|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет**  **имени Н.Э. Баумана (национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | *Робототехника и комплексная автоматизация* |
|  |  |
| КАФЕДРА | *Системы автоматизированного проектирования (РК-6)* |

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ***

***«Задача линейного упорядочивания: области применения и способы решения»***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | РК6-41М | |  |  | А. С. Антонов |
|  | (Группа) | |  | (подпись, дата) | (инициалы и фамилия) |
|  | |  |  |  |  |
| Руководитель | | |  |  | А. Н. Божко |
|  | |  |  | (подпись, дата) | (инициалы и фамилия) |
|  | |  |  |  |  |
|  | |  |  |  |  |

*Москва, 2025 г*

Оглавление

[1. ВВЕДЕНИЕ 3](#_Toc197949471)

[2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_Toc197949472)

[3. МЕТОДЫ РЕШЕНИЯ 9](#_Toc197949473)

[3.1. Метод границ и ветвей с LP-релаксацией 10](#_Toc197949474)

[3.2. Метод ветвей и отсечений 11](#_Toc197949475)

[3.3. Комбинированный алгоритм Митчелла и Борчерса 11](#_Toc197949476)

[3.4. Алгоритм Беккера 11](#_Toc197949477)

[3.5. Алгоритм локального поиска 12](#_Toc197949478)

[3.6. Алгоритм поиска с запретами 12](#_Toc197949479)

[3.7. Алгоритм поиска с рассеиванием 12](#_Toc197949480)

[3.8. Итеративный локальный поиск 12](#_Toc197949481)

[4. ОБЛАСТИ ПРИМЕНЕНИЯ 12](#_Toc197949482)

[5. ЗАКЛЮЧЕНИЕ 12](#_Toc197949483)

[6. СПИСОК ЛИТЕРАТУРЫ 13](#_Toc197949484)

1. ВВЕДЕНИЕ

Задача линейного упорядочения (Linear Ordering Problem, LOP) занимает центральное место в комбинаторной оптимизации благодаря своей универсальности и широкому спектру приложений. В условиях роста объема данных [1] и потребности в эффективном управлении сложными системами LOP становится инструментом для решения задач ранжирования, планирования и анализа зависимостей.

Гэри и Джонсон (1979) продемонстрировали, что LOP является NP-сложной задачей. Однако, благодаря её многочисленным применениям в различных областях, таких как археология (Гловер и др., 1972), экономика (Леонтьев, 2008), теория графов (Харон и Худри, 2007), машинный перевод (Тромб и Эйснер, 2009) мы можем найти большое количество работ, в которых бы LOP решался с помощью точных, эвристических и метаэвристических методов.

В экономике LOP используется для ранжирования инвестиционных проектов на основе их потенциальной доходности и рисков. В машинном обучении задача помогает строить консенсусные рейтинги в рекомендательных системах [2]. В спортивной аналитике с её помощью определяют рейтинги команд, учитывая исторические результаты матчей [3]. В биоинформатике LOP применяется для упорядочения геномных последовательностей, что критично для понимания эволюционных процессов [4]. В археологии LOP нашёл применение в задаче стратификации для определения наиболее вероятного хронологического порядка образцов, найденных в разных местах. Матрица, описывающая эту проблему, известна как матрица Харриса.

Рост интереса к Big Data и искусственному интеллекту усиливает потребность в алгоритмах, способных работать с высокоразмерными данными, что делает исследование LOP особенно актуальным.

Цель работы состоит в изучении теоретических основ задачи линейного упорядочения, исследовании областей применения и проведении сравнительного анализа методов её решения.

Работа включает в себя пять разделов. В теоретической части раскрывается постановка задачи и её связь с другими проблемами оптимизации. Далее анализируются методы решения, а в практическом разделе рассматриваются области применения данной задачи.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Для заданной матрицы размера задача линейного упорядочения (LOP), формулируется, как задача нахождения перестановки строк и столбцов , которая бы максимизировала функцию в формуле 1.

где обозначает индекс строки (и столбца), занимающей позицию в решении . Иными словами, цель заключается в нахождении такой перестановки строк и столбцов матрицы , которая бы максимизировала сумму элементов, находящихся выше главной диагонали. В данной постановке задача линейного упорядочивания известна, как triangulation problem of input-output matrices. Тем не менее существуют и альтернативные варианты представления LOP. Так Марти и Рейнельт в своей книге 2011-го года «The linear ordering problem: exact and heuristic methods in combinatorial optimization» дают интерпретацию в терминах теории графов с поиском ациклического турнира (полный ориентированный ациклический подграф), максимизирующего сумму весов дуг полного ориентированного графа.

Сами значения отображают силу предпочтения -го элемента -ому.

Рассмотрим пример для , который будет использоваться в дальнейшем. На рисунке 1 представлены три различные решения: и . Исходная матрица представлена перестановкой (рис. 1а) и значением её фитнес функции , равным 138. Решение (рис. 1b) иллюстрирует другой вариант решения, имеющий более оптимальное значение фитнеса . Наилучшее возможное решение представлено перестановкой (рис. 1c). с фитнесом .

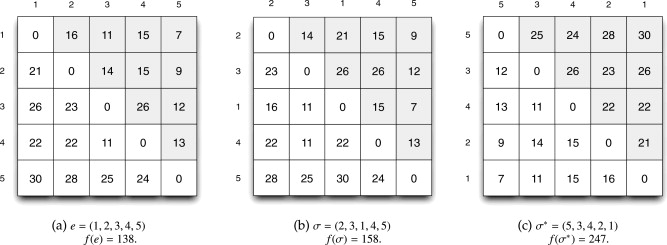


Рис. 1 – Три различных решения для матрицы : (а) исходная матрица; (b) неоптимальное решение; (c) оптимальное решение.

В ходе анализа проблемы было выявлено, что для любой перестановки индексов в матрице размером справедливо следующее:

* С каждым индексом связано записей из матрицы : из строки и из столбца .
* Набор связанных записей каждого индекса может быть сгруппирован в пары. Например, каждой записи из строки , , соответствует точка из столбца , , симметрично расположенная относительно главной диагонали.
* Все пары записей, связанные с индексом , остаются связанными с ним даже после перестановок.
* Каждая запись связана с двумя индексами, .
* Для каждой пары одна из записей всегда расположена выше главной диагонали, а другая ниже.

Утверждения проиллюстрированы примером на рисунке 2. В этом примере представлены два различных решения: на рис. 2a и на рис. 2b. На обоих рисунках записи, связанные с индексом 2, выделены жирным шрифтом. Мы видим, что, несмотря на разное упорядочивание, в обоих решениях набор записей, связанных с индексом 2, одинаков, то есть (21, 14, 15, 9, 16, 23, 22, 28). Несмотря на то, что позиция индекса 2 в различна , попарное отношение связанных с ним записей остается неизменным (см. обведенные индексы на рис. 2).

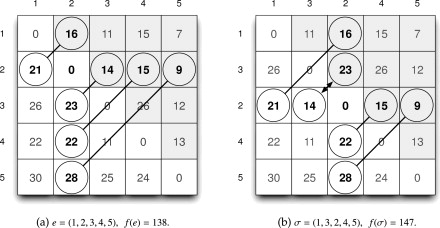


Рис. 2 – Два различных решения для экземпляра . Обведенные записи, соединенные ребрами, обозначают пары записей, связанные с индексом 2: (a) исходное состояние; (b) после перестановки.

Проверяя расположение связанных записей индекса 2 в , мы замечаем, что пара (14, 23) поменяла свои позиции в , 14 теперь ниже главной диагонали, а 23 выше неё. Введём новое понятие: вклад индекса в фитнес-функцию.

Когда индекс занимает позицию в , то есть , вклад индекса в функцию определяется суммой записей столбца в строках и суммой записей строки в столбцах . Иными словами, предыдущие индексы и последующие индексы определяют вклад индекса k в фитнес-функцию. Формальное описание представлено формулой 2.

Вернемся к примеру на рисунке 2, благодаря обмену позициями в пару (14, 23) вклад индекса 2 изменился с 54 (16 + 14 + 15 + 9) в (рис. 2a) до 63 (16 + 23 + 15 + 9) в (рис. 2b). В случае индекса 3 его вклад также увеличился, поскольку пара (14, 23) связана с обоими индексами, 2 и 3. И наоборот, в случае индексов 1, 4 и 5 их вклад не меняется от к .

Если мы внимательно посмотрим на уравнение (2), то поймем, что вклад индекса на самом деле не определяется конкретным упорядочиванием индексов в предыдущей и последующей позициях , но их группировкой в этих двух наборах позиций. Как показано в примере 2, вклад индексов 1, 4 и 5 не меняется от к , поскольку группировка остальных индексов в предыдущем и последующем наборах позиций, связанных с индексами 1, 4 и 5, была одинаковой.

Следовательно, при заданном решении вклад индекса , в фитнес-функцию не зависит от порядка предыдущих индексов и от порядка последующих индексов .

Пример из рисунка 3 иллюстрирует, как вклад индекса 3 не зависит от упорядочивания предыдущего и последующего наборов индексов. На рис. 3а вклад индекса 3 равен 63, как результат суммы (11 + 14 + 26 + 12). Если мы проверим вклад индекса 3 в (см. рис. 3б), то увидим, что он также равен 63, несмотря на то что индексы {1, 2} и {4, 5} поменялись местами.

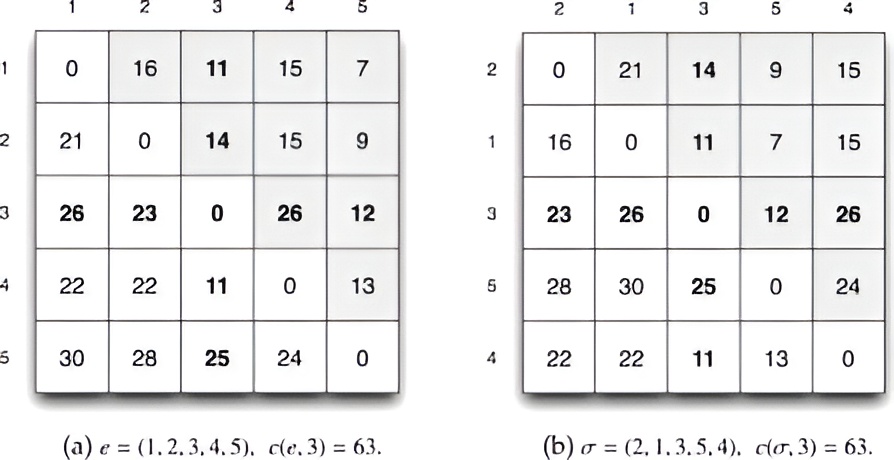


Рис. 3 – Эффект от замены индексов в позициях 1,2 и 4,5 для индекса 3: (a) исходное состояние; (b) после перестановки.

Как было сказано выше, вклад индекса не зависит от порядка следования индексов в предыдущих и последующих множествах. Но что произойдет, если индекс будет перемещен из предыдущего набора индексов в последующий набор индексов? В отличие от предыдущего случая, вклад индекса будет изменён. В этот момент стоит вспомнить, что каждая пара записей в матрице связана с двумя индексами, , а значит, любой обмен местоположением по определению влияет на вклад в функцию приспособленности и . Фактически, перемещение в позицию влияет на вклад всех индексов, расположенных между позициями и . Приведенный ниже пример (рисунок 4) иллюстрирует изменения в фитнес-функции, вызванные перемещением индекса.

На рисунке 4а и b показана матрица в соответствии с решениями . В этом примере мы анализируем последствия перемещения индекса в позицию 4. В результате этой модификации индексы 3 и 4 сдвигаются на одну позицию влево, что изменяет их вклад в фитнес-функцию. В частности, мы видим, что пары {14, 23} и {15, 22}, связанные с индексами 2-3 и 2-4, поменялись местами. Поэтому вклад индекса 3 меняется с 63 (11 + 14 + 26 + 12) на 72 (11 + 26 + 23 + 12). Аналогично, вклад индекса 4 меняется с 69 (15 + 15 + 26 + 13) до 76 (15 + 26 + 22 + 13). Что касается индекса 2, то его вклад также меняется с 54 (16 + 14 + 15 + 9) до 70 (16 + 23 + 22 + 9). Обратите внимание, что вариация фитнес-вклада индекса 2 равна сумме вариаций индексов 3 и 4.

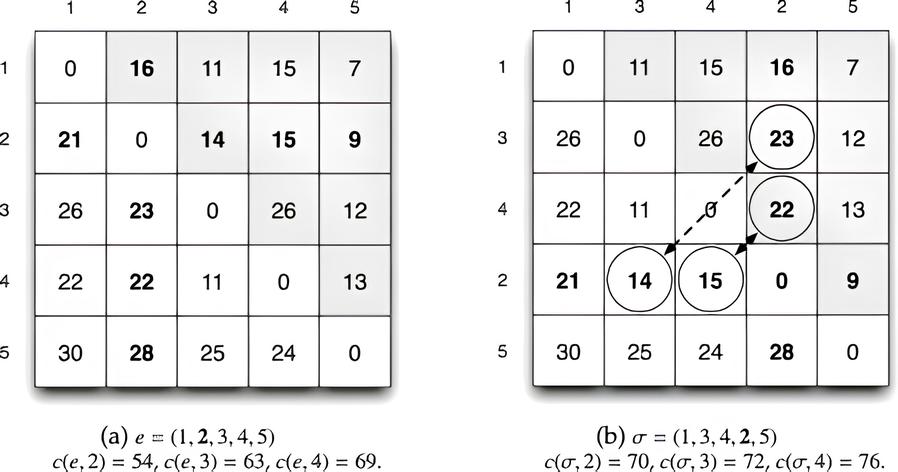


Рис. 4 – Иллюстрация влияния перемещения индекса 2 (из позиции 2 в позицию 4) на вклад индексов 2, 3 и 4 в фитнес-функцию. Числа, выделенные жирным шрифтом, обозначают записи, связанные с индексом 2. Обведенные кружком пары записей выделяют замененные записи: (a) исходное состояние; (b) после перестановки.

1. МЕТОДЫ РЕШЕНИЯ

LOP - это NP-полная задача, то есть мы не можем ожидать алгоритма, который решит ее за полиномиальное время. Однако LOP имеет множеств практических приложений [2, 3, 4], и поэтому алгоритмы для его эффективного решения востребованы. Было предложено несколько точных и эвристических алгоритмов. Точные алгоритмы включают в себя метод границ и ветвей(B&B), использующий LP-релаксацию для нижней границы, предложенный Каасом (A branch and bound algorithm for the acyclic subgraph problem), алгоритм ветвей и отсечений(B&C), предложенный Грёцшелем, Юнгером и Рейнелтом (A cutting plane algorithm for the linear ordering problem) и комбинированный алгоритм Митчелла и Борчерса (Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm). Современные точные алгоритмы могут решать достаточно большие задачи из определенных классов задач с числом столбцов и строк до нескольких сотен, в то время как на экземплярах из других классов гораздо меньшего размера они терпят неудачу. Независимо от типа решаемых задача, время вычислений точных алгоритмов сильно увеличивается с ростом размера матрицы.

LOP также решается с помощью ряда эвристических алгоритмов. К ним относятся жадный алгоритм Беккера, алгоритмы локального поиска, а также множество метаэвристических подходов, включая поиск с запретами (tabu search), scatter search и алгоритмы итеративного локального поиска (ILS). В частности, подходы на основе ILS в настоящее время считаются наиболее успешными метаэвристиками.

Существующие алгоритмы обычно тестировались на множестве классов реальных и случайно сгенерированных примеров. Однако до сих пор мало известно о том, как производительность современных алгоритмов зависит от специфических характеристик различных доступных классов задач LOP, а также о том, как различия между примерами влияют на особенности их пространства поиска. Первые шаги в решении этих открытых вопросов были сделаны в работах Скьявинотто и Шутцле от 2003-го и 2004-го года.

* 1. Метод границ и ветвей с LP-релаксацией

В данном методе задача формулируется в терминах целочисленного линейного программирования. Конкретная формулировка представлена формулой 3.

где если элемент предшествует , и 0 иначе. При этом накладывается ограничение на сумму парных элементов: , и на транзитивность: .

LP-релаксация переводит целочисленные переменные в непрерывные . Решение релаксированной задачи даёт верхнюю границу целевой функции. Если решение релаксации целочисленное и транзитивное, то оно оптимальное.

В иных случаях выбирается переменная значение которой в LP-решении ближе всего к 0.5 (наиболее неопределённая). Далее создаются две подзадачи или узла:

* Ветвь 1: ;
* Ветвь 2:

Подзадачи формируют список . Для каждого узла находится решение и происходит проверка на соответствие ограничениям. Ветви нарушающие ограничения транзитивности отсекаются. Для корректных узлов вычисляется верхняя граница, если она меньше текущего лучшего целочисленного решения, то её тоже необходимо отсечь.

Алгоритм:

1. Инициализация: корневой узел с LP-релаксацией.
2. Пока есть неисследованные узлы:
   1. Выбрать узел с наивысшей верхней границей.
   2. Решить LP-релаксацию.
      1. Если решение целочисленное и транзитивное → обновить лучшее решение.
      2. Иначе:
         1. Выбрать переменную для ветвления.
         2. Создать подзадачи, добавив новые ограничения.
         3. Добавить подзадачи в очередь узлов.

В ходе работы алгоритма могут быть исследованы все узлы. Поэтому он обладает экспоненциальной сложностью.

* 1. Метод ветвей и отсечений

Метод ветвей и отсечений является улучшением метода ветвей и границ. B&C обнаруживает циклы и добавляет их в набор ограничений что сужает пространство решений. Данная модификация особенно актуальна для задач с большим числом элементов.

Алгоритм:

1. Инициализация: корневой узел с LP-релаксацией.
2. Пока есть неисследованные узлы:
   1. Выбрать узел с наивысшей верхней границей.
   2. Решить LP-релаксацию.
      1. Если решение целочисленное и транзитивное → обновить лучшее решение.
      2. Иначе:
         1. Сгенерировать отсечения для устранения недопустимых решений.
         2. Если отсечения улучшили верхнюю границу → пересчитать LP.
         3. Если решение всё ещё нецелочисленное → выполнить ветвление.
         4. Добавить подзадачи в очередь узлов.

Данный метод сложнее оригинального в реализации, но в то же время и быстрее. Поэтому для задач с большим числом записей следует использовать модификацию. Более подробное сравнение представлено в таблице 1.

* 1. Комбинированный алгоритм Митчелла и Борчерса

Этот метод объединяет метод внутренней точки (Interior Point Method, IPM) и симплекс-метод в рамках алгоритма отсечений (Cutting Plane Algorithm) для решения задач линейного упорядочения (LOP). Его ключевая идея — использовать сильные стороны обоих методов:

* IPM — быстро находит приближённые решения на начальных этапах;
* Симплекс-метод — эффективно уточняет решение и генерирует отсечения.

В данном алгоритме существует три фазы:

1. Фаза внутренней точки: нахождение решения, приближённого к оптимальному. Решается LP-релаксация LOP с помощью IPM за счёт высокой эффективности для задач большой размерности;
2. Фаза генерации отсечений: устранение нарушений транзитивности, цикличности и нецелочисленных решений.
3. Фаза симплекс-метода: уточнение решения после добавления отсечений. Высокая эффективность для локального поиска точных решений в суженном пространстве.

Фазы 2 и 3 повторяются, пока не будет найдено решение.

Преимущества подхода:

1. Скорость и точность:
   1. IPM быстро приближается к оптимуму.
   2. Симплекс-метод точно обрабатывает добавленные отсечения.
2. Сокращение итераций:
   1. Динамические отсечения сужают пространство решений, уменьшая число ветвей.
3. Устойчивость к размерности:
   1. IPM хорошо работает для задач с тысячами переменных.

Комбинированный алгоритм Митчелла и Борчерса для решения LOP объединяет скорость метода внутренних точек (IPM) и точность симплекс-метода, дополненных динамическими отсечениями. IPM быстро приближается к оптимуму, решая LP-релаксацию, после чего симплекс-метод уточняет решение, добавляя отсечения для устранения циклов и нарушений транзитивности. Это позволяет эффективно решать крупномасштабные задачи линейного упорядочения, сокращая число итераций и вычислительные ресурсы за счёт гибридного подхода, который балансирует между скоростью IPM и точностью симплекса.

Сравнение точных методов представлено в таблице 1.

Таблица 1 –

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Теоретическая сложность** | **Практическая эффективность** | **Оптима для** |
| B&B с LP |  | Низкая | Малые задачи (n < 50) |
| B&C |  | Средняя | Средние задачи (n < 200) |
| Митчелл-Борчерс | + экспоненциальная (Симплекс) | Высокая | Крупные задачи (n > 200) |

* 1. Алгоритм Беккера

На первом шаге выбирается индекс, максимизирующий стоимость, представленную формулой 4.

Он помещается на первую позицию в перестановке. Затем этот индекс вместе с соответствующими столбцом и строкой удаляется, и из полученной подматрицы. Вычисляются новые значения для оставшихся индексов. Эти шаги повторяются до тех пор, пока список индексов не станет пустым, что приводит к вычислительным затратам A простая вариация этого алгоритма - вычислить значения только один раз в начале работы алгоритма, отсортировать эти значения в неувеличивающемся порядке, чтобы получить перестановку индексов. При использовании этого варианта решение может быть вычислено за

* 1. Алгоритм локального поиска

**Инициализация начального решения**  
Начните с произвольной перестановки элементов (например, случайной или жадной).

**Вычисление целевой функции**  
Для текущей перестановки π*π* вычислите сумму весов:

S=∑i=1n−1∑j=i+1nW[π(i),π(j)]*S*=*i*=1∑*n*−1​*j*=*i*+1∑*n*​*W*[*π*(*i*),*π*(*j*)]

**Определение окрестности**  
Соседние решения генерируются с помощью операции **вставки** элемента на новую позицию. Для элемента на позиции i*i* рассматриваются все возможные позиции j≠i*j*=*i*.

**Вычисление изменения целевой функции (дельта)**  
При перемещении элемента x=π(i)*x*=*π*(*i*) в позицию j*j*:

Если j<i*j*<*i*:

Δ=∑k=ji−1(W[x,π(k)]−W[π(k),x])Δ=*k*=*j*∑*i*−1​(*W*[*x*,*π*(*k*)]−*W*[*π*(*k*),*x*])

Если j>i*j*>*i*:

Δ=∑k=i+1j(W[π(k),x]−W[x,π(k)])Δ=*k*=*i*+1∑*j*​(*W*[*π*(*k*),*x*]−*W*[*x*,*π*(*k*)])

**Поиск улучшения**

Для каждого элемента x*x* найдите позицию j*j*, дающую максимальное ΔΔ.

Выберите перемещение с наибольшим Δ>0Δ>0.

**Обновление решения**  
Переместите элемент на новую позицию, обновите перестановку π*π* и сумму S*S*.

**Критерий остановки**  
Если улучшений нет (все Δ≤0Δ≤0), алгоритм завершается. Текущее решение — локальный оптимум.

* 1. Алгоритм поиска с запретами
  2. Алгоритм поиска с рассеиванием
  3. Итеративный локальный поиск

1. ОБЛАСТИ ПРИМЕНЕНИЯ
2. ЗАКЛЮЧЕНИЕ
3. СПИСОК ЛИТЕРАТУРЫ
4. Статья с сайта: Fabio Duarte Amount of Data Created Daily (2025) [электронный ресурс] // explodingtopics URL: <https://explodingtopics.com/blog/data-generated-per-day> (дата обращения: 06.05.2025)
5. Aparicio J., Landete M., Monge J. F. A linear ordering problem of sets //Annals of Operations Research. – 2020. – Т. 288. – №. 1. – С. 45-64.
6. Cameron T. R., Charmot S., Pulaj J. On the linear ordering problem and the rankability of data //arXiv preprint arXiv:2104.05816. – 2021.
7. Alcaraz J. et al. The linear ordering problem with clusters: a new partial ranking //Top. – 2020. – Т. 28. – С. 646-671.