# 5.1 Результаты и что они нам говорят

В таблице на Рисунке 5 показаны значения целевой функции для Simplex алгоритма вместе со значениями целевой функции на выходе LinSup при запуске на 3 различных значениях ядра . Также показаны значения относительных погрешностей RE (8). При больших значениях ядра его влияние уменьшается медленнее, оставляя больше места для возмущений целевой функции, что влияет на результат работы LinSup. При увеличении размерности проблемы относительная погрешность RE также возрастает.

В таблице на Рисунке 6 показано время выполнения в секундах для Simplex алгоритма и для LinSup алгоритма для 3-х различных значений ядра . Также показано отношение времени TR (9). При этом наблюдается, что LinSup работает быстре по сравнению с алгоритмом Simplex.

Рисунки 7-11 основаны на данных, содержащихся в Таблицах 5-6. Графики зависимости относительных погрешностей RE от различных размерностей проблемы для 3 различных значений ядра , основанны на данных таблицы на Рисунке 5, приводятся на Рисунке 7. Для каждого относительные ошибки увеличивается с увеличением размерности проблемы. Для всех размерностей проблемы относительные погрешности уменьшаются с ростом значения . Для всех размерностей проблемы относительная погрешность имеет наименьшее значение при наибольшем значения .

Графики зависимости отношения времени TR от размерности проблемы для LinSup по 3 различным ядра , основанные на данных из таблицы на Рисунке 6, приведены на Рисунке 8. Для каждого отношение времени уменьшается с увеличением размерности проблемы. Для всех размерностей проблемы значение соотношения времени уменьшается при уменьшении значения . Это приводит к проблеме при выбора ядра . Для лучшей (меньшей) относительной погрешности значение должно быть больше, а для лучшего соотношения (меньшего) времени оно должно быть меньше. Мы увидим некоторые компромиссы далее на следующих рисунках и таблицах.

На Рисунке 9 показаны значения целевой функции в зависимости от размерности проблемы для 3-х различных значений ядра . Наибольшее значение приводит к тому, что алгоритм LinSup производит большие шаги при снижении функции. Таким образом, это позволяет получить значения целевой функции, которые ближе всего к результатам, полученным из симплексного алгоритма. Рисунок 10 кратко показывает движение, совмещая рисунки 7 и 8. Этот рисунок наглядно показывает компромисс между уменьшением значения целевой функции и скоростью работы алгоритма LinSup.

Зависимость времени выполнения в тысячах секунд от размерности проблемы для алгоритмов Simplex и LinSup для 3 значений ядра изображены на Рисунке 11. Обратите внимание на резкое увеличение времени работы Simplex алгоритма (пунктирная линия) для задач большой размерности. LinSup отличается более умеренным ростом времени выполнения по сравнению с Simplex алгоритмом.

Результаты нашей работы показывают, что существует внутренний "конфликт" при выборе параметров, определяющих неустойчивое равновесие между усилиями, которые алгоритм LinSup прикладывает для работы алгоритма поиска, и снижением значений функции с возмущениями. Но поведение этих результатов на проблемах с большими размерностями позволяет надеяться, что при дальнейшем увеличении размерности LinSup алгоритмы получат больше возможностей и станут даже конкурентами для линейных алгоритмов минимизации. Заметим, что для задачи в последней строке таблицы на Рисунках 5 и 6 при , LinSup останавливается на значении целевой функции, которое довольно близко к полученному алгоритмом Simplex, но, при этом, алгоритм LinSup достигает этого значения всего за треть времени, которое потребовалось Simplex алгоритму.

# 5.2 Позволяем Simplex алгоритму останавливаться заранее

LinSup не предназначен для решения задач линейного программирования, а, как описано в Разделе 2, всего лишь позволяет найти допустимую точку с низким (не обязательно минимальным) значением линейной целевой функции. Тем не менее, с точки зрения задачи линейного программирования результат LinSup можно считать "достаточно близким решением задачи ЛП". В этой связи возникает вопрос, как бы это соотносится с не оптимально останавливающимся алгоритмом Simplex. Для того, чтобы получить предварительный обзор этой проблемы, мы породили задачу ЛП размерности 8000 x 10000 и запустили Simplex и LinSup алгоритмы работать по ней. LinSup остановился, когда его итерации не показали никаких существенных изменений (то есть, когда ), он был запущен с двумя различными значениями . Simplex алгоритму не разрешалось работать до нахождения оптимального решения, и он остановился практически через то же время, которое потребовалось алгоритму LinSup. Значения после остановки позволяют сравнить LinSup с неоптимально остановленным Simplex по одной и той же проблеме. Функция близости и количество вычислений значение целевой линейной функции Simplex алгоритмом считались после каждой итерации, потому что это не являются внутренней частью Simplex алгоритма.

Эти результаты показаны на Рисунках 12 и 13. Несмотря на то, что они изучены далеко не полностью, эти результаты показывают, что если Simplex алгоритм был бы остановлен неоптимально, скажем, после 5000 секунд, оба запуска LinSup дали бы более низкие значения целевой линейной функции, что показано на рисунке 12. В этот момент времени Simplex алгоритм показывал на выходе лучшие возможности, то есть более низкое значение функции близости. Тем не менее, в более поздний момент времени, например, через 20000 секунд, результат работы обоих LinSup алгоритмов будет иметь более низкое значение близости, чем Simplex, что показано на Рисунке 13, а один запуск с более высоким значением ядра будет иметь даже более низкое значение целевой линейной функции.

Такие результаты намекают на возможные преимущества LinSup для задач ЛП с большими размерностями. Если смотреть на результат работы LinSup как на "достаточно хорошее приближенное решение задачи ЛП", то LinSup не только сходится к такому решению быстрее, чем может сходиться Simplex метод, работающий с высокой точностью, но и быстрее, чем не оптимально остановленный Simplex алгоритм. Следует признать, что эти и другие эксперименты, представленные здесь, требуют дальнейшей работы, см. Раздел 6.

# 6. Выводы

Линейная superiorization (LinSup) не является, насколько мы знаем на данный момент, методом минимизации. Нахождение точки ограниченного минимума с помощью этого метода не может быть гарантирована. Однако, этот метод позволяет направить технические алгоритмы поиска в направлении точек с меньшими значениями функции (не обязательно минимальным) линейной целевой функции. Подобные применение LinSup в алгоритмах поиска является вычислительно-эффективным, оно использует проекции на выпуклые замкнутые множества ограничений, особенно успешно для линейного случая. Возмущения, применяемые для уменьшения значений целевой линейной функции не требуется никаких усилий, кроме как использования ***-с*** для корректировки направления спуска. Таким образом, предыдущая работа по методологии superiorization (см ссылки, упомянутые во введении и в приложении) наряду с доказательством концепции данной экспериментальной работы, позволяют предположить, что LinSup потенциально является жизнеспособным вариантом для решения задач ЛП больших размерностей.

Наши результаты показывают, что LinSup действительно находит приемлемую допустимую точку. И, при этом, увеличение времени работы на функциях проблемных размерностей в методологии LinSup является более умеренным, чем у алгоритма Simplex. Это позволяет нам сформулировать следующую гипотезу.

***Гипотеза 7.*** *Существуют такие разности проблем, над которыми LinSup будет работать лучше, чем линейные алгоритмы минимизации. Может быть, что это потребует использования сторонник методов поиска внутри LinSup, которые поддаются распараллеливанию, таких как блок-итерационных проекций (BIP) или проекций усреднения строк (SAP), указанных в разделе 3.1 выше.*

Возникает множество вопросов для дальнейших исследований, основанных на текущей работе. Вот список некоторых потенциально интересных направлений:

(1) Расширить вычислительные эксперименты для больших размерностей проблем и для по-разному порожденных проблем.

(2) Протестировать LinSup на более широком классе тестовых задач, чем те, которые используются в данной работе, например, задачи линейного программирования из NETLIB (http://www.netlib.org/) или других хранилищ.

(3) Исследовать LinSup с дополнительными проекционными методами поиска, которые поддаются распараллеливанию, такими как методы блок-итерационной проекции (BIP) или проекций усреднения строк (SAP).

(4) Исследовать влияние различных параметров на поведение LinSup, повторяя эксперименты с различными значениями: числа шагов возмущений N, которые выполняются перед каждым применением оператора осуществляющего поиск ищущий ***А***, параметров релаксации для встроенного базовом алгоритме поиска, ядра , с помощью которого генерируются размеры шагов .

(5) Углубленный математический анализ алгоритма LinSup.

(6) Повторить приведенные выше сравнения для дополнительных алгоритмов линейной оптимизации, такие как "interior-point" или "active-set" в пакете MATLAB или других.

(7) Исследовать противоречивый случай, в котором целевое множество М (2) пусто и заменяется, например, множеством ближайших точек для всех ограничений в соответствии с некоторой функцией близости. Алгоритмы линейного программирования могут не работать в таких условия, но LinSup вполне может дать полезный результат.

(8) Изучить работу LinSup для разреженных линейных ограничений, для которых некоторые проекционные методы уже доказали свою эффективность в качестве алгоритмов поиска.