# Введение

В данной работе мы предлагаем линейный метод superiorization в качестве инструмента для обработки задач линейного программирования. Метод линейной superiorization не гарантирует нахождение точки минимума линейной задачи оптимизации, но он регулирует работу линейных алгоритмов, которые ищут точки с наименьшим значением целевой функции. Этот алгоритм применим не только к задаче поиска минимума функции в задаче линейного программирования, но и в задачах с большими размерами, он является хорошей альтернативой симплекс-метода линейного программирования, с которым мы и будем сравнивать его здесь.

Эта работа опирается на более ранние теоретические работы о superiorization, которые включены в список используемых источников, в частности, [18]. В качестве используемых инструментов используются только экспериментальные расчеты. Несмотря на это, мы не обсуждаем вычислительные проблемы сами по себе, а используем вычисления в качестве инструмента для практической проверки, в стиле "доказательство концепции".

**Что такое superiorization?** Многие ограниченные методы оптимизации основаны на методах безусловной оптимизации, которые приспособлены для борьбы с ограничениями. Так, например, градиентные методы, в которых безусловная минимизация внутреннего шага "управляет" процессом и проекцией в допустимом множестве и выполняется после каждого шага минимизации. Эта проекция на ограничения внутри метода приводит к необходимости решения нетривиальной задачи оптимизации и необходимость ее решения в каждой итерации препятствует работе методов градиента и ограничивает их эффективность только на допустимых множествах. Барьерные или штрафные методы аналогичным образом основаны на безусловной оптимизации вместе с различными "дополнениями", которые гарантируют, что ограничения выполнятся. Методы регуляризации встраивают ограничения в целевую функцию и продолжают работать как методы решения без ограничения уже для новой целевой функции.

В отличие от этих подходов, методы superiorization можно рассматривать как противоположную точку зрения. Вместо того, чтобы адаптировать алгоритмы минимизации без ограничений для обработки ограничений, адаптируются сами алгоритмы для осуществления поисках минимума целевой функции. Это делается с сохранением исходного алгоритма, и не требует больших вычислений

**Польза подхода.** Полезность такого подхода зависит от двух особенностей: (i) **Вычислительной**: исходный алгоритм решения логической задачи менее требователен, чем поиск точки ограниченной минимизации в допустимом множестве. Таким образом, рассматриваемый метод позволяет эффективно модифицировать исходные алгоритмы с помощью недорогих дополнений, что хорошо работает на практике. (ii) **Прикладной**: в некоторых существенных реальных приложениях выбор целевой функции экзогенными для моделирования и сбора данных приводит к возникновению ограничений. В таких ситуациях, подобный выбор целевой функции часто приводит к тому, что нет никакой необходимости, ни оправдания, чтобы искать точку минимума в этих ограничениях. Для получения «правильных результатов», часто бывает достаточно найти допустимую точку, которая уменьшает значение целевой функции (обязательно точку минимума). В некоторых научных приложениях, операции с целевыми функциями являются затратными и играют центральную роль в модели, но в других приведенные выше рассуждения могут по-прежнему применяться.

**Текущее исследование.** Текущую работу по superiorization можно оценить из материалов на странице в сети Интернет [10]. В частности, [26] и [9] отзывы об интересах. Недавние исследования включает в себя множество отчетов, начиная от новых приложений в промышленной рентгеновской компьютерной томографии [39] и новые математические результаты на основе superiorization, таких как обоснование строгой монотонности фейеровских методов с использованием superiorization проекционных методов [18]. Недавнее подробное описание предыдущей работы, связанной с superiorization можно найти в [15, раздел 3].

**Линейная superiorization.** Линейная superiorization (далее сокращенно: LinSup) рассматривает проблемы линейного программирования (ЛП), в которых ограничения, а также целевая функция являются линейными. Два вопроса, которые мы хотим исследовать экспериментально: (i) Может ли LinSup обеспечить допустимую точку, в которой значение целевой функции ниже, чем в точке, полученной тем же алгоритмом но без superiorization, но при прочих равных условиях? и (ii) Какова «стоимость» LinSup алгоритма по сравнению с симплекс-методами для решения задач ЛП? На основании наших вычислительных экспериментов, представленных здесь, ответы на эти два вопроса: "да" и "очень хорошо", соответственно.

Интересным и перспективным аспектом современных экспериментов является зависимость результатов от размеров тестовой задачи. Мы обнаружили, что преимущества LinSup становятся монотонно более выраженным при увеличении размеров проблемы. Мы рассмотрели задачи с 8000 линейных неравенств и векторов по 10000 компонентов, и увидели тенденцию, которая также сохраняется за пределами этих размеров проблем. Таким образом, LinSup вполне может стать полезным инструментом для вычислений проблема больших размеров. Следует признать, что наша предварительная работа, представленная здесь, основывается на случайно сгенерированных проблемах, и что они не являются типичными проблемами для задачи линейного программирования, которые решаются на протяжении многих лет.

В разделе 4 мы покажем, что LinSup находит высокую допустимую точку, т.е. допустимую точку с более низким значением целевой функции. В разделе 5 мы покажем численное поведение LinSup в сравнении с классическим Simplex алгоритмом линейной оптимизации. Общие рамки superiorization показываются в разделе 2, а LinSup представлена в разделе 3. Наши экспериментальные результаты были произведены с промощью программы MATLAB [31] и представлены в разделах 4 и 5. Мы делаем заключительные замечания в разделе 6 и перечисляем различные вопросы для проведения дальнейших исследований по LinSup. Приложение (раздел 7) кратко описывает технические изменения и модификации в superiorized версии базового алгоритма, опубликованные в литературе за последние несколько лет с момента создания.

# Методология superiorization

Рассмотрим пару (*М,* ***А***), где *М*, называемая целевым набором, является заданным подмножеством заданного подмножества *Q* J-мерного евклидового пространства *M ⊂ Q ⊆ RJ*. Пусть ***А*** *: Q → Q* будет неким алгоритмический оператор, который определяет итерационный процесс, называемый базовым алгоритмом,

*x0 ∈ Q, xk+1 =* ***A****(xk), k = 1, 2, …* (1)

задачей которого является найти точку в целевом множестве *М*. Мы, в дальнейшем, будем называть такую пару (*M, A*) ка superiorization пару. Пусть ᵠ *: Q ⊆ RJ→ R* заданная действительная функция, называется базовой функцией.

Методология superiorization предназначена для ограниченных задач уменьшения функции следующего вида.

**Проблема 1. Проблема уменьшения ограниченной функции**. Пусть *(М,* ***А****)* superiorization пара и пусть ᵠ *: Q ⊆ RJ→ R* является базовой функцией. Найти точку х\* в М, в которой значение функции ᵠ меньше (но не обязательно минимально), чем в точке в М, которая может быть достигнута путем применения базового алгоритма поиска точки в М.

Методология superiorization подходит к этой проблеме путем исследования возмущений основного алгоритма, а затем с помощью предупреждения таких возмущений, "заставляет" возмущенный алгоритм, полученный из базового, делать в дополнение к своей первоначальной задаче уменьшение шагов функции. Таким образом, этот возмущенный алгоритм называется "superiorized версия базового алгоритма".

Если базовый алгоритм является вычислительно эффективным и полезным при нахождении точки М, и если это возмущение упругое и может быть просто и недорого вычислено. То преимущество этого способа состоит в том, что, по расчетной стоимости базового алгоритма, мы можем решить задачу условной редукции функции, регулируя итерации в соответствии с возмущениями по уменьшению целевой функции. Методология superiorization автоматически генерирует superiorized версию базового алгоритма. Вектор х \*, полученные путем применения superiorized версии базового алгоритма, не обязательно должны минимизировать ᵠ в *М*. Для получения более подробной информации о видах возмущений устойчивости, которые могут быть использованы обратитесь, например, к [9, определения 4 и 9] или [13, 15, 27].

Приведенные выше определения и терминология основаны на том, что мы подразумеваем под выражением "Найти точку *х\** в *М*" в Проблеме 1. В слабой superiorization "найти точку *М*" подразумевается, как получить бесконечную последовательность , которая сходится к точке х\*∈ *М*, при этом *М* должно быть непустым. При сильном superiorization "найти точку *М*" понимается, как найти точку х\*, которая ɛ-совместима с *M*, для некоторого положительного ɛ. То есть точку, функция в которой близка, или отличается от *М* на значение меньше или равно ɛ. Таким образом, непустоту *М* считать не нужно. Эти понятия были определены в работе [9].

Два важных частных случаев пар superiorization (*М,* ***А***) в указанных выше рамках приходят на ум, хотя и другие случаи также возможны.

**Случай 2**. Целевое множество М является множество решений выпуклой задачи следующего вида: Найти вектор х\* ∈ где C*i* ⊆ RJ замкнутые выпуклые подмножества, таким образом, : В этом случае алгоритмический оператор и базовый алгоритм (1) может быть применен к широкому спектру алгоритмов поиска значений функции, примеры [3, 4, 8, 11, 12, 19].

**Случай 3.** Целевое множество М является множеством решений другой задачи минимизации с ограничениями: минимизировать целевую функцию f в допустимой области , таким образом . В этом случае алгоритмический оператор и базовый алгоритм (1) может быть использованы к любому из широкого разнообразия ограниченных алгоритмов минимизации распространенных в литературе.

В данной работе мы производим линейную superiorization, которая концентрируется на особой ситуации **Случая 2**, в котором все ограничения устанавливает *Ci* также, как и целевая линейная функция ᵠ. Superiorization работет и с другими целевыми функциями, такими как функции полной вариации (TV), например, в [13, 15, 27]. Superiorization в Случае 3, где *М* представляет собой множество решений максимального правдоподобия задачи оптимизации, появляется в работах [23, 28, 30].

# Линейная superiorization

* 1. **Проблема и алгоритм**

Определим целевое множество *M:*

(2)

где даны: I x J матрица действительных чисел и вектор .

Для базового алгоритма мы выбираем ищущие проекционные методы. Проекции на наборы используются во многих методах в теории оптимизации, но здесь проекционные методы относятся к итерационным алгоритмам, которые используют проекции на наборы, опираясь на общий принцип - группы, как правило, замкнутых и выпуклых, наборов легче спроецировать на определенные наборы, чем на другие наборы (пересечений, наборов изображений при некоторой трансформации и т.д.), которые получены из отдельных наборов.

Проекционные методы могут иметь различные алгоритмические структуры, такие как блок-итерационные проекции (BIP), см, например, [22, 24] и ссылки в них, или усреднения строк по проекции (SAP), см, например, [17] и ссылки в них, некоторые из них особенно хорошо подходят для параллельных вычислений, и они демонстрируют хорошие показатели сходимости и/или хорошие начальные модели поведения. Этот класс алгоритмов стал свидетелем большого прогресса в последние годы, и его алгоритмы были с успехом применены для многих научных, технологических и математических задач. Смотрите, например, в обзор [3] 1996 года, за последнее время книги и обзоры [11] и ссылки, книги [8] или [12].

Важное замечание в этом месте. Выпуклая проблема, упомянутая в **Случае 2**, может быть переведена в безусловную минимизацию некоторой функции близости, которая измеряет изменение точек алгоритма. Например, с помощью взвешенной суммы квадратов евклидовых расстояний до множеств CFP как функция близости и применение наискорейшего спуска к нему приводит метод одновременной проекции для CFP типа CIMMINO. Тем не менее, нет никакой функции близости, что дало бы метод последовательных проекций на CFP типа Качмажа см [2]. Таким образом, исследование осуществимости алгоритмов поиска для CFP разработан независимо от методов минимизации, смотрите ссылки, упомянутые выше. На протяжении многих лет исследователи пытались использовать методы прогнозирования для выпуклой задачи алгоритмов Л.П. более чем одним способом, например, см. книгу Chinneckís [20]. Мини-обзор отношений между линейным программированием алгоритмами в [34, § 1] проливает больше света на это. Наши результаты позволяют нам задаться вопросом, может ли LinSup служить причиной для этого.

Пусть целевой функцией для линейной superiorization будет

(3)

где – скалярное произведение вектора x и данного вектора .

Следуя общим принципам методологии superiorization, которые были представлены для общих целевых функций ᵠ в предыдущих публикациях, смотрите, например, недавние обзоры [26] и [9], приведем следующий линейный алгоритм superiorization. Вход алгоритма состоит из исходных данных проблемы *A, b*, и *c* из (2) и (3), соответственно. Так же выбранная пользователем начальная точка и ядро 0 < *a* <1 (смотри пункт 1 в п.3.3), с помощью которых алгоритм генерирует размер шага , в соответствии с целым числом N (смотрите пункт 7 в п.3.3). Все величины в алгоритме, которые еще не были определены или объяснены - подробно описаны в п.3.3 ниже.

**Алгоритм 4. Линейная Superiorization (LinSup)**

1. **установим**

2. **установим**

3. **установим**

4. **пока** не выполняется правило остановки

5. **установим**

6. **установим**

7. **установим**

8. **пока** n<N

9. **установим**

10. **установим**

11. **установим**

12. **установим**

13. **установим**

14. **выход**

15. **установим**

16. **установим**

17. **установим**

18. **выход**

* 1. **Алгоритм Эгмон-Моцкин-Шенберга как базовый алгоритм**

Мы использовали проекционный метод Эгмон-Моцкин-Шенберга (AMS) [1,32], см. также [19, Алгоритм 5.4.2], в качестве базового алгоритма для уменьшающей функции ***A***  на шаге 16 Алгоритма 4. Обозначим полупространства представленные отдельным строками в (2) Hi,

, (4)

где является i-й строкой матрицы A и – i-я компонента вектора b в (2). Ортогональная проекция произвольной точки на *Hi* имеет закрытую форму:

(5)

**Алгоритм 5. Метод релаксации Эгмона, Моцкина и Шенберга.**

**Инициализация:**  задается произвольно.

**Итерация:** Следующая итерация вектора **x** расчитывается на основе текущей итерации xk следующим образом:

(6)

**Параметры релаксации:** Параметры λk выбраны такими, что , для всех , с некоторыми сколь угодно малыми .

**Контроль:** Контрольная последовательность зациклено на {1, 2…, I}.

Этот AMS циклический алгоритм поиска выполняется циклически по неравенствам в (2). Для того чтобы справиться с ограничениями неотрицательности в (2) мы просто берем текущий вектор итерации, после того, как сделать прошли полный цикл AMS метода через все I неравенства, и устанавливаем его отрицательные компоненты в нуль при сохранении остальных неизменными.

Краеугольным камнем методологии superiorization в целом, а также для линейного случая обсуждаемого здесь, является возмущение устойчивости базового алгоритма, который используется в ней. Алгоритм АМС, как известно, имеет ограниченное возмущение сопротивляемости, это может узнать из различных опубликованных ранее результатов, см. например [16, теорема 12], [33].