ISSN 2949-5598

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

Том 31 № 4 2024

Новосибирск
Издательство Института математики

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Октябрь–декабрь 2024. Т. 31, № 4. С. 186–212

УДК 519.8 DOI: 10.33048/daio.2024.31.794

МЕТОД ДЕКОМПОЗИЦИИ ДЛЯ УПРАВЛЕНИЯ ЗАПАСАМИ В ДВУХЭШЕЛОННОЙ СИСТЕМЕ СКЛАДОВ

А. Д. Юськов[1,a], *И. Н. Кулаченко*[2,b], *А. А. Мельников*[2,c], *Ю. А. Кочетов*[2,d]

[1] Новосибирский гос. университет,
ул. Пирогова, 2, 630090 Новосибирск, Россия
[2] Институт математики им. С. Л. Соболева,
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия
E-mail: [a] a.yuskov@g.nsu.ru, [b] ink@math.nsc.ru,
[c] melnikov@math.nsc.ru, [d] jkochet@math.nsc.ru

Аннотация. Склады первого эшелона в двухэшелонной системе предназначены для выполнения заказов клиентов. Во втором эшелоне находится центральный склад, пополняющий запасы на складах первого эшелона. Заказы клиентов можно выполнять частично, но общая доля выполненных заказов должна быть не меньше заданного порога. Требуется минимизировать общую стоимость хранения товаров на всех складах. Работа системы моделируется с помощью детерминированной имитационной модели, которая вычисляет долю удовлетворения заказов и стоимость хранения в течение планового периода в зависимости от параметров управления запасами на каждом складе по каждому типу товара. Разработан метод декомпозиции, основанный на решении подзадач для каждого типа товара. Предложены подходы для точного решения задачи. Приводятся результаты вычислительных экспериментов на примерах со 100 складами и 1000 типами товаров. На примерах с известным точным решением в двух случаях удалось найти оптимум, в остальных случаях отклонение от оптимума составило не более 1,9%. Табл. 5, ил. 1, библиогр. 23.

Ключевые слова: оптимизация «чёрного ящика», задача о рюкзаке, локальный поиск.

Введение

Оптимизация цепей поставок играет важную роль для сокращения издержек крупных компаний [1, 2]. Как правило, поступление заказов клиентов трудно предсказуемо, поэтому хранение запасов на складах

необходимо для своевременного удовлетворения запросов клиентов. Однако это ведёт к большим затратам на хранение. Управление политиками пополнения запасов направлено на поиск компромисса между долей удовлетворённых заказов и затратами на хранение. Основным способом управления запасами является использование на каждом складе политик хранения, которые регулируют запросы на пополнение склада. При разных политиках склады могут заказывать новые товары периодически или основываясь на текущих остатках. Размер заказа также может быть фиксированным или основываться на спросе клиентов. Обзор различных подходов, связанных с управлением запасами, можно найти в [3].

Многоэшелонные модели складов активно исследовались с середины прошлого столетия как в теоретическом [4, 5], так и в алгоритмическом плане [6]. В работах [7, 8] авторы предлагают математическую модель и методы решения с помощью эвристик, точных подходов и стохастического программирования. К сожалению, такой путь приводит либо к сильно упрощённым моделям, либо к высокой вычислительной сложности методов [9]. В данной работе рассматривается другой подход, который имеет ряд преимуществ, главным из которых является возможность сколь угодно точно моделировать бизнес-процессы. Он основан на имитационном моделировании и использовании таких моделей в численных методах. В этом случае алгоритм оптимизации использует имитационную модель в качестве «чёрного ящика» и оптимизирует её входные параметры. Недавние исследования продемонстрировали высокий потенциал этого подхода в детерминированном [10] и стохастическом [11] случаях. Однако, поскольку свойства целевой функции и ограничений теперь неизвестны, а алгоритм не может использовать специфическую информацию о задаче, эта схема порождает новые трудности: нет аналитических выражений для всех или части ограничений и целевой функции, нет производных, градиентов, свойств выпуклости и др. В некоторых подходах используются только входные данные и вывод симулятора [12, 13], другие сочетают моделирование с математическим программированием [14], стараясь привлечь как можно больше дополнительной информации об оптимизируемом бизнес-процессе. В последнее время растёт внимание ко второму подходу [15]. Насколько нам известно, постановка задачи, приведённая в данной работе, нова, хотя ранее встречались похожие варианты, например, в [16, 17].

В данной работе предлагаются идеи декомпозиции чёрного ящика на много маленьких, оптимизируемых независимо друг от друга, а потом собираемых воедино моделью о многовариантном рюкзаке. Исследуется задача управления запасами в двухэшелонной системе складов.

Склады первого эшелона предназначены для выполнения заказов клиентов. Во втором эшелоне находится центральный склад. Он не обслуживает заказы клиентов и предназначен только для пополнения запасов на складах первого эшелона. Заказы клиентов можно выполнять частично, но общая доля выполненных заказов должна быть не меньше заданного порога. Требуется минимизировать общую стоимость хранения товаров на всех складах при условии, что доля выполненных заказов не ниже указанного порога. Для вычисления стоимости хранения и доли выполненных заказов используется детерминированная имитационная модель. Для решения задачи предлагается эвристический алгоритм, основанный на идеях декомпозиции. Алгоритм решает подзадачи для одного типа товаров на всех складах при разных значениях порога, используя имитационную модель как чёрный ящик. Эти решения затем используются в модели о многовариантном рюкзаке для поиска оптимальной комбинации долей удовлетворения заказов клиентов по каждому типу товаров. Показано, что такой подход гарантирует получение точного решения всей задачи, если точно решать подзадачи для каждого типа товаров и правильно выбрать доли удовлетворения заказов. Приводится нижняя оценка минимума суммарных затрат, которая позволяет строить точные алгоритмы. Проведены вычислительные эксперименты на сгенерированных примерах со 100 складами и 1000 типами товаров. Эксперименты показали высокую эффективность предложенного подхода и его превосходство перед пакетом Nevergrad [18] и специализированным пакетом MEO для решения описанной задачи, предоставленным компанией вместе с имитационной моделью. На примерах с известным точным решением в двух случаях удалось найти оптимум, в остальных случаях отклонение от оптимума составило не более 1,9%. Данная статья является расширенной и исправленной версией работы [19], представленной на международной конференции MOTOR-2023 в Екатеринбурге. В новой версии используется улучшенная модель для одного типа товаров, чтобы гарантировать оптимальность финального решения при правильно подобранных долях выполнения заказов клиентов.

Статья организована следующим образом. В разд. 1 приводится постановка задачи. В разд. 2 описывается предлагаемый метод декомпозиции, в частности, в п. 2.2 представлен алгоритм координатора, а в п. 2.3 — алгоритм локального поиска для подзадач одного товара. В разд. 3 представлены некоторые свойства алгоритма и идеи для построения точного решения. В разд. 4 обсуждаются результаты вычислительных экспериментов на реальных данных, а в заключении формулируются основные выводы.

1. Двухэшелонная задача управления запасами

Политика управления запасами определяет, когда и в каком количестве отправлять заказы на пополнение. Ниже рассматривается двухэшелонная система хранения со складами, которая состоит из нескольких местных складов в первом эшелоне и одного центрального склада во втором эшелоне.

На местных складах применяется политика (s, S). Для каждого отдельного товара эта политика содержит два целочисленных положительных параметра s и S, представляющих минимальный и максимальный уровни запасов. Когда на склад приходит заказ на товар, он немедленно удовлетворяется товарами со склада, если таковые имеются в наличии; в противном случае он резервируется. Виртуальный уровень запасов — это количество товара в наличии, а также количество товаров, отправленных на склад, но ещё не полученных, за вычетом зарезервированных. Если виртуальный уровень запасов на складе опускается ниже порогового значения s, то на центральный склад отправляется запрос на пополнение запасов до уровня S.

На центральном складе применяется политика $(\overline{S} - 1, \overline{S})$. Данная политика содержит только один параметр $\overline{S}$, представляющий уровень пополнения. Этот параметр определяет количество товаров, хранящихся на складе. Если центральный склад получает запрос на пополнение, то он выполняет его и запрашивает такое же количество товаров у внешнего поставщика. Предполагается, что у этого поставщика имеется неограниченное количество товаров.

Если существует m местных складов, то должны быть определены $2m + 1$ параметров политик для управления уровнями запасов на всех складах для одного товара. Так как на складах одновременно хранится большое количество различных видов товаров, число переменных может быть довольно большим.

Показатели эффективности системы хранения обычно связаны с удовлетворённостью клиентов: например, с процентом заказов или спроса, выполненных в установленные сроки. Помимо удовлетворённости клиентов, для компании также важны затраты на хранение запасов. Стоимость хранения равна интегралу от реального уровня запасов по времени хранения, умноженному на стоимость хранения одной единицы товара.

Оптимизация системы хранения заключается в определении таких значений параметров управления запасами s и S для всех местных складов, а также параметров $\overline{S}$ центрального склада для каждого товара, чтобы процент заказов, выполненных сразу, был не меньше заданного значения, а затраты на хранение были минимальны. Табл. 1 содержит обозначения, используемые в статье.

Таблица 1

Список обозначений

Множества:	
I	множество индексов товаров
F	множество индексов местных складов
Функции:	
φ^L	суммарный спрос на местных складах
φ_i^L	суммарный спрос на товар $i \in I$ на местных складах
φ^C	суммарный спрос на центральном складе
φ_i^C	суммарный спрос на товар $i \in I$ на центральном складе
ψ_i^L	суммарный удовлетворённый спрос на товар $i \in I$ на мест. складах
ψ_i^C	суммарный удовлетворённый спрос на товар $i \in I$ на центр. складе
C	стоимость хранения, вычисляемая имитационной моделью
C_i	стоимость хранения товаров типа $i \in I$
ω^L	доля удовлетворённого спроса на местных складах
ω^C	доля удовлетворённого спроса на центральном складе
l	минимальное возможное значение переменной
u	максимальное возможное значение переменной
Константы:	
α	минимальная допустимая доля удовлетворённого суммарного спроса на местных складах
β	минимальная допустимая доля удовлетворённого суммарного спроса на центральном складе
Переменные задачи:	
x_{if}^1	мин. уровень запасов для товара $i \in I$ на местном складе $f \in F$
x_{if}^2	макс. уровень запасов для товара $i \in I$ на местном складе $f \in F$
x_i^3	уровень пополнения для товара $i \in I$ на центральном складе
X	вектор переменных параметров политик, состоящий из всех $x_{if}^1, x_{if}^2, x_i^3$
X_i	вектор параметров политик, состоящий из переменных $x_{if}^1, x_{if}^2, x_i^3$ для товара i

Функции $\varphi^C, \varphi_i^C, \psi_i^L, \psi_i^C, C_i, \omega^L, \omega^C$ зависят от параметров политик управления запасами.

В данных обозначениях задача управления запасами может быть записана следующим образом:

$$\min_X C(X), \tag{1}$$

$$\omega^L(X) \geqslant \alpha, \tag{2}$$

$$\omega^C(X) \geqslant \beta, \tag{3}$$

$$x_{if}^2 \geqslant x_{if}^1, \quad i \in I,\ f \in F, \tag{4}$$

$$l(X) \leqslant X \leqslant u(X), \quad X \in \mathbb{Z}. \tag{5}$$

Целевая функция (1) минимизирует общую стоимость хранения запасов. Показатели удовлетворённости обеспечиваются в силу ограничений (2)–(3). Неравенства (4) гарантируют, что минимальный уровень запасов не больше максимального. Область допустимых значений переменных определяется при помощи неравенств (5).

Будем рассматривать случай, в котором известно, что общая стоимость хранения вычисляется как сумма стоимостей хранения всех товаров. Тогда можно выписать следующие соотношения:

$$C(X) = \sum_i C_i(X),$$

$$\varphi^L = \sum_i \varphi_i^L,$$

$$\varphi^C(X) = \sum_i \varphi_i^C(X),$$

$$\psi^L(X) = \sum_i \psi_i^L(X),$$

$$\psi^C(X) = \sum_i \psi_i^C(X),$$

$$\omega^L(X) = \psi^L(X)/\varphi^L,$$

$$\omega^C(X) = \psi^C(X)/\varphi^C(X).$$

Для получения значений φ_i^L, $\varphi_i^C(X)$, $C_i(X)$, $\psi_i^L(X)$ и $\psi_i^C(X)$ при заданных параметрах X используется детерминированная имитационная модель.

2. Предлагаемый метод решения

Проведём декомпозицию представленной задачи, которая позволит разбить её на независимые подзадачи, соответствующие отдельным товарам. Выгода от использования такой декомпозиции заключается в том, что моделирование работы склада для разных товаров может выполняться независимо, а это открывает возможности выполнения параллельных вычислений. На основе предложенной декомпозиции построим многоагентную систему, в которой подзадачи решаются отдельными агентами, управляемыми центральным координатором.

2.1. Декомпозиция задачи. Можно заметить, что целевая функция (1) и ограничения (2), (3) представляют сумму независимых величин

по товарам. Также известно, что функции $C_i(X)$, $\varphi_i^C(X)$, $\psi_i^L(X)$ и $\psi_i^C(X)$ могут быть вычислены независимо для каждого товара. Тем самым выглядит разумной схема, в которой алгоритм пытается подобрать оптимальный вклад каждого товара в ограничения (2), (3), а затем независимо находит наилучшие в плане стоимости параметры политик хранения для каждого товара так, чтобы удовлетворялись заданные пороги удовлетворения спроса.

Ограничение (2) можем переписать следующим образом:

$$\sum_{i\in I}\psi_i^L(X_i) \geqslant \alpha\sum_{i\in I}\varphi_i^L, \tag{6}$$

а (3) запишем в таком виде:

$$\sum_{i\in I}\left(\psi_i^C(X_i) - \beta\varphi_i^C(X_i)\right) \geqslant 0. \tag{7}$$

Введём следующие обозначения: переменные γ_i, $i \in I$, определяют нижнюю границу удовлетворённого спроса клиентов на i-й товар (значение под суммой в левой части ограничения (6)); переменные δ_i, $i \in I$, определяют нижнюю границу значения под суммой в ограничении (7) для i-го товара: $\psi_i^C(X_i) - \beta\varphi_i^C(X_i) \geqslant \delta_i$. С помощью этих обозначений задачу можно переписать следующим образом:

$$\min_{(\gamma_i),\,(\delta_i)}\sum_{i\in I}C_i(\gamma_i,\delta_i), \tag{8}$$

$$\sum_{i\in I}\gamma_i \geqslant \alpha\sum_{i\in I}\varphi_i^L, \tag{9}$$

$$\sum_{i\in I}\delta_i \geqslant 0, \tag{10}$$

где стоимость $C_i(\gamma_i,\delta_i)$ хранения товара i на всех складах находится из решения задачи $\texttt{OneItem}_i(\gamma_i,\delta_i)$

$$\min_{X_i} C_i(X_i), \tag{11}$$

$$\psi_i^L(X_i) \geqslant \gamma_i, \tag{12}$$

$$\psi_i^C(X_i) - \beta\varphi_i^C(X_i) \geqslant \delta_i, \tag{13}$$

$$x_{if}^2 \geqslant x_{if}^1, \quad f \in F, \tag{14}$$

$$l(X_i) \leqslant X_i \leqslant u(X_i), \quad X_i \in \mathbb{Z}. \tag{15}$$

Задача координатора (8)–(10) направлена на нахождение порогового значения удовлетворённого спроса для каждого товара таким образом, чтобы суммарная доля удовлетворённого спроса составляла не менее α для местных складов и не менее β для центрального склада. Заметим,

что ограничения (9), (10) гарантируют это требование при условии выполнения ограничений (12), (13). На нижнем уровне рассматриваются $|I|$ задач оптимизации, направленных на минимизацию затрат на хранение для каждого товара при условии, что должен быть удовлетворён определённый объём спроса. Значения целевой функции (11) и ограничений (12), (13) вычисляются при помощи имитационной модели. В разд. 3 будет показано, что оптимальные решения задач (1)–(5) и (8)–(15) совпадают.

Заметим, что при данных политиках хранения местные склады всегда запрашивают у центрального не больше предметов, чем суммарный спрос от клиентов, поэтому $\varphi_i^C(X_i) \leqslant \varphi_i^L$. Так как $\psi_i^C(X_i) \leqslant \varphi_i^C(X)$, можно установить границы для значений δ:

$$-\beta\varphi_i^L \leqslant \psi_i^C(X_i) - \beta\varphi_i^C(X_i) \leqslant (1-\beta)\varphi_i^L.$$

Для допустимых запросов γ_i и δ_i введём множества $\Gamma_i = \{0, \ldots, \varphi_i^L\}$, $\Delta_i = \{-\beta\varphi_i^L, \ldots, (1-\beta)\varphi_i^L\}, U_i = \Gamma_i \times \Delta_i$. Минимальные и максимальные возможные значения запросов используются в алгоритме, представленном далее, а обозначения Γ_i и Δ_i используются при теоретических обоснованиях в разд. 3.

2.2. Многоагентная система. В этом пункте опишем эвристический подход к решению задачи (8)–(15). Подход основан на том факте, что задачи одного товара (11)–(15) могут обрабатываться независимо для каждого отдельного $i \in I$. Это позволяет разработать многоагентную схему (MAS), где несколько *рабочих агентов* параллельно решают задачи для одного товара с использованием алгоритма локального поиска, в то время как *координирующий агент* направляет поиск, запрашивая решения для некоторых конкретных γ_i и δ_i, и агрегирует решения задач для одного товара в решение для исходной задачи.

Решения для отдельных товаров. Во время вычисления сохраняем наборы наилучших найденных решений $P_i = \{X_i^1, \ldots, X_i^{k_i}\}$ для каждой задачи $\texttt{OneItem}_i$, $i \in I$, и разных пар (γ, δ). Пусть имеется набор номеров запомненных решений $K_i = \{1, \ldots, k_i\}$, $i \in I$. После решения подзадачи для каждого решения X_i^k, $i \in I$, $k \in K_i$, известны связанные с ним величины $d_{ik}^C = \varphi_i^C(X_i^k)$, $s_{ik}^L = \psi_i^L(X_i^k)$, $s_{ik}^C = \psi_i^C(X_i^k)$, и $c_{ik} = C_i(X_i^k)$. Кроме того, сохраняется пара $(\gamma_{ik}, \delta_{ik})$ соответствующих значений параметров задачи $\texttt{OneItem}$.

Скажем, что решение X_i^j *доминирует* решение X_i^k, если $\gamma_{ij} \geqslant \gamma_{ik}$, $\delta_{ij} \geqslant \delta_{ik}$ и $c_{ij} \leqslant c_{ik}$. Решение X_i' назовём *недоминируемым*, если в P_i нет X_i'', доминирующего X_i'. Всякий раз при сохранении нового решения

проверяем, доминирует ли оно некоторое из существующих и является ли доминируемым. Все доминируемые решения удаляются.

На начальном шаге для вычисления запрашиваются решения задачи для одного товара, соответствующие парам (γ_i, δ_i) таким, что $\gamma_i = \delta_i + \beta\varphi_i^L = \left\lceil \frac{l}{p}\varphi_i^L \right\rceil$. Здесь $p \in \mathbb{Z}^+$ является параметром алгоритма, а l принимает целочисленные значения от 1 до p. Следует отметить, что решение, полученное для некоторой пары (γ, δ), допустимо для задачи с одним элементом с параметрами (γ', δ') такими, что $\gamma \geqslant \gamma'$ и $\delta \geqslant \delta'$. Таким образом, разумно начать вычисление решений с задачи для одного товара с пары (γ, δ), соответствующей значению $l = p$. Полученное решение используется позже для инициализации локального поиска при решении задачи с $l = p - 1$. Затем процесс повторяется дальше, обрабатывая значения l в порядке убывания. В наших экспериментах использовано $p = 20$. Если для элемента требуется менее 10^5 оцениваний целевой функции, чтобы полностью перечислить все возможные конфигурации значений политик, то применяем полный перебор к этим элементам и сохраняем все решения, не являющиеся доминируемыми.

Координирующий агент. Решение исходной задачи (1)–(5) получаем, беря одно решение для каждой из задач одного товара из набора решений и объединяя эти решения вместе. Для заданного набора решений $\mathbf{P} = (P_1, \ldots, P_{|I|})$ наилучшая комбинация может быть получена путём решения вспомогательной задачи булева программирования. Введём булевы переменные (z_{ik}), $i \in I$, $k \in P_i$, равные единице, если решение X_i^k входит в итоговое решение, и нулю в противном случае. Используя эти обозначения, сформулируем вспомогательную задачу `coordinator`$(\mathbf{P})$, решив которую, можно получить ответ к исходной задаче:

$$\min_{(z_{ik})} \sum_{i \in I} \sum_{k \in K_i} c_{ik} z_{ik}, \tag{16}$$

$$\sum_{k \in K_i} z_{ik} = 1, \quad i \in I, \tag{17}$$

$$\sum_{i \in I} \sum_{k \in K_i} s_{ik}^L z_{ik} \geqslant \alpha \sum_{i \in I} \varphi_i^L, \tag{18}$$

$$\sum_{i \in I} \sum_{k \in K_i} s_{ik}^C z_{ik} \geqslant \beta \sum_{i \in I} \sum_{k \in K_i} d_{ik}^C z_{ik}, \tag{19}$$

$$z_{ik} \in \{0, 1\}, \quad i \in I,\ k \in K_i. \tag{20}$$

Целевая функция (16) состоит в минимизации стоимости хранения запасов в полном решении. В силу равенств (17) для каждой из задач одного товара в итоговое решение будет взято единственное решение.

Неравенства (18), (19) гарантируют допустимый процент удовлетворённых заказов. При изложенной ранее схеме запроса частичных решений данная задача всегда имеет допустимое решение. Действительно, решение с коэффициентами выполнения на местных и центральном складах, равными единице, всегда помещается в историю на шаге инициализации, когда решается задача одного товара, соответствующая $l = p$. По определению доминирования такое решение может быть удалено из истории только в том случае, если в него будет помещено другое решение, имеющее тот же процент выполнения заказов.

Алгоритм координирующего агента итеративный: на каждом шаге решается задача верхнего уровня для текущего набора частичных решений задач одного товара, а затем координатор посылает запросы на вычисление решений подзадач с установленными процентами выполнения заказов в окрестности тех, которые соответствуют решениям, выбранным в итоговое решение.

Пусть (z_{ik}), $i \in I$, $k \in K_i$, — оптимальное решение задачи координатора на текущем шаге, и пусть $i \in I$, $k \in K_i$ таковы, что $z_{ik} = 1$. Тогда X_i^k является частью итогового решения, построенного на текущем шаге. Рассмотрим пару $(\gamma_{ik}, \delta_{ik})$, относящуюся к этому решению. Перед началом следующего шага координатор запросит найти решения задач $\texttt{OneItem}_i$ со схожими уровнями.

Через ε обозначим величину шага, на который в задачах одного товара изменяются уровни γ и δ. Изначально $\varepsilon = \varepsilon_0$, и эта величина остаётся постоянной, пока полное решение изменяется. Когда полное решение

Алгоритм 1. Многоагентная система (MAS)

function `MAS`
  $\mathbf{Q}^0 = \big(Q_i^0\big)$, $i \in I$; ▷ Начальное множество пар (γ, δ)
  $P_i = \varnothing$, $i \in I$; ▷ Начальный набор решений для каждого товара
  for all $(\gamma_i, \delta_i) \in Q_i^0$ **do** `oneItemLS`$(\gamma_i, \delta_i, P_i)$;
  $\gamma^0, \delta^0 \leftarrow$ `solveMP`(P);
  $t \leftarrow 1$, $\varepsilon \leftarrow \varepsilon_0$;
  while не достигнут критерий остановки **do**
    for $i \in I$ **do**
      $Q_i^t \leftarrow Q_i^{t-1} \cup A\big(\gamma_i^t, \delta_i^t\big)$;
      for all $(\gamma_i, \delta_i) \in Q_i^t \setminus Q_i^{t-1}$ **do** `oneItemLS`$(\gamma_i, \delta_i, P_i)$;
    $\gamma^t, \delta^t \leftarrow$ `solveMP`(P);
    if $\gamma^t = \gamma^{t-1}$ и $\delta^t = \delta^{t-1}$ **then** $\varepsilon \leftarrow \varepsilon/2$;
    $t \leftarrow t + 1$;
  return лучшее найденное решение.

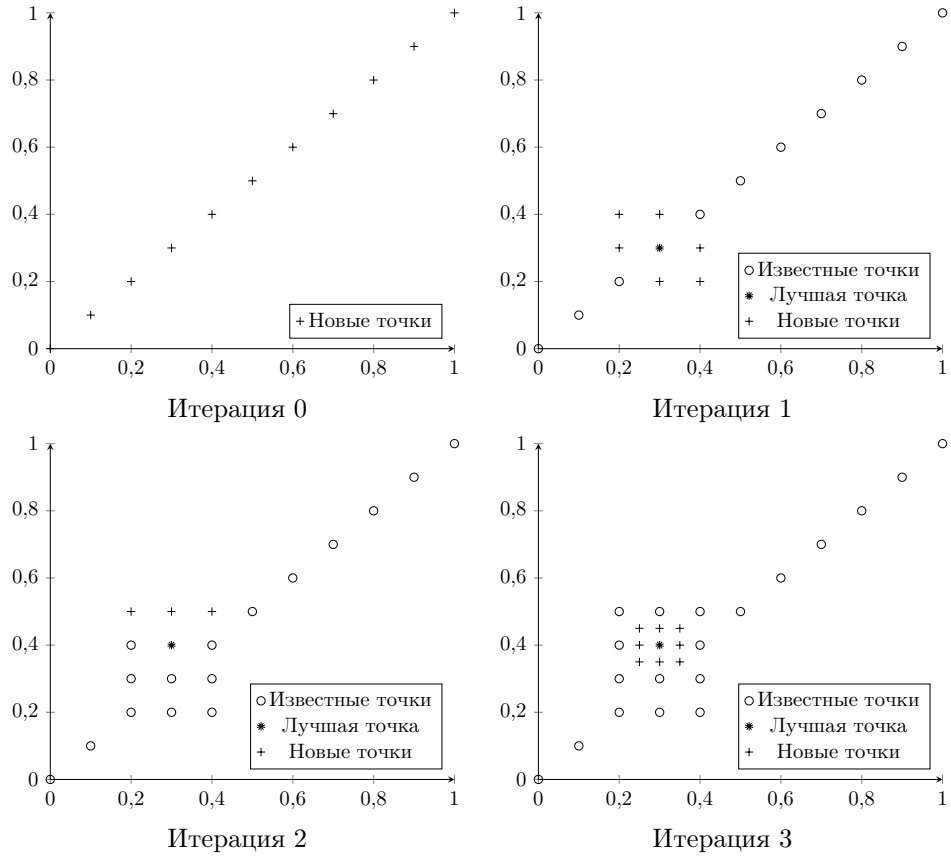

Рис. 1. Пример изменения множества Q_i

оказывается таким же, как и на предыдущем шаге, изменяем величину шага $\varepsilon \leftarrow \varepsilon/2$ — аналогично алгоритмам координатного поиска [20]. Таким образом, если ε — величина шага на итерации t, γ_i^t, δ_i^t — лучшие найденные на текущий момент запросы процентов выполнения заказов, то на следующей итерации запрашиваются решения задач для уровней из множества $A\big(\gamma_i^t, \delta_i^t\big) = \big\{\gamma_i^t \pm \lceil \varepsilon\varphi_i^L \rceil\big\} \times \big\{\delta_i^t \pm \lceil \varepsilon\varphi_i^L \rceil\big\}$. Пусть Q_i^t — множество запросов локального поиска для товара i на шаге t. Тогда $Q_i^0 = \big\{\big(\lceil \frac{l}{p}\varphi_i^L \rceil, \lceil \frac{l}{p}\varphi_i^L \rceil - \beta\varphi_i^L\big) \mid l = 0, \ldots, p\big\}$, $Q_i^{t+1} = Q_i^t \cup A\big(\gamma_i^t, \delta_i^t\big)$.

Если $\lceil \varepsilon\varphi_i^L \rceil \leqslant 1$ для всех $i \in I$ и объединённое решение с предыдущего шага не меняется, то алгоритм завершается. Изложенная многоагентная система представлена в виде алгоритма 1. На рис. **??** приведён пример эволюции множества Q_i для некоторого товара в ходе работы этого алгоритма.

2.3. Задача для одного товара. Агенты могут независимо решать задачи, поставленные координатором. Чтобы найти решения для этих задач, агенты используют рандомизированный локальный поиск с дополнительными компонентами, повышающими его производительность. При заданном типе товаров $i \in I$ и паре ограничений (γ_i, δ_i) задача $\texttt{OneItem}_i(\gamma_i, \delta_i)$, определённая как (11)–(15), направлена на то, чтобы найти параметры X_i политик для товара i на всех складах.

В случае, когда множество P_i найденных на данный момент решений без доминирования не пусто, рандомизированный локальный поиск начинается с наиболее подходящего решения из этого множества. Это решение $X_i^{k'} \in P_i$, где

$$k' = \arg\min_{k \in K_i} \big\{ C_i\big(X_i^k\big) \mid \gamma_{ik} \geqslant \gamma \text{ и } \delta_{ik} \geqslant \delta \big\}.$$

На этапе инициализации MAS набор решений может быть пустым. В соответствии с порядком запросов на этапе инициализации, описанным в п. 2.2, первым запрашивается то решение, в котором удовлетворены все заказы. Такое решение можно построить с помощью простой эвристики. Эта эвристика использует предположение о независимости каждого местного склада при некотором большом значении $\overline{S}$ на центральном складе и необходимости полностью удовлетворить весь спрос на местных складах. Используя предположение о монотонности стоимости хранения запасов на отрезке $[s, S]$, находим наилучшие решения с помощью алгоритма дихотомии. После этого корректируем политику $\overline{S}$ центрального склада, чтобы минимизировать затраты на хранение на нём.

Окрестность $N(X_i)$, используемая в рандомизированном локальном поиске, состоит из полудопустимых векторов (т. е. удовлетворяющих ограничениям (14)), которые определяют параметры политики, различающиеся не более чем на двух складах (местных или центральном), и разница составляет не более двух единиц. Рандомизированная окрестность решения X_i размера $r \in \mathbb{Z}^*$ будет обозначаться через $N_r(X_i)$. Она состоит из r случайно выбранных элементов множества $N(X_i)$.

Также используем список запретов, в котором сохраняем десять последних посещённых решений, чтобы избежать попадания в цикл, когда набор $N(X_i)$ мал и рандомизация не приводит к выходу из локальных оптимумов. Во время рассмотрения окрестности $N_r(X_i)$ алгоритм переходит к лучшему соседу, которого нет в списке запретов. При этом используем штрафную функцию

$$\texttt{InfPenalty}\big(X_i^k\big) = M\big(\max\big\{\gamma_i - \psi^L\big(X_i^k\big), 0\big\} + \max\big\{\delta_i - \psi^C\big(X_i^k\big), 0\big\}\big),$$

где M — большое число, чтобы предотвратить переход в недопустимое решение.

По мере вычислений обновляем множества P_i, сохраняя новые найденные недоминируемые решения. Когда лучшее из найденных решений не меняется в течение T итераций, поиск возвращает его как лучшее решение. Общая схема рандомизированного локального поиска представлена алгоритмом 2.

Алгоритм 2. Алгоритм локального поиска для задачи одного товара

function `oneItemLS`$(\gamma_i, \delta_i, P_i)$
  $X_i = \arg\min\limits_{X_i^k \in P_i} \{C(X_i^k) \mid \gamma_{ik} \geqslant \gamma_i \text{ и } \delta_{ik} \geqslant \delta_i\}$;
  while не выполнен критерий остановки **do**
    $X_i \leftarrow \arg\min\limits_{X_i^k \in N_r(X_i)} \{C(X_i^k) + \texttt{InfPenalty}(X_i^k, v)\}$;
    обновить множество P_i, добавив решение X_i;
    $X_i^* \leftarrow$ лучшее из X_i, X_i^*;
    if выполнен критерий перезапуска **then**
      $X_i \leftarrow X_i^*$;
      очистить список запретов;
  return X_i^*.

3. Точное решение задачи

Лемма 1. *Пусть X^* — оптимальные параметры политик хранения, $p_i^* = (\gamma_i^*, \delta_i^*)$ — соответствующие значения для задачи агента* (11)–(15). *Тогда если на некоторой итерации t для любого товара $p_i^* \in Q_i^t$, т. е. оптимальные значения γ_i^* и δ_i^* попали в множество запросов к локальному поиску, и алгоритм нижнего уровня вернул некоторые оптимальные решения для соответствующих задач* (11)–(15) *в этих точках, то координатор вернёт оптимальное решение исходной задачи* (1)–(5).

Доказательство. Пусть алгоритм вернул решения $X_i^{k'}$ с соответствующими ему уровнями удовлетворённости $p_i^{k'} = \left(s_{ik'}^L, s_{ik'}^C - \beta d_{ik'}^C\right)$ и стоимостями $c_{ik'}$. Так как решение агента допустимо, то $p_i^{k'} \succeq p_i^*$. Значит, решение

$$z_{ik'} = 1, \quad z_{ik} = 0, \quad k \neq k', \tag{21}$$

допустимо для задачи координатора. Кроме того, поскольку решение X_i^* допустимо для задачи агента и он вернул оптимальное решение для своей задачи, то $c_{ik'} \leqslant C_i(X_i^*)$. Тогда для стоимости решения (21) имеет место неравенство $\sum\limits_{i \in I} c_{ik'} \leqslant C(X^*)$. Таким образом, $\sum\limits_{i \in I} c_{ik'} = C(X^*)$ и решение $X^{k'}$, полученное из (21), оптимально. Значит, решение, возвращённое координатором, также оптимально. Лемма 1 доказана.

Лемма 1 говорит о том, что если мы умеем точно решать задачу агента, то для получения оптимального решения достаточно определить правильные значения γ_i и δ_i. Это позволяет полностью перейти к задаче координатора. Обозначим через $\texttt{coordinator}(\mathbf{Q}, \mathbf{H})$ процедуру, которая точно решает задачу координатора с множествами запросов $\mathbf{Q} = (Q_1, \ldots, Q_n)$ и функциями $\mathbf{H} = (H_1, \ldots, H_n)$, выдающими стоимость удовлетворения ограничений с порогами $p_i = (\gamma_i, \delta_i)$, и возвращает вектор оптимальных точек p^* в пространстве ограничений. Пусть $C_i(\gamma_i, \delta_i)$ обозначает оптимальное значение целевой функции при заданных ограничениях:

$$C_i(\gamma_i, \delta_i) = \min_{X_i} \left\{ C_i(X_i) \mid \psi_i^L(X_i) \geqslant \gamma_i,\ \psi_i^C(X_i) - \beta\varphi_i^C(X_i) \geqslant \delta_i \right\}.$$

Для $p_i^1 = \left(\gamma_i^1, \delta_i^1\right)$, $p_i^2 = \left(\gamma_i^2, \delta_i^2\right)$ положим $p_i^1 \succeq p_i^2$, если $\gamma_i^1 \geqslant \gamma_i^2$ и $\delta_i^2 \geqslant \delta_i^2$. Тогда можно сформулировать следующие утверждения.

Свойство 1. *Функция $C_i(p)$ монотонна, т. е. из отношения $p^1 \succeq p^2$ следует $C_i(p^1) \geqslant C_i(p^2)$.*

Свойство монотонности позволяет нам предложить метод для подсчёта нижней оценки для оптимального решения, если известны значения функции в некоторых точках. Пусть U_i — множества всех допустимых значений пар $(\gamma_i, \delta_i) = p_i$ и Q_i — множества точек, в которых значения функций C_i известны.

Лемма 2. *Пусть $\left(0, -\beta\varphi_i^L\right) \in Q_i$. Если для каждого товара $i \in I$ доопределить функцию C_i меньшим значением, т. е. $C_i'(p) = C_i(p')$, $p' \preceq p$, $p' \in Q_i$, $p \in U_i \setminus Q_i$, точно решить полученную задачу координатора $(Q_i' = U_i, C_i')$, то полученное решение будет нижней оценкой для оптимального решения исходной задачи.*

Доказательство. Действительно, пусть p_i^*, $i \in I$, — запросы для оптимального решения. Тогда решение (p_i^*) допустимо и $C_i'(p_i^*) \leqslant C_i(p_i^*)$. Значит, стоимость оптимального решения изменённой задачи не превосходит $C(p^*)$. Лемма 2 доказана.

Заметим, что точке $\left(0, -\beta\varphi_i^L\right)$ соответствует тривиальное решение, в котором все параметры политик равны 0.

Зная нижнюю оценку для данной задачи, нетрудно построить простой точный алгоритм, который находит оптимальное решение. Алгоритм 3 описывает такой метод. Он начинает с некоторого множества точек Q_i, в которых известны значения функции C_i. На каждом шаге алгоритм, зная точное значение C_i в точках Q_i, доопределяет функцию C_i до C_i' во всех точках $U_i \setminus Q_i$ и решает задачу координатора, получая нижнюю оценку. Если эта оценка получена в точке, в которой значение C_i известно

точно, то это решение является решением и для исходной задачи, и алгоритм возвращает это решение. Иначе алгоритм вычисляет значение C_i в полученной точке с целью улучшить оценку.

Алгоритм 3. Точный алгоритм

1: Выбрать и оценить начальные точки $\mathbf{Q}$;
2: **while** критерий остановки не выполнен **do**
3: **for all** $i \in I$, $p_i^j \in Q_i$ **do** вычислить точные значения $C_i(p_i^j)$;
4: доопределить функции C_i до C_i' во всех неизвестных точках;
5: $p \leftarrow \texttt{coordinator}(\mathbf{U}, \mathbf{C}')$;
6: **if** все значения $C_i(p_i)$ известны точно **then** СТОП; ▷ Найдено оптимальное решение
7: $Q_i \leftarrow Q_i \cup \{p_i\}$.

Теорема 1. *Алгоритм* 3 *находит оптимальное решение задачи.*

Доказательство. Из доказанного ранее $\sum\limits_i C_i'(p_i)$ является нижней оценкой на стоимость оптимального решения исходной задачи. Если все значения $C_i(p_i)$ известны на шаге 6, то $\sum\limits_i C_i'(p_i) = \sum\limits_i C_i(p_i)$, т. е. реальная стоимость решения p совпадает с нижней оценкой, и оно оптимально. В противном случае ходя бы для одного $i \in I$ точка p_i неизвестна, и на следующей итерации значение $C_i(p_i)$ вычисляется хотя бы в одной новой точке. Так как множество возможных точек конечно, алгоритм конечен. Теорема 1 доказана.

Кроме того, можно заметить, что способ получения нижних оценок годится в случае, когда рассматривается не всё пространство допустимых значений, а некоторое его подмножество.

Замечание 1. Лемма 2 остаётся верной, если рассматривать не всё пространство решений U_i, а некоторое его прямоугольное подмножество $V_i = \{(\gamma_i, \delta_i) \in U_i \mid a \leqslant \gamma_i \leqslant b, c \leqslant \delta_i \leqslant d\}$ при условии, что известно значение C_i в точке $\min V_i = \big(\min\limits_{(\gamma_i,\delta_i)\in V_i} \gamma_i, \min\limits_{(\gamma_i,\delta_i)\in V_i} \delta_i\big)$. В таком случае полученное решение будет являться нижней оценкой для оптимального решения среди всех $p_i \in V_i$. Доказательство этого факта аналогично доказательству леммы 2.

Наличие способа построения верхних и нижних оценок для подмножеств пространства решений позволяет построить для данной задачи алгоритм, основанный на методе ветвей и границ. Компоненты такого алгоритма выглядят следующим образом. Для текущей подобласти $\mathbf{V} = (V_1, \dots V_n)$ имеем следующее.

• Верхняя граница: оптимальное решение $\texttt{coordinator}(\mathbf{V} \cap \mathbf{Q}, \mathbf{C})$.

• Нижняя граница: оптимальное решение $\texttt{coordinator}(\mathbf{V}, \mathbf{C}')$.

• Ветвление: например, для некоторого товара i выбираем порог γ_i' и разбиваем область на две:

$$V_i' = \{(\gamma_i, \delta_i) \in V_i \mid \gamma_i \leqslant \gamma_i'\}, \quad V_i'' = \{(\gamma_i, \delta_i) \in V_i \mid \gamma_i > \gamma_i'\},$$

или аналогично для δ_i.

• Дополнительно на каждом шаге происходит вычисление функции $C_i(\min V_i)$ для обеспечения корректности леммы 2, а для получения перспективных решений опционально вычисляется $C(\texttt{coordinator}(\mathbf{V}, \mathbf{C}'))$.

Описанный алгоритм может демонстрировать более высокую производительность, чем алгоритм 3.

Заметим, что в процессе работы алгоритма приходится много раз решать задачу целочисленного программирования (16)–(20), в которой множества P_i состоят из всех возможных точек. Это может оказаться трудоёмкой задачей, поэтому для уменьшения размерности задачи координатора можно использовать следующее изменение.

Замечание 2. Для некоторых точек p_i^1, p_i^2 и функции оценки стоимости H определим полное доминирование относительно H:

$$p_i^1 \preceq_H p_i^2, \quad \text{если } p_i^1 \preceq p_i^2 \text{ и } H\big(p_i^1\big) \geqslant H\big(p_i^2\big),$$

т. е. точка p_i^1 хуже точки p_i^2 по всем параметрам, включая значение целевой функции. Тогда в лемме 2 достаточно оставить только не полностью доминируемые точки относительно C_i'. Обозначим это сужение через U_i'.

Доказательство. Если $p_i^* \notin U_i'$, то существует точка $p_i^2 \in U_i'$ такая, что $p_i^2 \succeq_{C_i'} p_i^*$. Значит, $p_i^2 \succeq p_i^*$, и решение с p_i^2 допустимо для исходной задачи, а также $C_i'\big(p_i^2\big) \leqslant C_i'(p_i^*) \leqslant C_i(p_i^*)$. Замечание 2 доказано.

Замечание 3. Введём функции для получения «предыдущих» значений: $\texttt{prev}_i(\gamma) = \gamma - 1$, $\texttt{prev}_i(\delta) = \max\{\delta' \in \Delta_i \mid \gamma' < \gamma\}$. Пусть

$$Q_i^\gamma = \{\gamma \mid (\gamma, \delta) \in Q_i\} \cup \{\max \Gamma_i\}, \quad Q_i^\delta = \{\delta \mid (\gamma, \delta) \in Q_i\} \cup \{\max \Delta_i\}.$$

Тогда для корректности леммы 2 функцию C_i достаточно доопределить в точках

$$U_i' = \big\{(\texttt{prev}_i(\gamma), \texttt{prev}_i(\delta)),\ (\texttt{prev}_i(\gamma), \delta),\ (\gamma, \texttt{prev}_i(\delta)),\ (\gamma, \delta) \mid \gamma \in Q_i^\gamma,\ \delta \in Q_i^\delta\big\}.$$

Доказательство. Выберем оптимальную точку $p_i^* = (\gamma_i^*, \delta_i^*)$ и положим $D = \{p \in U_i' \mid p \succeq p_i^*\}$ — множество всех точек, доминирующих p_i^*.

Множество D непусто, так как $\max U_i' \in D$. Достаточно показать, что существует точка $p' \in D$ такая, что $C_i'(p') \leqslant C_i(p_i^*)$. Рассмотрим множество «самых левых» точек из D:

$$D^l = \{(\gamma_0, \delta) \in D \mid \gamma_0 = \min\{\gamma' \mid (\gamma', \delta') \in D\}\}$$

и возьмём «самую нижнюю» из них: $p_0 = (\gamma_0, \delta_0 = \min\{\delta' \mid (\gamma, \delta') \in D^l\})$. Тогда $C_i'(p_0) = C_i(p_1)$ для некоторой известной точки p_1. Если $p_1 \preceq p_i^*$, то условие выполнено. Значит, $\delta_1 > \delta^*$ или $\gamma_1 > \gamma^*$.

При $\delta_1 > \delta^*$ с необходимостью известно значение функции C_i в точке $p_2 = (\gamma_0, \mathtt{prev}_i(\delta_1)) \succeq p^*$. Если это не так, то указанное значение должно быть доопределено; противоречие с условием «минимальности» p_0. Возможны несколько вариантов расположения точки p_2.

1) В случае $p_2 = p^*$ известно значение $C_i(p^*)$, и требуемое условие выполнено.

2) Если $\delta_2 > \delta^*$ и $\gamma_2 > \gamma^*$, то по условию утверждения должна быть доопределена точка $p_3 = (\mathtt{prev}_i(\delta_2), \mathtt{prev}_i(\gamma_2))$. Повторяем рассуждения для неё.

3) Если $\delta_2 > \delta^*$ и $\gamma_3 = \gamma^*$, то аналогично должна быть доопределена точка $p_4 = (\mathtt{prev}(\delta_2), \gamma_2)$.

4) Случай $\delta_2 = \delta^*$ и $\gamma_2 > \gamma^*$ рассматривается аналогично.

При $\gamma_1 > \gamma^*$ аналогично должна была быть доопределена точка $p_5 = (\mathtt{prev}_i(\gamma_0), \delta_1) \succeq p^*$.

Таким образом, существует точка $p' \in U_i'$, $p' \succeq p^*$, такая, что значение $C_i'(p')$ доопределено значением из некоторой точки $p_2' \preceq p^*$, а значит, $C_i'(p') \leqslant C_i(p^*)$. Замечание 3 доказано.

4. Вычислительные эксперименты

Алгоритм MAS был реализован на языке Julia [21]. Все эксперименты проводились на компьютере, оснащённом процессором Intel Core i7-8700 с частотой 3,20 ГГц и имеющем 32 ГБ оперативной памяти, под управлением операционной системы Microsoft Windows 10 Pro.

4.1. Описание тестовых примеров. Тестовые примеры сгенерированы аналогично [22]. Для формирования запросов клиентов на товары применяется распределение Пуассона. Чтобы смоделировать неравномерность спроса и цен, используется процедура, предложенная в [23]. Сначала для каждого товара выбирается непрерывная случайная величина u_{di} из равномерного распределения на отрезке $[0, 1]$. Затем по формуле

$$\nu_i = \frac{\lambda}{\rho_d} u_{di}^{\frac{1-\rho_d}{\rho_d}},$$

где λ — средняя интенсивность спроса, а ρ_d — параметр скошенности для спроса, вычисляется интенсивность спроса на товар i. Затем для каждого склада вновь выбирается значение h_{dif} из равномерного распределения на отрезке $[0, 2]$. Итоговая интенсивность спроса в распределении Пуассона на товар i на складе f равна $\lambda_{if} = \nu_i h_{dif}$.

Стоимость хранения товара i на складе f вычисляется аналогично:

$$c_{if} = \frac{c}{\rho_c} u_{ci}^{\frac{1-\rho_c}{\rho_c}} h_{cif},$$

где c — средняя стоимость хранения, ρ_c — параметр скошенности для стоимости, u_{ci} и h_{cif} — реализации случайных величин из отрезков $[0, 1]$ и $[0, 2]$ соответственно. В проведённых экспериментах использованы параметры $\rho_d = 0{,}139$, $\rho_c = 0{,}097$. Кроме того, величины λ_{if} и c_{if} не могут принимать значения меньше 10^{-3}.

Времена поставок для каждого предмета и склада выбираются случайно из равномерного распределения в диапазоне от 1 до 7 дней для местных складов и от 10 до 16 для центрального склада. Для тестирования сгенерированы два семейства примеров: малые и большие. Малые примеры имеют 100 типов товаров и 10 местных складов. У товара есть вероятность 0,025 наличия спроса на каждом складе. Принимая во внимание центральный склад, получаем около 500 переменных на пример. Большие примеры имеют 1000 типов товаров, 100 местных складов и вероятность в 0,045 получения запросов с определённого склада. Это даёт около 10000 переменных. Уровни целевых показателей удовлетворённости были установлены на уровне $\alpha = 0{,}95$, $\beta = 0{,}95$.

4.2. Сравнительный анализ. Предложенный алгоритм исследован в сравнении с универсальным пакетом Nevergrad [18] и многоэшелонным оптимизатором MEO. Для Nevergrad используем оптимизатор NGOpt39 с бюджетом в $3 \cdot 10^5$ оцениваний и предоставляем ему первоначальное решение, найденное с помощью нашей эвристики для случая удовлетворения всех заказов. Если не предоставить первоначального решения, то Nevergrad потребуется гораздо больший вычислительный бюджет, чтобы показать результаты, аналогичные представленным далее. Nevergrad и MAS использовали все доступные потоки, а MEO работает только в однопоточном режиме. Кроме того, Nevergrad использует полную симуляцию, в то время как MAS и MEO используют симуляции одного товара. Для MAS установлен параметр $\varepsilon_0 = 0{,}04$. Критерием остановки является достижение $\varepsilon = 1$ и $z^t = z^{t-1}$ (п. 2.2). В локальном поиске $r = 21$, $T = \max\{10, n\}$, где n — число переменных в X_i. Алгоритм локального поиска завершается после 500 оцениваний целевой функции или когда лучшее найденное решение не меняется в течение $10T$ итераций. В алгоритме MAS для товара $i \in I$ и каждой пары (γ_i, δ_i) мы не тратим в общей

Таблица 2

Сравнение MAS, MEO и Nevergrad на малых примерах

№	$\mathrm{MAS}_{\min}$	$\mathrm{MAS}_{\mathrm{avg}}$	MEO	$\mathrm{NG}_{\min}$	$\mathrm{NG}_{\mathrm{avg}}$	t_{MAS}	t_{MEO}
1	6485	6487	6912	11 857	12 562	36	20
2	5421	5431	5581	11 536	12 181	33	18
3	5056	5061	5503	8413	8954	25	18
4	5590	5592	5868	8705	9475	20	18
5	12 230	12 258	12 870	16 993	18 540	25	24
6	7122	7125	7518	13 493	14 253	25	19
7	10 846	10 852	11 565	15 747	16 472	22	18
8	11 560	11 571	11 882	17 443	18 028	29	17
9	13 383	13 389	13 743	20 084	21 290	23	20
10	8968	8999	9356	12 999	13 291	26	24

сложности более 2000 оцениваний при решении задач $\texttt{OneItem}_i(\gamma_i, \delta_i)$. Хотя следует отметить, что изменение параметров может сместить баланс между временем вычислений и качеством результата. В табл. 2 и 3 представлены значения затрат на хранение запасов для соответствующих примеров и алгоритмов.

В табл. 2 приведены результаты для малых примеров. Для каждого примера было выполнено 5 запусков алгоритма MAS и 5 запусков

Таблица 3

Сравнение MAS, MEO и Nevergrad на больших примерах

№	$\mathrm{MAS}_{\min}$	$\mathrm{MAS}_{\mathrm{avg}}$	MAS_t	MEO	$\mathrm{NG}_{\min}$	$\mathrm{NG}_{\mathrm{avg}}$	t_{MAS}	t_{MEO}
1	133 591	133 664	135 809	140 153	209 120	215 842	1625,790	617
2	127 924	128 103	129 725	136 991	217 382	219 338	1462,747	622
3	118 993	125 220	120 654	127 671	204 631	208 680	1405,229	892
4	145 825	146 274	147 470	154 442	231 940	233 772	1792,402	892
5	125 180	125 220	126 123	134 837	210 121	213 743	1579,075	903
6	126 049	126 097	128 579	133 663	217 160	219 995	1728,637	620
7	96 095	96 160	102 472	104 676	183 820	186 964	1332,990	356
8	134 922	134 999	137 072	141 286	226 715	229 002	1466,459	879
9	127 997	128 043	133 151	137 233	222 444	224 208	1637,495	366
10	134 605	134 713	136 435	144 226	238 641	240 459	1773,003	909

Nevergrad. В столбцах «$\mathrm{MAS}_{\min}$», «$\mathrm{MAS}_{\mathrm{avg}}$», «$\mathrm{NG}_{\min}$» и «$\mathrm{NG}_{\mathrm{avg}}$» представлены наилучшие и средние результаты для этих запусков. Результаты MEO приведены в колонке «MEO». Столбцы «t_{MAS}» и «t_{MEO}» показывают среднее время выполнения в секундах за один запуск MAS и MEO.

MAS демонстрирует результаты в среднем на 6% лучше, чем MEO, за аналогичное время выполнения, а Nevergrad показывает значительно более плохие результаты, хотя его время выполнения было примерно в 100 раз больше, чем у MAS и MEO. Можно заметить, что разница между минимальными и средними значениями не велика, и алгоритм демонстрирует стабильные результаты.

В табл. 3 представлены результаты для больших примеров. Значение столбцов такое же, как и для табл. 2. Результаты, достигнутые MAS за время выполнения $t = t_{\mathrm{MEO}}$, представлены в столбце MAS_t. Время работы Nevergrad составляло примерно 4 ч на запуск.

Результаты в столбцах $\mathrm{MAS}_{\mathrm{avg}}$ и MAS_t на 5% и 4% лучше, чем соответствующие для MEO. Nevergrad не смог найти сопоставимого решения ни на одном из примеров. По сравнению с предыдущей версией алгоритма результаты улучшились в среднем на 0,6%.

Как оказалось, вспомогательная подзадача `coordinator`($\mathbf{P}$) достаточно лёгкая для точного решения пакетом целочисленного программирования. Её решение занимает менее 1 с. Отметим также, что из-за особенностей устройства имитационной модели почти всё время тратится на её работу, а время работы остальной части алгоритма незначительно. Время полной симуляции равно сумме времён, затраченных на симуляцию каждого товара.

Таблица 4

Сравнение результатов MAS с точным решением

№	$\mathrm{MAS}_{\min}$	$\mathrm{MAS}_{\max}$	$\mathrm{MAS}_{\mathrm{avg}}$	Точное значение
1	5514	5514	5514	5514
2	5539	5539	5539	5539
3	5827	5827	5827	5825
4	6456	6461	6459	6455
5	5265	5265	5265	5265
6	6402	6402	6402	6391
7	4774	4775	4775	4750
8	5175	5175	5175	5170
9	8292	8292	8292	8292
10	4208	4208	4208	4208

Таблица 5

Сравнение результатов MAS без перебора с точным решением

№	MAS_{min}	MAS_{max}	MAS_{avg}	Точное значение
1	5547	5547	5547	5514
2	5598	5610	5601	5539
3	5849	5849	5849	5825
4	6540	6576	6566	6455
5	5265	5272	5271	5265
6	6421	6421	6421	6391
7	4785	4805	4789	4750
8	5264	5175	5292	5170
9	8308	8308	8308	8292
10	4208	4208	4208	4208

4.3. Сравнение с точным решением. Также проведено сравнение решений полученных алгоритмом MAS с точными решениям на маленьких примерах, на которых возможен полный перебор в подзадачах одного товара (около 70 складов и 400 переменных в сумме). Результаты данного сравнения приведены в табл. 4.

Видно, что в 5 из 10 примеров алгоритм MAS нашёл точное решение, а в остальных случаях отклонение составило не более 0,5%.

Однако стоит заметить, что алгоритм MAS включает в себя полный перебор в тех подзадачах одного товара, в которых число вариантов мало. Если полностью отключить перебор, то получатся результаты, представленные в табл. 5.

В данном случае оптимальное решение находится только в двух примерах, а максимальное отклонение возрастает до 1,88%.

Заключение

Для задачи управления запасами в двухэшелонной системе складов разработан алгоритм декомпозиции, основанный на многоагентной схеме. Алгоритм использует задачу координатора и однотипные задачи рабочих агентов. Каждый рабочий агент оптимизирует затраты на хранение одного товара на всех складах при заданных уровнях удовлетворения заказов в обоих эшелонах. Координатор задаёт эти уровни агентам и строит решение исходной задачи из решений рабочих агентов, используя модель о многовариантном рюкзаке. Для вычисления целевой функции и ограничений рабочие агенты используют детерминированную имитационную модель. Показано, что такой подход гарантирует получение

точного решения задачи, если точно решать подзадачи для каждого товара и координатор правильно выбрал уровни удовлетворения заказов. Разработан итерационный алгоритм, который последовательно уточняет эти уровни. Приводится нижняя оценка минимума суммарных затрат, которая позволяет строить точные алгоритмы. Проведены вычислительные эксперименты на примерах большой размерности. Эксперименты показали высокую эффективность предложенного подхода и его превосходство над пакетами Nevergrad и MEO. На примерах с известным точным решением в двух случаях удалось найти оптимум, в остальных случаях отклонение от оптимума составило не более 1,9%.

Финансирование работы

Исследование выполнено в рамках гос. задания Института математики им. С. Л. Соболева (проект № FWNF–2022–0019). Дополнительных грантов на проведение или руководство этим исследованием получено не было.

Конфликт интересов

Авторы заявляют, что у них нет конфликта интересов.

Литература

1. **Wassick J. M., Agarwal A., Akiya N.** [et al.]. Addressing the operational challenges in the development, manufacture, and supply of advanced materials and performance products // Comput. Chem. Eng. 2012. V. 47. P. 157–169. DOI: 10.1016/j.compchemeng.2012.06.041.
2. **Grossmann I.** Enterprise-wide optimization: A new frontier in process systems engineering // AIChE J. 2005. V. 51, No. 7. P. 1846–1857. DOI: 10.1002/aic.10617.
3. **Zhang S., Huang K., Yuan Y.** Spare parts inventory management: A literature review // Sustainability (Switz.). 2021. V. 13, No. 5. Paper ID 2460. 23 p. DOI: 10.3390/su13052460.
4. **Lee H. L.** A multi-echelon inventory model for repairable items with emergency lateral transshipments // Manag. Sci. 1987. V. 33, No. 10. P. 1302–1316. DOI: 10.1287/mnsc.33.10.1302.
5. **Axsäter S.** Modelling emergency lateral transshipments in inventory systems // Manag. Sci. 1990. V. 36, No. 11. P. 1329–1338. DOI: 10.1287/mnsc.36.11.1329.
6. **Wong H., van Houtum G. J., Cattrysse D., Van Oudheusden D.** Simple, efficient heuristics for multi-item multi-location spare parts systems with lateral transshipments and waiting time constraints // J. Oper. Res. Soc. 2005. V. 56, No. 12. P. 1419–1430. DOI: 10.1057/palgrave.jors.2601952.
7. **Yue D., You F.** Planning and scheduling of flexible process networks under uncertainty with stochastic inventory: MINLP models and algorithm // AIChE J. 2013. V. 59, No. 5. P. 1511–1532. DOI: 10.1002/aic.13924.

8. **Zapata J. C., Pekny J., Reklaitis G. V.** Simulation-optimization in support of tactical and strategic enterprise decisions // Planning production and inventories in the extended enterprise. A state of the art handbook. V. 1. New York: Springer, 2011. P. 593–627. (Int. Ser. Oper. Res. Manag. Sci.; V. 151). DOI: 10.1007/978-1-4419-6485-4_20.
9. **Peidro D., Mula J., Poler R., Lario F. C.** Quantitative models for supply chain planning under uncertainty // Int. J. Adv. Manuf. Technol. 2009. V. 43. P. 400–420. DOI: 10.1007/s00170-008-1715-y.
10. **Köchel P., Nieländer U.** Simulation-based optimisation of multi-echelon inventory systems // Int. J. Prod. Econ. 2005. V. 93–94. P. 505–513. DOI: 10.1016/j.ijpe.2004.06.046.
11. **Chu Y., You F., Wassick J. M., Agarwal A.** Simulation-based optimization framework for multi-echelon inventory systems under uncertainty // Comput. Chem. Eng. 2015. V. 73. P. 1–16. DOI: 10.1016/j.compchemeng.2014.10.008.
12. **Mele F. D., Guillén G., Espuña A, Puigjaner L.** A simulation-based optimization framework for parameter optimization of supply-chain networks // Ind. Eng. Chem. Res. 2006. V. 45, No. 9. P. 3133–3148. DOI: 10.1021/ie051121g.
13. **Moncayo-Martínez L. A., Zhang D. Z.** Multi-objective ant colony optimisation: A meta-heuristic approach to supply chain design // Int. J. Prod. Econ. 2011. V. 131, No. 1. P. 407–420. DOI: 10.1016/j.ijpe.2010.11.026.
14. **Nikolopoulou A., Ierapetritou M. G.** Hybrid simulation based optimization approach for supply chain management // Comput. Chem. Eng. 2012. V. 47. P. 183–193. DOI: 10.1016/j.compchemeng.2012.06.045.
15. **Blum C., Puchinger J., Raidl G. R., Roli A.** Hybrid metaheuristics in combinatorial optimization: A survey // Appl. Soft Comput. 2011. V. 11, No. 6. P. 4135–4151. DOI: 10.1016/j.asoc.2011.02.032.
16. **Noordhoek M., Dullaert W., Lai D. S. W., de Leeuw S.** A simulation–optimization approach for a service-constrained multi-echelon distribution network // Transp. Res. Part E: Logist. Transp. Rev. 2018. V. 114. P. 292–311. DOI: 10.1016/j.tre.2018.02.006.
17. **Chen H., Dai B., Li Y.** [et al.]. Stock allocation in a two-echelon distribution system controlled by (s, S) policies // Int. J. Prod. Res. 2022. V. 60, No. 3. P. 894–911. DOI: 10.1080/00207543.2020.1845915.
18. **Rapin J., Teytaud O.** Nevergrad — A gradient-free optimization platform. 2018. URL: GitHub.com/FacebookResearch/Nevergrad (accessed: 24.01.2024).
19. **Yuskov A. D., Kulachenko I. N., Melnikov A. A., Kochetov Yu. A.** Decomposition approach for simulation-based optimization of inventory management // Mathematical optimization theory and operations research: Recent trends. Rev. Sel. Pap. 22nd Int. Conf. (Yekaterinburg, Russia, July 2–8, 2023). Cham: Springer, 2023. P. 259–273. (Commun. Comput. Inf. Sci.; V. 1881). DOI: 10.1007/978-3-031-43257-6_20.

20. Bajaj I., Arora A., Hasan M. M. F. Black-box optimization: Methods and applications // Black box optimization, machine learning, and no-free lunch theorems. Cham: Springer, 2021. P. 35–65. (Springer Optim. Its Appl.; V. 170). DOI: 10.1007/978-3-030-66515-9_2.

21. Bezanson J., Edelman A., Karpinski S., Shah V. B. Julia: A fresh approach to numerical computing // SIAM Rev. 2017. V. 59, No. 1. P. 65–98. DOI: 10.1137/141000671.

22. Topan E., Bayındır Z. P., Tan T. Heuristics for multi-item two-echelon spare parts inventory control subject to aggregate and individual service measures // Eur. J. Oper. Res. 2017. V. 256. P. 126–138. DOI: 10.1016/j.ejor.2016.06.012.

23. Thonemann U. W., Brown A. O., Hausman W. H. Easy quantification of improved spare parts inventory policies // Manag. Sci. 2002. V. 48, No. 9. P. 1213–1225. DOI: 10.1287/mnsc.48.9.1213.173.

Юськов Александр Дмитриевич
Кулаченко Игорь Николаевич
Мельников Андрей Андреевич
Кочетов Юрий Андреевич

Статья поступила
25 января 2024 г.
После доработки —
10 марта 2024 г.
Принята к публикации
22 июня 2024 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
October–December 2024. Vol. 31, No. 4. P. 186–212

UDC 519.8 DOI: 10.33048/daio.2024.31.794

DECOMPOSITION APPROACH FOR A TWO ECHELON INVENTORY MANAGEMENT SYSTEM

A. D. Yuskov[1,a], *I. N. Kulachenko*[2,b], *A. A. Melnikov*[2,c], and *Yu. A. Kochetov*[2,d]

[1] Novosibirsk State University,
2 Pirogov Street, 630090 Novosibirsk, Russia

[2] Sobolev Institute of Mathematics,
4 Acad. Koptyug Avenue, 630090 Novosibrsk, Russia

E-mail: [a] a.yuskov@g.nsu.ru, [b] ink@math.nsc.ru,
[c] melnikov@math.nsc.ru, [d] jkochet@math.nsc.ru

Abstract. Warehouses of the first echelon in a two-echelon system are designed to satisfy customer orders. In the second echelon, we have a central warehouse for restocking the first echelon warehouses. Customer orders can be partially satisfied, but the total fraction of completed orders should not be less than the specified threshold. We need to minimize the total cost of storing the items in all warehouses. We use a deterministic simulation to calculate the order satisfaction ratio and the storage cost during the planning period. The simulation depends on inventory management policies at each warehouse for each type of items. We develop a decomposition method for solving the problem. It is based on solution of subproblems for each type of items. Also, we propose some approaches for exact solution of the problem. The results of numerical experiments with instances with 100 warehouses and 1000 types of items are presented. On instances with known exact solutions, we have the optimum in two cases, while in the other cases the deviation from the optimal values is at most 1.9%. Tab. 5, illustr. 1, bibliogr. 23.

Keywords: gray-box optimization, knapsack problem, local search.

English transl.: *Journal of Applied and Industrial Mathematics* **18** (4), 919–935 (2024), DOI: 10.1134/S1990478924040239.

References

1. **J. M. Wassick, A. Agarwal, N. Akiya,** [et al.]. Addressing the operational challenges in the development, manufacture, and supply of advanced materials and performance products, *Comput. Chem. Eng.* **47**, 157–169 (2012), DOI: 10.1016/j.compchemeng.2012.06.041.
2. **I. Grossmann,** Enterprise-wide optimization: A new frontier in process systems engineering, *AIChE J.* **51** (7), 1846–1857 (2005), DOI: 10.1002/aic.10617.
3. **S. Zhang, K. Huang,** and **Y. Yuan,** Spare parts inventory management: A literature review, *Sustainability (Switz.)* **13** (5), ID 2460 (2021), DOI: 10.3390/su13052460.
4. **H. L. Lee,** A multi-echelon inventory model for repairable items with emergency lateral transshipments, *Manag. Sci.* **33** (10), 1302–1316 (1987), DOI: 10.1287/mnsc.33.10.1302.
5. **S. Axsäter,** Modelling emergency lateral transshipments in inventory systems, *Manag. Sci.* **36** (11), 1329–1338 (1990), DOI: 10.1287/mnsc.36.11.1329.
6. **H. Wong, G. J. van Houtum, D. Cattrysse,** and **D. Van Oudheusden,** Simple, efficient heuristics for multi-item multi-location spare parts systems with lateral transshipments and waiting time constraints, *J. Oper. Res. Soc.* **56** (12), 1419–1430 (2005), DOI: 10.1057/palgrave.jors.2601952.
7. **D. Yue** and **F. You,** Planning and scheduling of flexible process networks under uncertainty with stochastic inventory: MINLP models and algorithm, *AIChE J.* **59** (5), 1511–1532 (2013), DOI: 10.1002/aic.13924.
8. **J. C. Zapata, J. Pekny,** and **G. V. Reklaitis,** Simulation-optimization in support of tactical and strategic enterprise decisions, in *Planning Production and Inventories in the Extended Enterprise. A State of the Art Handbook,* Vol. 1 (Springer, New York, 2011), pp. 593–627 (Int. Ser. Oper. Res. Manag. Sci., Vol. 151), DOI: 10.1007/978-1-4419-6485-4_20.
9. **D. Peidro, J. Mula, R. Poler, F. C. Lario,** Quantitative models for supply chain planning under uncertainty, *Int. J. Adv. Manuf. Technol.* **43**, 400–420 (2009), DOI: 10.1007/s00170-008-1715-y.
10. **P. Köchel** and **U. Nieländer,** Simulation-based optimisation of multi-echelon inventory systems, *Int. J. Prod. Econ.* **93–94**, 505–513 (2005), DOI: 10.1016/j.ijpe.2004.06.046.
11. **Y. Chu, F. You, J. M. Wassick,** and **A. Agarwal,** Simulation-based optimization framework for multi-echelon inventory systems under uncertainty, *Comput. Chem. Eng.* **73**, 1–16 (2015), DOI: 10.1016/j.compchemeng.2014.10.008.
12. **F. D. Mele, G. Guillén, Espuña A** and **L. Puigjaner,** A simulation-based optimization framework for parameter optimization of supply-chain networks, *Ind. Eng. Chem. Res.* **45** (9), 3133–3148 (2006), DOI: 10.1021/ie051121g.
13. **L. A. Moncayo-Martínez** and **D. Z. Zhang,** Multi-objective ant colony optimisation: A meta-heuristic approach to supply chain design, *Int. J. Prod. Econ.* **131** (1), 407–420 (2011), DOI: 10.1016/j.ijpe.2010.11.026.

14. **A. Nikolopoulou** and **M. G. Ierapetritou,** Hybrid simulation based optimization approach for supply chain management, *Comput. Chem. Eng.* **47**, 183–193 (2012), DOI: 10.1016/j.compchemeng.2012.06.045.
15. **C. Blum, J. Puchinger, G. R. Raidl,** and **A. Roli,** Hybrid metaheuristics in combinatorial optimization: A survey, *Appl. Soft Comput.* **11** (6), 4135–4151 (2011), DOI: 10.1016/j.asoc.2011.02.032.
16. **M. Noordhoek, W. Dullaert, D. S. W. Lai,** and **S. de Leeuw,** A simulation–optimization approach for a service-constrained multi-echelon distribution network, *Transp. Res., Part E: Logist. Transp. Rev.* **114**, 292–311 (2018), DOI: 10.1016/j.tre.2018.02.006.
17. **H. Chen, B. Dai, Y. Li,** [et al.]. Stock allocation in a two-echelon distribution system controlled by (s, S) policies, *Int. J. Prod. Res.* **60** (3), 894–911 (2022), DOI: 10.1080/00207543.2020.1845915.
18. **J. Rapin** and **O. Teytaud,** Nevergrad — A gradient-free optimization platform. 2018. URL: GitHub.com/FacebookResearch/Nevergrad (accessed: 24.01.2024).
19. **A. D. Yuskov, I. N. Kulachenko, A. A. Melnikov,** and **Yu. A. Kochetov,** Decomposition approach for simulation-based optimization of inventory management, in *Mathematical Optimization Theory and Operations Research: Recent Trends* (Rev. Sel. Pap. 22nd Int. Conf., Yekaterinburg, Russia, July 2–8, 2023) (Springer, Cham, 2023), pp. 259–273 (Commun. Comput. Inf. Sci., Vol. 1881), DOI: 10.1007/978-3-031-43257-6_20.
20. **I. Bajaj, A. Arora,** and **M. M. F. Hasan,** Black-box optimization: Methods and applications, in *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems* (Springer, Cham, 2021), pp. 35–65 (Springer Optim. Its Appl., Vol. 170), DOI: 10.1007/978-3-030-66515-9_2.
21. **J. Bezanson, A. Edelman, S. Karpinski,** and **V. B. Shah,** Julia: A fresh approach to numerical computing, *SIAM Rev.* **59** (1), 65–98 (2017), DOI: 10.1137/141000671.
22. **E. Topan, Z. P. Bayındır** and **T. Tan,** Heuristics for multi-item two-echelon spare parts inventory control subject to aggregate and individual service measures, *Eur. J. Oper. Res.* **256**, 126–138 (2017), DOI: 10.1016/j.ejor.2016.06.012.
23. **U. W. Thonemann, A. O. Brown,** and **W. H. Hausman,** Easy quantification of improved spare parts inventory policies, *Manag. Sci.* **48** (9), 1213–1225 (2002), DOI: 10.1287/mnsc.48.9.1213.173.

Aleksandr D. Yuskov
Igor N. Kulachenko
Andrey A. Melnikov
Yury A. Kochetov

Received January 25, 2024
Revised March 10, 2024
Accepted June 22, 2024