затраты во всех секторах. Конкуренция моделируется с помощью модели Курно, в которой фирмы выбирают объемы производства для своего нижележащего рынка таким образом, чтобы максимизировать прибыль, учитывая цены на все производственные ресурсы и все взаимодополняющие продукты. Объемы производства в секторах, поставляющих одного и того же преемника, координируются с помощью механизмов ценообразования, так что дополнительные продукты производятся в нужных пропорциях. При этих предположениях равновесные цены для любой многошаговой сборочной сети характеризуются системой линейных уравнений. Авторы заключили что, большие структуры могут быть решены либо прямым решением характеристических линейных уравнений, либо путем итерационного сокращения (сжатия) до меньших структур основной цепи поставок.

Одними из недавно опубликованных работ была статья Жоу, Кармаркара и Джанга увидевшая свет в 2015 году [6], о которой было сказано ранее, обобщившая исследования Кармакара [15-16] для случая дистрибутивной сети поставок и влияния вида функции спроса. А также отечественная диссертация за авторством Лонягиной Ю.Е, которая подробнейшим образом рассмотрела задачу координации многоуровневых сетей поставок различной структуры с точки зрения теоретико-игрового подхода, сравнила различные подходы и их результаты [17]. Однако, несмотря на это, ее формализация уровней узлов, взятая из работ Кармакара, кажется, не совсем корректной, из-за невозможности на программном уровне поддерживать структуру хоть сколько-нибудь масштабной цепи поставок. Также ею не была рассмотрена ситуация, когда принятое за принцип оптимальности равновесие по Нэшу в децентрализованной сети могло выдавать при больших значениях издержек фирм отрицательные объёмы поставок, как значения наилучшего результата.

Данная магистерская работа рассматривает ситуации, в которых исход зависит от принимаемых решений каждой из участвующих сторон, конкурирующих по модели Курно. В рамках данного исследования, теория игр является оптимальным выбором для изучения и анализа указанных положений. Исследователи, занимающиеся управлением цепями поставок, все чаще обращаются к теории игр в целях прогнозирования, изучения и поддержки стратегического принятия решений. Одним из ключевых преимуществ теории игр является ее способность учитывать воздействие каждого участника цепи на общую прибыль или иные показатели эффективности работы цепи поставок, а также способность предсказывать результаты, которые могут возникнуть при применении различных стратегий участниками.

Теория игр, как математическая отрасль стратегического взаимодействия различных сторон, сформировалась в сороковых годах прошлого века. Ее основополагающей целью является объяснение логики рационального поведения индивидуумов при конфликтах интересов, т. е. в условиях столкновения сторон, каждая из которых стремится воздействовать на развитие конфликта в своих собственных интересах. Появление теории игр принято связывать с публикацией в 1944 г. монографии Джона фон Неймана и Оскара Моргенштерна «Теория игр и экономическое поведение». В статье Дегтерева Д. А. «Зарубежные работы по теории игр» [1] автор подробно описывает, когда и какие шаги были предприняты учеными в развитии теории игр, а также какие политические или научные события служили причиной для начала исследований. Денис Андреевич приводит реальные примеры из жизни, чтобы читателю было проще понять, о чем именно идет речь и как устроены механизмы решения в игре. Но, несмотря на это, математические аспекты он не затрагивает, в статье нет ни одной формулы, так что ее можно рассматривать исключительно, как введение в данную область и краткую ретроспективу научных работ.

Авторы Л.А.Петросян, Н.А.Зенкевич и Е.В.Шевкопляс в своей книге «Теория игр / 2-ое издание» дают следующие пояснения, насчет природы изучаемой проблемы: «Неопределенность, с которой мы встречаемся в теории игр, может иметь различное происхождение. Однако, как правило, она является следствием сознательной деятельности другого лица (лиц), отстаивающего свои интересы. В связи с этим под теорией игр часто понимают теорию математических моделей принятия оптимальных решений в условиях конфликта. Таким образом, моделями теории игр можно в принципе содержательно описывать весьма разнообразные явления: экономические, правовые и классовые конфликты, взаимодействие человека с природой, биологическую борьбу за существование и т. д. Все такие модели в теории игр принято называть играми. Математическое описание игры сводится к перечислению всех действующих в ней игроков, указанию для каждого игрока всех его стратегий, а также численного выигрыша, который он получит после того, как игроки выберут свои стратегии. В результате игра становится формальным объектом, который поддается математическому анализу» [2].

Учебник за авторством Петросяна, Зенкевича и Шевкопляса является прекрасным пособием, пригодным как для первоначального, так и для углубленного изучения теории игр. Авторы постарались дать максимально систематическое изложение единой теории статистических и динамических игр. Имеются исчерпывающие формулы с подробным пояснением, приведены доказательства математических утверждений. А также дается обширная информация о различных видах и типов игр, способов их формализации и решения. Данные знания являются базой для дальнейшего продвижения и развития моей научной работы.

Одним из способов классификации игр является разделения их по типу взаимоотношений между игроками. Таким образом, игры могут быть кооперативными, где действующие лица могут заключать союзы для максимизации общего выигрыша, беря на себя некоторые обязательства перед другими игроками и координируя свои действия, а также некооперативными, иначе говоря, бескоалиционными, где каждый игрок действует обособленно от других в угоду, исключительно, собственного интереса.

Наиболее актуальной и значимой проблемой теории игр является установление того, что стоит подразумевать под оптимальным поведением в игре. «Наиболее распространенным принципом оптимального поведения или принципом оптимальности считается выбор в качестве наилучшей некоторой ситуации равновесия, которая названа в честь Джона Нэша, сформулировавшего указанный принцип оптимальности в 1951 году. Этот принцип определяет в качестве оптимальных такие ситуации, для которых любые индивидуальные отклонения игроков от входящих в эту ситуацию стратегий не могут увеличить выигрыша отклонившегося игрока при условии, что все остальные игроки придерживаются зафиксированных в этой ситуации стратегий» [3].

Учебник за авторством Н. А. Зенкевича, Л. А. Петросяна, Д. В. К. Янга можно рассматривать, как более узкоспециализированную и углубленную версию предыдущего материала [2]. Они направили свои усилия, не только на объяснения новых концепций теории игр и принципов оптимальности, но и связали их с управленческой отраслью. Представили менеджмент, как игру, а работников, фирмы и компании – как игроков. В книге рассматриваются статические и динамические игры, кооперативные и некооперативные, а также различные модели поведения в условиях конкуренции. Основное внимание уделяется анализу кооперативных динамических игр и принципам динамической устойчивости их решений. Но не рассматриваются многоуровневые задачи со сложной структурой, которые и являются объектом текущих изучений.

Статьи [11] и [12] могут быть полезны при дальнейшем развитии магистерской работы, при условии перехода от некооперативных игр, к кооперативным, где могут заключаться договора.

Исходя из всего этого, можно сказать, что продолжение работы предшественников и доработка их методов более, чем актуальна.

# Математическая модель цепи поставок

Разберем ориентированный граф, имеющий древовидную структуру, в котором каждый узел представляет несколько компаний, функционирующих между собой по экономической модели рыночной конкуренции Курно.

Корневой узел, или узел-дистрибьютор, делит произведенный им товар между своими дочерними узлами. При условии того, что дальнейшие узлы не являются терминальными, в них происходит модернизация исходного продукта, а затем продажа произведенного результата своим дочерним узлам. Терминальные узлы реализуют товар на рынках.

Данный способ устройства экономических отношений является наиболее распространённым в современном бизнесе. В качестве примера можно рассмотреть цепочку поставок в производстве электроники.

На корневой вершине ориентированного древовидного графа расположены производители полупроводниковых материалов и компонентов, например, Qualcomm, Intel, Micron Technologies, Broadcom и т.д. Они поставляют свои продукты, такие как процессоры, чипы памяти, модули связи, на следующий уровень цепочки поставок.

На следующем уровне расположены производители микросхем и компонентов для электроники, такие как Texas Instruments, Broadcom, Analog Devices и т.д. Они используют продукты, полученные от производителей полупроводниковых материалов, чтобы создавать микросхемы, процессоры и т.д., которые в свою очередь поставляются на следующий уровень цепочки поставок.

Наконец, на терминальном уровне находятся производители конечной электроники, такие как Apple, Samsung, Dell и т.д., которые используют поставленные дистрибьюторами микросхемы и компоненты, чтобы создать итоговые продукты, такие как смартфоны, ноутбуки, планшеты и т.д.

Введем ряд определений и понятий, чтобы упростить понимание дальнейших положений.

Будем рассматривать ориентированный граф,который *обозначим* через , где *X* – множество вершин, а *U* – множество ребер.

**Определение 1.** Если существует ребро  связывающее вершины и  где  тогда вершина  называется дочерней вершиной  а вершина  будет называться **родительской** по **отношению к вершине**

*Обозначим* через  множество индексов дочерних узлов, для вершины : ,  где  – количество узлов в графе.

**Замечание 1.** Если  является дочерней вершиной тогда 

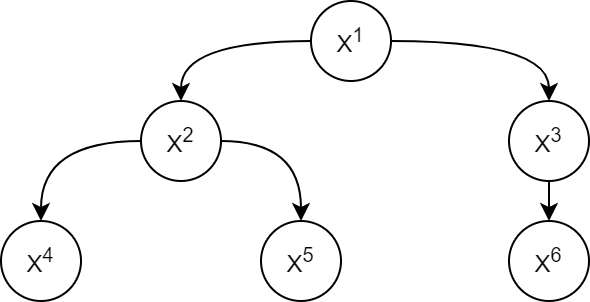


Рисунок 3 – Древовидная модель цепи поставок с номерными индексами.

Для цепи поставок, изображенной на рис. 3, имеем:

тогда 

 тогда 

тогда 

**Замечание 2.** Если для  множество  и существуетнекоторая вершина  такая, что  тогда можно реализовать следующую запись узла: 

Дополним рисунок 3, исходя из замечания 2.

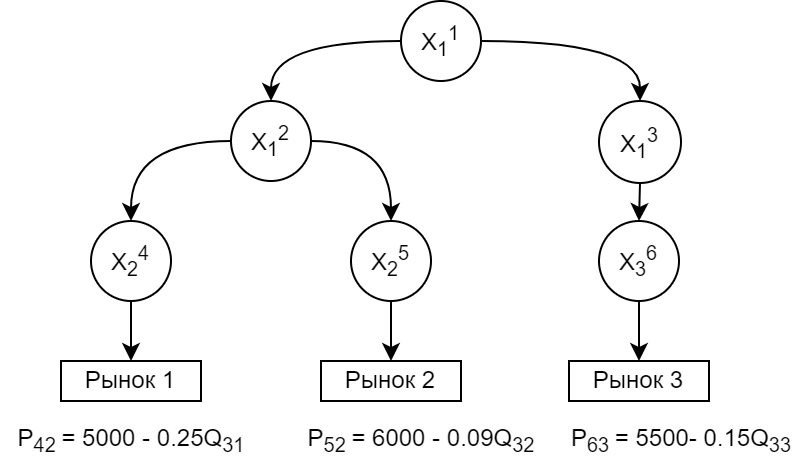
****

Рисунок 4 – Древовидная модель цепи поставок с номерными верхними индексами и нижними ссылками на родителей.

Для классификации узлов внутри цепи поставок введем ряд обозначений.

*Обозначим* корневую вершину через: 

Множество терминальных узлов обозначим через :



Множество промежуточных узлов обозначим через :



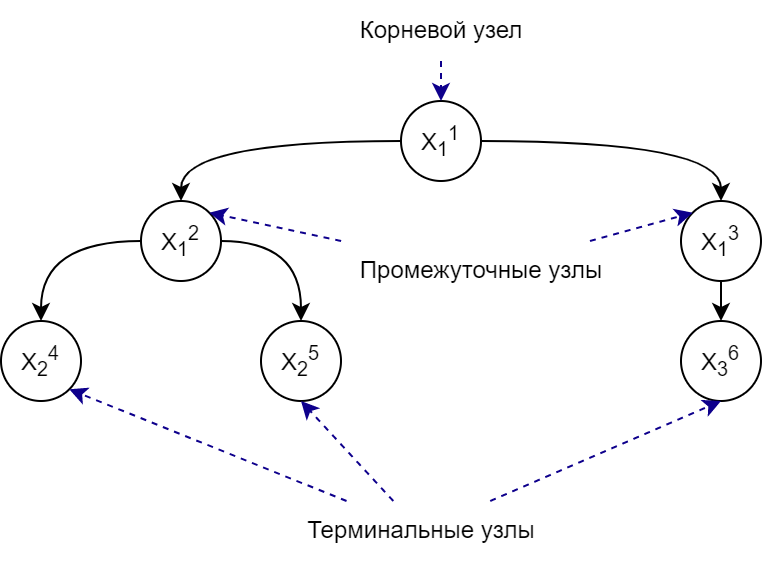
****

Рисунок 5 – Классификация узлов на примере

**Определение 2.** Под **мощностью**  **узла**  будем понимать количество фирм в узле , где .

*Построим* теоретико-игровую модель для рассматриваемой задачи [18].

Определим **множество игроков,** как , где  фирма внутри узла , , где  – количество узлов в графе. Иначе говоря, каждый игрок представляет собой компанию, находящуюся внутри каждогоузла.

Определим **множество стратегий** для каждого игрока из множества *N* следующим образом:

Для игроков, находящихся в узлах ***корневого*** и ***промежуточного*** типа:



Для игроков, находящихся в узлах ***терминального*** типа:

,

где  — это объем однородной продукции выпускаемой фирмой  узла, а  — это цена за единицу товара узла .

Определим **функции выигрыша** для каждого из игроков, как прибыль, которую может получить фирма, следующим образом.

Для игроков, находящихся в узлах *терминального* типа

.

Для игроков, находящихся в узлах *корневого*типа

.

Для игроков, находящихся в узлах *промежуточного* типа



где

 ***–*** значение издержек определенной фирмы при реализации одной единицы товара.

***–*** свободный член, константа, обозначающая цену товара на узле  при нулевом количестве продаж.

 ***–*** коэффициент пропорциональности, обозначает скорость изменения цены товара в зависимости от изменения количества продаж.

Рассматривается игра с полной информацией, то есть все игроки имеют полную информацию о характеристиках игры и выбранных стратегиях других игроков.

Таким образом, теоретико-игровая модель для задачи может быть представлена следующим образом:

 (1)

где  – множество игроков, участвующих в игре. Каждый игрок имеет свои собственные цели и интересы и выбирает свою стратегию, чтобы достичь их.

– множество стратегий, доступных каждому игроку. Стратегия — это полный набор действий, которые игрок может выбрать в любой ситуации в игре.

 – функция выигрыша каждого игрока, зависящая от выбранных ими стратегий.

Также представим формулы линейной функции цены и условия отсутствия излишков и дефицита. Каждая из фирм стремитсяк максимизации собственной прибыли [2].

Линейная функция цены:

(2)

Функция условия отсутствия излишков и дефицита:

 (3)

Условие отсутствия излишков и дефицита в цепи поставок — это ситуация, когда производитель способен оценить спрос на свой товар, точно распланировать производственную программу, чтобы покупателям хватало товара без излишков или дефицита. То есть значение объема родительского узла представляет собой сумму объемов дочерних узлов. Обеспечивается эффективный поток товаров без задержек на всех этапах поставки. Таким образом, гарантируется, что количество товаров, поступившее на вход, равно количеству товаров, перешедшему на следующий уровень, что позволяет избежать потерь и оптимизировать производственный процесс.

*Опишем процесс принятия решения в рассматриваемой модели*.

Шаг 1. Корневой узел определяет цену, по которой он продает товар *своим дочерним узлам*.

Шаг 2. Дочерние узлы корневого узла цепи поставок, если они не являются терминальными узлами, получая информацию от родителя, назначают цену товара уже *своим дочерним вершинам*.

Шаг 3. Терминальные узлы на основе цен, полученных от своих родителей, и функций спроса определяют объемы выпуска товара на рынок.

Шаг 4. Происходит процедура распределения объемов между фирмами в каждом из терминальных узлов.

Шаг 5. Информация об объемах поступает на все верх лежащие узлы и внутри каждого происходит процедура распределения объемов между фирмами.

Шаг 6. Происходит подсчет прибыли каждого из участников цепи поставок.

Данная процедура принятия решений может быть представлена многошаговой некооперативной иерархической игрой *n* лиц. Игроками являются фирмы, находящиеся в каждом из узлов. Стратегиями игроков, располагающихся в корневом и промежуточных узлах, являются объемы производства и цены, по которым продается товар, а для игроков концевых вершин – только объёмы. Функции выигрыша – это функции прибыли каждой фирмы-игрока. При этом внутри каждого узла фирмы участники конкурируют по модели Курно [22-23]. В качестве принципа оптимальности было выбрано равновесие по Нэшу [21]. Принцип оптимальности был выбран, исходя из того, что нас интересует оптимизация действий на уровне отдельного игрока (фирмы) и в рассматриваемой задаче на данный момент не предусматриваются коалиции, т.е. каждый игрок действует независимо от других в угоду только собственным интересам

*Равновесие по Нэшу* [2]:

Ситуация в игре  называется равновесием по Нэшу, если для каждого игрока  и любой стратегии этого игрока выполняется неравенство:



# Анализ существующих средств решения задачи

В процессе поиска программного обеспечения, решающего поставленную задачу, были найдены следующие представители на рынке.

* **Oracle Supply Chain Management (SCM)**

Это программное обеспечение, разработанное компанией Oracle, которое предназначено для автоматизации и управления цепочкой поставок предприятия – от закупки сырья и комплектующих до производства готовой продукции и ее доставки заказчикам. Oracle SCM включает в себя такие модули, как управление запасами, закупками, логистикой и транспортировкой, планирование производства и другие. Это мощный инструмент, который позволяет компаниям оптимизировать свои производственные и логистические процессы, улучшить эффективность и повысить прибыльность.

Ниже приведены некоторые преимущества и недостатки использования Oracle SCM:

Плюсы:

* Улучшенное управление запасами и уменьшение затрат на складское хранение, благодаря точному прогнозированию спроса, оптимизации планирования и координации всех этапов процесса поставок.
* Ускоренная логистика и оптимизация процесса поставки товаров путем координации и управления цепочкой поставок, начиная от производства до доставки товаров к конечному потребителю.
* Улучшенное управление рисками, благодаря повышенной прозрачности, точности и своевременности информации обо всех этапах процесса поставок.

Минусы:

* Высокая стоимость приобретения, настройки и поддержки Oracle SCM может стать финансовым бременем для малых и средних предприятий.
* Необходимость внедрения и адаптации к сложной системе управления цепями поставок может потребовать значительных усилий со стороны команды проекта.
* Некоторые функциональные возможности Oracle SCM могут быть излишними для ряда компаний, и они могут предпочесть более простые и доступные решения.
* **Visary SCM**

Отечественное ПО Visary SCM, которое автоматизирует закупочный цикл: от процесса изготовления до транспортировки товара в необходимом количестве и в установленные сроки, способствуя повышению спроса на продукцию компании и снижению расходов на материальные ресурсы. Оно позволяет планировать пополнения запасов, а также спроса на продукцию.

Ниже приведены некоторые преимущества и недостатки использования Visary SCM:

Плюсы:

* Основные возможности включают в себя автоматизацию планирования, управление запасами, поставками, производственными процессами и логистикой, а также мониторинг.
* Также имеются инструменты анализа данных и прогнозирования спроса, что помогает компаниям принимать более обоснованные решения и повышать эффективность своей деятельности.
* Одним из главных преимуществ Visary SCM является его высокая гибкость и возможность настроить под конкретные нужды компании.
* Визуализация цепочки поставок. Visary SCM обеспечивает прозрачность процесса управления цепочкой поставок, отображая данные в виде графиков, таблиц и диаграмм.

Минусы:

* Реализация и внедрение могут потребовать значительных затрат времени и ресурсов.
* Ограниченная функциональность. Visary SCM может не обладать некоторыми функциями, которые могут быть необходимы для определенных компаний.
* **Kinaxis RapidResponse**

Kinaxis RapidResponse — это программное обеспечение для планирования ресурсов предприятия, предназначенное для управления цепочкой поставок и операционной деятельности. Оно объединяет в себе функции планирования, прогнозирования спроса, управления запасами, закупками, выпуска продукции и другие важные возможности, необходимые для эффективного управления производственными процессами [24].

Ниже приведены некоторые преимущества и недостатки использования Kinaxis RapidResponse:

Плюсы:

* Система быстро реагирует на изменения в планах производства и спроса, что позволяет предприятию оперативно реагировать на рыночные изменения.
* Одна из главных преимуществ Kinaxis RapidResponse – это многофункциональность системы. Она позволяет управлять всеми процессами предприятия в одной системе.
* Большое количество отчётов, которые можно настроить под отдельные нужды.

Минусы:

* Для полноценного использования системы требуется обучение персонала, что может занять некоторое время и привести к дополнительным затратам.
* Система может быть дорогой для малых и средних предприятий, которые не могут позволить себе значительные затраты на приобретение и настройку системы.
* Перед использованием системы необходимо интегрировать ее с другим программным обеспечением компании.
* Связи с необходимостью глубокого изучения сложного и устаревшего интерфейса программы, перемещение и мониторинг различных ресурсов может проходить со значительными трудностями.

Однако, ключевыми минусами рассмотренных ранее программных продуктов является, тот факт, что поиск решения осуществляется на ограниченных по размерности цепочках поставок, а также они не анализируют их в целом. Не каждое ПО внутри себя имеет необходимый теоретико-игровой аппарат для моделирования конкуренции по Курно. Тем временем, одной из целей текущей работы является разработка ПО, которое позволяло бы оптимизировать цепь поставок по прибыли для произвольного числа участников и узлов используя принципы теории игр и оптимальности Нэша.

# Выводы к разделу

В настоящим разделе была проанализирована предметная область и проведен обзор актуальной литературы по теме, в процессе которого выявлены основные достижения и проблемы на данный момент.

Также была построена математическая модель для рассматриваемой задачи.

Был произведен сравнительный анализ аналогов, установлены их сильные и слабые стороны, а также выявлено отсутствие прямых эквивалентов разрабатываемого программного продукта.

# Разработка алгоритма решения

Опишем алгоритм построения оптимального решения.

1. Представим исходные данные в качестве древовидного графа, где в узлах хранится изначальная информация о количестве фирм и их издержек. Терминальные узлы выходят на рынки, где нам известна функция спроса на товар. Выразим из функции спроса функцию цены.
2. Будем производить обратный обход дерева:
   1. Если текущая вершина принадлежит к терминальному множеству:
      1. Функция цены нам уже известна.
      2. Формируем систему функций прибыли.
      3. Применяем к системе условия максимума.
      4. Из полученной системы производных выражаем функции оптимальных объемов фирм.
      5. Суммируем функции оптимальных объемов фирм, чтобы получить функцию объема текущего узла.
   2. Если текущая вершина принадлежит к промежуточному множеству:
      1. Применяем условие отсутствия дефицита и излишков.
      2. Выражаем функцию цены текущего узла.
      3. Формируем систему функций прибыли.
      4. Применяем к системе условия максимума.
      5. Из полученной системы производных выражаем функции оптимальных объемов фирм.
      6. Суммируем функции оптимальных объемов фирм, чтобы получить функцию объема текущего узла.
   3. Если текущая вершина принадлежит к корневому множеству:
      1. Применяем условие отсутствия дефицита и излишков.
      2. Выражаем функцию цены текущего узла.
      3. Формируем систему функций прибыли.
      4. Применяем к системе условия максимума.
      5. Получаем значения объемов фирм.
      6. Рассчитываем значение цены.
      7. Рассчитываем значения прибыли фирм.
3. Спускаясь по дереву вниз, рассчитываем значения цены, объёмов и прибыли для всех оставшихся узлов.

Покажем, что данный алгоритм приведёт к построению равновесия по Нэшу. Рассмотрим терминальную вершину 2 цепи поставок, являющейся дочерней к корневой вершине 1. Будем считать для определённости, что имеем дело со случаем дуополии, т.е. предположим, что в узле действуют всего две фирмы, т.е. . Пусть эти фирмы конкурируют в соответствии с моделью Курно.

Функция цены для обоих игроков, в соответствии с формулой (2), имеет вид:  А функции прибыли, для игроков 1 и 2, находящихся в узлах терминального типа, имеют следующий вид:



Применим необходимое условие экстремума первого порядка и получим систему (4):

 (4)

Решая систему (4), получим функции реакции конкурирующих фирм (5) [3]:

 (5)

Решим систему (4), подставив функции (5). Имеем:

 (6)

Проанализируем вторые производные от функций прибыли:

,

Они отрицательны. Это означает, что функции прибыли обеих фирмы строго вогнуты по объемам выпуска ,, соответственно, этих фирм. Это означает, что условия первого порядка являются необходимыми и достаточными для равновесия по Нэшу [3].

Таким образом, мы показали, что найденные объёмы в системе (6) являются равновесием по Нэшу:

,

Можно показать, что данный результат имеет место как для большего числа фирм внутри одной вершины, т.е., когда , так и для всех остальных множеств вершин в рассматриваемой цепи, а именно корневых и промежуточных.

Таким образом, реализация, описанного выше алгоритма, позволит рассчитать оптимальные объёмы в цепи поставок и соответствующие цены.

# Пример решения задачи

Рассмотрим теоретико-игровую модель цепи поставок c шестью узлами изображенную на рисунке 6. В таблице 2 представлены исходные данные.

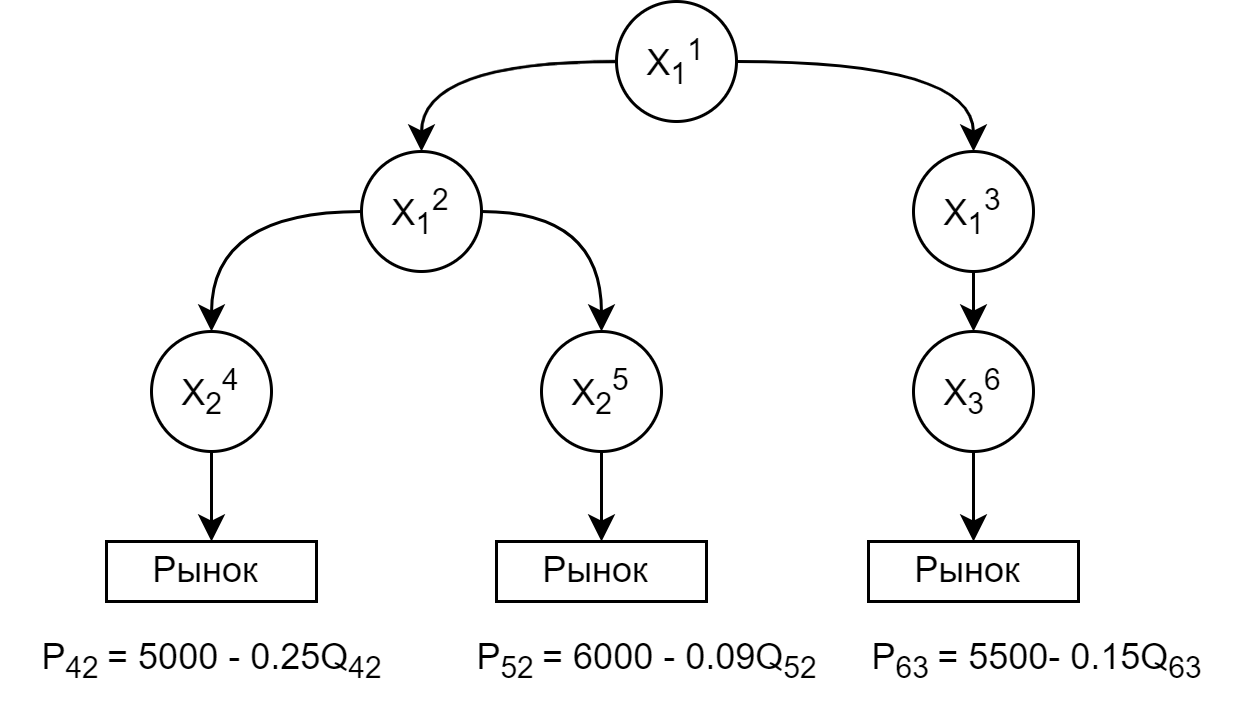


Рисунок 6 – Цепь поставок с тремя узлами дистрибьюторов и тремя узлами ретейлеров

Таблица 1 — Обозначения

|  |  |
| --- | --- |
| Обозначения | Пояснение |
|  | Узел номер  с родителем |
|  | Фирм *k* внутри узла |
|  | Суммарный объём однородной продукции в узле |
|  | Объем выпуска продукции фирмы *k* в узле |
|  | Прибыль фирмы *k* в узле |
|  | Цена в узле |

Таблица 2 — Исходные данные

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Узел |  |  |  |  |  |  |
| Кол-во фирм в узле |  |  |  |  |  |  |
| Затраты |  |  |  |  |  |  |

Для построения равновесия по Нэшу в рассматриваемой игрераспишем функции прибыли для всех фирм из терминальных узлов:

 (7)

 (8)

 (9)

Применив к функциям (7), (8) и (9) необходимое условие максимума, получаем системы уравнений:

 (10)

 (11)

 (12)

Также возьмем вторую производную от функций (7), (8) и (9), чтобы показать ее отрицательность, это доказывает, что необходимое условия максимума являются достаточными:







В результате решения систем (10) – (12) получим выражения для объёма поставок  Затем, исходя из условия отсутствия дефицита и излишков, получаем соотношение:

 (13)

 (14)

Из соотношения (13) выразим:

 (15)

Из соотношения (14) выразим :

 (16)

Зная функции цен, мы можем выражать функции прибыли для фирм родительских узлов. Дальнейшие действия осуществляются по аналогии с приведенными, вплоть до множества корневых узлов, при достижении которого получаем численные значения объёмов для фирм корневого узла. Зная данные значения, подставляем их в ранее выведенные формулы для получения информации об объёмах и прибыли.

Таблица 3 — Результаты

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Узел |  |  |  |  |  |  |
| Объём выпуска продукции для фирмы |  |  |  |  |  |  |
| Цена |  |  |  |  |  |  |
| Прибыль |  |  |  |  |  |  |

Ручное решение данной проблемы потребовало значительного количества затраченного времени и вычисления математических операций. Однако, время, необходимое для получения решения и сложность выполняемых человеком операций, несомненно, будет значительно сокращено с разработкой программного обеспечения для получения оптимального, с точки зрения прибыли, решения в таких цепях.

# Поведение при отрицательных объемах

При поиске равновесия Нэша в игре с нелинейными функциями выигрыша для каждого участника мы решаем задачу оптимизации с ограничениями, где каждый участник максимизирует свою функцию выигрыша, учитывая действия других участников. При этом существуют дополнительные ограничения в виде неравенств, которые гарантируют, что объемы производства (или цены, если они являются стратегиями) не могут быть отрицательными. В общем виде можно записать задачу нелинейного программирования следующим образом:



,, (17)

, ,,

где  – количество узлов в графе, – мощность конкретного узла.

Сформулированная задача (17) является стандартной задачей нелинейной оптимизации. Необходимыми условиями оптимальности для неё являются условия Куна-Таккера [3, 31]. Из этих условий следует, что если при решении задачи мы получаем отрицательное значение объема производства или цены, то оно приравнивается к нулю. С практической точки зрения это может происходить в связи с разными факторами, такими как, существенные издержки по сравнению с конкурентами, завышенные ценны у фирм родительского узла и т.д. Но приравняв к нулю объём производства, мы фактически исключаем из процесса одного из игроков, что может привести к несбалансированности в игре и необходимости пересчета решения, так как участник был учтен в исходной модели, и его исключение может нарушить равновесие между оставшимися игроками.

Объяснение положений исходя из логики предметной области:

* *Ситуация 1.* Фирме невыгодно производить и участвовать в рыночных отношениях. Связи с этим она выпадает из товарно-денежного оборота. Она ничего не тратит, ничего не производит, ничего не продает.
* *Ситуация 2.1.* В рамках цепи поставок целому узлу невыгодно производить свой товар и участвовать в рыночных отношениях. Связи с этим он упраздняется из цепочки, а его дочерние узлы получая сырье от нового родительского узла изменяют собственное производство, не изменяя своих издержек.
* *Ситуация 2.2.* В рамках цепи поставок нерентабельным является выход терминального узла на определенный рынок товаров. В таком случае, если нет возможности перейти на другой рынок, целое поддерево цепи поставок может быть закрыто.
* *Ситуация 2.3.* В рамках цепи поставок корневому узлу невыгодно производить свой товар и участвовать в рыночных отношениях. Это приводит к невозможности функционирования низлежащих узлов, т.к. отсутствует исходное сырье.

Рассмотрим возникающие ситуации и предлагаемые действия устранения проблемы:

1. *Если хотя бы одна фирма в узле имеет неотрицательный объем.*  
   Предлагается вычеркнуть фирмы с отрицательным объёмом из рассмотрения и провести поиск решения сначала изменив исходные данные. В результирующем ответе считать объем вычеркнутых фирм равным нулю.
2. *Если все фирмы узла имеют отрицательные объёмы.*
   1. *Если узел является промежуточным.*

Предлагается вычеркнуть данный узел из дальнейшего рассмотрения. Произвести реструктуризацию цепи поставок, таким образом, чтобы родительский узел удаляемой вершины, стал родителем для всех его дочерних узлов. Провести поиск решения с изменёнными значениями и реконструированной цепочкой поставок. В результирующем ответе считать объем вычеркнутых фирм равным нулю.

* 1. *Если узел является терминальным*

Предлагается вычеркнуть узел и рынок, на который он выходит из дальнейшего рассмотрения. А также родительские узлы, если у них нет альтернативных выходов на рынки. Провести поиск решения с изменёнными значениями и реконструированной цепочкой поставок. В результирующем ответе считать объем вычеркнутых фирм равным нулю.

* 1. *Если узел является корневым.*

Предлагается считать всю цепочку поставок нерентабельной. Обозначить объемы всех фирм равными нулю. Не производить дополнительных вычислений.

# Пример решения задачи с отрицательными объемами

Рассмотрим ситуации возникновения отрицательных объемов. Для этого изменим исходные данные задачи из предыдущего пункта, таким образом, чтобы издержки фирмы  возросли с 720 до 2000 усл. ед.

Таблица 4 — Исходные данные

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Узел |  |  |  |  |  |  |
| Кол-во фирм в узле |  |  |  |  |  |  |
| Затраты |  |  |  |  |  |  |

Ход решения аналогичен ранее рассмотренной задачи. В результате вычислений получим значения, записанные в таблице 5.

Таблица 5 — Результаты

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Узел |  |  |  |  |  |  |
| Объём выпуска продукции для фирмы |  |  |  |  |  |  |
| Цена |  |  |  |  |  |  |
| Прибыль |  |  |  |  |  |  |

Как мы можем видеть, объем для фирмы составляет -567, что является недопустимым в меру условия неотрицательности данного параметра.

Вычеркнем данную фирму из рассмотрения и снова проведем вычисления. Итоговые результаты представлены в таблице 6, также учтем ранее удаленную компанию, присвоив ей нулевое значение объемов продаж и прибыли соответственно.

Таблица 6 — Результаты

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Узел |  |  |  |  |  |  |
| Объём выпуска продукции для фирмы |  |  |  |  |  |  |
| Цена |  |  |  |  |  |  |
| Прибыль |  |  |  |  |  |  |

# Выводы к разделу

В настоящем разделе был разработан алгоритм построения равновесия по Нэшу в цепи поставок древовидной структуры.

Также рассмотрено возможное поведение при ситуациях возникновения отрицательных объемов в фирмах цепи поставок.

Для демонстрации эффективности предложенного алгоритма и его практической ценности был осуществлен разбор аналитических примеров решения.

# Разработка и проектирование программного обеспечения

# Функциональные требования к разрабатываемой системе

В таблице 7 представлены основные функциональные требования к разрабатываемому программному продукту. Реализация спецификации осуществлялась исходя из требований к ПО [25;26;27].

Таблица 7 — Функциональные требования

|  |  |
| --- | --- |
| FUN-ID | Описание |
| FUN-01 | Система должна иметь графический интерфейс (GUI). |
| FUN-02 | Пользователь должен иметь возможность добавлять новые узлы в цепь поставок. |
| FUN-03 | Пользователь должен иметь возможность удалять узлы из цепи поставок. |
| FUN-04 | Пользователь должен иметь возможность по двойному клику на узел цепи вносить в него данные о фирмах (Название компании, значение издержек за единицу товара). |
| FUN-05 | Пользователь должен иметь возможность по двойному клику на терминальных узел цепи вносить данные о коэффициентах линейной функции цены. |
| FUN-06 | Система должна иметь возможность сбросить всю внесенную в нее информацию по нажатию одной кнопки (Сбросить). После чего на экране отображается состояние системы по умолчанию (пустой шаблон). |
| FUN-07 | Пользователь должен иметь возможность внешне моделировать цепь поставок (перемещать узлы зажав ЛКМ по плоскости). |
| FUN-08 | Система должна рассчитывать и выводить на экран оптимальные значения для смоделированной пользователем цепи поставок используя алгоритм, описанный в пункте 1.3. |
| FUN-09 | Система должна осуществлять валидацию вводимых пользователем данных в соответствующие поля. |
| FUN-10 | Пользователь должен иметь возможность изменять вводимые ранее данные о фирмах внутри узлов. |
| FUN-11 | Пользователь должен иметь возможность закрыть поле ввода, не изменяя его, то есть только просмотреть его. |
| FUN-12 | Система должна отображать цепь поставок в виде древовидного графа. |
| FUN-13 | Система должна иметь возможность сохранения структуры цепи поставок и информации о фирмах, содержащихся в ней в формат json. |
| FUN-14 | Система должна иметь возможность загрузки структуры цепи поставок и информации о фирмах, содержащихся в ней из формата json. |
| FUN-15 | Система должна иметь возможность сохранения рассчитанных результатов в формате xlsx. |
| FUN-16 | Пользователь должен иметь возможность по одинарному клику выбирать узел. |

# Нефункциональные требования к разрабатываемой системе

Нефункциональные требования (NF-требования) определяют, как программное обеспечение должно работать и какими свойствами оно должно обладать, чтобы удовлетворять нужды пользователей [28]. В таблице 8 представлены основные нефункциональные требования к разрабатываемому программному продукту.

Таблица 8 — Нефункциональные требования

|  |  |
| --- | --- |
| NF-ID | Описание |
| NF-01 | Программа должна быть стабильной и надежной, не должна выдавать ошибки или аварийно завершаться (Надежность и безопасность). |
| NF-02 | Программа должна быстро загружаться, быстро открывать и обрабатывать файлы, работать без промедлений и нареканий (Производительность). |
| NF-03 | Интерфейс программы должен быть интуитивно понятным и минималистичным, что позволит пользователю быстро освоить программу и начать ее использовать (Удобство использования). |
| NF-04 | Программа должна хорошо работать с другими программами и операционными системами, чтобы пользователи могли легко обмениваться информацией и файлами. |
| NF-05 | Приложение должно работать на различных операционных системах. |
| NF-06 | Программа должна уметь адаптироваться к разным размерам экранов и разрешениям, чтобы пользователи могли использовать ее на разных устройствах (например, на мониторе, на ноутбуке). |

# Требования к организации входных и выходных данных

Для организации входных данных необходимо разработать графический интерфейс, который позволит пользователю схематично добавлять новые узлы и заполнять такие поля как: Название фирмы, Издержки фирмы для каждого узла, а также рыночные коэффициенты А и Б для терминальных узлов. Необходимо предусмотреть возможность редактирования и удаления уже существующих узлов.  Данные, вводимые вручную, проверяются на корректность после попытки сохранения, также предусмотрен запрет в конкретных полях на определенные символы.

Для организации выходных данных нужно создать таблицу, которая будет содержать информацию о всех фирмах во всех узлах цепочки поставок, включая их цены, объемы товаров и прибыль. Возможно, также стоит предусмотреть возможность сортировки и фильтрации данных в таблице для удобства пользователя.

При организации входных и выходных данных необходимо учитывать, что они должны быть четко структурированы и удобны для работы пользователя. Необходимо предусмотреть все возможные варианты заполнения полей и обработки данных, чтобы избежать ошибок и искажений информации. Используемые типы данных представлены в таблице 9.

Таблица 9 — Принятые обозначения и типы данных

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | float | Представляет собой числа с плавающей точкой двойной точности (64 бита) и имеет ограничения на точность и диапазон значений. Диапазон значений типа float составляет от приблизительно -1,8 × 10^308 до приблизительно 1,8 × 10^308, что обеспечивает возможность представления очень больших и очень маленьких чисел. Однако точность типа float ограничена 16 десятичными знаками. |
| 2 | string | Представляет собой последовательность символов Unicode произвольной длины, которые используются для хранения текстовой информации. |
| 3 | int | Целочисленные значения в диапазоне от -231 (-2 147 483 648) до 231-1 (2 147 483 647) |

Таблица 10 — Словарь данных

|  |  |  |  |
| --- | --- | --- | --- |
| Структура или  элемент данных | Тип данных | Длина | Пояснение |
| nodeNumber | int | max | Генерируемый системой порядковый номер узла в цепи поставок, начиная с 1. (Not Null) |
| nodeParent | int | max | Номер родительской вершины, необходимый для индексирования наследственности. Принимает значения из множества номеров узлов. (Not Null) |
| firmName | string | 400 символов | Название фирмы, может содержать буквы латиницы и кириллицы числа и символы из следующего после двоеточия списка:  [0-9] / ! \*’ ; : @ & = + $ ? # \_ |
| firmCost | float | max | Издержки фирмы, которые являются строго положительным числом с плавающей запятой |
| a | float | max | Свободный член, константа, обозначающая цену товара на узле при нулевом количестве продаж, который является строго положительным числом с плавающей запятой |
| b | float | max | Коэффициент пропорциональности, обозначает скорость изменения цены товара в зависимости от изменения количества продаж, который является строго положительным числом с плавающей запятой |
| price | float | max | Цена за ед. Продукцию фирмы, которые являются строго положительным числом с плавающей запятой. |
| volume | float | max | Объем однородной продукции фирмы, которые являются строго положительным числом с плавающей запятой. |
| profit | float | max | Итоговая прибыль фирмы, которые являются строго положительным числом с плавающей запятой. |

# Варианты использования системы

Словесные представления вариантов использования приведены в таблице 11.

Таблица 11 — Варианты использования системы.

|  |  |
| --- | --- |
| Идентификатор ВИ | UC-01 |
| Наименование: | Создание дочерней вершины |
| Автор: | Владислав Чепусов |
| Дата создания | 14.04.2023 |
| Требования: | FUN-01, FUN-02, FUN-12, FUN-16 |
| Действующее лицо: | Пользователь |
| Описание: | Создает новую дочернюю вершину для выбранной исходной вершины |
| Условие-триггер: | Пользователь вошел в приложение, чтобы смоделировать цепь поставок. |
| Предварительные условия: | Пользователь выбрал уже имеющуюся вершину в цепи. |
| Входные данные | Выбранная вершина, ее номер и расположение в цепи. |
| Выходные условия: | Создание новой вершины в цепи поставок. |
| Основной поток: | 1. Пользователь, используя графическое представление цепи поставок, выбирает конкретный узел и нажимает по нему 1 раз ЛКМ. 2. Пользователь выбирает опцию создания новой вершины на панели инструментов. 3. Приложение добавляет новую вершину в древовидный граф. |
| Исключения: | Пользователь пытается использовать функцию добавления без выбора узла.   1. Система ничего не делает, ожидает дальнейшего выбора пользователя. |
| Идентификатор ВИ | UC-02 |
| Наименование: | Удаление вершины |
| Автор: | Владислав Чепусов |
| Дата создания | 14.04.2023 |
| Требования: | FUN-01, FUN-03, FUN-12, FUN-16 |
| Действующее лицо: | Пользователь |
| Описание: | Позволяет пользователю удалить выбранную вершину из древовидного графа. |
| Условие-триггер: | Пользователь вошел в приложение, чтобы смоделировать цепь поставок. |
| Предварительные условия: | Пользователь выбрал уже имеющуюся вершину в цепи. |
| Входные данные | Выбранная вершина, ее номер и расположение в цепи. |
| Выходные условия: | Удаление вершины и ее связей, а также всех дочерних узлов |
| Основной поток: | 1. Пользователь выбирает опцию удаления вершины на панели инструментов 2. Пользователь выбирает вершину для удаления 3. Система удаляет выбранную вершину из графа и отображения на экране, а также ее дочерние узлы если те имеются. |
| Исключения: | Пользователь пытается использовать функцию удаления без выбора узла.   1. Система ничего не делает, ожидает дальнейшего выбора пользователя. |
| Идентификатор ВИ | UC-03 |
| Наименование: | Управление графом |
| Автор: | Владислав Чепусов |
| Дата создания | 14.04.2023 |
| Требования: | FUN-01, FUN-07, FUN-12, FUN-16 |
| Действующее лицо: | Пользователь |
| Описание: | Изменение расположения в пространстве вершин графа |
| Условие-триггер: | Пользователь вошел в приложение |
| Предварительные условия: | Пользователь вошел в приложение, чтобы смоделировать цепь поставок. |
| Выходные условия: | Измененный вид древовидного графа |
| Основной поток: | 1. Пользователь выбирает опцию управления на панели инструментов 2. Пользователь выбирает вершину для перемещения 3. Пользователь перемещает вершину 4. Пользователь повторяет пункт 2-3 пока его не устроит внешний вид 5. Система фиксирует изменение графа и отображает их на экране |
| Исключения: | — |
| Идентификатор ВИ | UC-04 |
| Наименование: | Внесение данных о цепи поставок |
| Автор: | Владислав Чепусов |
| Дата создания | 14.04.2023 |
| Требования: | FUN-01, FUN-04, FUN-05, FUN-09, FUN-16 |
| Действующее лицо: | Пользователь |
| Описание: | Пользователь вносит данные о фирмах, находящихся внутри узлов цепи поставок |
| Условие-триггер: | Пользователь вошел в приложение, чтобы получить оптимальные по критерию прибыли значения для фирм определенной цепи поставок. |
| Предварительные условия: | Пользователь сформировал древовидный граф цепи поставок |
| Входные условия: | Название фирмы;  Издержки фирмы:  Коэффициенты a, b линейных функций цен терминальных рынков; |
| Выходные условия: | Сохраненные данные о фирмах хранящийся внутри узлов |
| Основной поток: | 1. Пользователь дважды нажимает на выбранную *корневую* или *промежуточную* вершину ЛКМ 2. В открывшимся окне он вбивает в таблицу названия фирм и значения их издержек 3. После того как все данные были внесены, пользователь нажимает «ОК» 4. Система сохраняет информацию о фирмах и привязывает их к определенному узлу цепи поставок |
| Альтернативный поток: | 1. Пользователь дважды нажимает на выбранную *терминальную* вершину ЛКМ 2. В открывшимся окне он вбивает коэффициенты линейной функции цены для рынка, на который выходит данный узел 3. После того как все данные были внесены, пользователь нажимает «ОК» 4. В открывшимся окне он вбивает в таблицу названия фирм и значения их издержек 5. После того как все данные были внесены, пользователь нажимает «ОК» 6. Система сохраняет информацию о фирмах и привязывает их к определенному узлу цепи поставок. Также сохраняет данные рынка. |
| Исключения: | Пользователь не ввел никаких данных.   1. Система ничего не делает, ожидает дальнейшего выбора пользователя. |
| Идентификатор ВИ | UC-05 |
| Наименование: | Расчет оптимальных значений |
| Автор: | Владислав Чепусов |
| Дата создания | 14.04.2023 |
| Требования: | FUN-08 |
| Действующее лицо: | Пользователь |
| Описание: | Расчет оптимальных значений объемов и цен с точки зрения прибыли. |
| Условие-триггер: | Пользователь вошел в приложение, чтобы получить оптимальные по критерию прибыли значения для фирм определенной цепи поставок |
| Предварительные условия: | Пользователь сформировал древовидный граф цепи поставок. Также внес информацию о всех фирмах и рынках в цепи |
| Выходные условия: | Таблица с рассчитанные значениями цен, объемов, прибыли для каждой фирмы всех узлов |
| Основной поток: | 1. Пользователь сформировал и заполнил данные в цепи поставок 2. Пользователь выбирает опцию «Рассчитать» на панели инструментов 3. Открывается таблица с результирующими данными и двумя кнопками «Закрыть» и «Экспорт в Excel» |
| Исключения: | — |
| Идентификатор ВИ | UC-06 |
| Наименование: | Сохранение рассчитанных данных в Excel-файл |
| Автор: | Владислав Чепусов |
| Дата создания | 14.04.2023 |
| Требования: | FUN-15 |
| Действующее лицо: | Пользователь |
| Описание: | Экспортирование значений, рассчитанных программой в Excel-файл |
| Условие-триггер: | Пользователь захотел сохранить данные для дальнейших манипуляций в среде Excel |
| Предварительные условия: | Был произведен расчет оптимальных значений программой |
| Выходные условия: | Excel-файл с данными |
| Основной поток: | 1. Пользователь выбирает опцию экспорта данных. 2. Приложение отображает диалоговое окно для выбора имени и места сохранения файла. 3. Пользователь выбирает имя и место сохранения файла и нажимает кнопку "Сохранить". 4. Приложение экспортирует данные в формат .xlsx и сохраняет файл. |
| Исключения: | Файл с таким именем уже существует   1. Система предлагает заменить существующий файл или переименовать текущий. 2. В случае замены файла система спрашивает точно ли пользователь этого хочет. |
| Идентификатор ВИ | UC-07 |
| Наименование: | Загрузить цепь поставок |
| Автор: | Владислав Чепусов |
| Дата создания | 14.04.2023 |
| Требования: | FUN-14 |
| Действующее лицо: | Пользователь |
| Описание: | Позволяет пользователю загружать ранее сохраненные цепи поставок из файла. |
| Условие-триггер: | Пользователь вошел в приложение и хочет продолжить работу с ранее сохраненной цепочкой поставок. |
| Предварительные условия: | — |
| Выходные условия: | Отображается древовидный граф цепи поставок, внутри узлов хранятся данные о фирмах. |
| Основной поток: | 1. Пользователь выбирает на опцию «Загрузить» на панели инструментов. 2. Приложение отображает диалоговое окно для выбора файла в формате json. 3. Система отображает загруженный граф на экране. |
| Исключения: | Ошибка при загрузке файла.   1. Выводится сообщение об ошибке. 2. Пользователю предлагают попробовать загрузить снова файл. |
| Идентификатор ВИ | UC-08 |
| Наименование: | Сохранить цепь поставок |
| Автор: | Владислав Чепусов |
| Дата создания | 14.04.2023 |
| Требования: | FUN-13 |
| Действующее лицо: | Пользователь |
| Описание: | Позволяет пользователю сохранить данные о текущей цепи поставок в формате json для дальнейшей работы с ней. |
| Условие-триггер: | Пользователь хочет сохранить исходные данные. |
| Предварительные условия: | Пользователь сформировал древовидный граф цепи поставок. |
| Выходные условия: | Файл с сохраненным графом формата json. |
| Основной поток: | 1. Пользователь выбирает опцию «Сохранить» на панели инструментов. 2. Приложение отображает диалоговое окно для выбора имени и места сохранения файла. 3. Пользователь выбирает имя и место сохранения файла и нажимает кнопку "Сохранить". 4. Приложение сохраняет файл формата json. |
| Исключения: | Файл с таким именем уже существует   1. Система предлагает заменить существующий файл или переименовать текущий. 2. В случае замены файла система спрашивает точно ли пользователь этого хочет. |
| Идентификатор ВИ | UC-09 |
| Наименование: | Сброс в начальное состояние |
| Автор: | Владислав Чепусов |
| Дата создания | 14.04.2023 |
| Требования: | FUN-06 |
| Действующее лицо: | Пользователь |
| Описание: | Система сбрасывает всю внесенную в нее информацию о цепи поставок. Отображая граф по умолчанию. |
| Условие-триггер: | Пользователь вошел в приложение |
| Предварительные условия: | — |
| Выходные условия: | Начальный граф, состоящий из одной корневой вершины |
| Основной поток: | 1. Пользователь выбирает опцию «Заново» на панели инструментов. 2. Приложение удаляет информацию о текущей цепи поставок 3. Система отображает граф по умолчанию из одной вершины |
| Исключения: | — |
| Идентификатор ВИ | UC-10 |
| Наименование: | Изменить данные в узле |
| Автор: | Владислав Чепусов |
| Дата создания | 14.04.2023 |
| Требования: | FUN-01,FUN-04,FUN-05,FUN-09 FUN-10,FUN-11 |
| Действующее лицо: | Пользователь |
| Описание: | Позволяет пользователю изменять данные вершины в цепи поставок |
| Условие-триггер: | Пользователь решил изменить исходные данные, чтобы получить оптимальные по критерию прибыли значения для фирм определенной цепи поставок. |
| Предварительные условия: | Данные в узле были внесены до текущей операции |
| Входные условия: | Название фирмы;  Издержки фирмы:  Коэффициенты a, b линейных функций цен терминальных рынков; |
| Выходные условия: | Измененные сохраненные данные о фирмах хранящийся внутри узлов |
| Основной поток: | 1. Пользователь дважды нажимает на выбранную *корневую* или *промежуточную* вершину ЛКМ 2. В открывшимся окне он изменяет в таблице названия фирм и/или значения их издержек 3. После того как все данные были внесены, пользователь нажимает «ОК» 4. Система сохраняет информацию о фирмах и привязывает их к определенному узлу цепи поставок |
| Альтернативный поток: | 1. Пользователь дважды нажимает на выбранную *терминальную* вершину ЛКМ 2. В открывшимся окне он изменяет коэффициенты линейной функции цены для рынка, на который выходит данный узел 3. После того как все данные были внесены, пользователь нажимает «ОК» 4. В открывшимся окне он изменяет в таблице названия фирм и/или значения их издержек 5. После того как все данные были внесены, пользователь нажимает «ОК» 6. Система сохраняет измененную информацию о фирмах и привязывает их к определенному узлу цепи поставок. Также сохраняет данные рынка. |
| Исключения: | Пользователь не захотел изменять данные.   1. Пользователь может нажать на кнопку отмены, чтобы отменить внесенные им изменения. 2. Система сохранит исходные данные |

# Выбор и обоснование используемых инструментов

При изучении и анализе возможных инструментов для разработки программного продукта, был сделан выбор в пользу следующих средств, библиотек и языка программирования:

* ЯП **—** Python (версия 3.9.13)

Python является одним из наиболее популярных языков программирования для разработки научных и инженерных приложений, включая приложения для оптимизации различных алгоритмов. Ниже приведены несколько причин, по которым Python может быть хорошим выбором для разработки десктопного приложения:

1. *Экономия рабочего времени*: Python имеет простой и понятный синтаксис, который позволяет быстрее создавать программы. Кроме того, Python имеет множество библиотек и фреймворков, помогающих автоматизировать задачи и ускорять разработку.
2. *Мощность и гибкость*: Python является мощным языком программирования, который поддерживает многопоточность и многозадачность, что делает его очень гибким и универсальным для разработки различных приложений. Python также может использоваться для создания приложений с графическим интерфейсом, что делает его подходящим для создания десктопных приложений.
3. *Простота и удобство в использовании*: Python имеет простой и понятный синтаксис, что делает его очень легким для изучения и использования. Python также имеет обширную библиотеку, которая содержит множество инструментов и функций, которые могут использоваться для разработки научных и инженерных приложений.
4. *Многофункциональность*: Python может использоваться для разработки широкого спектра приложений, включая создание веб-сайтов, научных вычислений, игр и многого другого.
5. *Математические и графические библиотеки*: Python имеет множество библиотек, включая NumPy, SciPy и Matplotlib, которые позволяют работать с математическими функциями и создавать красивые графики и диаграммы.
6. *Большое и активное сообщество:* Python имеет огромное сообщество разработчиков, которые создают и поддерживают множество библиотек и инструментов. Это делает Python идеальным выбором для разработки научных и инженерных приложений, так как вы можете легко получить доступ к широкому спектру библиотек и инструментов, которые могут помочь в оптимизации математических алгоритмов.
7. *Переносимость*: Python является кроссплатформенным языком программирования, что означает, что приложения, созданные на Python, могут работать на различных операционных системах и архитектурах, включая Windows, macOS и Linux.
8. *Библиотеки и фреймворки*: Python имеет огромную коллекцию библиотек и фреймворков, которые позволяют упростить разработку и повысить производительность приложения.
9. *Популярность*: Python является одним из самых популярных языков программирования в мире, и его использование может существенно улучшить карьерные перспективы программиста.

Python — это язык программирования, который часто используется при работе с математическими и научными расчетами. Он имеет широкий набор библиотек, таких как NumPy, SciPy, Matplotlib, которые облегчают решение сложных математических и инженерных задач. Кроме того, Python — это интерпретируемый язык, что означает, что он может очень быстро выполнить простые операции и не требует компиляции перед запуском программы.

Все эти преимущества делают его отличным выбором для математических и инженерных задач благодаря широкому набору библиотек, простому синтаксису и скорости интерпретации. Он делает решение задач легким и эффективным для программистов на любом уровне опыта и навыков.

* IDE **—** PyCharm (версия 2022.3.3)

PyCharm - это интегрированная среда разработки (IDE) для языка программирования Python, разработанная компанией JetBrains. Вот ряд преимуществ, по которым PyCharm может быть лучше аналогов [29]:

1. *Интеллектуальная подсветка синтаксиса и автодополнение:* PyCharm предоставляет широкие возможности, такие как автоматическое завершение кода, быстрое переходы по определениям функций, классов и модулей, более точную подсветку синтаксиса, которая позволяет различать типы переменных (локальные, глобальные и т. д.).
2. *Инструменты анализа кода*: PyCharm включает в себя инструменты анализа кода, которые помогают выявить ошибки во время написания. Такие инструменты как Linting, проверка кода на PEP-стандарты, статический анализ и другие, повышают качество и читабельность кода.
3. *Интеграция с Git и другими системами контроля версий*: PyCharm имеет встроенную поддержку Git, Mercurial, Subversion и других систем контроля версий, что позволяет быстро и удобно работать с кодом, отслеживать изменения и управлять версиями.
4. *Интеграция с виртуальными окружениями*: PyCharm поддерживает создание и управление виртуальными средами Python. Это позволяет изолировать приложение от других и повысить его устойчивость.
5. *Наличие ряда полезных плагинов*: PyCharm предоставляет большое количество плагинов, например, Pytest, Django Support, Docker Support, и многие другие плагины, которые упрощают разработку и автоматизацию работы в IDE.
6. *Облегчение отладки*: PyCharm имеет встроенный отладчик, который помогает предотвращать ошибки, и обеспечивает более быстрый и удобный процесс отладки. Также предоставляется множество иных возможностей для анализа кода, включая поддержку отладчика PDB, условных точек останова, встроенную поддержку юнит-тестирования и многие другие инструменты. Это позволяет разработчикам быстро находить и устранять ошибки.
7. *Поддержка различных операционных систем*: PyCharm работает на Windows, Mac OS и Linux, что позволяет использовать её на различных платформах.
8. *Популярность*: PyCharm обладает широкой пользовательской базой и активным сообществом, что делает его более удобным и популярным для разработчиков различных уровней.
9. *Удобство использования:* PyCharm имеет удобный интерфейс, интуитивно понятные команды и шаблоны кода, что упрощает процесс разработки и снижает количество ошибок.
10. *Поддержка многопоточности*: PyCharm поддерживает разработку многопоточных приложений и обеспечивает удобные инструменты для отслеживания и отладки многопоточных приложений.

Таким образом, PyCharm — это один из лучших выборов для разработки информационного продукта, так как он обладает множеством инструментов, является мощной и удобной средой разработки для языка Python, идеально подходит для командной работы и имеет широкое сообщество разработчиков, которые могут помочь в случае необходимости.

* Графическая библиотека – PyQt (Версия 5.15.7)

PyQt — это библиотека Python, которая предоставляет инструменты для создания графических пользовательских интерфейсов (GUI) на основе фреймворка Qt. К значительным преимуществам PyQt можно отнести:

1. *Кроссплатформенность*: PyQt работает на всех платформах, где есть Python и Qt. Это позволяет создавать приложения для Windows, Mac, Linux, Android и iOS.
2. *Большое количество возможностей:* PyQt позволяет создавать различные типы приложений, от простых форм до сложных мультимедийных интерфейсов. Библиотека предоставляет разнообразные инструменты, такие как дизайнеры интерфейсов и инструменты редактирования CSS, которые упрощают и ускоряют процесс разработки.
3. *Открытое программное обеспечение*: PyQt имеет открытый исходный код, который может быть доступен бесплатно каждому.
4. *Широкий выбор виджетов:* PyQt предоставляет большое количество готовых виджетов, которые можно использовать в своих приложениях: кнопки, поля ввода, графические элементы, таблицы и многое другое.
5. *Простота использования*: PyQt имеет простой и понятный API, благодаря которому создание приложений становится проще и быстрее.
6. *Актуальные возможности:* PyQt постоянно обновляется и улучшается, что позволяет использовать новые возможности Qt и Python.
7. *Большое сообщество разработчиков*: PyQt имеет большое сообщество разработчиков, которые поддерживают его и разрабатывают новые библиотеки и расширения, а также имеется значительное количество форумов в интернете для обсуждения возникших проблем.
8. *Обширная документация*: PyQt имеет обширную и понятную документацию, содержащую примеры кода и руководства для различных задач.
9. *Легкая интеграция с другими инструментами:* PyQt может легко интегрироваться с другими библиотеками и инструментами Python, что делает его еще более мощным и удобным в использовании.

PyQT является Python-оберткой QT, поэтому обладает большинством его функций. За счет своих возможностей и преимуществ PyQt позволяет быстро создавать сложные приложения с элегантным пользовательским интерфейсом, обладающие хорошей производительностью и широкими возможностями настройки.

* Математический пакет - Sympy (Версия 1.11.1)

Sympy — это библиотека для символьных математических вычислений на языке Python. Она предоставляет широкий функционал для математических операций, таких как алгебраические упрощения, дифференцирование, интегрирование, решение уравнений, построение графиков, численное интегрирование, и многое другое.

Преимущества Sympy:

1. *Символьные вычисления*: Sympy может выполнять все операции обычной математики, но используя символьные переменные вместо чисел. Это позволяет удобно работать с выражениями, которые нельзя вычислить в явном виде, или если значения переменных неизвестны.
2. *Широкий набор функций*: Sympy содержит более 100 функций для работы с символьными выражениями, включая алгебру, геометрию, теорию чисел и многое другое.
3. *Легко использовать*: Sympy поставляется с документацией и множеством примеров использования, которые помогут быстро освоить библиотеку и начать ей пользоваться.
4. *Можно интегрировать с другими библиотеками*: Sympy легко интегрируется с другими библиотеками для научных вычислений, например, NumPy и SciPy, что позволяет использовать их функции в выражениях Sympy.
5. *Open source*: Sympy является бесплатной и открытой для использования библиотекой, что делает ее доступной для всех.
6. *Кроссплатформенность*: SymPy работает на различных операционных системах, таких как Windows, Linux и macOS.

SymPy – это мощная библиотека для символьных вычислений в Python, которая может быть использована как профессионалами, так и новичками. Она имеет множество преимуществ, которые делают ее предпочтительным выбором для математических вычислений в Python.

# Инструкция к запуску программы

Шаг 1: установите Python на свой компьютер

Перед тем, как можно будет запустить программу, нам нужно убедиться, что интерпретатор Python установлен и готов к работе на вашем компьютере. Можно скачать последнюю версию Python с официального сайта <https://www.python.org/downloads/>

Шаг 2: скачайте программу с Github / Склонируйте репозиторий

* Для скачивания программы с Github нужно открыть URL https://github.com/VladislavChepusov/supply\_chains в веб-браузере. Далее необходимо выбрать зеленую кнопку "Code" и выбрать "Download ZIP", чтобы скачать папку с программой в ZIP-архиве на свой компьютер.
* Для клонирования сначала запустите командную строку и перейдите в каталог, в который вы хотите склонировать репозиторий. Затем выполните следующую команду:

*git clone* [*https://github.com/VladislavChepusov/supply\_chains*](https://github.com/VladislavChepusov/supply_chains)

Шаг 3: перейдите в папку с программой

После этого, нужно открыть терминал или командную строку (в зависимости от операционной системы) и перейти в папку с программой с помощью команды "cd" (например, "cd Desktop/supply\_chains-master").

Шаг 4: установите все зависимости

Перейдите в каталог репозитория, установите все необходимые зависимости, выполнив следующие команды:

*cd supply\_chains*

*pip install -r requirements.txt*

Шаг 5: запустите программу

После установки всех зависимостей, мы можем запустить программу. В папке с программой нужно открыть терминал или командную строку и запустить файл "main.py" с помощью Python с помощью команды:

*python main.py*

Программа должна начать работать.

# Пример работы ПО со скриншотами интерфейса

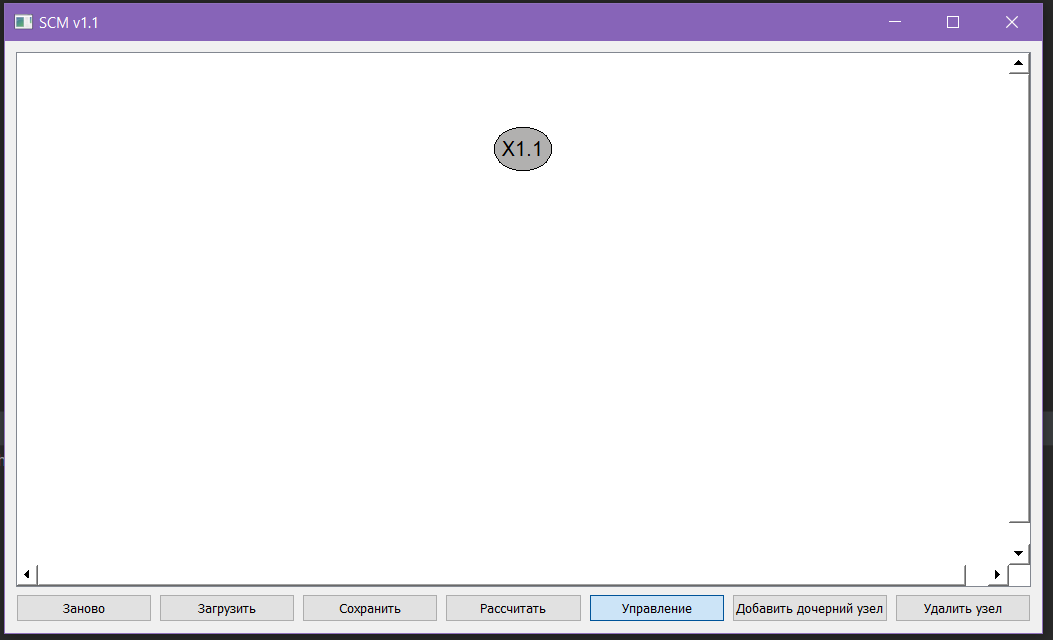


Рисунок 7 – Начальное состояние системы

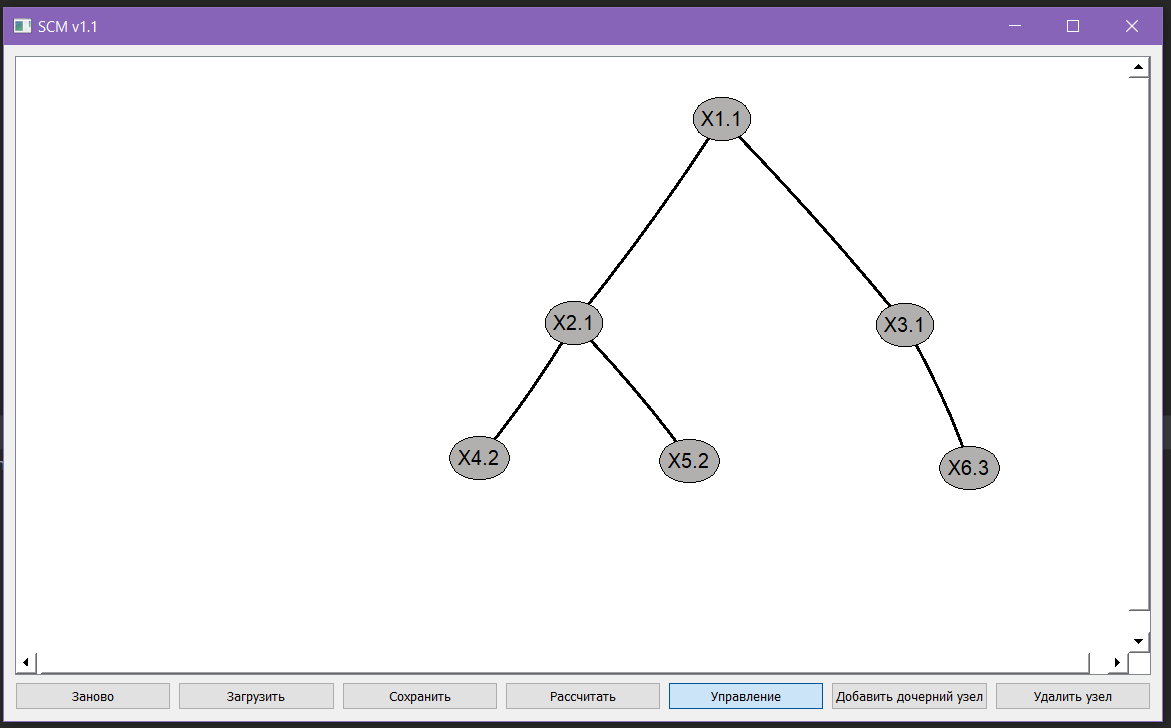


Рисунок 8 – Добавили несколько узлов в цепь

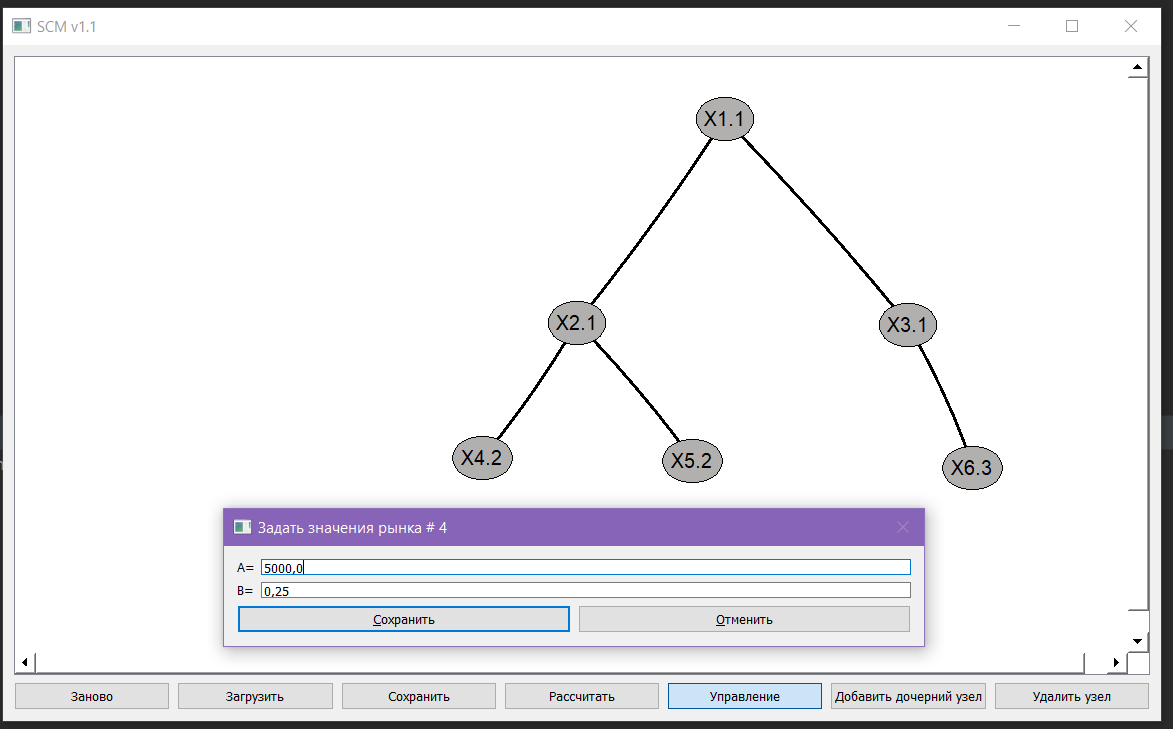


Рисунок 9 – Задаем значения коэффициентов (а,в) линейный функций цены для терминального рынка №4

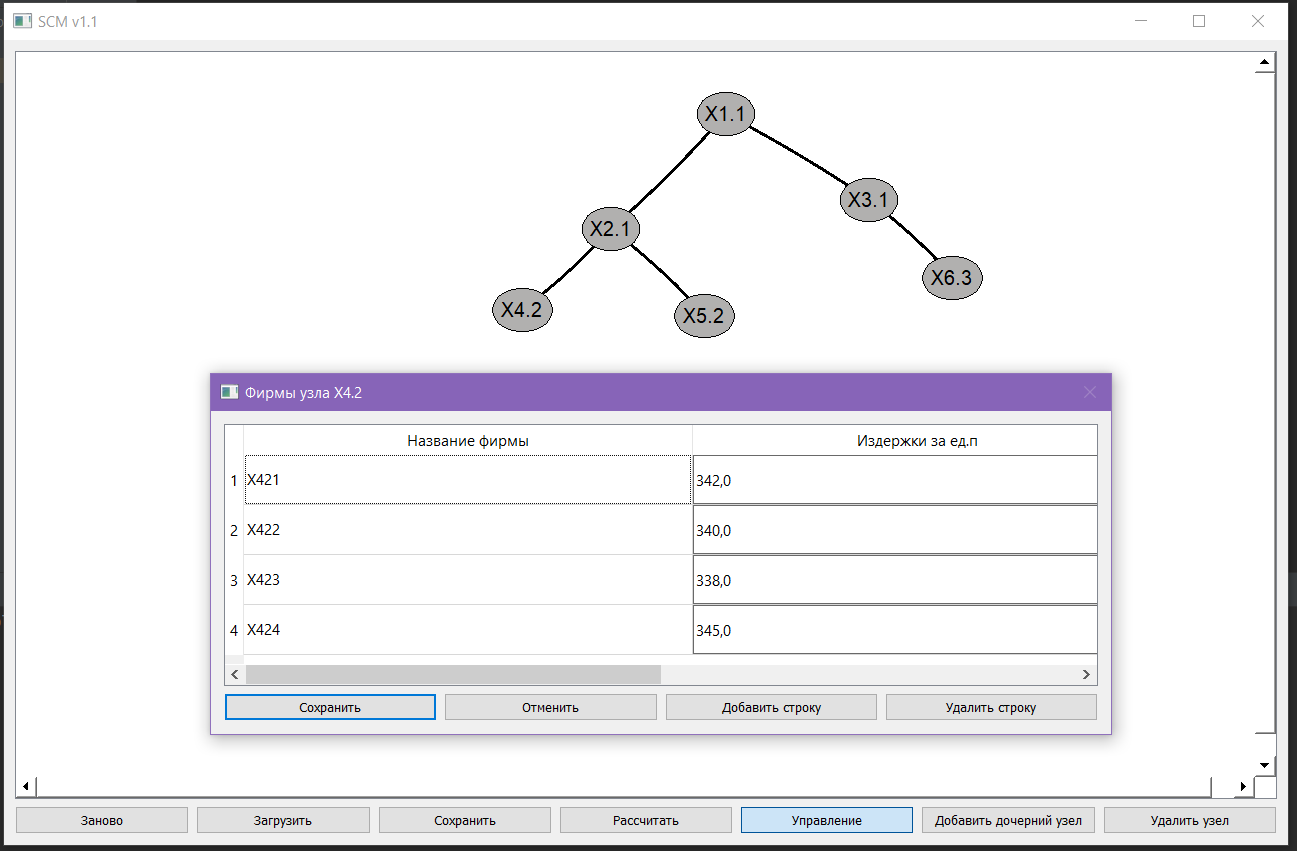


Рисунок 10 – Заполняем данные об узле 4(сколько содержится фирм в узле, их название и издержки каждой компании)

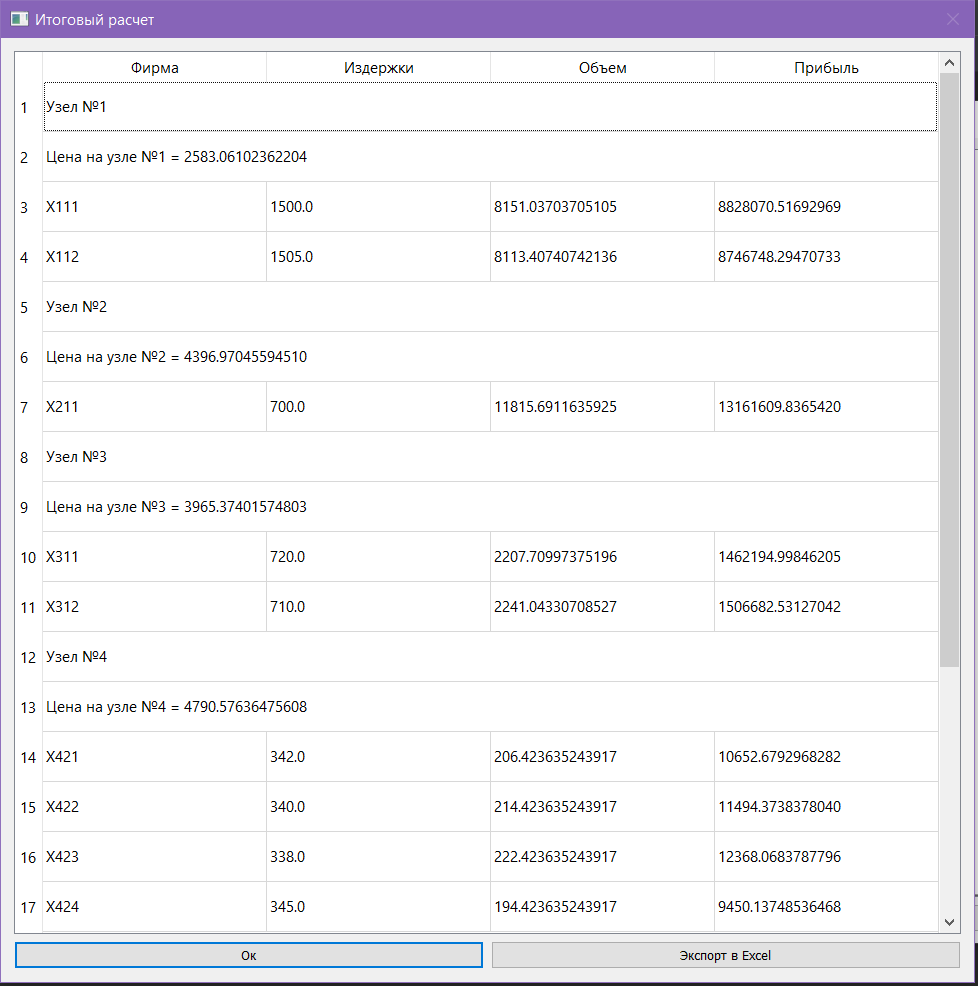


Рисунок 11 – Получения итоговых значений после заполнения всей доступной информации о цепи

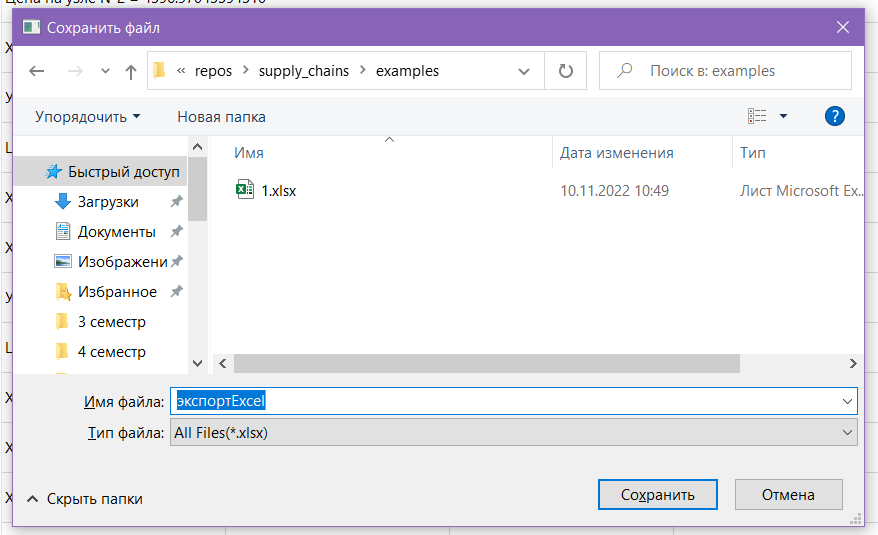


Рисунок 12 – Сохранение полученных результатов в формате .xlsx

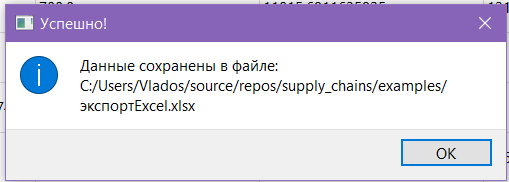


Рисунок 13 – Сообщение об успехе + полное имя файла

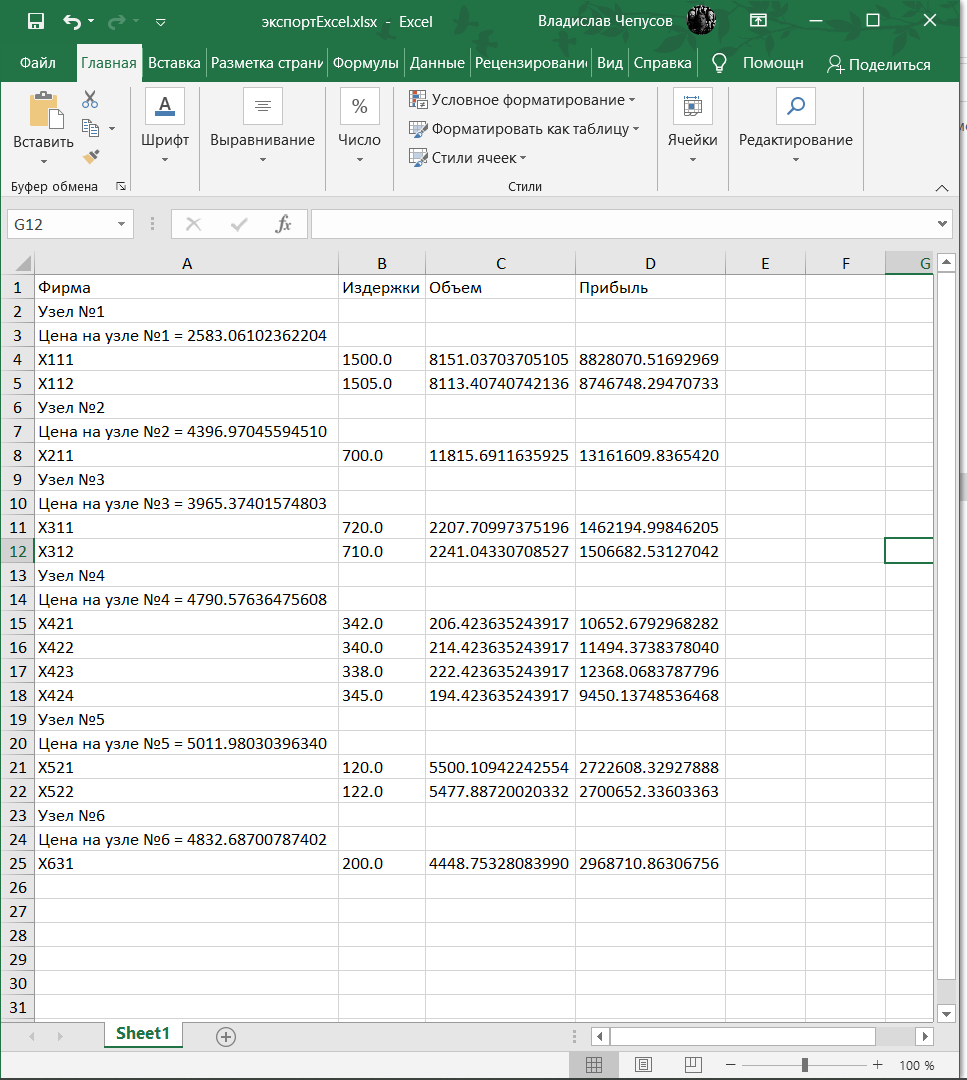


Рисунок 14 – Экспорт данных в Excel. Отображение данных

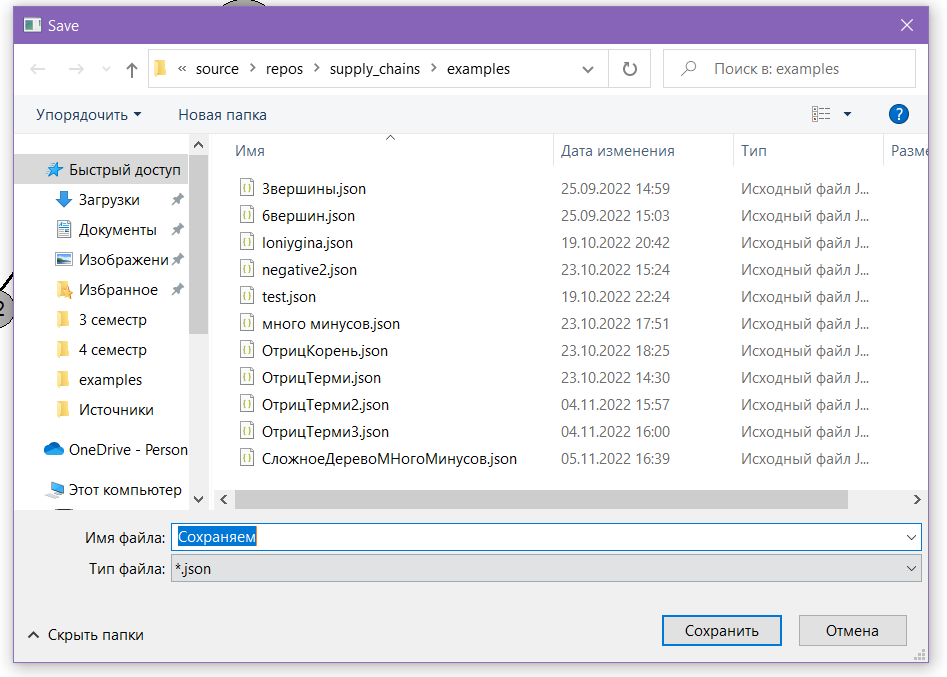


Рисунок 15 – Сохранение структуры цепи поставок и информации о фирмах, содержащихся в ней в формат json

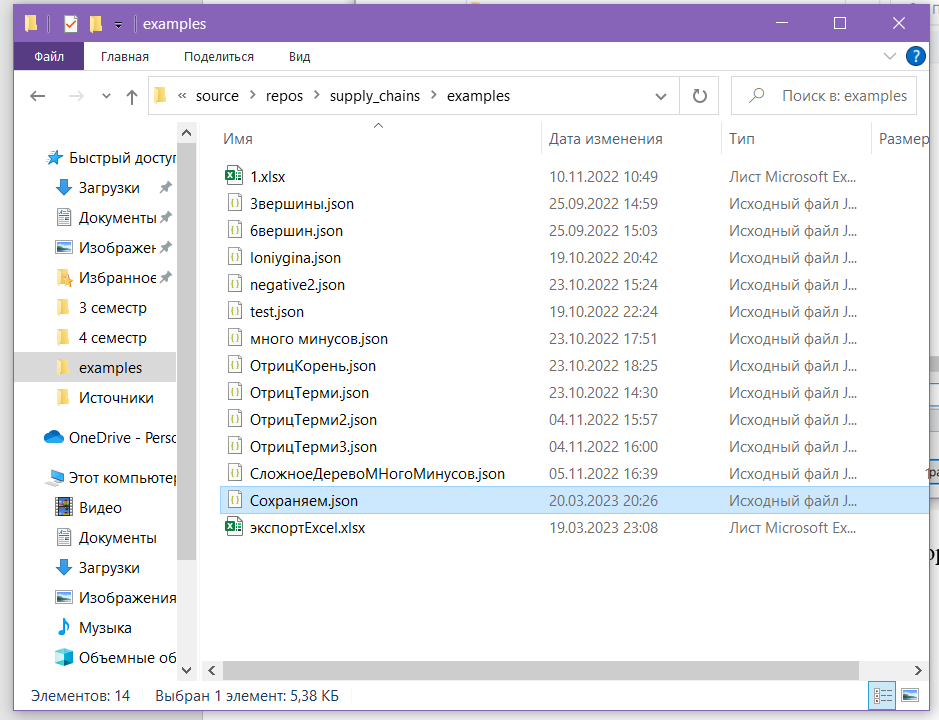


Рисунок 16 – Скриншот директории куда сохранили структуру с демонстрацией самого файла

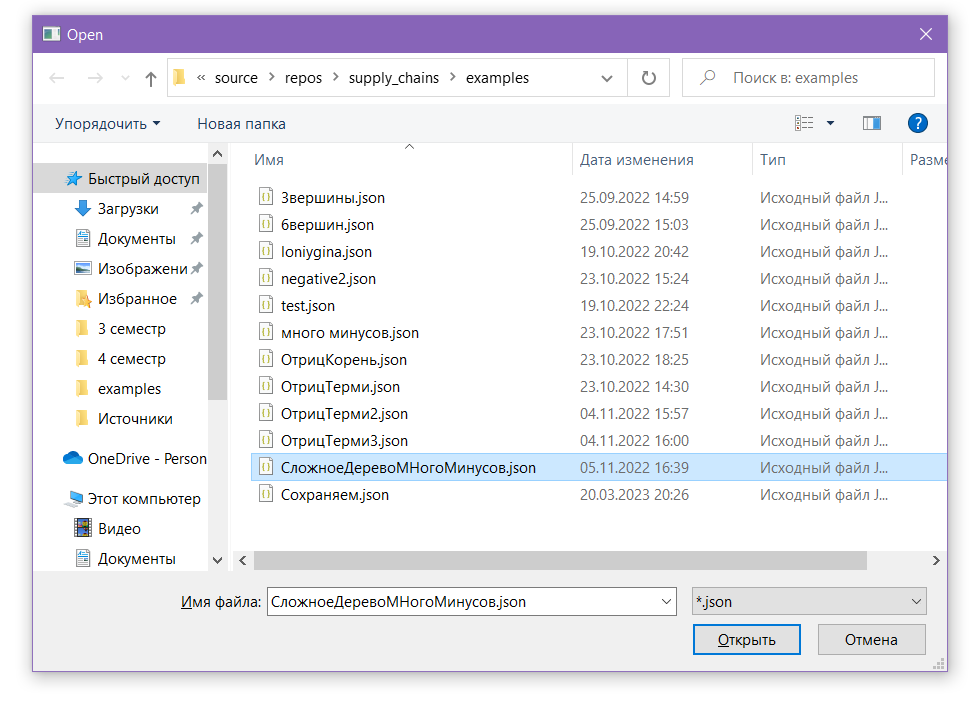


Рисунок 17– загрузка структуры цепи поставок и информации о фирмах, содержащихся в ней из формата json

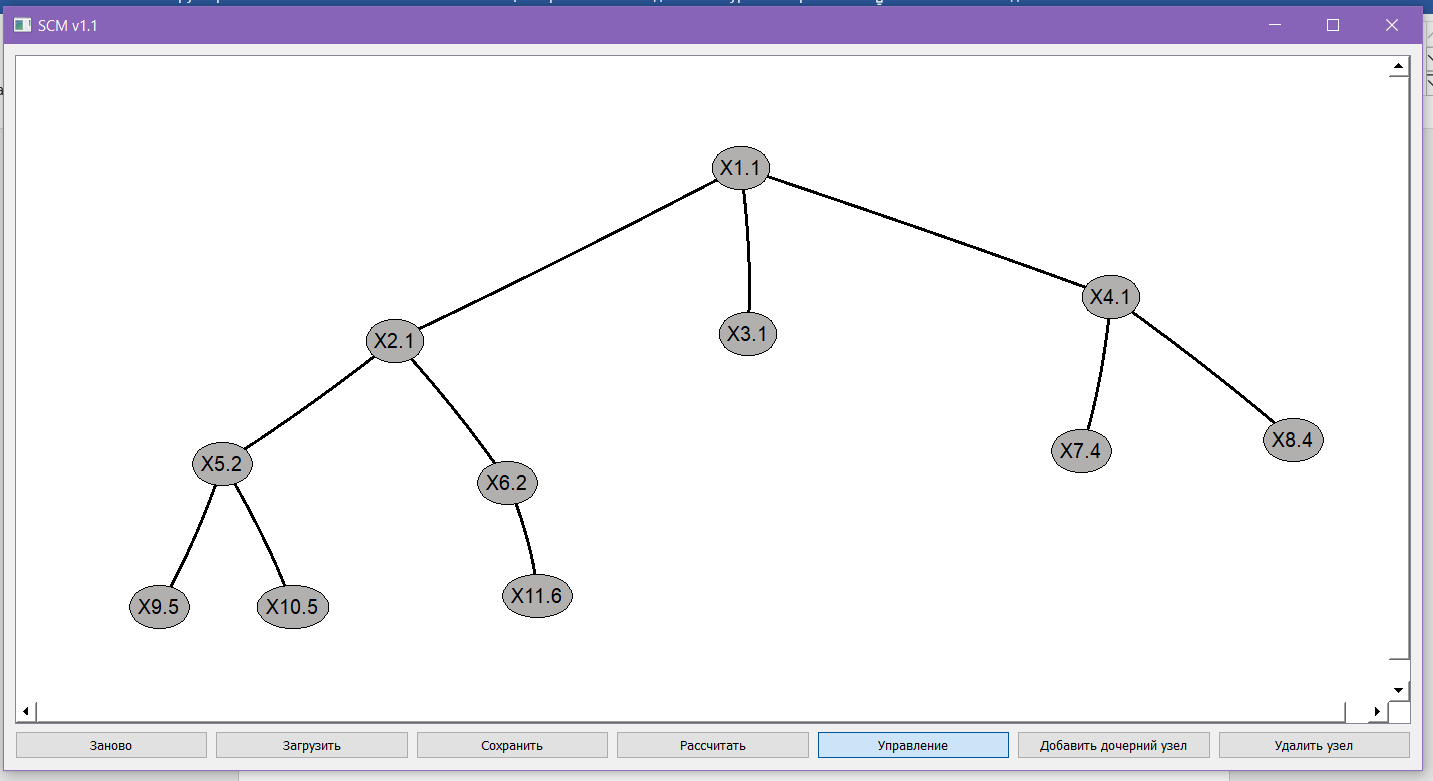


Рисунок 18 – визуальный результат загрузки данных

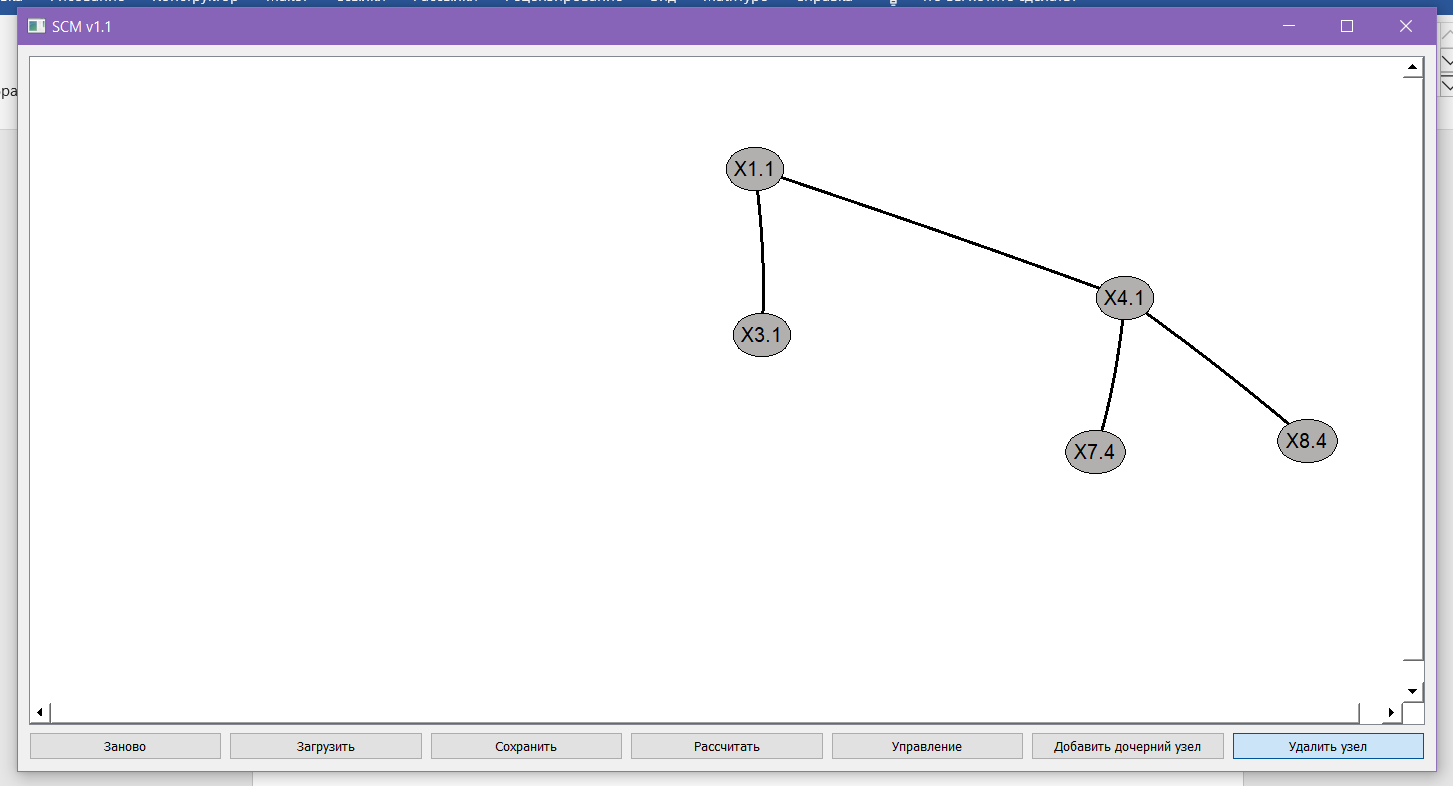


Рисунок 19 – функция удаление узла на примере Х2.1

# Выводы к разделу

В настоящем разделе был проведен комплексный анализ разрабатываемой системы. Были приведены функциональные, нефункциональные, а также требования к организации входных и выходных данных.

Разработаны варианты использования системы, показывающие различные сценарии, в которых она может быть применена, и демонстрирующие ее практическую ценность для пользователей.

Был осуществлен и обоснован выбор инструментов разработки, составлена инструкция по запуску программы и продемонстрированы результаты работы системы посредством скриншотов.

# ЗАКЛЮЧЕНИЕ

В данном исследовании была создана математическая модель для древовидной цепи поставок, а также был разработан способ поиска оптимального решения относительно прибыли участников, конкурирующих по экономической модели Курно и реализовано программное обеспечение по данному алгоритму.

Программный инструментарий прошел проверку на основе тестовых данных.

Подводя итоги, можно сказать, что в результате успешного выполнения всех поставленных в данной работе задач, цель дипломного проектирования была достигнута.

Выделим несколько потенциальных направлений, которые могут быть исследованы в дальнейшем:

1. Расширение цепи поставок благодаря добавлению нового типа узлов-поставщиков.
2. Использование централизованного подхода для улучшения эффективности цепи поставок.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Дегтерев Д.А. Зарубежные работы по теории игр / Д.А. Дегтерев // Международные процессы. - 2009. - № 2 (20), май-август.
2. Петросян Л.А., Зенкевич Н.А., Шевкопляс Е.В., 2014. Теория игр / 2-ое издание, Санкт-Петербург
3. Н. А. Зенкевич, Л. А. Петросян, Д. В. К. Янг. ДИНАМИЧЕСКИЕ ИГРЫ И ИХ ПРИЛОЖЕНИЯ В МЕНЕДЖМЕНТЕ /Издательство: «Высшая школа менеджмента», 2009 – 417 С.
4. Крылатков П. П., Прилуцкая М. А., 2018. Управление цепью поставок (SCM) / 2-ое издание, Екатеринбург
5. Christopher, M. Logistics and Supply Chain Management: Strategies for Reducing Cost and Improving Service / M. Christopher –London, 2011.
6. Deming Zhou Uday S. Karmarkar. Competition in MultiEchelon Distributive Supply Chains with Linear Demand // International journal of production research, London,2015.31 c
7. Парфёнов Михаил Александрович Эволюция управления цепями поставок: от интегрированной логистики к сетевым структурам // Вестник РГЭУ РИНХ. 2010. №32.
8. Щербаков В. В., Логистика и управление цепями поставок : учебник для академического бакалавриата / В. В. Щербаков [и др.] ; под редакцией В. В. Щербакова. — Москва / Издательство: Юрайт, 2019. — 582 с.
9. Tyagi R.K. On the effect of downstream entry // Management science, 1999. № 45. P. 59-73
10. Ziss S. Vertical separation and horizontal mergers // Journal of industrial economics, 1995. №43. P. 63-75
11. Cachon G.P. Supply chain coordination with contracts // Handbooks in Operations Research & Management Science, 2003. Vol. 11. P. 227-339
12. Kaya M., Ozer O. Pricing in business-to-business contracts: sharing risk, profit and information // The Oxford Handbook of Pricing Management. Oxford: Oxford University Press, 2012. P. 738-783.
13. Adida E., DeMiguel V. Supply Chain competition with multiple manufacturers and retailers // Operation Research, 2011. Vol. 59, №1. P.156- 172.
14. Lee H. Decentralized multi-echelon supply chains: incentives and information / H. Lee, S. Whang // Management Science – 1999. – Vol. 54, №5. – P.633-640.
15. Corbett, C.J., & Karmarkar, U.S. Competition and Structure in Serial Supply Chains with Deterministic Demand / C.J. Corbett, U.S. Karmarkar // Manag. Sci. - 2001. - Т. 47. - С. 966-978.
16. Carr, S.M. & Karmarkar, U.S. Competition in Multiechelon Assembly Supply Chains / S.M. Carr, U.S. Karmarkar // Manag. Sci. - 2005. - Т. 51. - С. 45-59.
17. Ю.Е. Лонягина, Н. А. Зенкевич. Модели конкуренции и кооперации многоуровневых сетей поставок / Издательство: «Санкт-Петербургский Государственный Университет»,2018 – 90 с.
18. Захаров, А. В. Теория игр в общественных науках: учебник для вузов / А. В. Захаров; Нац. исслед. ун-т «Высшая школа экономики». — М.: Изд. дом Высшей школы экономики, 2015. — (Учебники Высшей школы экономики). — 304 с. — 1500 экз.
19. Mas-Collel A., Whinston M.D., Green J.R. Microeconomic theory. — N.-Y.: Oxford Univ. Press, 1995. — 981 p.
20. The Theory of Industrial Organization / Jean Tirole. – Cambridge, Massachusetts: The MIT Press, 1988. – 496 с. – ISBN 978-0-262-20071-4.
21. Васин А. А., Морозов В. В. Теория игр и модели математической экономики. — М.: МГУ, 2005, 272 с
22. Лонягина Ю., Никольченко Н., Зенкевич Н. Конкурентное и кооперативное поведение в распределительных сетях // Вклад в теорию игр и менеджмент. 2018. Т. 11. С. 73–102.
23. Дюсуше О. М. Статичное равновесие Курно–Нэша и рефлексивные игры олигополии: случай линейных функций спроса и издержек // Экономический журнал ВШЭ. 2006. №1.
24. Компания GMDH [Электронный ресурс] – Нью-Йорк, NY 10006 Соединенные Штаты. Автор: Кошулько А. – URL: <https://gmdhsoftware.com/ru/scm-software/>? (дата обращения: 20.03.2023).
25. Вигерс Карл, Битти Джой Разработка требований к программному обеспечению. 3-е изд., дополненное/ Пер. с англ. – М.: Издательство «Русская редакция»; СПБ.: БХВ-Петербург, 2014. -736стр.: ил.
26. 830-1998 - IEEE Recommended Practice for Software Requirements Specifications. - NY: The Institute of Electrical and Electronics Engineers, 2009. - 31 c
27. Writing Software Requirements Specifications (SRS) [Электронный ресурс]. — INKtopia Limited, 2020 — URL: <https://techwhirl.com/writing-software-requirements-specifications/> (дата обращения: 20.03.2023)
28. SimbirSoft. Нефункциональные требования: как не пустить систему ко дну [Электронный ресурс]. Блог компании SimbirSoft. Автор: Елена Иванова, ведущий аналитик ИТ-компании SimbirSoft. URL: <https://habr.com/ru/company/simbirsoft/blog/688428/> (дата обращения: 20.03.2023).
29. JetBrains. PyCharm Documentation [Электронный ресурс]: Quick Start Guide. URL: <https://www.jetbrains.com/help/pycharm/quick-start-guide.html> (дата обращения: 21.03.2023).
30. Мовсесян А.А., Стародубцева О.А. ОТЛИЧИТЕЛЬНЫЕ ОСОБЕННОСТИ МАРКЕТИНГА НА РЫНКАХ B2B И B2C // Инфраструктурные отрасли экономики: проблемы и перспективы развития. 2016.
31. Воронцова, Е. А., Хильдебранд, Р. Ф., Гасников, А. В., & Стонякин, Ф. С. Выпуклая оптимизация: учебное пособие / Е. А. Воронцова, Р. Ф. Хильдебранд, А. В. Гасников, Ф. С. Стонякин. - Москва: МФТИ, 2021. - 364 с. - ISBN 978-5-7417-0776-0.

# ПРИЛОЖЕНИЕ А

Листинг + Ссылка на Github

URL: <https://github.com/VladislavChepusov/supply_chains>

**Файл main.py**

import sys  
from IPython.external.qt\_for\_kernel import QtCore, QtGui  
from PyQt5.QtGui import QFont, QDoubleValidator  
from PyQt5.QtWidgets import QFileDialog, QDialog, QApplication, QWidget, QMainWindow, QVBoxLayout, QHBoxLayout, \  
 QFormLayout, QPushButton, QLineEdit, QLabel, QTableWidgetItem  
from QGraphViz.DotParser import Graph  
from QGraphViz.Engines import Dot  
from QGraphViz.QGraphViz import QGraphViz, QGraphVizManipulationMode  
from calculations import all\_calculation\_fun, type\_node, daughters\_map  
from tableWidget import TableWidget  
  
Parent\_node = ''  
Number\_node = 1  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 # Создание приложения QT  
 app = QApplication(sys.argv)  
  
 # Выделение узла  
 def node\_selected(node):  
 if qgv.manipulation\_mode == QGraphVizManipulationMode.Node\_remove\_Mode:  
 print(f"Удалили {node.name}")  
 else:  
 global Parent\_node  
 Parent\_node = node  
  
 # Событие двойного нажатия на вершину  
 def node\_invoked(node):  
 global dlg  
 print(f"Двойное нажатие на вершины {node.name}")  
 validator = QDoubleValidator(0.99, 99.99, 4)  
 if type\_node(node.name, daughters\_map(qgv.engine.graph.toDICT()["edges"])) == 3:  
 dlg = QDialog()  
 dlg.ok = False  
 dlg.setWindowTitle(f'Задать значения рынка # {node.name}')  
  
 dlg.A = ""  
 dlg.B = ""  
  
 main\_layout = QVBoxLayout()  
 l = QFormLayout()  
 buttons\_layout = QHBoxLayout()  
 main\_layout.addLayout(l)  
 main\_layout.addLayout(buttons\_layout)  
 dlg.setLayout(main\_layout)  
 leA = QLineEdit()  
 leB = QLineEdit()  
  
 l.setWidget(0, QFormLayout.LabelRole, QLabel("A="))  
 l.setWidget(0, QFormLayout.FieldRole, leA)  
 l.setWidget(1, QFormLayout.LabelRole, QLabel("B="))  
 l.setWidget(1, QFormLayout.FieldRole, leB)  
 # Установить валидатор ток для чисел с плавающей запятой  
 # до 4-х символов после запятой  
 leA.setValidator(validator)  
 leB.setValidator(validator)  
 # Загрузка старых цен  
 try:  
 leA.setText(str(node.kwargs['level\_price'][0]).replace('.', ','))  
 leB.setText(str(node.kwargs['level\_price'][1]).replace('.', ','))  
 except:  
 pass  
 buttOK = QPushButton()  
 buttCancel = QPushButton()  
 buttOK.setText("&Сохранить")  
 buttCancel.setText("&Отменить")  
  
 def cancel():  
 if dlg.B != '' and dlg.A != '':  
 dlg.ok = True  
 else:  
 dlg.ok = False  
 dlg.close()  
  
 def ok():  
 if leA.text() != "" and leB.text() != "":  
 dlg.ok = True  
 dlg.A = leA.text()  
 dlg.B = leB.text()  
 dlg.close()  
  
 buttOK.clicked.connect(ok)  
 buttCancel.clicked.connect(cancel)  
 buttons\_layout.addWidget(buttOK)  
 buttons\_layout.addWidget(buttCancel)  
  
 dlg.setFixedSize(700, 100)  
 dlg.setWindowFlags(QtCore.Qt.WindowCloseButtonHint)  
 dlg.setWindowFlag(QtCore.Qt.WindowCloseButtonHint, False)  
 dlg.exec\_()  
 if dlg.ok and dlg.A != '' and dlg.B != '':  
 A = float(dlg.A.replace(',', '.'))  
 B = float(dlg.B.replace(',', '.'))  
 node.kwargs['level\_price'] = [A, B]  
  
 if type\_node(node.name, daughters\_map(qgv.engine.graph.toDICT()["edges"])) != 3 or dlg.ok:  
 dlg = QDialog()  
 dlg.ok = False  
 dlg.setWindowTitle(f'Фирмы узла {node.kwargs["label"]}')  
 main\_layout = QVBoxLayout()  
 l = QFormLayout()  
 buttons\_layout = QHBoxLayout()  
 table = TableWidget(1, 2, ["Название фирмы", "Издержки за ед.п"])  
 main\_layout.addWidget(table)  
 main\_layout.addLayout(l)  
 main\_layout.addLayout(buttons\_layout)  
 dlg.setLayout(main\_layout)  
 dlg.resize(900, 900)  
 firm\_list = []  
  
 def cancel():  
 dlg.ok = False  
 dlg.close()  
  
 def ok():  
 if table.isEmptys():  
 dlg.ok = True  
 rowCount = table.rowCount()  
 for i in range(rowCount):  
 firm\_list.append({'name\_firm': table.item(i, 0).text(),  
 'cost\_firm': float(table.cellWidget(i, 1).text().replace(',', '.'))}  
 )  
 dlg.close()  
  
 # Загрузка старых данных  
 if node.kwargs['firms']:  
 for i in range(len(node.kwargs['firms'])):  
 table.setItem(i, 0, QTableWidgetItem(node.kwargs['firms'][i]['name\_firm']))  
 table.cellWidget(i, 1).setText(str(node.kwargs['firms'][i]['cost\_firm']).replace('.', ','))  
 table.\_addRow()  
 table.\_removeRow()  
  
 pbOK = QPushButton()  
 pbCancel = QPushButton()  
 pbAdd = QPushButton()  
 pbDelete = QPushButton()  
 pbOK.setText("&Сохранить")  
 pbCancel.setText("&Отменить")  
 pbAdd.setText("&Добавить строку")  
 pbDelete.setText("&Удалить строку")  
 pbOK.clicked.connect(ok)  
 pbCancel.clicked.connect(cancel)  
 pbAdd.clicked.connect(table.\_addRow)  
 pbDelete.clicked.connect(table.\_removeRow)  
 buttons\_layout.addWidget(pbOK)  
 buttons\_layout.addWidget(pbCancel)  
 buttons\_layout.addWidget(pbAdd)  
 buttons\_layout.addWidget(pbDelete)  
 dlg.setWindowFlags(QtCore.Qt.WindowCloseButtonHint)  
 dlg.setWindowFlag(QtCore.Qt.WindowCloseButtonHint, False)  
 dlg.exec\_()  
  
 if dlg.ok and firm\_list:  
 node.kwargs['firms'] = firm\_list  
  
 # Удаление вершин от листьев к родителям (снизу вверх)  
 def node\_removed(node):  
 map\_ = daughters\_map(qgv.engine.graph.toDICT()["edges"])  
 if node.name in map\_: # node in dict  
 list\_ = deletionList(map\_, node.name, [])  
 for \_ in reversed(list\_):  
 qgv.removeNode(qgv.engine.graph.findNode(\_))  
 qgv.build()  
  
 # Список последовательного удаления  
 def deletionList(dict\_, parent\_name, list\_=None):  
 if list\_ is None:  
 list\_ = []  
 if parent\_name in dict\_:  
 for node in dict\_[parent\_name]:  
 list\_.append(node)  
 if node in dict\_:  
 deletionList(dict\_, node, list\_)  
 else:  
 continue  
 return list\_  
  
  
 # QGraphViz виджет  
 ShowSubGraphs = False  
 qgv = QGraphViz(  
 show\_subgraphs=ShowSubGraphs,  
 auto\_freeze=True,  
 node\_selected\_callback=node\_selected,  
 node\_invoked\_callback=node\_invoked,  
 node\_removed\_callback=node\_removed,  
 hilight\_Nodes=True,  
 hilight\_Edges=False  
 )  
  
 qgv.setStyleSheet("background-color:white;")  
 # Создайте новый график, используя механизм компоновки Dot  
 qgv.new(Dot(Graph("Main\_Graph"), show\_subgraphs=ShowSubGraphs, font=QFont("Arial", 12), margins=[20, 20]))  
 # Добавляем первый узел  
 qgv.addNode(qgv.engine.graph, 1, label=f"X1.1", firms=[], level\_price=[], fillcolor="#b1b0af")  
 # Постройте граф (механизм компоновки организует расположение узлов и соединений)  
 qgv.build()  
 # Задание формочки  
 w = QMainWindow()  
 w.setWindowTitle('SCM v1.1')  
 # Create a central widget to handle the QGraphViz object  
 wi = QWidget()  
 wi.setLayout(QVBoxLayout())  
 w.setCentralWidget(wi)  
 # Add the QGraphViz object to the layout  
 wi.layout().addWidget(qgv)  
 # добавить горизонтальный макет (панель для выбора того, что делать)  
 hpanel = QHBoxLayout()  
 wi.layout().addLayout(hpanel)  
  
 # Манипуляция рисунком  
 def manipulate():  
 off\_button()  
 btnManip.setCheckable(True)  
 btnManip.setChecked(True)  
 qgv.manipulation\_mode = QGraphVizManipulationMode.Nodes\_Move\_Mode  
  
 # Сохранение данных в json  
 def save():  
 fname = QFileDialog.getSaveFileName(qgv, "Save", "", "\*.json")  
 if fname[0] != "":  
 qgv.saveAsJson(fname[0])  
  
 # получить структуру данных  
 def CalculationsOfIndicators():  
 graph\_dic = qgv.engine.graph.toDICT()  
 calculation = all\_calculation\_fun(graph\_dic)  
  
 # for key, value in calculation.items():  
 # print("{0}: {1}".format(key, value))  
 dlg = QDialog()  
 dlg.ok = False  
 dlg.setWindowTitle(f'Итоговый расчет')  
 main\_layout = QVBoxLayout()  
 l = QFormLayout()  
 buttons\_layout = QHBoxLayout()  
 table = TableWidget(0, 4, ["Фирма", "Издержки", "Объем", "Прибыль"])  
  
 def add\_visual(items, gluing):  
 rowPosition = table.rowCount()  
 table.insertRow(rowPosition)  
 [item.setFlags(QtCore.Qt.ItemIsEnabled) for item in items]  
 if gluing:  
 table.setSpan(rowPosition, 0, 1, 4)  
 table.setItem(rowPosition, 0, items[0])  
 else:  
 for i in range(len(items)):  
 table.setItem(rowPosition, i, items[i])  
  
 for i in calculation:  
 level = [QtGui.QTableWidgetItem(f"Узел №{i}")]  
 add\_visual(level, True)  
 price1 = [QtGui.QTableWidgetItem(f"Цена на узле №{i} = {calculation[i]['price']}")]  
 add\_visual(price1, True)  
  
 le = len(calculation[i]['cost'])  
 for j in range(le):  
 data = [QtGui.QTableWidgetItem(f"{calculation[i]['name\_firm'][j]} "),  
 QtGui.QTableWidgetItem(f"{calculation[i]['cost'][j]}"),  
 QtGui.QTableWidgetItem(f"{calculation[i]['value'][j]}"),  
 QtGui.QTableWidgetItem(f"{calculation[i]['profit'][j]}")]  
 add\_visual(data, False)  
  
 main\_layout.addWidget(table)  
 main\_layout.addLayout(l)  
 main\_layout.addLayout(buttons\_layout)  
 dlg.setLayout(main\_layout)  
 dlg.resize(900, 900)  
 dlg.setWindowFlags(QtCore.Qt.WindowCloseButtonHint)  
 dlg.setWindowFlag(QtCore.Qt.WindowCloseButtonHint, False)  
  
 pbCancel = QPushButton()  
 pbCancel.setText("&Ок")  
 pbCancel.clicked.connect(dlg.close)  
 buttons\_layout.addWidget(pbCancel)  
  
 pbExcel = QPushButton()  
 pbExcel.setText("&Экспорт в Excel")  
 pbExcel.clicked.connect(table.tableSave)  
 buttons\_layout.addWidget(pbExcel)  
 dlg.exec\_()  
  
 # Стирание всей цепи  
 def new():  
 off\_button()  
 btnManip.setCheckable(True)  
 btnManip.setChecked(True)  
 qgv.engine.graph = Graph("MainGraph")  
 global Number\_node  
 Number\_node = 1  
 qgv.addNode(qgv.engine.graph, 1, label=f"X1.1", firms=[], level\_price=[], fillcolor="grey")  
 qgv.build()  
 qgv.repaint()  
  
 # Загрузка данных из json  
 def load():  
 off\_button()  
 btnManip.setCheckable(True)  
 btnManip.setChecked(True)  
 fname = QFileDialog.getOpenFileName(qgv, "Open", "", "\*.json")  
 if fname[0] != "":  
 qgv.loadAJson(fname[0])  
 graph\_js = qgv.engine.graph.toDICT()  
 global Number\_node  
 Number\_node = len(graph\_js['nodes'])  
  
 # Удаление вершины(включить режим)  
 def rem\_node():  
 off\_button()  
 qgv.manipulation\_mode = QGraphVizManipulationMode.Node\_remove\_Mode  
 btnRemNode.setCheckable(True)  
 btnRemNode.setChecked(True)  
  
 # Добавление дочернего узла  
 def add\_node\_child():  
 off\_button()  
 btnAddChild.setCheckable(False)  
 if Parent\_node != '':  
 global Number\_node  
 Number\_node += 1  
 new\_node = qgv.addNode(qgv.engine.graph, Number\_node, label=f"X{Number\_node}.{Parent\_node.name}",  
 firms=[], level\_price=[], fillcolor="#b1b0af")  
 qgv.addEdge(Parent\_node, new\_node, {"width": 3})  
 qgv.build()  
  
  
 def off\_button():  
 qgv.manipulation\_mode = QGraphVizManipulationMode.Nodes\_Move\_Mode  
 for btn in buttons\_list:  
 btn.setChecked(False)  
 btn.setCheckable(False)  
  
  
 # Добавить кнопки  
 btnNew = QPushButton("Заново")  
 btnNew.clicked.connect(new)  
 btnOpen = QPushButton("Загрузить")  
 btnOpen.clicked.connect(load)  
 btnSave = QPushButton("Сохранить")  
 btnSave.clicked.connect(save)  
 btndddd = QPushButton("Рассчитать")  
 btndddd.clicked.connect(CalculationsOfIndicators)  
  
 hpanel.addWidget(btnNew)  
 hpanel.addWidget(btnOpen)  
 hpanel.addWidget(btnSave)  
 hpanel.addWidget(btndddd)  
  
 buttons\_list = []  
  
 btnManip = QPushButton("Управление")  
 btnManip.setCheckable(True)  
 btnManip.setChecked(True)  
 btnManip.clicked.connect(manipulate)  
 hpanel.addWidget(btnManip)  
 buttons\_list.append(btnManip)  
  
 btnAddChild = QPushButton("Добавить дочерний узел")  
 btnAddChild.setCheckable(True)  
 btnAddChild.clicked.connect(add\_node\_child)  
 hpanel.addWidget(btnAddChild)  
 # buttons\_list.append(btnAddChild)  
  
 btnRemNode = QPushButton("Удалить узел")  
 btnRemNode.setCheckable(True)  
 btnRemNode.clicked.connect(rem\_node)  
 hpanel.addWidget(btnRemNode)  
 buttons\_list.append(btnRemNode)  
  
 w.showMaximized()  
 sys.exit(app.exec\_())

Файл TableWidget.py

from PyQt5.QtGui import QDoubleValidator  
from PyQt5.QtWidgets import QHeaderView, QTableWidget, \  
 QTableWidgetItem, QFileDialog, QMessageBox, QLineEdit  
from xlsxwriter import Workbook  
  
  
# Кастомная таблица (произвольные размеры и название).Удаление/Добавление/Сохранение/Проверка на пустоту ячеек  
class TableWidget(QTableWidget):  
 def \_\_init\_\_(self, row, col, name):  
 super().\_\_init\_\_(row, col)  
 self.setHorizontalHeaderLabels(name)  
 self.verticalHeader().setDefaultSectionSize(50)  
 self.horizontalHeader().setDefaultSectionSize(899 / col)  
 self.horizontalHeader().setSectionResizeMode(QHeaderView.Fixed)  
  
 float\_element = QLineEdit()  
 float\_element.setValidator(QDoubleValidator(0.99, 99.99, 4))  
 self.setCellWidget(0, 1, float\_element)  
  
 # Добавить строку таблицы  
 def \_addRow(self):  
 self.insertRow(self.rowCount())  
 float\_element = QLineEdit()  
 float\_element.setValidator(QDoubleValidator(0.99, 99.99, 4))  
 self.setCellWidget(self.rowCount() - 1, 1, float\_element)  
  
 # Удалить строку таблицы  
 def \_removeRow(self):  
 if self.rowCount() > 1:  
 self.removeRow(self.rowCount() - 1)  
  
 def \_copyRow(self):  
 self.insertRow(self.rowCount())  
 rowCount = self.rowCount()  
 columnCount = self.columnCount()  
  
 for j in range(columnCount):  
 if not self.item(rowCount - 2, j) is None:  
 self.setItem(rowCount - 1, j, QTableWidgetItem(self.item(rowCount - 2, j).text()))  
  
 # Проверка ячеек на пустоту  
 def isEmptys(self):  
 rowCount = self.rowCount()  
 for i in range(rowCount):  
 try:  
 if not (self.item(i, 0).text() and self.cellWidget(i, 1).text()):  
 return False  
 except:  
 return False  
 return True  
  
 # Сохранение таблицы в Excel  
 def tableSave(self):  
 fileName, ok = QFileDialog.getSaveFileName(  
 self,  
 "Сохранить файл",  
 ".",  
 "All Files(\*.xlsx)"  
 )  
 if not fileName:  
 return  
  
 model = self.model()  
 \_list = [[self.horizontalHeaderItem(i).text() for i in range(model.columnCount())]]  
 for row in range(model.rowCount()):  
 \_r = []  
 for column in range(model.columnCount()):  
 \_r.append("{}".format(model.index(row, column).data() or ""))  
 \_list.append(\_r)  
 workbook = Workbook(fileName)  
 worksheet = workbook.add\_worksheet()  
 for r, row in enumerate(\_list):  
 for c, col in enumerate(row):  
 worksheet.write(r, c, col)  
 workbook.close()  
 QMessageBox.information(  
 self,  
 "Успешно!",  
 f"Данные сохранены в файле: \n{fileName}"  
 )

Файл calculations.py

import sympy as sym  
import copy  
  
  
# Вернуть ключ по значению  
def get\_key(d, value):  
 for k, v in d.items():  
 for j in v:  
 if j == value:  
 return k  
  
  
# Возвращает список ячеек которые можно удалить тк их терминальные вершины банкроты  
def delete\_market(mapgs, rec):  
 mapping = copy.deepcopy(mapgs)  
 del\_little\_graph = []  
 for x1 in rec:  
 if type\_node(x1, mapping) == 3:  
 papa = get\_key(mapping, x1)  
 mapping[papa].remove(x1)  
 Next = True  
 while Next:  
 for x2 in mapping:  
 if not mapping[x2]:  
 del\_little\_graph.append(x2)  
 for num in mapgs[x2]:  
 del\_little\_graph.append(num)  
 mapping.pop(x2)  
 papa = get\_key(mapping, x2)  
 if papa is not None:  
 mapping[papa].remove(x2)  
 else:  
 Next = False  
 break  
 break  
 for x2 in mapping:  
 if not mapping[x2]:  
 Next = True  
 break  
 else:  
 Next = False  
 return del\_little\_graph  
  
  
# Должен возвращать граф очищенный от фирм у которых отрицательный объем  
# Если корневая вершина отрицательна то вся цепь нулевая  
# если терминальная вершина отрицательна то его родительский граф банкрот  
def CleaningNegativeVolume(graph, calculation):  
 old\_graph = copy.deepcopy(graph)  
 old\_calculation = copy.deepcopy(calculation)  
 del\_mass\_map = []  
 # поиск узлов и фирм с отрицательными объемами  
 for k in old\_calculation:  
 for i in reversed(range(len(old\_calculation[k]["value"]))):  
 if old\_calculation[k]["value"][i] < 0:  
 del\_mass\_map.append([k, i])  
 # Удаление из глобального графа информации о фирмах с отрицательным объемом  
 # Сохранение узлов без фирм  
 reconstruction = []  
 for elem in old\_graph['nodes']:  
 for x\_mark in del\_mass\_map:  
 if elem["name"] == x\_mark[0]:  
 elem["kwargs"]["firms"].pop(x\_mark[1])  
 if len(elem["kwargs"]["firms"]) < 1:  
 reconstruction.append(elem["name"])  
  
 # Если будет ошибка, то наверное тут. Манипуляция именно с узлами ниже  
 reconstruction = list(reversed(reconstruction))  
 # Если корневая вершина вся отрицательна (переписать хардкод)  
 if 1 in reconstruction:  
 old\_graph = {'name': 'MainGraph',  
 'graph\_type': 0,  
 'kwargs': {},  
 'nodes': [  
 {'name': 1, 'kwargs': {'label': 'X1.1', 'firms': [], 'level\_price': [], 'fillcolor': 'grey'}}],  
 'edges': []}  
 return old\_graph  
  
 # Если рынки банкроты, то поддерево выходящее на эти рынки тоже удаляется  
 old\_mapping = daughters\_map(old\_graph["edges"])  
 d = delete\_market(old\_mapping, reconstruction)  
 for x4 in d:  
 # Удаление связей  
 for edg1 in old\_graph["edges"]:  
 if edg1['source'] == x4 or edg1['dest'] == x4:  
 old\_graph["edges"].remove(edg1)  
 for nj1 in old\_graph["nodes"]:  
 if nj1["name"] == x4:  
 old\_graph["nodes"].remove(nj1)  
  
 reconstruction = [x for x in reconstruction if x not in d]  
 # Если нужно изменять структура(удалить одну вершины из рассмотрения)  
 if reconstruction:  
 # удаление пустой вершины и соединение с дочерей с родителями  
 for i in reconstruction:  
 parent = -1  
 child = []  
 for edg in old\_graph["edges"]:  
 # Если наша вершина родитель  
 if edg['source'] == i:  
 child.append(edg['dest'])  
 old\_graph["edges"].remove(edg)  
 elif edg['dest'] == i:  
 parent = edg['source']  
 old\_graph["edges"].remove(edg)  
 # Выстраивание связи  
 if child:  
 for ch in child:  
 old\_graph["edges"].append({'source': parent, 'dest': ch, 'kwargs': {'width': 3}})  
 # Удаление из основного графа вершины  
 for nj in old\_graph["nodes"]:  
 if nj["name"] == i:  
 old\_graph["nodes"].remove(nj)  
  
 return old\_graph  
  
  
# Проверка на наличие отрицательных значений объемов  
def isNegative(deci):  
 for node in deci.values():  
 if sum(1 for x in node["value"] if x < 0) > 0:  
 return True  
 return False  
  
  
# Функция вставляющая новые значения в старую структуру (изначальную)  
def OldButGold(old, new):  
 skeleton = copy.deepcopy(old)  
 for zero in skeleton.values():  
 zero["profit"] = [0] \* len(zero["value"])  
 zero["value"] = [0] \* len(zero["value"])  
 for vice in new:  
 for name in new[vice]["name\_firm"]:  
 skeleton\_index = skeleton[vice]['name\_firm'].index(name)  
 new\_index = new[vice]["name\_firm"].index(name)  
 skeleton[vice]["profit"][skeleton\_index] = new[vice]["profit"][new\_index]  
 skeleton[vice]["value"][skeleton\_index] = new[vice]["value"][new\_index]  
 skeleton[vice]["price"] = new[vice]["price"]  
 return skeleton  
  
  
# Корневая функция  
def all\_calculation\_fun(graph):  
 decision = CalculationForTheChain(graph)  
 if isNegative(decision):  
 pure\_graph = CleaningNegativeVolume(graph, decision)  
 new\_decision = CalculationForTheChain(pure\_graph)  
 result = OldButGold(decision, new\_decision)  
 return result  
 else:  
 return decision  
  
  
def CalculationForTheChain(graph):  
 cart\_maps = daughters\_map(graph["edges"]) # Связи  
 stack = calculation\_queue(cart\_maps) # очередь расчета  
 cart\_nodes = node\_firm(graph["nodes"]) # ТАБЛИЦА ДАННЫЙ  
 for nodes in stack:  
 if nodes != 1:  
 parent = get\_key(cart\_maps, nodes)  
 else:  
 parent = 1  
 if nodes in cart\_maps:  
 cart\_nodes[nodes]['price'] = price\_function(cart\_nodes, nodes, cart\_maps)  
  
 v1 = (volume\_calculation(cart\_nodes, nodes, parent, cart\_maps))  
 cart\_nodes[nodes]['value'] = v1[0].args[0][:-1]  
 cart\_nodes[nodes]['profit'] = v1[1]  
  
 revers\_stack = stack[::-1]  
 for new\_result in revers\_stack:  
 q = sym.symbols(f"q{new\_result}\_1:{len(cart\_nodes[new\_result]['value']) + 1}")  
  
 if new\_result == 1:  
 cart\_nodes[new\_result]['price'] = sum(cart\_nodes[new\_result]['value']) \* -(  
 cart\_nodes[new\_result]['price'][1]) + (cart\_nodes[new\_result]['price'][0])  
 cart\_nodes[new\_result]['profit'] = [x.subs([(u, y) for u, y in zip(q, cart\_nodes[new\_result]['value'])]) for  
 x in cart\_nodes[new\_result]['profit']]  
 else:  
 parent = get\_key(cart\_maps, new\_result)  
 p = sym.symbols(f'p{parent}') # цена родителя  
 cart\_nodes[new\_result]['value'] = [v.subs(p, cart\_nodes[parent]['price']) for v in  
 cart\_nodes[new\_result]['value']]  
  
 cart\_nodes[new\_result]['price'] = sum(cart\_nodes[new\_result]['value']) \* -(  
 cart\_nodes[new\_result]['price'][1]) + (cart\_nodes[new\_result]['price'][0])  
  
 cart\_nodes[new\_result]['profit'] = [  
 x.subs([(u, y) for u, y in zip(q, cart\_nodes[new\_result]['value'])]).subs(p,  
 cart\_nodes[parent]['price'])  
 for  
 x in cart\_nodes[new\_result]['profit']]  
  
 return cart\_nodes  
  
  
# Функция расчета цены  
def price\_function(nf, name\_node, maps):  
 size\_ = len(nf[name\_node]['cost'])  
 child\_node = maps[name\_node]  
 q = sym.symbols(f'q{name\_node}\_1:{size\_ + 1}') # Объемы текущего узла для расчета цены  
 value\_child = [nf[x]['value'] for x in child\_node]  
 Q = sum(sum(i) for i in value\_child)  
 new\_p = sym.solve(sum(q) - Q, f'p{name\_node}')  
 A = new\_p[0].subs([(i, 0) for i in q])  
 B = -((new\_p[0].subs([(i, 1) for i in q]) - A) / size\_) # ТУТ ПОКА ПРОБЛЕМА СО ЗНАКОМ(УЖЕ НЕТ)  
 return [A, B]  
  
  
# Расчет объемов (и прибыли)  
def volume\_calculation(nf, name\_node, parent, cart\_maps):  
 size\_ = len(nf[name\_node]['cost'])  
 q = sym.symbols(f'q{name\_node}\_1:{size\_ + 1}') # объем  
 p = sym.symbols(f'p{parent}') # цена родителя  
 c = nf[name\_node]['cost']  
 e = nf[name\_node]['price'] # функция цены текущего узла  
 switch = {  
 1: [q[i] \* (e[0] - e[1] \* sum(q) - c[i]) for i in range(size\_)], # прибыль корневого уровня  
 2: [q[i] \* (e[0] - e[1] \* sum(q) - p - c[i]) for i in range(size\_)], # прибыль промежуточного уровня  
 3: [q[i] \* ((e[0] - e[1] \* sum(q)) - p - c[i]) for i in range(size\_)] # прибыль терминального уровня  
 }  
 P = switch.get(type\_node(name\_node, cart\_maps))  
 Pi = [sym.diff(P[i], q[i]) for i in range(size\_)]  
 return [sym.linsolve([\*Pi], (\*q, p)), P]  
  
  
# проверить тип узла  
def type\_node(name, cart\_maps):  
 if name == 1:  
 return 1 # корневая  
 elif name in cart\_maps:  
 return 2 # промежуточная  
 else:  
 return 3 # терминальная  
  
  
# Очередь расчета вершин  
def calculation\_queue(map1):  
 stack = []  
 for i in map1:  
 if i not in stack:  
 stack.insert(0, i)  
 for x in map1.get(i):  
 stack.insert(0, x)  
 else:  
 for q in map1.get(i):  
 if q not in stack:  
 stack.insert(0, q)  
 return stack  
  
  
# Множество дочерних узлов  
def daughters\_map(edges):  
 maps = {}  
 for i in edges:  
 if i['source'] in maps:  
 (maps.get(i['source'])).append(i['dest'])  
 else:  
 maps.update({i['source']: [i['dest']]})  
 return maps  
  
  
# вся информация будет храниться тут  
def node\_firm(nodes):  
 data = {}  
 for i in nodes:  
 data.update({i['name']: {  
 'cost': [c['cost\_firm'] for c in i['kwargs']['firms']],  
 'price': i['kwargs']['level\_price'], # Массив из 2-х значений А и Б тк p = a - b\*SUM(value)  
 'profit': [],  
 'value': [],  
 'name\_firm': [c['name\_firm'] for c in i['kwargs']['firms']],  
 }  
 })  
 return data