

DOI: 10.17323/2587-814X.2025.2.77.88

Параллельная реализация симплекс-метода в матричной форме средствами библиотеки PyTorch для задач экономики и менеджмента

Ю.С. Эзрох^a 

E-mail: ezrokh@corp.nstu.ru

А.В. Снытников^{a,b} 

E-mail: aleksej.snytnikov@klgtu.ru

Е.Ю. Скоробогатых^b 

E-mail: elena.skorobogatykh@klgtu.ru

^a Новосибирский государственный технический университет, Новосибирск, Россия^b Калининградский государственный технический университет, Калининград, Россия

Аннотация

Симплекс-метод имеет широкое применение в задачах экономического планирования и прогнозирования. Однако, этот метод используется в реальной экономической деятельности для поиска решения масштабных задач, скорость выполнения процедур которых не является критичным фактором. Это в существенной степени ограничивает прикладное значение симплекс-метода в экономической сфере, поскольку в настоящее время сформировалась определенная тенденция перехода к более подробным экономическим моделям, что делает актуальной задачу ускорения расчетов на основе симплекс-метода. В этих условиях важнейшим средством ускорения расчетов становятся графические ускорители вычислений GPU (Graphical Processor Unit). Авторами предлагается реализация симплекс-метода в матричной форме для вычислений на графических процессорах средствами библиотеки PyTorch, которая позволяет перейти к использованию вычислительных мощностей графических процессоров

* Статья опубликована при поддержке Программы НИУ ВШЭ «Университетское партнерство»

простым и надежным способом. Задача линейного программирования с 900 ограничениями решается на графическом ускорителе в 6–9 раз быстрее по сравнению с решением на обычном процессоре. В работе выделены группы прикладных экономических задач, для которых предложенные алгоритмы и методы могут быть актуальны.

Ключевые слова: модифицированный симплекс-метод, ускорение вычислений, графические процессоры, задачи линейного программирования в экономике

Цитирование: Эзрох Ю.С., Снытников А.В., Скоробогатых Е.Ю. Параллельная реализация симплекс-метода в матричной форме средствами библиотеки PyTorch для задач экономики и менеджмента // Бизнес-информатика. 2025. Т. 19. № 2. С. 77–88. DOI: 10.17323/2587-814X.2025.2.77.88

Введение

Симплекс-метод [1] широко используется в задачах экономического планирования и прогнозирования [2, 3]. В широком смысле задача линейного программирования [4] состоит в том, что необходимо максимизировать или минимизировать некоторый линейный функционал на многомерном пространстве при заданных линейных ограничениях [5–7].

Роль линейного программирования, в частности, симплекс-метода, в экономическом анализе и планировании описана в [8]. Там говорится, что линейное программирование рассматривалось как инструмент реализации неоклассических экономических принципов как раз в то время, когда сама концепция рыночной экономики подвергалась атаке с нескольких направлений. Линейное программирование стало широко использоваться в народнохозяйственном планировании [9–11], особенно для развивающихся стран, и для изучения отдельных отраслей, особенно энергетики. В статье [12] описывается опыт применения линейного программирования к нефтеперерабатывающей промышленности США. Использование линейного программирования в сочетании с более тщательным выбором источников информации обеспечивает большую предсказательную способность по сравнению с использованием прогнозов на основании временных рядов. Также [13] подчеркивает необходимость полного понимания специфики производственных процессов, в противном случае методы математической экономики [14], в частности, линейное программирование дают лишь очень приблизительные результаты.

Работы [15, 16] характеризуют использование симплекс-метода следующим образом: эту процедуру можно интерпретировать как поиск рыночных цен, уравнивающих спрос на факторы производства с их предложением.

В то же время практически отсутствуют работы, в которых бы апробировалось применение симплекс-метода для решения большого числа однотипных задач с множеством условий в короткое время для индивидуальных пользователей. Иными словами, симплекс-метод используется в реальной экономической деятельности для поиска решения масштабных задач, скорость выполнения процедур которого не является критичным фактором. Это в существенной степени ограничивает прикладное значение симплекс-метода в экономической сфере.

Таким образом, на основании того, что симплекс-метод активно используется и существует определенная тенденция перехода к более подробным моделям, можно сделать вывод, что существует необходимость ускорения расчетов на его основе, причем, в первую очередь, речь идет о расчетах, проводимых на бытовых/офисных компьютерах, а не на суперЭВМ [17–19]. В таком случае важнейшим средством ускорения расчетов становятся графические ускорители вычислений (или графические процессоры, англ. Graphical Processor Unit, GPU) [20, 21].

Графические процессоры представляют собой набор из очень большого количества (до 10 тыс.) упрощенных процессорных ядер с общей памятью, на которых может быть запущено несколько десятков тысяч параллельных процессов. Это означает, что графические ускорители очень хорошо подходят

для решения задач линейной алгебры (типа умножения матриц). Время вычисления на графических процессорах обычно рассматривается в сравнении с временем вычислений на основном процессоре компьютера (англ. Central Processor Unit, CPU).

1. Обзор параллельных реализаций симплекс-метода

В статье [22] представлено распараллеливание симплекс-метода в табличном варианте для систем с общей памятью с целью решения крупномасштабных задач линейного программирования с плотной матрицей. Авторами описана общая схема метода и объяснены стратегии, принятые для распараллеливания каждого шага стандартного симплексного алгоритма. Проанализировано ускорение и параллельная эффективность по сравнению со стандартным симплекс-методом при использовании системы с общей памятью и 64 вычислительными ядрами. Эксперименты были выполнены для нескольких различных задач, до 8192 переменных и ограничений. Максимальное достигнутое ускорение — порядка 19.

Работа [23] рассматривает распараллеливание симплекс-метода на основе технологии OpenMP. Такой подход используется для того, чтобы улучшить ускорение и эффективность. Полученные результаты работы параллельного алгоритма сравниваются с обычным симплекс-методом. Также в этой работе использовалась такая особенность современных процессоров, как многопоточность. Предложенный алгоритм распараллеливания легко масштабируется на другое количество процессорных ядер. В общем случае на примере решения задачи о распределении вагонов между грузовыми терминалами железнодорожной станции проведен ряд вычислительных экспериментов. Сообщается об ускорении в 7 раз для 8 потоков OpenMP при размере данных более 10^2 .

В статье [24] рассматриваются предыдущие попытки распараллелить симплекс-метод в контексте улучшения производительности последовательных реализаций симплекс-метода и характера практических задач линейного программирования (ЛП). Для основной задачи решения общих больших разреженных задач ЛП автором статьи не было найдено распараллеливания симплекс-метода, обеспечивающего значительное улучшение производительности по сравнению с хорошей последовательной реализацией. Однако был достигнут определенный

успех в разработке параллельных решателей для плотных или обладающих особыми структурными свойствами задач ЛП. В результате проведенного обзора в данной работе определены направления будущей работы, направленной на разработку параллельных реализаций симплекс-метода, имеющих практическую ценность. Эти направления связаны с использованием параллельных методов факторизации и параллельных методов обращения разреженных матриц.

В работе [25] отмечено, что исследования, посвященные реализации методов математического программирования на GPU все еще находятся в зачаточном состоянии. Один из вариантов заключается в том, чтобы изменить существующие алгоритмы таким образом, чтобы получить значительный прирост производительности за счет выполнения на графическом процессоре. В статье [25] решается именно эта проблема, через представление эффективной реализации для симплекс-метода, адаптированного для GPU. В статье описывается, как выполнять шаги адаптированного симплексного метода, чтобы в полной мере использовать возможности графического процессора. Проведенные эксперименты демонстрируют значительное ускорение по сравнению с последовательной реализацией, что подчеркивает огромный потенциал GPU.

Приведенный обзор показывает, во-первых, актуальность работы по параллельной реализации симплекс-метода, и во-вторых, что эта работа должна быть в основном ориентирована на вычислительные системы с общей памятью, в частности, на GPU, что подтверждает правильность постановки задачи для данной статьи.

2. Описание симплекс-метода в матричной форме

В связи с необходимостью провести подробный анализ производительности алгоритма, а также для того, чтобы подчеркнуть отличие от более распространенного симплекс-метода в форме таблиц, приведем описание симплекс-метода в матричной форме (или «модифицированный симплекс-метод», англ. revised simplex method), следуя книге [26].

Рассмотрим задачу линейного программирования в следующем виде: требуется минимизировать cx при условии $Ax = b$, $x \geq 0$, где A — матрица размера $m \times n$, имеющая ранг m . Алгоритм решения задачи представлен на *рисунке 1*.

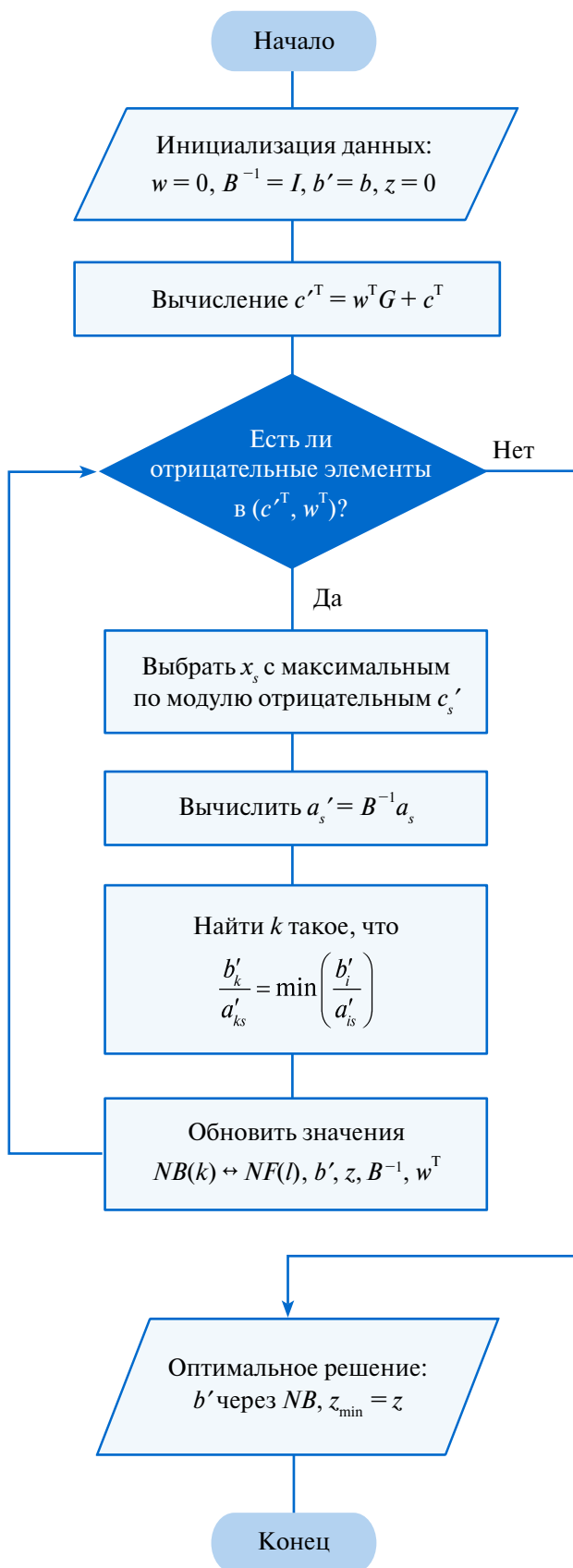


Рис. 1. Алгоритм модифицированного симплекс-метода.

Далее приведем пошаговое описание алгоритма.

2.1. Шаг инициализации

Выбрать начальное основное допустимое решение с базисом B .

2.2. Основная часть алгоритма

1. Решить систему линейных алгебраических уравнений $Bx_B = b$.

2. Решить систему линейных алгебраических уравнений $wB = c_b$ с единственным решением $w = c_b B^{-1}$. Вектор w принято называть симплекс-множителем из-за того, что его компоненты используются как множители для строк матрицы A при приведении ее в каноническую форму. Далее необходимо вычислить $z_j - c_j = wa_j - c_j$ для всех небазисных переменных. Пусть $z_j - c_j = \max_{j \in J} \{z_j - c_j\}$, где J – текущий набор индексов, связанных с небазисными переменными. Если $z_k - c_k \leq 0$, то текущее решение представляет собой оптимальное решение. В противном случае, выполняется шаг 3, при этом x_k .

3. Решить систему $By_k = a_k$. Если $y_k \leq 0$, тогда расчет прекращается, и делается вывод, что оптимальное решение неограниченно и находится на прямой

$$\left\{ \begin{bmatrix} \bar{b} \\ 0 \end{bmatrix} + x_k \begin{bmatrix} -y_k \\ e_k \end{bmatrix} : x_k \geq 0 \right\},$$

где e_k – вектор длины $(n - m)$, состоящий из нулей, за исключением компоненты с номером k , которая равна 1.

Если $y_k \geq 0$, то далее выполняется шаг 4.

4. Пусть x_k входит в базис. Тогда r – индекс блокирующей переменной x_{B_r} , такой что базис остается неизменным в результате следующей проверки на минимальное отношение

$$y_{r,k} = \min_{1 \leq i \leq m} \left\{ \frac{\bar{b}_r}{y_{r,k} : y_{i,k} > 0} \right\}.$$

Далее необходимо обновить базис B , где a_k заменяет a_{B_r} , обновить набор индексов J и повторить шаг 1.

2.3. Производительность реализации симплекс-метода в матричной форме

Время работы симплекс-метода в матричной форме было измерено на нескольких модельных задачах на компьютере с процессором Intel Core i7 950, 3.07 GHz (*таблица 1*). Модельные задачи были сгенерированы с помощью онлайн-генератора задач линейного программирования [27].

Таблица 1.

Время работы симплекс-метода в матричной форме

Количество ограничений	Размер матрицы A	Время, мс
10	10×20	9,3
10	10×910	27,1
100	100×600	36,7
900	900×1800	222,3

3. Программная реализация симплекс-метода в матричной форме

Современные средства языка программирования Python, а именно библиотека NumPy, дают возможность исключительно удобной реализации вычислительных алгоритмов, основанных на операциях с матрицами. Вследствие этого реализация симплекс-метода была выполнена в соответствии с процедурой, описанной в разделе 2.

Одна из задач данной работы в том, чтобы показать, что использовать мощности высокопроизводительных вычислений для решения прикладных задач можно с помощью очень простых в использовании инструментов. В связи с этим для реализации симплекс-метода на GPU была ис-

пользована библиотека PyTorch, предназначенная для работы с нейронными сетями. Библиотека PyTorch обладает двумя важными с точки зрения решаемой задачи достоинствами: большой набор инструментов для матричных вычислений и возможность перенести вычисления на GPU без изменения программы, просто добавкой одной инструкции, имеющей смысл «передать матрицу на GPU». Все дальнейшие вычисления с указанной матрицей будут проводиться на GPU. Как видно из *таблицы 2*, при переносе расчета на GPU с использованием библиотеки PyTorch время счета заметно уменьшается.

Не приводя листинги кода, тем не менее можно сказать, что при использовании альтернативных вариантов реализации симплекс-метода, основным из которых является технология CUDA, необходимо перед вычислением, например, матричного произведения выполнить несколько подготовительных операций, а именно переместить матрицы из памяти компьютера в память GPU и преобразовать в формат, оптимальный с точки зрения вычислений на GPU. Затем выполняется, собственно, реализация произведения матриц на основе асинхронных вычислений с помощью максимально возможного количества параллельных процессов.

Таким образом, видно, что перенос вычислительных алгоритмов на GPU с помощью библиотеки PyTorch оказываются значительно проще, чем при использовании специализированных инструментов программирования, таких как CUDA, OpenACC, cuBLAS и пр.

Производительность симплекс-метода, реализованного на GPU, была измерена на графическом ускорителе Nvidia TITAN X, а также на Nvidia Volta (вычислительный кластер НИВЦ НГУ), результаты измерений представлены в *таблице 2*.

Таблица 2.

Производительность симплекс-метода, реализованного на GPU

Количество ограничений	Размер матрицы A	Время на Titan, мс	Время на Volta, мс	Время на CPU, мс
10	10×20	1,9	1,3	9,3
10	10×910	4,4	4,4	27,1
100	100×600	5,2	5,2	36,7
900	900×1800	81,9	23,7	222,3

Меньшее время счета для Nvidia Volta достигается за счет, так называемых, «тензорных ядер» (англ. tensor cores), которые дополнительно ускоряют операции типа матричного умножения.

Также производительность программы была проанализирована на графическом ускорителе Nvidia A100 Ampere, установленном на вычислительном комплексе «Эйлер» КГТУ. На *рисунке 2* показана зависимость времени работы процедуры обращения матрицы, как математически наиболее трудоемкой операции, от параметра N , который с точки зрения симплекс-метода представляет собой количество переменных. Здесь важно обратить внимание на то, что обращение матрицы реализовано на CUDA средствами библиотеки PyTorch, и приведенный график не означает, что получена линейная зависимость времени работы обращения матрицы от количества переменных. Для прояснения этого тезиса рассмотрим количество потоков CUDA (CUDA threads). В силу того, что алгоритм обращения матрицы не реализуется вручную, на CUDA выбор количества потоков выполняется автоматически, тем не менее профилировщик дает возможность увидеть, сколько потоков используется, результат показан на *рисунке 3*.

Кроме того, интересно выглядит зависимость времени обращения матрицы от количества ограничений симплекс-метода M (*рис. 4*).

4. Направления прикладного экономического применения программной реализации симплекс-метода в матричной форме

Вышеописанные алгоритмы и подходы могут быть использованы при решении трех основных групп прикладных экономических задач, в том числе в области маркетинга, менеджмента (в рамках управления клиентским опытом, а также реализации ассортиментной и продуктовой политики), а также финансов.

Первая группа — коммерческие задачи в области маркетингового продвижения товаров/услуг, требующие мгновенного решения в виде онлайн отклика для получения положительного сверхэффекта спонтанных покупок. Примером этого может являться приложение продуктовых гипермаркетов, которое в режиме пресейл формирует итоговую потребительскую корзину текущего заказа. Учитывая возрастающий уровень детализа-

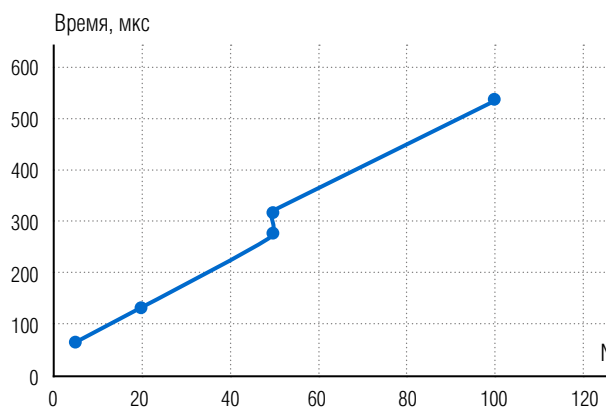


Рис. 2. Зависимость времени обращения матрицы на графическом ускорителе Nvidia A100 от количества переменных симплекс-метода.

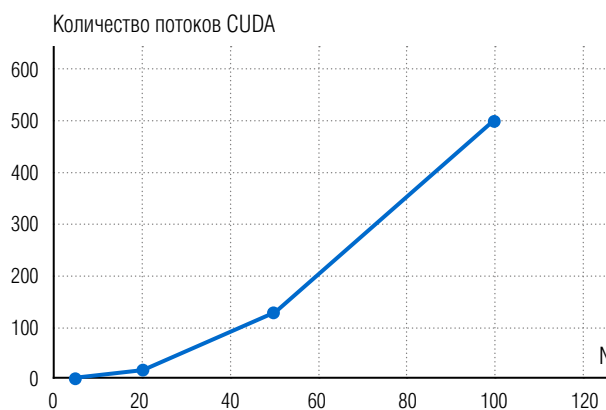


Рис. 3. Количество потоков CUDA, используемых при выполнении обращения матрицы при реализации симплекс-метода, на графическом ускорителе Nvidia A100.

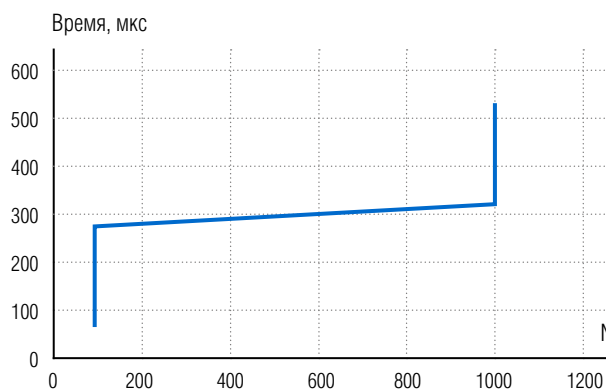


Рис. 4. Зависимость времени обращения матрицы на графическом ускорителе Nvidia A100, от количества ограничений симплекс-метода.

ции информации о клиенте (его личные данные, а также история покупок), число параметров, выступающих ограничениями в рамках симплекс-метода, становится достаточно большим, что оправдывает его применение. В такой ситуации клиент не готов ожидать время, требуемое для вычисления CPU (с учетом очереди клиентов), а также на передачу информации. Учитывая скорость современной жизни, клиент ориентирован на получение исключительно мгновенного результата, достигаемого с применением графических ускорителей, который сопровождается соответствующим объяснением-обоснованием.

Вторая группа экономических задач — интеллектуализация работы высокопроизводительных чат-ботов, использующих fuzzy logic, то есть генерирующих нестандартные ответы на заранее незапрограммированные вопросы. Это позволит значительно увеличить способность выполнения функции личного помощника, в том числе: а) расширить горизонт планирования расписания, которое сможет учитывать ограничения членов семьи, рабочих календарных планов, климатическую сезонность и т.д.; б) готовить обоснованные рекомендации в области здорового образа жизни, в том числе состав и рецептуру комбинированных продуктов, учитывающих геронтологические, антропометрические, личностные данные, вкусовые предпочтения и т.д. Очевидно, что увеличивающийся масштаб применения ботов-помощников позволит осуществлять более таргетированное взаимодействие с потребителями, а также использовать селлерам алгоритмы рекомендательных систем (задачи на стыке с описанными выше в первой группе). Так, например, на основании характеристик продуктов питания (от простейшего состава белков, жиров и углеводов, до характеристик взаимного дополнения отдельных продуктов или невозможности их совместного употребления) может решаться проблема формирования наиболее оптимального портфеля заказов в гипермаркетах, осуществляющих доставку своей продукции клиентам. В такой ситуации матрица линейных ограничений может существенно выходить за рамки, которые определены в статье, что подчеркивает важность практического применения. Вместе с тем, стоит подчеркнуть, что клиентопоток крупных субъектов электронной коммерции имеет тенденцию к резкому росту, тренд

на который может сохраниться как минимум в среднесрочной перспективе. При этом даже снижение темпов прироста интернет-бизнеса через 3–5 лет приведет, скорее всего, не к спаду, а к выходу соответствующих экономических показателей на плато с сохранением значительного объема удаленного взаимодействия компаний с клиентами в долгосрочной перспективе.

Третья группа объединяет финансово-экономические задачи, где скорость принятия решения является одним из ключевых факторов успешной алгоритмической торговли на биржевом и внебиржевом рынках. Стоит отметить, что данная группа задач является, в настоящее время, с одной стороны, одной из наиболее крупных по количеству участников ввиду, так как подавляющее большинство трейдеров при организации своей деятельности ориентированы на использование различных математических моделей. С другой стороны, конкуренция имеющихся способов алгоритмизации и высокая «цена ошибки» трейдинга в принципе, который осуществляется зачастую на средства клиентов или с использованием схем маржинальной торговли, затрудняют тестирование гипотез, неизбежным образом происходящее в рамках применения новых методов машинного обучения.

Стоит подчеркнуть, что указанные выше группы не исчерпывают задачи, где может быть эффективно применен симплекс-метод в матричной форме на GPU. К ним относятся традиционные экономические вопросы в области оптимизации с учетом ограниченности ресурсов, а также иные, например, творческие задачи (Basadur Simplex).

Заключение

Проведенное исследование показало, что перенос реализации симплекс-метода в матричной форме на GPU с помощью библиотеки PyTorch делает решение задачи значительно проще, чем при использовании специализированных инструментов программирования, таких как CUDA, OpenACC, cuBLAS и пр. Возникает вопрос о том, насколько хорошие результаты дает этот более простой метод. Для проведенных тестовых расчетов, при переносе расчета на GPU с использованием библиотеки PyTorch время счета уменьшается в 3–5 раз в зависимости от размера задачи.

С другой стороны, важно получить возможность ускорить расчеты по симплекс-методу для сравнительно небольших задач и для обычных рабочих станций, используемых в реальных экономических расчетах, а не для сверхбольших постановок задач, которые могут решаться только на суперЭВМ, что соответствует реальной практике, поскольку даже расчеты, проводимые в рамках экономической науки, не дают примеров очень крупных задач для симплекс-метода. В работе представлены три основные группы прикладных экономических задач, которые могут быть эффек-

тивно решены с использованием обоснованного в статье инструментария.

Также необходимо отметить, что технология CUDA позволяет почти во всех случаях получить наибольшую эффективность реализации на GPU по сравнению с PyTorch, тем не менее, CUDA настолько сложна, что ее использование для решения какой-то реальной задачи представляет обычно отдельный большой вопрос. В данной работе используется менее эффективный, но значительно более простой инструмент. ■

Литература

1. Канторович Л.В. Математические методы организации и планирования производства. Ленинград: ЛГУ, 1939.
2. Макаров В.Л., Бахтизин А.Р., Сушко Е.Д., Сушко Г.Б. Создание суперкомпьютерной имитации общества с активными агентами разных типов и ее апробация // Вестник Российской академии наук. 2022. Т. 92. № 5. С. 458–466.
3. Kolev M., Georgiadou S. On simulation and modeling in economics // Asian-European Journal of Mathematics. 2022. Vol. 15. No. 10. Article 2250239. <https://doi.org/10.1142/s1793557122502394>
4. Hybrid variable neighborhood search for automated warehouse scheduling / I. Davydov [et al.] // Optimization Letters. 2023. Vol. 17. No. 9. P. 2185–2199. <https://doi.org/10.1007/s11590-022-01921-6>
5. Чистякова Т.Б., Шашихина О.Е. Интеллектуальный программный комплекс моделирования процесса планирования многоассортиментных промышленных производств // Прикладная информатика. 2022. Т. 17. № 5(101). С. 41–50. <https://doi.org/10.37791/2687-0649-2022-17-5-41-50>
6. Бекларян Л.А., Бекларян Г.Л., Акопов А.С., Хачатрян Н.К. Динамические и агентные модели интеллектуальных транспортных систем // Экономика и математические методы. 2024. Т. 60. № 2. С. 105–122. <https://doi.org/10.31857/S0424738824020091>
7. Mochurad L., Boyko N., Sheketa V. Parallelization of the method of simulated annealing when solving multicriteria optimization problems // Proceedings of the CEUR Workshop. Lviv, 2020. P. 12–24.
8. Arrow K.J. George Dantzig in the development of economic analysis // Discrete Optimization. 2008. Vol. 5. No. 2. P. 159–167. <https://doi.org/10.1016/j.disopt.2006.11.007>
9. Ratushnyi A., Kochetov Y. A column generation based heuristic for a temporal bin packing problem // Proceedings of the 20th International Conference Mathematical Optimization Theory and Operations Research (MOTOR 2021) (eds. P. Pardalos, M. Khachay, A. Kazakov), Irkutsk, Russia, July 5–10, 2021. Lecture Notes in Computer Science. Vol. 12755. P. 96–110. Springer, Cham. https://doi.org/10.1007/978-3-030-77876-7_7
10. Кочетов Ю.А., Шамрай Н.Б. Оптимизация размещения и передислокации бригад скорой медицинской помощи // Дискретный анализ и исследование операций. 2021. Т. 28. № 2(148). С. 5–34. <https://doi.org/10.33048/daio.2021.28.702>
11. Ковешников В.А., Мехтиев А.Я. Исследование накопительно-сортировочного метода решения задач параметрической оптимизации // Проблемы управления. 2020. № 2. С. 28–35. <https://doi.org/10.25728/ru.2020.2.3>
12. Manne A.S. A linear programming model of the U.S. petroleum refining industry // Econometrica. 1958. No. 26. P. 67–196.
13. Chenery H.B. Engineering production functions // Quarterly Journal of Economics. 1949. No. 63. P. 507–531.
14. Макаров В.Л., Бахтизин А.Р., Бекларян Г.Л., Акопов А.С. Цифровой завод: методы дискретно-событийного моделирования и оптимизации производственных характеристик // Бизнес-информатика. 2021. Т. 15. № 2. С. 7–20. <https://doi.org/10.17323/2587-814X.2021.2.7.20>
15. Scarf H.E. Mathematical programming and economic theory // Operations Research. 1990. Vol. 38. No. 3. P. 377–385. <https://doi.org/10.1287/opre.38.3.377>
16. Апалькова Т.Г., Косоруков О.А., Мищенко А.В., Цурков В.И. Математические модели управления производственно-финансовой деятельностью предприятия // Известия Российской академии наук. Теория и системы управления. 2024. № 2. С. 107–129. <https://doi.org/10.31857/S0002338824020109>

17. Gergel V., Grishagin V., Liniyov A., Shumikhin S. Parallel computations in integrated environment of engineering modeling and global optimization // Proceedings of the 16th International Conference (PaCT 2021), Kaliningrad, Russia, September 13–18, 2021 (ed. V. Malyshev). Lecture Notes in Computer Science. 2021. Vol. 12942. P. 413–419. Springer, Cham. https://doi.org/10.1007/978-3-030-86359-3_31
18. Gergel V., Kozinov E. Parallel computations for solving multicriteria mixed-integer optimization problems // Communications in Computer and Information Science. 2021. Vol. 1437. P. 92–107. https://doi.org/10.1007/978-3-030-81691-9_7
19. Shichkina Y., Kupriyanov M., Awadh A.M.M.H. Application of methods for optimizing parallel algorithms for solving problems of distributed computing systems // Proceedings of the International Conference on Cyber-Physical Systems and Control (CPS&C'2019), St. Petersburg, Russia, June 10–12, 2019 (eds. D. Arseniev, L. Overmeyer, H. Kälviäinen, B. Katalinić). Lecture Notes in Networks and Systems. 2020. Vol. 95. P. 212–224. Springer, Cham. https://doi.org/10.1007/978-3-030-34983-7_21
20. Моделирование миграционных и демографических процессов с использованием FLAME GPU / В.Л. Макаров и [др.] // Бизнес-информатика. 2022. Т. 16. № 1. С. 7–21. <https://doi.org/10.17323/2587-814X.2022.1.7.21>
21. Beklaryan A.L., Akopov A.S., Beklaryan L.A. Implementation of the Deffuant model within the FLAME GPU framework // Advances in Systems Science and Applications. 2021. Vol. 21. No. 4. P. 87–99. <https://doi.org/10.25728/assa.2021.21.4.1161>
22. Coutinho D.A.M., Lins e Silva F.O., Aloise D., Xavier-de-Souza S. A scalable shared-memory parallel simplex for large-scale linear programming // arXiv:1804.04737v2. 2019. <https://doi.org/10.48550/arXiv.1804.04737>
23. Mochurad L. Parallelization of the Simplex method based on the OpenMP technology // Proceedings of the 4th International Conference on Computational Linguistics and Intelligent Systems (COLINS 2020), Lviv, Ukraine, April 23–24, 2020. P. 952–963.
24. Hall J.A.J. Towards a practical parallelization of the Simplex method // Computational Management Science. 2010. Vol. 7. P. 139–170. <https://doi.org/10.1007/s10287-008-0080-5>
25. Bieling J. An efficient GPU implementation of the revised simplex method // Proceedings of the IEEE International Symposium on Parallel Distributed Processing, Workshops and PhD Forum (IPDPSW), Atlanta, GA, USA, 2010. P. 1–8. <https://doi.org/10.1109/IPDPSW.2010.5470831>
26. Bazaraa M.S., Jarvis J.J., Sherali H.D. Linear programming and network flows. John Wiley Sons, Inc., Hoboken, New Jersey, 2010.
27. LP random problem generator // Instituto Superior Técnico, Universidade de Lisboa, 2025. [Электронный ресурс]: <https://web.tecnico.ulisboa.pt/~mcasquilho/compute/or/Fx-LP-generator.php> (дата обращения 20.04.2025).

Об авторах

Эзрох Юрий Семенович

доктор экономических наук, доцент;

заведующий кафедрой, кафедра экономической информатики, Новосибирский государственный технический университет, Россия, 630073, г. Новосибирск, улица Карла Маркса, д. 6;

E-mail: ezrokh@corp.nstu.ru

ORCID: 0000-0002-8367-1840

Снытников Алексей Владимирович

доктор технических наук;

профессор, кафедра прикладной информатики, Калининградский государственный технический университет, Россия, 236022, г. Калининград, проспект Советский, д. 1;

E-mail: aleksej.snytnikov@klgtu.ru

ORCID: 0000-0003-4111-308X

Скоробогатых Елена Юрьевна

кандидат педагогических наук, доцент;

доцент, кафедра прикладной информатики и информационных технологий, Калининградский государственный технический университет, Россия, 236022, г. Калининград, проспект Советский, д. 1;

E-mail: elena.skorobogatykh@klgtu.ru

ORCID: 0000-0001-6050-4831

Parallel implementation of the simplex method in matrix form using the PyTorch library for economics and management problems

Yuriy S. Ezrokh^a

E-mail: ezrokh@corp.nstu.ru

Alexey V. Snytnikov^{a,b}

E-mail: aleksej.snytnikov@klgtu.ru

Elena Yu. Skorobogatykh^b

E-mail: elena.skorobogatykh@klgtu.ru

^a Novosibirsk State Technical University, Novosibirsk, Russia

^b Kaliningrad State Technical University, Kaliningrad, Russia

Abstract

The simplex method is widely used in economic planning and forecasting tasks. However, this method is used in real economic activity to find solutions to large-scale tasks, the speed of which is not a critical factor. This significantly limits the applied value of the simplex method in the economic sphere, since currently there is a certain tendency to move to more detailed economic models, which makes it urgent to accelerate calculations based on the simplex method. In these conditions, GPU (Graphical Processor Unit) computing accelerators become the most important means of accelerating calculations. The authors propose the implementation of the simplex method in matrix form for computing on GPUs using the PyTorch library. This allows you to switch to using the computing power of graphics processors in a simple and reliable way. A linear programming problem with 900 constraints is solved on a graphics accelerator 6–9 times faster than the solution on a conventional processor. This paper identifies groups of applied economic problems for which the proposed algorithms and methods can be relevant.

Keywords: modified simplex method, acceleration of calculations, graphics processors, linear programming problems in economics

Citation: Ezrokh Y.S., Snytnikov A.V., Skorobogatykh E.Yu. (2025) Parallel implementation of the simplex method in matrix form using the PyTorch library for economics and management problems. *Business Informatics*, vol. 19, no. 2, pp. 77–88. DOI: 10.17323/2587-814X.2025.2.77.88

References

1. Kanorovich L.V. (1939) *Mathematical methods of organization and planning of production*. Leningrad: Leningrad State University Press (in Russian).
2. Makarov V.L., Bakhtizin A.R., Sushko E.D., Sushko G.B. (2022) Creating a supercomputer simulation of society with active agents of different types and its testing. *Herald of the Russian Academy of Sciences*, vol. 92, no. 5, pp. 458–466 (in Russian).
3. Kolev M., Georgiadou S. (2022) On simulation and modeling in economics. *Asian-European Journal of Mathematics*, vol. 15, no. 10, article 2250239. <https://doi.org/10.1142/s1793557122502394>
4. Davydov I., Kochetov Yu., Tolstykh D., et al. (2023) Hybrid variable neighborhood search for automated warehouse scheduling. *Optimization Letters*, vol. 17, no. 9, pp. 2185–2199. <https://doi.org/10.1007/s11590-022-01921-6>
5. Chistyakova T.B., Shashikhina O.E. (2022) Intelligent software complex for modeling the process of planning multi-assortment industrial productions. *Applied Informatics*, vol. 17, no. 5(101), pp. 41–50 (in Russian). <https://doi.org/10.37791/2687-0649-2022-17-5-41-50>
6. Beklaryan L.A., Beklaryan G.L., Akopov A.S., Khachatryan N.K. (2024) Dynamic and agent-based models of intelligent transport systems. *Economics and Mathematical Methods*, vol. 60, no. 2, pp. 105–122 (in Russian). <https://doi.org/10.31857/S0424738824020091>
7. Mochurad L., Boyko N., Sheketa V. (2020) Parallelization of the method of simulated annealing when solving multicriteria optimization problems. Proceedings of the *CEUR Workshop*, Lviv, May 21, 2020, pp. 12–24.
8. Arrow K.J. (2008) George Dantzig in the development of economic analysis. *Discrete Optimization*, vol. 5, no. 2, pp. 159–167. <https://doi.org/10.1016/j.disopt.2006.11.007>
9. Ratushnyi A., Kochetov Y. (2021) A column generation based heuristic for a temporal bin packing problem. Proceedings of the *20th International Conference Mathematical Optimization Theory and Operations Research (MOTOR 2021)*, Irkutsk, Russia, July 5–10, 2021 (eds. P. Pardalos, M. Khachay, A. Kazakov), Lecture Notes in Computer Science, vol. 12755, pp. 96–110. Springer, Cham. https://doi.org/10.1007/978-3-030-77876-7_7
10. Kochetov Yu.A., Shamray N.B. (2021) Optimization of placement and relocation of emergency medical teams. *Discrete Analysis and Operation Research*, vol. 28, no. 2(148), pp. 5–34 (in Russian). <https://doi.org/10.33048/daio.2021.28.702>
11. Kovesnikov V.A., Mehtiev A.Ya. (2020) Study of the accumulative-sorting method for solving parametric optimization problems. *Control Issues*, no. 2, pp. 28–35 (in Russian). <https://doi.org/10.25728/pu.2020.2.3>
12. Manne A.S. (1958) A linear programming model of the U.S. petroleum refining industry. *Econometrica*, no. 26, pp. 67–196.
13. Chenery H.B. (1949) Engineering production functions. *The Quarterly Journal of Economics*, no. 63, pp. 507–531.
14. Makarov V.L., Bakhtizin A.R., Beklaryan G.L., Akopov A.S. (2021) Digital plant: methods of discrete-event modeling and optimization of production characteristics. *Business Informatics*, vol. 15, no. 2, pp. 7–20. <https://doi.org/10.17323/2587-814X.2021.2.7.20>
15. Scarf H.E. (1990) Mathematical programming and economic theory. *Operations Research*, vol. 38, no. 3, pp. 377–385.
16. Apalkova T.G., Kosorukov O.A., Mishchenko A.V., Tsourkov V.I. (2024) Mathematical models for managing the production and financial activities of an enterprise. *Herald of the Russian Academy of Sciences. Theory and Systems of Control*, no. 2, pp. 107–129. <https://doi.org/10.31857/S0002338824020109>
17. Gergel V., Grishagin V., Liniov A., Shumikhin S. (2021) Parallel computations in integrated environment of engineering modeling and global optimization. Proceedings of the *16th International Conference (PaCT 2021)*, Kaliningrad, Russia, September 13–18, 2021 (ed. V. Malyshekin), Lecture Notes in Computer Science, vol. 12942, pp. 413–419. Springer, Cham. https://doi.org/10.1007/978-3-030-86359-3_31
18. Gergel V., Kozinov E. (2021) Parallel computations for solving multicriteria mixed-integer optimization problems. *Communications in Computer and Information Science*, vol. 1437, pp. 92–107. https://doi.org/10.1007/978-3-030-81691-9_7
19. Shichkina Y., Kupriyanov M., Awadh AM.M.H. (2020) Application of methods for optimizing parallel algorithms for solving problems of distributed computing systems. Proceedings of the *International Conference on Cyber-Physical Systems and Control (CPS&C'2019)*, St. Petersburg, Russia, June 10–12, 2019 (eds. D. Arseniev, L. Overmeyer, H. Kälviäinen, B. Katalinić), Lecture Notes in Networks and Systems, vol. 95, pp. 212–224. Springer, Cham. https://doi.org/10.1007/978-3-030-34983-7_21
20. Makarov V.L., Bakhtizin A.R., Beklaryan G.L., Akopov A.S., Strelkovskii N.V. (2022) Simulation of migration and demographic processes using FLAME GPU. *Business Informatics*, vol. 16, no. 1, pp. 7–21. <https://doi.org/10.17323/2587-814X.2022.1.7.21>
21. Beklaryan A.L., Akopov A.S., Beklaryan L.A. (2021) Implementation of the Deffuant model within the FLAME GPU framework. *Advances in Systems Science and Applications*, vol. 21, no. 4, pp. 87–99. <https://doi.org/10.25728/assa.2021.21.4.1161>
22. Coutinho D.A.M., Lins e Silva F.O., Aloise D., Xavier-de-Souza S. (2019) A scalable shared-memory parallel simplex for large-scale linear programming. *arXiv:1804.04737v2*. <https://doi.org/10.48550/arXiv.1804.04737>
23. Mochurad L. (2020) Parallelization of the Simplex method based on the OpenMP technology. Proceedings of the *4th International Conference on Computational Linguistics and Intelligent Systems (COLINS 2020)*, Lviv, Ukraine, April 23–24, 2020, pp. 952–963.

24. Hall J.A.J. (2010) Towards a practical parallelization of the Simplex method. *Computational Management Science*, vol. 7, pp. 139–170. <https://doi.org/10.1007/s10287-008-0080-5>
25. Bieling J. (2010) An efficient GPU implementation of the revised simplex method. *Proceedings of the IEEE International Symposium on Parallel Distributed Processing, Workshops and PhD Forum (IPDPSW), Atlanta, GA, USA*, pp. 1–8. <https://doi.org/10.1109/IPDPSW.2010.5470831>
26. Bazaraa M.S., Jarvis J.J., Sherali H.D. (2010) *Linear Programming and Network Flows*. Hoboken, NJ: John Wiley & Sons, Inc.
27. Instituto Superior Técnico, Universidade de Lisboa (2025) *LP random problem generator*. Available at: <https://web.tecnico.ulisboa.pt/~mcasquilho/compute/or/Fx-LP-generator.php> (accessed 20 April 2025).

About the authors

Yuriy S. Ezrokh

Doctor of Sciences (Economics), Associate Professor;

Head of Department, Department of Economic Informatics, Novosibirsk State Technical University, 6, Karl Marx St., Novosibirsk 630073, Russia;

E-mail: ezrokh@corp.nstu.ru

ORCID: 0000-0002-8367-1840

Alexey V. Snytnikov

Doctor of Sciences (Technology);

Professor, Department of Applied Informatics, Kaliningrad State Technical University, 1, Sovetskiy Ave., Kaliningrad 236022, Russia;

E-mail: aleksej.snytnikov@klgtu.ru

ORCID: 0000-0003-4111-308X

Elena Yu. Skorobogatykh

Candidate of Sciences (Pedagogy), Associate Professor;

Associate Professor, Department of Applied Informatics, Kaliningrad State Technical University, 1, Sovetskiy Ave., Kaliningrad 236022, Russia;

E-mail: elena.skorobogatykh@klgtu.ru

ORCID: 0000-0001-6050-4831