А. А. Сиднев, В. П. Гергель, Автоматический выбор наиболее эффективных реализаций алгоритмов, *Выч. мет. программирование*, 2014, том 15, выпуск 4, 579–592

УДК 004.852; 004.272.43

АВТОМАТИЧЕСКИЙ ВЫБОР НАИБОЛЕЕ ЭФФЕКТИВНЫХ РЕАЛИЗАЦИЙ АЛГОРИТМОВ

А. А. Сиднев[1], **В. П. Гергель**[2]

Предложен подход к прогнозированию времени распределенных высокопроизводительных вычислений, не требующий проведения экспериментов на всех целевых вычислительных системах. Выбор оптимального алгоритма производится по информации об асимптотической сложности оцениваемых алгоритмов на основе использования методов машинного обучения. Предложенный в работе подход позволяет значительно сократить как количество экспериментов, так и размерность задач, которые решаются при оценке производительности вычислительной системы. Оценка времени выполнения алгоритмов по параметрам вычислительной системы позволяет определить, насколько вычислительная система эффективна при решении определенного класса задач без проведения экспериментов на ней. Возможно быстрое уточнение прогноза по минимальному числу экспериментов с малоразмерными задачами на целевой вычислительной системе. Предложенное решение может применяться и при автоматической настройке библиотеки перед ее использованием (подобно автоматической настройке в библиотеке ATLAS (Automatically Tuned Linear Algebra Software)). Выполнен сравнительный анализ результатов предсказания времени решения ряда задач на 84 вычислительных системах. Использование случайного леса в сочетании с методом наименьших квадратов показывает, что средняя относительная ошибка оценки времени выполнения составляет 17% при обучении на данных, соответствующих задачам малой размерности, и 9% при обучении на данных из всего диапазона изменения параметров алгоритма тестовой выборки. Полученные оценки позволяют выполнять выбор наиболее эффективной реализации алгоритма более чем в 80% случаев, а потери времени от ошибочного выбора не превосходят 6%.

Ключевые слова: выбор реализации алгоритма, время выполнения программ, асимптотические оценки сложности, характеристики вычислительных систем, машинное обучение, восстановление регрессии.

1. Введение. За годы развития высокопроизводительных и распределенных вычислений появилось понимание необходимости и широкая практика оптимизации алгоритмов базовых вычислительных задач под конкретные вычислительные архитектуры и под размер решаемой задачи. Следствием этого процесса явились современные оптимизированные библиотеки высокопроизводительных вычислений от производителей вычислительной техники (Intel MKL, AMD ACML, NVIDIA CUDA SDK, Sun Performance Library и др.). Такие библиотеки способны перед запуском алгоритма автоматически выбирать его параметры или реализацию, оптимальные для конкретной вычислительной системы и величин аргументов задачи. Общеизвестным примером подобной библиотеки является библиотека ATLAS, представляющая собой переносимый самооптимизирующийся пакет BLAS (Basic Linear Algebra Subprograms).

Одной из важнейших проблем современных распределенных и высокопроизводительных вычислений, возникающих при организации решения прикладных задач, является проблема прогноза времени вычислений, тесно связанная с выбором оптимального варианта алгоритма вычислений. Решение этой проблемы имеет определенную историю. Автор статьи [1] один из первых формализовал задачу выбора алгоритма в виде задачи оптимизации. Для ее решения предлагалось использовать методы аппроксимации, служащие для построения оценок эффективности алгоритмов, и выбирать лучший алгоритм на основе этих оценок. На текущий момент данный подход является наиболее распространенным.

В [2, 3] рассматривается проблема выбора лучшего алгоритма для решения задачи планирования (в частности, построение маршрута перевозки продуктов по городу). Подобные задачи тяжело решать в

[1] Нижегородский государственный университет им. Н. И. Лобачевского, факультет вычислительной математики и кибернетики, просп. Гагарина, д. 23, 603950, г. Нижний Новгород; ассистент, e-mail: sidnev@vmk.unn.ru

[2] Нижегородский государственный университет им. Н. И. Лобачевского, факультет вычислительной математики и кибернетики, просп. Гагарина, д. 23, 603950, г. Нижний Новгород; декан, профессор, e-mail: gergel@unn.ru

общем случае из-за высокой вычислительной трудоемкости. Чаще всего задачи данного типа решаются алгоритмами перебора, при этом в зависимости от выбора начальных параметров допустимое решение находится быстро или его поиск занимает длительный промежуток времени. В подобных ситуациях алгоритм останавливается по временно́му порогу [4]. Известны эвристические алгоритмы выбора начальных параметров, использование которых в алгоритмах перебора позволяет находить решения для частных случаев (задача при этом может иметь большую размерность). У подобных алгоритмов есть один недостаток: изменение параметров задачи, даже при сохранении размерности, может существенно влиять на время работы [5]. Таким образом, каждый алгоритм имеет смысл применять только для определенного класса параметров задачи, а реализация планировщика, способного выбрать подходящий алгоритм, очень востребована. Авторы публикации [6] рассматривают один из планировщиков для задачи разрешимости булевых формул SAT (Boolean SATisfiability). В [7] сравниваются 28 планировщиков на 4726 наборах тестовых данных. Таким образом, на текущий момент проблема автоматического выбора наиболее приемлемого алгоритма решения задачи получила широкое распространение [8].

Оценить время выполнения алгоритма можно посредством предварительного тестирования. В [9] рассматривается оценка времени выполнения алгоритма на суперкомпьютерах за счет построения линейной модели на основе результатов тестирования. Примерно с середины 90-х годов для решения задачи оценки времени выполнения успешно используются алгоритмы машинного обучения. Среди активно применяющихся на практике можно выделить: линейную регрессию и ее модификации (гребневая регрессия (ridge regression) или метод LASSO (Least Absolute Shrinkage and Selection Operator), предложенный Тибширани) деревья решений и их ансамбли, метод опорных векторов, нейронные сети (см., например, [10]) и использование гауссовских процессов [11]. В [12] выполнено сравнение результатов работы некоторых методов машинного обучения на задаче выбора алгоритма. Результаты показали, что в большинстве случаев лучшими являются методы восстановления линейной регрессии и деревья решений.

По результатам нашего обзора можно отметить, что проблема прогноза времени вычислений в целом успешно решается на основе методов машинного обучения, однако во всех случаях прогноз строится на анализе пробных запусков задачи только на целевой вычислительной системе. Тем не менее, остается актуальным: уменьшение количества экспериментов; прогнозирование для задач большого размера по эксперименту с задачами малого размера (относительно целевого); прогноз по экспериментам, выполненным на вычислительных системах, которые отличаются от целевой. Последнее особенно актуально для распределенных вычислений на GRID-системах. В настоящей статье формулируется другая постановка задачи, позволяющая выполнять оценки без проведения экспериментов на целевой вычислительной системе. Предлагается метод решения задачи выбора оптимального алгоритма на основании информации об асимптотической сложности оцениваемых алгоритмов, который может быть использован и для классической задачи на целевой платформе.

2. Задача выбора наиболее эффективной реализации алгоритма. Пусть имеется r реализаций алгоритмов, которые предназначены для решения одной и той же задачи. Будем предполагать, что все реализации имеют одинаковый набор параметров. Например, для задачи умножения матриц в качестве параметров будут выступать размеры матриц. Обозначим через $p = \left(p^1, p^2, \ldots, p^m\right)$, $p \in R^m$, — вектор входных параметров алгоритма, через $x = \left(x^1, x^2, \ldots, x^k\right)$, $x \in R^k$, — вектор характеристик вычислительной системы. Тогда время выполнения алгоритма y при заданных параметрах p на вычислительной системе с характеристиками x задается в виде $y = f(p, x)$, где f — неизвестная функциональная зависимость, при этом i-я реализация алгоритма характеризуется временем решения задачи $y_i = f_i(p, x)$. Выбор наиболее эффективной реализации алгоритма i^* заключается в нахождении такой реализации, которая решает задачу за минимальное время:

$$i^* = \arg\min_{i=1,\ldots,r} f_i(p, x). \tag{1}$$

Для решения задачи выбора (1) необходимо решить задачу оценки времени выполнения алгоритма, состоящую в построении функции $g(p, x)$, которая наилучшим образом аппроксимирует f. Тогда качество построенной оценки может быть определено функцией потерь $Q(g) = \sum_{p \in P} \left(g(p, x) - y\right)^2$, где P — тестовая выборка параметров. Таким образом, задача оценки времени выполнения алгоритма на конкретной вычислительной системе заключается в построении такой функции g^*, которая минимизирует функцию потерь:

$$g^* = \arg\min_{g} \sum_{p \in P} \left(g(p, x) - y\right)^2. \tag{2}$$

Для построения функции g^* будем использовать информацию о времени выполнения алгоритма на

близких программно-аппаратных платформах. Совокупность $\left(x_i, (y_{ij}, p_{ij})_{j=1}^{e_i}\right)_{i=1}^{n}$ называется обучающей выборкой, где n — количество вычислительных систем, для которых доступна информация о времени выполнения реализации алгоритма, и e_i — количество экспериментов, проведенных на i-й вычислительной системе. Отметим, что оценку времени выполнения необходимо осуществлять не только в интервале изменения значений параметра p обучающей выборки, но и за его пределами (например, для параметров соответствующих задаче большей размерности), т.е. требуется выполнять экстраполяцию. Характеристики вычислительных систем из тестовой и обучающей выборок не сильно отличаются, поэтому требование к качеству экстраполяции по характеристикам вычислительных систем не столь существенно. Если же архитектура тестовой системы принципиально отличается от представленных в обучающей выборке, то предсказать время работы программы на ней крайне сложно; тем не менее, будут приведены результаты экспериментов и для подобных ситуаций.

В настоящей статье рассматриваются только вычислительные системы с общей памятью. Результаты экспериментов будут приведены для вычислительных систем архитектуры IA-32 на база процессоров Intel и AMD под управлением ОС семейства Windows, так как именно эти процессоры и ОС широко доступны для проведения экспериментов. Однако предлагаемый подход может быть использован в системах под управлением других ОС и на процессорах других производителей и архитектуры. Ниже будут рассматриваться такие реализации алгоритмов, асимптотическая оценка сложности которых зависит только от размерности входных данных (например, умножение матриц, преобразование Фурье, решение линейных систем прямыми методами и др.). Кроме того, предполагаются следующие ограничения: алгоритм выполняется в рамках одного процесса (допускается многопоточная реализация); не используются коммуникационные сети, жесткий диск или хранилища данных; запуски выполняются в условиях, когда вычислительная система обладает достаточными ресурсами для выполнения программы (свободной оперативной памяти достаточно для хранения всех структур данных алгоритма, не запущено других вычислительных программ).

3. Метод решения. Если известны функции y_i, описывающие время выполнения алгоритма, то решение задачи выбора (1) сводится к выбору наименьшего из r значений. Данный раздел статьи посвящен построению функций y_i, т.е. решению задачи (2) оценки времени выполнения.

3.1. Оценка времени выполнения. Задача оценки времени выполнения (2) является частным случаем задачи восстановления регрессии, которая успешно решается широким кругом методов машинного обучения. Метод наименьших квадратов (МНК) [13] позволяет восстанавливать линейную регрессию (при этом допускается нелинейное преобразование параметров) и успешно применяется для задач, в которых восстанавливаемая функция имеет соответствующий вид. Однако зависимость времени выполнения алгоритма от параметров вычислительной системы сложно представить в виде линейной функции с достаточной точностью. Для восстановления подобных зависимостей используются нелинейные методы, такие как метод случайных деревьев [14], метод опорных векторов с нелинейным ядром [15] и нейронные сети [16]. Существенным недостатком этих методов является слабая способность к экстраполяции [17], что делает их малопригодными для решения задачи (2). В нашей работе предлагается использовать подход, который является комбинацией линейной и нелинейной регрессии. Полученное решение обладает способностью к экстраполяции, успешному восстановлению сложной зависимости эффективности выполнения программы от параметров вычислительной системы и не требует подбора параметров или преобразования исходных данных.

Алгоритм обучения состоит из двух этапов.

1. На каждой вычислительной системе i для каждой точки p_{ij} определяются коэффициенты кривой, на которой лежит точка p_{ij}, где $i = \overline{1,n}$ и $j = \overline{1,e_i}$. Иными словами, с помощью линейного МНК восстанавливаются коэффициенты функции $h(p) = C_0 + C_1\varphi_1(p) + C_2\varphi_2(p) + \ldots + C_t\varphi_t(p)$, где φ_i — произвольная функция от параметров алгоритма. Вид функции h соответствует асимптотической оценке сложности алгоритма, который использовался при проведении экспериментов.

2. Строится обучающая выборка из коэффициентов, полученных на предыдущем шаге, и соответствующих характеристик вычислительной системы: $\left(x_i, (C_0^{ij}, C_1^{ij}, \ldots, C_t^{ij})_{j=1}^{e_i}\right)_{i=1}^{n}$. Выполняется обучение нелинейной модели на полученной выборке.

Алгоритм предсказания тоже состоит из двух этапов.

1. По характеристикам вычислительной системы и параметрам алгоритма выполняется восстановление коэффициентов функции $h(p)$.

2. По параметрам алгоритма и полученным на предыдущем шаге коэффициентам вычисляется время выполнения алгоритма.

Отметим, что в линейном МНК, который используется на первом шаге обучения для конкретной вы-

числительной системы, минимизируется рассмотренная выше функция качества $Q(g)$, при этом функция g принимает вид функции h.

Таким образом, предлагаемое решение задачи (2) состоит в том, что выполняется предсказание не значения функции в точке, а коэффициентов функции асимптотической сложности, которая проходит через заданную точку. В результате предлагаемое решение способно выполнять экстраполяцию за пределами обучающих данных.

3.2. Восстановление функции асимптотической сложности. При проведении вычислительных экспериментов могут наблюдаться отклонения времени выполнения алгоритма от асимптотической кривой, что значительно затрудняет восстановление коэффициентов функции асимптотической сложности $h(p)$. Отклонение от асимптотической кривой может быть вызвано как особенностью реализации алгоритма, так и ошибками при измерении времени выполнения программ. Отметим, что МНК, использующийся для восстановления линейной регрессии, успешно подавляет небольшие ошибки измерения, распределенные по нормальному закону.

Таблица 1

Особенности реализации библиотеки MKL на некоторых вычислительных системах

Вычислительная система	Количество итераций основного цикла	Инструкции реализаций
Intel Pentium D 945 2 CPU DualCore AMD Opteron 270	899 002	16 x mulpd/addpd
Intel Core i5 3330 Intel Core i7 3820	968 752	32 x vmulpd/vaddpd
Intel Core 2 Duo E6300 Intel Core 2 Duo E6550	868 001	64 x mulpd/addpd
2 CPU QuadCore Intel Xeon L5630 2 CPU HexaCore Intel Xeon X5670	1 937 501	32 x mulpd/addpd
2 CPU DualCore Intel Xeon 5150	840 001	64 x mulpd/addpd

Многие высокопроизводительные библиотеки перед запуском алгоритма автоматически выбирают его параметры или содержат несколько реализаций этого алгоритма, выбирая лучший для конкретной системы и аргументов задачи.

В табл. 1 приведены результаты проведенного нами анализа исполнения матричного умножения средствами библиотеки MKL на 9 вычислительных системах (умножались квадратные матрицы размером 500×500). Запуск на каждой вычислительной системе выполнен в режиме инструментации с помощью Intel Pin [18]. Для каждого запуска детектирован основной цикл алгоритма умножения матриц, выполнен подсчет его итераций, определен тип векторных инструкций сложения и умножения внутри тела цикла и определено их количество. В итоге, в исследуемой функции умножения матриц библиотеки MKL выявлены 5 различных реализаций, которые отличались набором инструкций (AVX, SSE), структурой матричного умножения [19] (различный размер блока и порядок обхода циклов) и, возможно, типом алгоритма (например, вместо классического алгоритма может использоваться алгоритм Штрассена [20]). Оценка сложности классического алгоритма матричного умножения определяется формулой $2n^3$. На основе данных из таблицы можно получить степень при n, вычислив $\log_{500}\langle$количество операций умножения$\rangle$. Для вычислительных систем, соответствующих первой строке таблицы, количество обра-

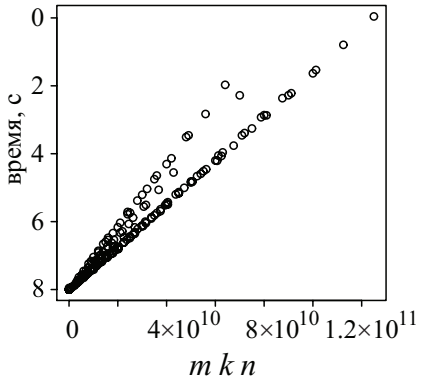

Рис. 1. Время выполнения алгоритма умножения матриц из библиотеки OpenBLAS на одной вычислительной системе в зависимости от произведения размеров матриц

батываемых данных имеет порядок $n^{2.76}$, в то время как для всех остальных n^3.

В рамках одной вычислительной системы функция может исполнять одну из нескольких реализаций, которые оптимизированы для определенного набора параметров (например, для задач малой размерности и для задач большой размерности). В реализациях может быть представлен одинаковый алгоритм, но с различным коэффициентом перед асимптотической оценкой за счет отличий в оптимизации, поэтому время выполнения может отличаться даже для близких параметров (рис. 1). Кроме того, реализация алгоритма может быть оптимизирована только для некоторых параметров, для остальных — реализация менее эффективна.

На первом шаге алгоритма обучения из раздела 3.1 для каждой точки обучающей выборки выполняется восстановление функции, которая описывает результаты экспериментов и проходит через выбранную точку p_{ij} (в дальнейшем эту точку будем называть базовой). Вид восстанавливаемой функции определяется асимптотической сложностью алгоритма, для которого проведены эксперименты. Сначала линейный МНК применяется для всех точек обучающей выборки для текущей вычислительной системы. Если восстановленная функция слабо согласована с обучающими данными, то это свидетельствует о наличии нескольких кривых, описывающих результаты экспериментов (на рис. 1 присутствуют три кривые). Для базовой точки необходимо выделить только те точки, которые лежат с ней на одной прямой. Этого можно достигнуть за счет удаления точек, отклонение которых от восстановленной кривой максимально. Оценка согласованности модели и экспериментальных данных выполняется с помощью коэффициента детерминации с поправкой на число параметров модели: $R^2_{\text{adj}} = R^2 - \left(1 - R^2\right) \dfrac{t}{n - t - 1}$, где n — количество точек обучающей выборки на заданной вычислительной системе, R^2 — коэффициент детерминации [21]. Если величина скорректированного коэффициента R^2_{adj} близка к 1, то модель согласована. На каждой итерации выполняется исключение из обучающей выборки точки с максимальной величиной ошибки с учетом взаимного расположения восстановленной кривой и базовой точки (если значение в базовой точке больше значения на кривой, то исключается точка, расположенная ниже кривой с максимальной величиной отклонения, иначе — выше кривой). Алгоритм фильтрации выполняется до тех пор, пока коэффициент R^2_{adj} не станет больше заданного порога. В результате отфильтрованные данные содержат только те точки, которые вместе с базовой точкой образуют одну кривую. На основе этих точек с помощью МНК восстанавливаются коэффициенты функции асимптотической сложности $h(p)$.

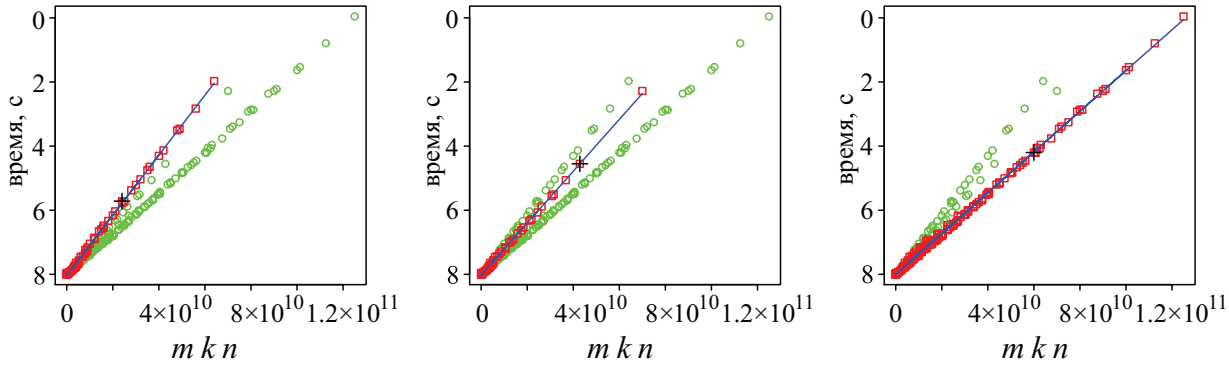

Рис. 2. Время выполнения алгоритма умножения матриц из библиотеки OpenBLAS на одной вычислительной системе в зависимости от произведения $m \times k \times n$ размеров матриц с разными базовыми точками

На рис. 2 показаны базовые точки, а также выделенные для них точки со значением скорректированного коэффициента детерминации 0.999 и полученные кривые. Перекрестие — базовая точка, красные прямоугольники — выделенные точки, использующиеся для восстановления функции асимптотической сложности, зеленые окружности — исключенные точки.

4. Выбор признаков. Предложенный метод оценки времени выполнения алгоритма требует применения двух алгоритмов машинного обучения. Линейный МНК восстанавливает функцию асимптотической сложности, используя только параметры алгоритма. Метод восстановления нелинейной модели предсказывает коэффициенты функции $h(p)$ на новой вычислительной системе на основе параметров алгоритма и характеристик системы. В настоящей статье приводятся результаты экспериментов с использованием 4 алгоритмов и 133 характеристик вычислительных систем, которые рассматриваются ниже.

4.1. Параметры алгоритмов. В табл. 2 приведены примеры алгоритмов, их параметры и вид восстанавливаемой функции, полученный на основе асимптотической оценки сложности алгоритма. Рассмотренные алгоритмы использовались при проведении экспериментов.

Таблица 2

Алгоритмы, их параметры и вид восстанавливаемой функции

Алгоритм	Параметры алгоритма	Вид восстанавливаемой функции
Умножение вещественных матриц, $C = A \times B$, $A \in R^{m \times k}$, $B \in R^{k \times n}$	Размерности матриц: m, k, n	$T = C_1 mkn + C_0$
Сортировка	Размер массива: n	$T = C_1 n \log(n) + C_0$
Решение СЛАУ прямыми методами	Порядок квадратной матрицы: n	$T = C_1 n^3 + C_0$
Быстрое преобразование Фурье	Порядок квадратной матрицы: n	$T = C_1 n^2 \log(n) + C_0$

4.2. Статические характеристики вычислительных систем. Выбор признаков для алгоритма машинного обучения — важная задача, поскольку от их описательной способности зависит точность прогноза. Кроме того, следует учитывать, что не все параметры вычислительной системы могут быть получены или измерены программным путем. В связи с этим был сформирован список признаков вычислительной системы, которые оказывают влияние на предсказание времени.

Статические характеристики вычислительных систем — это такие характеристики, которые не меняются во время работы всей системы и могут быть получены считыванием соответствующей информации из какого-либо хранилища (например, из регистров процессора после выполнения команды cpuid [22]). Ниже приведена часть статических характеристик, которые учитываются в работе как параметры вычислительной системы и собирались во время проведения экспериментов.

1. Пиковая теоретическая производительность вычислительной системы для чисел с плавающей запятой двойной точности (с учетом количества векторных инструкций, которое процессор способен выполнить за такт [23]).
2. Информация о каждом уровне кэш-памяти процессора:
   a) размер кэш-памяти;
   b) является кэш общим для нескольких ядер или принадлежит только одному;
   c) количество портов доступа к кэш-памяти.
3. Тактовая частота процессора.
4. Тактовая частота процессора в turbo-режиме.
5. Тактовая частота системной шины.
6. Информация о подсистеме памяти:
   a) количество каналов доступа;
   b) тайминг памяти: CAS Latency, RAS to CAS Delay, RAS Precharge, Cycle Time.
7. Количество вычислительных ядер, количество процессоров.
8. Включена ли технология Hyper-Threading.
9. Тип процессора: для мобильных устройств, настольный, серверный.
10. Производитель процессора: Intel, AMD.
11. Тип ядра: Ivy Bridge, Sandy Bridge, Conroe, Haswell, Wolfdale, Arrandale, Brisbane, Yorfield, Pineview, Penryn, Lynnfield.
12. Поддерживаемый набор инструкций: AVX, FMA4, SSE5, SSE4a, SSE4.2 и др.

В общей сложности используется 77 статических характеристик. Все характеристики являются либо числовыми, либо бинарными (все строковые характеристики преобразуются к бинарному вектору).

4.3. Измеряемые характеристики вычислительных систем. Статические характеристики не позволяют в достаточной степени описать подсистему памяти (как кэш-памяти, так и оперативной), поэтому требуются дополнительные метрики. Одна из важных характеристик — это количество данных, которое может быть записано в память или прочитано из нее процессором в единицу времени (с учетом иерархии памяти). Эту величину будем называть пропускной способностью памяти.

Измерениям подвергается пропускная способность подсистемы памяти при выполнении операций чтения и записи. Размер обрабатываемого блока задается равным степени двойки в диапазоне от 8 КБ до 64 МБ (14 вариантов). Подобный диапазон позволяет оценить пропускную способность всех элементов иерархии памяти. Доступ к элементам памяти осуществляется как последовательно, так и случайно. Таким образом, выполняется измерение пропускной способности памяти при 4 вариантах доступа к ней:

1) последовательный на запись;
2) последовательный на чтение;
3) случайный на запись;
4) случайный на чтение.

В общей сложности используется 56 измеряемых признаков.

5. Результаты экспериментов. На каждой вычислительной системе измерялось время выполнения алгоритма в зависимости от входных параметров с использованием библиотек MKL [24], OpenBLAS [25], TBB [26], FFTW [27]. В табл. 3 приведены алгоритмы, использующиеся библиотеки, количество запусков, выполненных на каждой вычислительной системе, и диапазон изменения параметров алгоритмов.

Таблица 3

Реализации алгоритмов и их параметры, использующиеся при проведении экспериментов

Алгоритм	Реализация	Количество запусков на одной системе	Параметры алгоритма	
			Минимальные	Максимальные
Умножение матриц	MKL, OpenBLAS	200	100×100	5000×5000
Решение СЛАУ	MKL, OpenBLAS	46	100	10 000
Сортировка	MKL, TBB	49	100 000	40 000 000
Двухмерное быстрое преобразование Фурье	MKL, FFTW	30	100×100	5000×5000

Статистика запусков получена на 84 вычислительных системах, которые построены на процессорах разных поколений (от Pentium 4 до современных на базе ядра Haswell), разных производителей (Intel, AMD), разного назначения (мобильные, настольные, серверные) и, как следствие, существенно отличающейся производительности. Так, решение СЛАУ с 10 000 переменными с помощью библиотеки MKL занимало от нескольких секунд (на процессоре Intel Core i5-4570) до 1500 секунд (на процессоре Intel Atom N475).

Каждый эксперимент состоял из двух этапов. Сначала оценивалось время выполнения алгоритма. Далее выполнялся выбор самой быстрой реализации на основе предсказанных времен выполнения. Для сравнения с общепринятыми алгоритмами оценки часть экспериментов выполнялась при обучении только на данных с одной вычислительной системы. Основной эксперимент состоял в оценке времени выполнения алгоритма на целевой вычислительной системе по статистике запусков этого алгоритма на других системах. Кроме того, было выполнено сравнение предложенного алгоритма выбора наиболее эффективной реализации с простыми подходами, часто применяющимися на практике.

Для оценки качества предсказания времени выполнения и выбора реализаций на одной вычислительной системе использовались следующие характеристики.

1. Средняя абсолютная ошибка предсказанных времен выполнения алгоритма и реальных (MAE — Mean Absolute Error).

2. Среднеквадратическое отклонение (RMSE — Root-Mean-Square Error) предсказанных времен выполнения алгоритма и реальных:

$\mathrm{RMSE} = \sqrt{\dfrac{1}{t}\sum\limits_{i=1}^{t}\big(g(p_i) - f(p_i)\big)^2}$, t — количество точек тестовой выборки.

3. Средний процент относительной ошибки (MRE — Mean Relative Error) для реализаций, работающих больше 1 секунды:

$\mathrm{MRE} = \dfrac{100}{t}\sum\limits_{i=1}^{t}\dfrac{\big|g(p_i) - f(p_i)\big|}{f(p_i)}$, t — количество точек тестовой выборки, $f(p_i) > 1$.

4. Процент правильно выбранных реализаций, т.е. тех, которые соответствуют минимальному времени выполнения алгоритма (CP).

5. Процент потери времени из-за ошибочного выбора реализации относительно оптимального выбора:

$\mathrm{RAL} = 100\left(\sum\limits_{i=1}^{t}\big(g(p_i) - s(p_i)\big)\right)\left(\sum\limits_{i=1}^{t} s(p_i)\right)^{-1}$, $s(p_i)$ — минимальное время выполнения среди всех реализаций для заданных параметров, t — количество точек тестовой выборки.

Отметим, что при вычислении MRE реализации, работающие менее 1 секунды, не рассматриваются, так как для них погрешность измерения времени может быть сопоставима со временем выполнения программы. В результате, значение MRE может достигать очень больших величин, в то время как абсолютная величина ошибки MAE достаточно мала.

Для восстановления нелинейной регрессии в предложенном в работе подходе оценки времени выполнения использовался случайный лес (500 регрессионных деревьев) — Random Forest [28]. Кроме того, были проведены эксперименты с использованием метода опорных векторов с нелинейным ядром и нейронных сетей, но полученные результаты оказались хуже, чем при использовании случайного леса, поэтому приведены результаты только для последнего метода. Во всех экспериментах при восстановлении коэффициентов функции использовался порог скорректированного коэффициента детерминации, равный 0.999.

В табл. 4 представлены средние метрики качества для четырех алгоритмов оценки времени выполнения при обучении на данных с одной вычислительной системы на задаче умножения матриц. Задача выбора успешно решается на одной вычислительной системе, поэтому для сравнения были выбраны три алгоритма, которые используются на практике [12]: линейная регрессия, гребневая регрессия и случайный лес. Результаты приведены только для задачи умножения матриц, поскольку только для этой задачи достаточно экспериментальных данных. В качестве обучающей выборки в первом эксперименте использовались данные, соответствующие задачам малой размерности: матрицы до 1000×1000 (36 точек). Во втором эксперименте обучающая выборка содержала половину экспериментальных данных, выбранных случайным образом (на каждой вычислительной системе выборка была одинаковая).

Таблица 4

Средние метрики качества предсказания и выбора при использовании общепринятых алгоритмов оценки времени выполнения и предложенного в работе подхода при обучении на данных с одной вычислительной системы на задаче умножения матриц

Обучающая выборка	Алгоритм оценки времени	Реализация	MAE, с.	MRE, %	RMSE, с.	CP, %	RAL, %
Матрицы до 1000×1000	Линейная регрессия	MKL	**0.42**	**12.1**	**0.71**	92.1	1.37
		OpenBLAS	**0.42**	**10.9**	**0.69**	92.1	1.37
	Гребневая регрессия	MKL	1.68	26.9	2.46	84.1	3.41
		OpenBLAS	1.51	32.1	2.17	84.1	3.41
	Случайный лес	MKL	5.97	96.3	8.01	**92.9**	**1.13**
		OpenBLAS	4.70	96.45	6.28	**92.9**	**1.13**
	Предлагаемый подход	MKL	0.44	13.48	0.73	85.3	2.11
		OpenBLAS	0.44	12.08	0.73	85.3	2.11
50 % экспериментальных данных (случайная выборка)	Линейная регрессия	MKL	0.17	6.31	0.28	89.4	0.48
		OpenBLAS	0.15	4.80	0.24	89.4	0.48
	Гребневая регрессия	MKL	0.21	7.70	0.33	84.7	0.70
		OpenBLAS	0.17	5.73	0.26	84.7	0.70
	Случайный лес	MKL	0.56	16.7	0.82	**92.8**	0.48
		OpenBLAS	0.44	16.6	0.64	**92.8**	0.48
	Предлагаемый подход	MKL	**0.15**	**5.71**	**0.26**	90.0	**0.44**
		OpenBLAS	**0.13**	**4.24**	**0.22**	90.0	**0.44**

Лучшие результаты по оценке времени выполнения показывают линейная регрессия и предлагаемый в работе подход. Гребневая регрессия и случайный лес демонстрируют бóльшую величину ошибки при обучении на данных, соответствующих задачам малой размерности. Однако все алгоритмы показывают достаточно высокие показатели качества выбора и низкую величину потерь. Таким образом, предлагаемый в работе подход не уступает существующим при решении задачи оценки времени выполнения и выбора лучшей реализации при обучении на данных с одной вычислительной системы.

Основной эксперимент состоял в оценке времени выполнения алгоритма на целевой вычислительной системе по статистике запусков этого алгоритма на других системах. Иными словами, выполнялся скользящий контроль по отдельным объектам [10] (каждая вычислительная система последовательно выступала в роли тестовой выборки, а в роли обучающей — все остальные).

Для оценки времени выполнения реализации алгоритма на целевой вычислительной системе использовалось два подхода на основе методов восстановления нелинейной регрессии:

— предсказание времени выполнения алгоритма на основе характеристик вычислительной системы и параметров алгоритма с использованием случайного леса;

— предсказание коэффициентов функции асимптотической сложности, описывающих время выполнения алгоритма; вычисление по предсказанным коэффициентам времени выполнения.

В табл. 5 представлены средние метрики качества по каждой библиотеке и каждому алгоритму. Эксперименты приведены для двух ситуаций: в обучающей выборке нет данных о времени выполнения алгоритма на целевой вычислительной системе (столбец “No test”), в обучающей выборке есть данные (добавлялось 5 случайных точек — из тестовой выборки эти точки исключались) о времени выполнения алгоритма на целевой вычислительной системе (столбец “Test”). Буквой “T” обозначен алгоритм, выполняющий предсказание времени выполнения алгоритма, а буквой “C” — алгоритм, выполняющий предсказание коэффициентов асимптотических кривых.

Таблица 5

Средние метрики качества предсказания времени выполнения и выбора наиболее эффективной реализации при обучении на данных из всего диапазона изменения параметров алгоритмов

Алгоритм	Реализация	MAE, с.				MRE, %				RMSE, с.				CP, %				RAL, %			
		No test		Test		No test		Test		No test		Test		No test		Test		No test		Test	
		T	C	T	C	T	C	T	C	T	C	T	C	T	C	T	C	T	C	T	C
Умножение матриц	MKL	2.85	2.97	1.43	**0.21**	50.0	46.0	29.8	**7.47**	3.93	4.42	2.36	**0.47**	80.9	80.2	79.2	**86.3**	3.19	3.98	1.89	**0.89**
	OpenBLAS	1.21	1.29	0.80	**0.16**	30.1	30.1	21.8	**4.40**	1.78	1.95	1.32	**0.39**	80.9	80.2	79.2	**86.3**	3.19	3.98	1.89	**0.89**
Решение СЛАУ	MKL	10.9	8.63	9.74	**1.03**	85.4	37.4	107	**10.1**	25.7	24.8	21.2	**3.18**	81.6	75.5	**82.3**	80.8	3.56	3.70	**1.50**	1.66
	OpenBLAS	5.07	4.41	4.08	**0.58**	47.7	34.0	59.2	**6.98**	10.4	10.4	7.39	**1.80**	81.6	75.5	**82.3**	80.8	3.56	3.70	**1.50**	1.66
Сортировка	MKL	0.62	0.55	0.72	**0.11**	12.3	12.5	17.6	**1.63**	1.60	1.58	1.38	**0.76**	98.6	98.4	**99.6**	98.5	0.67	0.06	**0.02**	0.05
	TBB	0.15	0.33	0.14	**0.03**	7.57	23.0	7.48	**2.43**	0.30	0.65	0.30	**0.09**	98.6	98.4	**99.6**	98.5	0.67	0.06	**0.02**	0.05
БПФ	MKL	0.14	0.12	0.14	**0.06**	24.5	29.5	22.7	**14.7**	0.38	0.35	0.38	**0.21**	81.2	81.8	**83.4**	82.6	6.98	9.31	3.62	**3.31**
	FFTW	0.14	0.13	0.15	**0.09**	**21.2**	25.6	21.6	22.3	0.30	0.30	0.29	**0.22**	81.2	81.8	**83.4**	82.6	6.98	9.31	3.62	**3.31**

Алгоритм предсказания коэффициентов асимптотических кривых и алгоритм предсказания времени показывают близкие показатели качества при отсутствии в обучающей выборке данных с целевой вычислительной системы. Однако добавление всего 5 точек с целевой вычислительной системы при использовании алгоритма “C” приводит к существенному уменьшению абсолютной и относительной ошибок в отличие от алгоритма “T”. Так, при использовании алгоритма “C” для умножения матриц средняя абсолютная ошибка MAE уменьшается в 14 и 8 раз для библиотек MKL и OpenBLAS соответственно, а средняя относительная ошибка MRE уменьшается более чем в 6 раз для обеих библиотек. Каждый из алгоритмов оценки времени выполнения позволяет выполнять выбор наиболее эффективной реализации с точностью более 80%. Отметим, что большая величина относительной ошибки для быстрого преобразования Фурье связана с тем, что время выполнения алгоритма для большинства систем не превышает 1 секунды. В результате, метрика MRE вычисляется на основании малого количества данных, которые наблюдаются на малом количестве вычислительных систем.

В табл. 6 приведены результаты экспериментов для ситуации, когда обучение выполнялось на данных, соответствующих задачам малой размерности, а тестовая выборка содержала точки из всего диапазона изменения параметров алгоритма. На каждой вычислительной системе для задачи матричного умножения для обучения использовалось 36 точек (матрицы до 1000×1000), для решения СЛАУ — 18 точек (системы до 3000 неизвестных), для сортировки — 10 точек (массивы до 1 000 000 элементов), для быстрого преобразования Фурье — 10 точек (двухмерные массивы до 1000×1000). Столбец “Test” содержит результаты экспериментов для случая, когда в обучающую выборку добавлялось 5 случайных точек с целевой вычислительной системы (как и ранее, эти точки соответствуют задачам малой размерности).

Результаты, представленные в табл. 6, демонстрируют, что предложенный подход способен строить качественную модель даже при обучении на данных малой размерности (в отличие от алгоритма предсказания времени). При этом, как и в табл. 5, добавление 5 точек, соответствующих задачам малой размерно-

Таблица 6

Средние метрики качества предсказания времени выполнения и выбора наиболее эффективной реализации при обучении на данных, соответствующих малой размерности задачи

Алгоритм	Реализация	MAE, с.				MRE, %				RMSE, с.				CP, %				RAL, %			
		No test		Test		No test		Test		No test		Test		No test		Test		No test		Test	
		T	C	T	C	T	C	T	C	T	C	T	C	T	C	T	C	T	C	T	C
Умножение матриц	MKL	4.88	2.98	5.00	**0.39**	95.7	49.0	95.0	**11.2**	7.24	4.44	8.28	**0.75**	**85.7**	81.8	84.4	**82.1**	2.86	3.18	**3.14**	3.28
	OpenBLAS	3.82	1.27	3.92	**0.39**	95.2	33.9	95.3	**11.6**	5.67	1.93	6.20	**0.71**	**85.7**	81.8	84.4	**82.1**	2.86	3.18	**3.14**	3.28
Решение СЛАУ	MKL	16.3	9.83	18.2	**1.99**	82.1	45.8	85.2	**17.9**	41.2	28.6	44.2	**4.83**	84.9	82.2	**85.7**	84.6	2.38	4.79	**2.23**	3.36
	OpenBLAS	13.1	5.15	14.7	**1.97**	80.1	38.7	82.8	**15.4**	27.5	11.8	29.4	**4.40**	84.9	82.2	**85.7**	84.6	2.38	4.79	**2.23**	3.36
Сортировка	MKL	3.15	0.52	3.51	**0.22**	95.6	10.9	95.5	**4.81**	4.88	1.52	5.21	**0.99**	99.6	99.6	99.7	**99.7**	0.02	0.02	0.02	**0.02**
	TBB	1.15	0.32	1.28	**0.10**	96.9	23.8	96.6	**8.07**	1.79	0.60	1.93	**0.19**	99.6	99.6	99.7	**99.7**	0.02	0.02	0.02	**0.02**
БПФ	MKL	0.36	0.23	0.43	**0.17**	96.8	46.9	96.8	**38.2**	0.85	0.56	1.00	**0.46**	84.4	82.1	88.8	**90.2**	6.84	8.30	5.91	**5.62**
	FFTW	0.47	0.17	0.56	**0.16**	96.7	36.4	96.5	**29.8**	0.93	0.40	1.07	**0.37**	84.4	82.1	88.8	**90.2**	6.84	8.30	5.91	**5.62**

сти, с целевой вычислительной системы приводит к существенному улучшению качества предсказания и, как следствие, к лучшему выбору реализаций. Средняя абсолютная ошибка MAE уменьшается более чем в 7 раз для библиотеки MKL, более чем в 3 раза для OpenBLAS. Средняя относительная ошибка MRE уменьшается более чем в 4 раза для библиотеки MKL, почти в 3 раза для OpenBLAS. Следует отметить, что MRE для алгоритма предсказания времени "T" составляет почти 100% из-за того, что этот метод обладает слабой способностью к экстраполяции.

При наличии данных с целевой вычислительной системы в обучающей выборке алгоритм "C" оценивает время выполнения значительно точнее, чем алгоритм "T". Тем не менее, оба алгоритма осуществляют выбор лучшей реализации в более чем 80% случаев. Использующиеся реализации из библиотек таковы, что в большинстве случаев при любых значениях параметров запуска одна библиотека имеет более эффективную реализацию, чем другая на конкретной вычислительной системе. За счет данной особенности экспериментальных данных использование алгоритма "T" позволяет выполнять выбор наиболее эффективной реализации с высокой точностью, несмотря на бóльшую погрешность оценки.

В табл. 7 представлены метрики качества простых алгоритмов выбора, применяющихся на практике, в сравнении с предложенном в работе подходом на задаче умножения матриц.

Таблица 7

Средние метрики качества выбора наиболее эффективной реализации в сравнении с простыми алгоритмами выбора

Стратегия выбора реализации			CP, %	RAL, %
Только MKL			73.6	47.2
Только OpenBLAS			26.3	15.5
На процессорах Intel: MKL, на процессорах AMD: OpenBLAS			80.1	44.1
Предлагаемый подход	Обучение на данных, соответствующих задачам малой размерности	No test	81.8	3.18
		Test	82.1	3.28
	Обучение на данных из всего диапазона изменения параметров	No test	80.2	3.98
		Test	**86.3**	**0.89**

Стратегия, которая заключается в использовании на вычислительных системах с процессорами Intel библиотеки MKL, а на AMD — OpenBLAS, позволяет выбирать в более чем 80% случаев наиболее эффективную реализацию. Однако средние потери времени от ошибочного выбора при такой стратегии превосходят 44%. Предлагаемый в работе подход демонстрирует процент потери времени менее 4%, несмотря на то что точность выбора может быть сопоставима с простыми стратегиями. Это достигается за счет того, что ошибочный выбор в большинстве случаев проявляется на запусках, которые работают сопоставимое время и содержат ошибки измерения. В подобной ситуации может быть выбрана более медленная реализация, но сама величина ошибки является небольшой. В то же время простые стратегии могут совершать большие ошибки, выбирая значительно более медленную реализацию по сравнению с оптимальной.

На рис. 3а представлена диаграмма средней абсолютной ошибки алгоритма предсказания коэффициентов асимптотических кривых для задачи умножения матриц с использованием библиотеки MKL по каждой вычислительной системе. Обучение алгоритма выполнялось на данных из всего диапазона изменения параметров тестовой выборки. На рис. 3б показано уменьшение средней абсолютной ошибки после добавления 5 точек целевой системы в обучающую выборку.

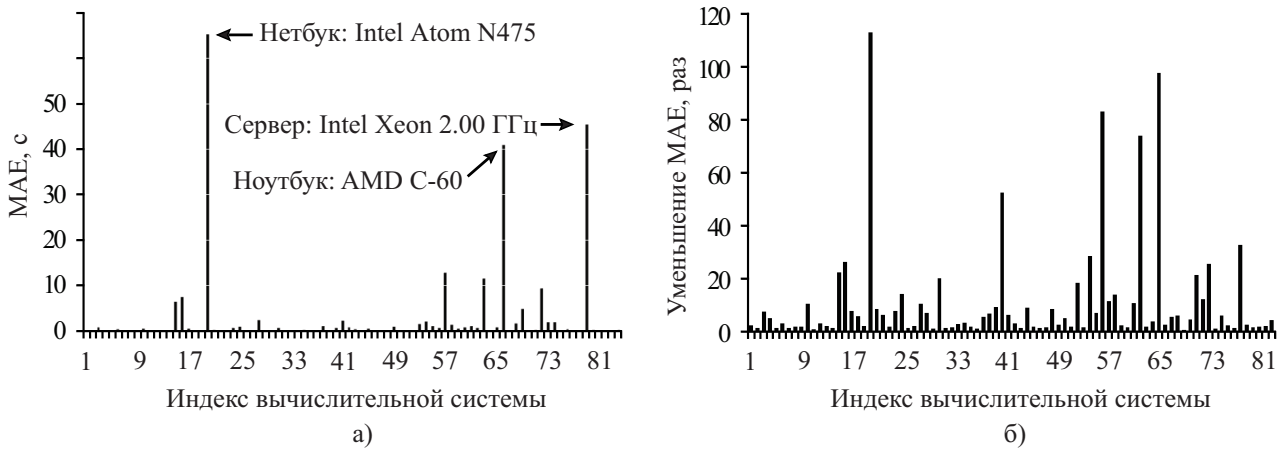

Рис. 3. Средняя абсолютная ошибка (а) и ее уменьшение (б) после добавления данных с целевой вычислительной системы в обучающую выборку алгоритма предсказания коэффициентов асимптотических кривых для задачи умножения матриц с использованием библиотеки MKL по каждой вычислительной системе

На системах, которые широко представлены в обучающей выборке (на основе вычислительных ядер Ivy Bridge, Sandy Bridge, Haswell), средняя абсолютная ошибка MAE достаточно мала и составляет для этих систем 0.32 с. На ряде вычислительных систем ошибка предсказания велика из-за того, что эти системы были плохо представлены в обучающей выборке (например, системы на базе процессоров AMD, нетбуки, сервера). Добавление в обучающую выборку небольшого количества данных с целевой вычислительной системы позволяет значительно уменьшить ошибку как при обучении на данных малой размерности, так и при обучении на данных из всего диапазона изменения параметров алгоритма. Абсолютная ошибка может уменьшаться более чем в 100 раз для систем, плохо представленных в обучающей выборке (рис. 3б).

6. Заключение. В настоящей статье рассмотрена задача выбора наиболее эффективной реализации алгоритма на основе оценки времени выполнения программы на вычислительной системе по результатам экспериментов на других вычислительных системах. Построена модель оценки времени выполнения программы в виде функциональной зависимости времени выполнения от параметров алгоритма и характеристик вычислительной системы. Выделено 133 признака вычислительных систем, по которым восстанавливалась функциональная зависимость: 77 статических признаков описывают вычислительные способности процессора, 56 измеряемых признаков описывают пропускную способность подсистемы памяти. Предложен метод оценки времени выполнения программ на основе предсказания коэффициентов асимптотических кривых с использованием комбинации методов восстановления регрессии: линейный метод наименьших квадратов и метод восстановления нелинейной модели. Эксперименты на 84 вычислительных системах и 4 библиотеках с использованием случайного леса показали, что средняя относительная ошибка предсказания не превосходит 8% для матричного умножения, 3% для сортировки, 11% для решения СЛАУ и 23% для быстрого преобразования Фурье. Предложенный подход позволяет выбрать наиболее эффективную реализацию в более чем 80% случаев, а потери времени от ошибочного выбора не превосходят 6%.

Рассмотренный в работе подход способен выполнять экстраполяцию за пределами тестовой выборки и, как следствие, успешно применяться при наличии небольшого количества экспериментальных данных для задач малой размерности. Эксперименты при обучении на данных малой размерности показали, что средняя относительная ошибка предсказания не превосходит 12% для матричного умножения, 9% для сортировки, 18% для решения СЛАУ и 39% для быстрого преобразования Фурье.

Оценка производительности вычислительных систем — важная задача, которая решается сегодня путем тестирования вычислительных систем с использованием тестовых пакетов Linpack [29], NAS Benchmark [30], Graph 500 Benchmark [31] и др. Предложенный в работе подход позволяет значительно сократить как количество экспериментов, так и размерность задач, которые решаются при оценке производительно-

сти вычислительной системы. Кроме того, оценка времени выполнения алгоритмов по параметрам вычислительной системы позволяет определить, насколько вычислительная система эффективна при решении определенного класса задач без проведения экспериментов на ней. Предложенное решение может применяться при автоматической настройке библиотеки перед ее использованием (подобно автоматической настройке в библиотеке ATLAS [32]).

Предложенный алгоритм оценки времени выполнения программ и выбора наиболее эффективной реализации является также основой для автоматического выбора наиболее эффективной реализации в подходе к макромодульной разработке программ, изложенном в [33].

Полученные результаты демонстрируют работоспособность подхода. Тем не менее, представляет интерес задача оценки применимости других методов восстановления регрессии и задача расширения и уточнения признаков вычислительных систем для еще более точного построения оценок: выбор значимых признаков и добавление новых.

Авторы благодарят Турлапова В.Е., Меерова И.Б. и Золотых Н.Ю. за полезные обсуждения и внимание к работе.

список литературы

1. *Rice J.R.* The algorithm selection problem // Advances in Computers. 1976. **15**. 65–118.
2. *Fink E.* How to solve it automatically: selection among problem-solving methods // Proc. of the Fourth International Conference on Artificial Intelligence Planning Systems. Palo Alto: AAAI Press, 1998. 128–136.
3. *Roberts M., Howe A.* Learning from planner performance // Artificial Intelligence. 2009. **173**, N 5–6. 536–561.
4. *Howe A.E., Dahlman E., Hansen C., Scheetz M., von Mayrhauser A.* Exploiting competitive planner performance // Lecture Notes in Computer Science. Vol. 1809. Heidelberg: Springer, 2000. 62–72.
5. *Gomes C.P., Selman B., Crato N., Kautz H.* Heavy-tailed phenomena in satisfiability and constraint satisfaction problems // Journal of Automated Reasoning. 2000. **24**, N 1. 67–100.
6. *Xu L., Hutter F., Hoos H.H., Leyton-Brown K.* SATzilla2009: an automatic algorithm portfolio for SAT. Solver description. SAT competition 2009 (http://www.satcompetition.org/2009/spec2009.html).
7. *Gagliolo M., Schmidhuber J.* Learning dynamic algorithm portfolios // Annals of Mathematics and Artificial Intelligence. 2006. **47**, N 3–4. 295–328.
8. *Hutter F., Xu L., Hoos H.H., Leyton-Brown K.* Algorithm runtime prediction: methods & evaluation // Artificial Intelligence. 2014. **206**. 79–111.
9. *Brewer E.A.* High-level optimization via automated statistical modeling // Proc. of the 5th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPOPP-95). Vol. 30, Issue 8. New York: ACM Press, 1995. 80–91.
10. *Hastie T., Tibshirani R., Friedman J.* The elements of statistical learning: data mining, inference, and prediction. New York: Springer, 2001.
11. *Rasmussen C.E., Williams C.K.I.* Gaussian processes for machine learning. Cambridge: MIT Press, 2006.
12. *Kotthoff L., Gent I.P., Miguel I.* An evaluation of machine learning in algorithm selection for search problems // Artificial Intelligence Communications. 2012. **25**, N 3. 257–270.
13. *Charnes A., Frome E.L., Yu P.L.* The equivalence of generalized least squares and maximum likelihood estimates in the exponential family // Journal of the American Statistical Association. 1976. **71**, N 353. 169–171.
14. *Breiman L.* Random forests // Machine Learning. 2001. **45**, N 1. 5–32.
15. *Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P.* Numerical recipes: the art of scientific computing. New York: Cambridge Univ. Press, 2007.
16. *Broomhead D.S., Lowe D.* Multivariable functional interpolation and adaptive networks // Complex Systems. 1988. **2**, N 3. 321–355.
17. *Marcus G.F.* Rethinking eliminative connectionism // Cognitive Psychology. 1998. **37**, N 3. 243–282.
18. Pin — A Dynamic Binary Instrumentation Tool. (https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool; дата обращения: 22.08.2014).
19. *Goto K., van de Geijn R.A.* Anatomy of high-performance matrix multiplication // ACM Transactions on Mathematical Software. 2008. **34**, N 3. 1–25.
20. *Cormen T.H., Leiserson C.E., Rivest R.L., Stein C.* Introduction to algorithms. Cambridge: MIT Press, 2009.
21. *Дрейпер Н., Смит Г.* Прикладной регрессионный анализ. М.: Издательский дом “Вильямс”, 2007.
22. Intel®64 and IA-32 Architectures Software DeveloperTs Manual. Volume 2A: Instruction Set Reference, A–M. 2014 (http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-2a-manual.html; дата обращения: 22.08.2014).
23. *Agner F.* Instruction tables: lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs (http://www.agner.org/optimize/instruction_tables.pdf; дата обращения 17.07.2014).
24. Intel Math Kernel Library 11.1 (https://software.intel.com/en-us/intel-mkl; дата обращения: 6.08.2014).
25. OpenBLAS 0.2.9 (http://www.openblas.net; дата обращения: 15.07.2014).

26. Intel Threading Building Blocks 4.2 (https://www.threadingbuildingblocks.org; дата обращения: 6.08.2014).
27. FFTW 3.3.4 (http://www.fftw.org; дата обращения: 6.08.2014).
28. randomForest: Breiman and Cutler's random forests for classification and regression (http://cran.r-project.org/web/packages/randomForest/index.html; дата обращения: 6.08.2014).
29. *Dongarra J.J., Luszczek P., Petitet A.* The LINPACK benchmark: past, present and future // Concurrency and Computation: Practice and Experience. 2003. **15**, N 9. 803–820.
30. NAS Parallel Benchmarks (http://www.nas.nasa.gov/publications/npb.html; дата обращения: 22.08.2014).
31. Graph 500 Benchmark (http://www.graph500.org/specifications; дата обращения: 22.08.2014).
32. Automatically Tuned Linear Algebra Software (ATLAS) (http://math-atlas.sourceforge.net; дата обращения: 15.07.2014).
33. *Гергель В.П., Сиднев А.А.* Методы и программные средства макромодульной разработки программ // Вестник Нижегородского университета им. Н.И. Лобачевского. 2012. № 2. 294–300.

Поступила в редакцию
24.08.2014

Automatic Selection of the Fastest Algorithm Implementations

A. A. Sidnev [1] **and V. P. Gergel** [2]

[1] *Lobachevsky State University of Nizhni Novgorod, Faculty of Computational Mathematics and Cybernetics; prospekt Gagarina 23, Nizhni Novgorod, 603950, Russia; Assistant, e-mail: sidnev@vmk.unn.ru*

[2] *Lobachevsky State University of Nizhni Novgorod, Faculty of Computational Mathematics and Cybernetics; prospekt Gagarina 23, Nizhni Novgorod, 603950, Russia; Dean of Faculty, Dr. Sci., Professor, e-mail: gergel@unn.ru*

Received August 24, 2014

Abstract: We propose an approach for the runtime prediction of distributed high-performance computation. This approach does not require experimentation on all target computer systems. The selection of an optimal algorithm is performed according to the asymptotic complexity of the algorithms under evaluation using machine learning methods. The proposed approach can significantly reduce the number of experiments and the dimension of the problems to be solved during the process of evaluating the performance of a computer system. The evaluation of algorithm execution time based on the known parameters of the system allows determining the computer system efficiency for solving certain classes of problems without performing experiments on it. This allows one to quickly update the forecast by a minimum number of experiments with small-size tasks on the target computer system. The proposed solution can be used for the automatic library turning before using it (like the autosetting in the ATLAS (Automatically Tuned Linear Algebra Software) library. A comparative analysis of runtime prediction results obtained when solving several problems on 84 computers is given. The use of a random forest combined with the linear least square method shows the average relative error of the estimated execution time 17% for the training data corresponding to the problems of small dimension and the average relative error 9% when training was performed on data from the entire range of algorithm parameters in the test samples. The resulting estimates allow one to select the most efficient implementation of the algorithm in more than 80% of cases.

Keywords: selection of algorithm implementation, program running time, asymptotic estimates of complexity, characteristics of computing systems, machine learning, regression analysis.

References

1. J. R. Rice, "The Algorithm Selection Problem," Adv. Comput. **15**, 65–118 (1976).

2. E. Fink, "How to Solve It Automatically: Selection Among Problem-Solving Methods," in *Proc. 4th Int. Conf. on Artificial Intelligence Planning Systems* (AAAI Press, Palo Alto, 1998), pp. 128–136.

3. M. Roberts and A. Howe, "Learning from Planner Performance" Artif. Intell. **173** (5–6), 536–561 (2009).

4. A. E. Howe, E. Dahlman, C. Hansen, M. Scheetz, and A. von Mayrhauser, "Exploiting Competitive Planner Performance," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2000), Vol. 1809, pp. 62–72.

5. C. P. Gomes, B. Selman, N. Crato, and H. Kautz, "Heavy-Tailed Phenomena in Satisfiability and Constraint Satisfaction Problems," J. Automat. Reason. **24** (1), 67–100 (2000).

6. L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "SATzilla2009: An Automatic Algorithm Portfolio for SAT. Solver Description," http://www.satcompetition.org/2009/spec2009.html. Cited August 11, 2014.

7. M. Gagliolo and J. Schmidhuber, "Learning Dynamic Algorithm Portfolios," Ann. Math. Artif. Intell. **47** (3–4), 295–328 (2006).

8. F. Hutter, L. Xu, H. H. Hoos, K. Leyton-Brown, "Algorithm Runtime Prediction: Methods & Evaluation," Artif. Intell. **206**, 79–111 (2014).

9. E. A. Brewer, "High-Level Optimization via Automated Statistical Modeling," in *Proc. 5th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming* (ACM Press, New York, 1995), Vol. 30, Issue 8, pp. 80–91.

10. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction* (Springer, New York, 2001).

11. C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (MIT Press, Cambridge, 2006).

12. L. Kotthoff, I. P. Gent, and I. Miguel, "An Evaluation of Machine Learning in Algorithm Selection for Search Problems," AI Commun. **25** (3), 257–270 (2012).

13. A. Charnes, E. L. Frome, and P. L. Yu, "The Equivalence of Generalized Least Squares and Maximum Likelihood Estimates in the Exponential Family," J. Am. Stat. Assoc. **71** (353), 169–171 (1976).

14. L. Breiman, "Random Forests," Mach. Learn. **45** (1), 5–32 (2001).

15. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific computing* (Cambridge Univ. Press, New York, 2007).

16. D. S. Broomhead and D. Lowe, "Multivariable Functional Interpolation and Adaptive Networks," Comp. Syst. **2** (3), 321–355 (1988).

17. G. F. Marcus, "Rethinking Eliminative Connectionism," Cogn. Psychol. **37** (3), 243–282 (1998).

18. Pin — A Dynamic Binary Instrumentation Tool. https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool. Cited August 22, 2014.

19. K. Goto and R. A. van de Geijn, "Anatomy of High-Performance Matrix Multiplication," ACM Trans. Math. Softw. **34** (3), 1–25 (2008).

20. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (MIT Press, Cambridge, 2009).

21. N. R. Draper and H. Smith, *Applied Regression Analysis* (New York, Wiley, 1998; Vil'yams, Moscow, 2007).

22. Intel®64 and IA-32 Architectures Software Developer's Manual. Volume 2A: Instruction Set Reference. http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-2a-manual.html. Cited August 22, 2014.

23. F. Agner, "Instruction Tables: Lists of Instruction Latencies, Throughputs and Micro-Operation Breakdowns for Intel, AMD and VIA CPUs," http://www.agner.org/optimize/instruction_tables.pdf. Cited July 17, 2014.

24. Intel Math Kernel Library 11.1. https://software.intel.com/en-us/intel-mkl. Cited August 6, 2014.

25. OpenBLAS 0.2.9. http://www.openblas.net. Cited July 15, 2014.

26. Intel Threading Building Blocks 4.2. https://www.threadingbuildingblocks.org. Cited August 6, 2014.

27. FFTW 3.3.4. http://www.fftw.org. Cited August 6, 2014.

28. RandomForest: Breiman and Cutler's Random Forests for Classification and Regression. http://cran.r-project.org/web/packages/randomForest/index.html. Cited August 16, 2014.

29. J. J. Dongarra, P. Luszczek, and A. Petitet, "The LINPACK Benchmark: Past, Present and Future," Concurr. Comp.-Pract. E. **15** (9), 803–820 (2003).

30. NAS Parallel Benchmarks. http://www.nas.nasa.gov/publications/npb.html. Cited August 22, 2014.

31. Graph 500 Benchmark. http://www.graph500.org/specifications. Cited August 22, 2014.

32. Automatically Tuned Linear Algebra Software (ATLAS). http://math-atlas.sourceforge.net. Cited July 15, 2014.

33. V. P. Gergel and A. A. Sidnev, "Methods and Software of Macromodule Programming," Vestn. Lobachevsky Gos. Univ. Nizhni Novgorod, No. 5, 294–300 (2012).