ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ т. 9, №4(39), с. 35–52

ББК 24.стд1-53
ГРНТИ 50.39.19, 53.03.13
УДК 004.75

И. А. Чернов, Н. Н. Никитина

Эффективное сканирование пространства параметров в Desktop Grid для идентификации модели разложения гидридов металлов

Аннотация. В работе рассматривается задача идентификации параметров модели разложения гидридов металлов методом параллельного сканирования пространства параметров. Задача является вычислительноемкой, но при этом хорошо подходит для выполнения в среде Desktop Grid. Выбор заданий рассматривается как математическая игра. В работе показано, что процесс поиска можно оптимизировать таким образом, чтобы получать решения быстрее по сравнению со случайным поиском, не требуя дополнительных накладных расходов.

Ключевые слова и фразы: Desktop Grid, планирование заданий, BOINC, Enterprise Desktop Grid, высокопроизводительные вычисления, распределенные вычисления, моделирование разложения гидридов металлов.

1. Введение

Обратные задачи идентификации параметров на основе экспериментальных данных имеют высокую актуальность в материаловедении и физической химии. Способность модели воспроизводить наблюдения — главный критерий ее качества. Обычно модель имеет множество параметров, обладающих физическим смыслом. Нелинейное взаимодействие различных процессов делает независимое определение параметров методами суперпозиции сложным или невозможным. Распространенный подход выбора единственной ограничивающей реакции может быть применен только в том случае, когда данная реакция протекает значительно медленнее остальных, что является редкостью для процессов, близких к равновесному состоянию. Хорошая

Работа выполнена при финансовой поддержке РФФИ в рамках научных проектов №16-07-00622-А и №18-07-00628-А.

10.25209/2079-3316-2018-9-4-35-52

модель должна соответствовать не одной, а целому ряду экспериментальных кривых, тем лучше, чем более близки наборы параметров. Основанная на базовых принципах (например, законах сохранения) и описаниях элементарных реакций, модель обычно содержит более одного параметра.

Модель — это функция M, которая преобразует набор параметров (область значений D из конечномерного пространства R^k) в кривую (непрерывную функцию или, в случае численной модели, массив). В материаловедении физические явления описываются дифференциальными уравнениями, обыкновенными или в частных производных. Некоторая функция от фазовых переменных описывает измеряемую величину как функцию от времени.

Оценка этой функции M в заданной точке области D — это прямая задача. Обратная задача, называемая также идентификацией параметров, заключается в поиске таких точек из D, которые приближают заданные экспериментальные кривые. Если кривые сравниваются в пространстве L_2, то используется метод наименьших квадратов, хотя могут применяться и другие подходящие метрические пространства. Набор параметров $s \in R^k$, который приближает заданную кривую $f(t)$ с заданной точностью ϵ, т.е.

$$F(s) = \|M(s) - f\| \leq \epsilon, \tag{1}$$

называется решением обратной задачи. Точность ϵ также называется порогом.

Не существует общепринятых методов решения обратных задач. Эти задачи обычно нелинейны, даже если прямая задача линейна, поскольку решения даже линейных дифференциальных уравнений — нелинейны. Кроме того, такие задачи обычно являются некорректными [1, 2]. Существующие методы обеспечивают, в лучшем случае, локальную сходимость, поэтому существование альтернативных решений остается открытым вопросом. Различные попытки обеспечить уникальность решения требуют на практике знания о решении.

Качество решения измеряется его способностью приближать ряд кривых с разумной точностью. Кроме того, решение должно быть физически обоснованным, т. е. соответствовать общему физическому смыслу и основным свойствам. Для вероятностей или долей некоторой величины заданы очевидные неравенства. Иногда физические оценки не являются строгими; например, известно, что некоторая скорость намного ниже скорости звука, но отсутствуют более точные оценки. Такие мягкие оценки приводят к необходимости проверки потенциальных априори маловероятных решений.

Область D допустимых значений параметров может содержать гораздо больше, чем одно решение обратной задачи. Если размерность пространства параметров k не слишком высока, то плотный поиск (также называемый сканированием) на сетке внутри D может оказаться плодотворным. Узлы с наилучшими решениями могут служить в качестве начальных приближений для алгоритмов спуска или аналогичных.

Такая задача хорошо подходит для выполнения в распределенных вычислительных системах [3, 4], включая Desktop Grid, благодаря следующим свойствам:

(1) Отдельные задачи (отдельные решения прямой задачи, т. е. оценка M в узле сетки и сравнение двух кривых) полностью независимы, могут выполняться параллельно и не нуждаются в синхронизации.

(2) Задачи требуют умеренного количества вычислительной мощности, оперативной памяти и дискового пространства.

(3) Объем данных, необходимых для решения задачи, мал: k-мерный массив значений и экспериментальных кривых (отправляется один раз).

(4) Результатом является одно действительное число (качество приближения), поэтому скорость сетевых соединений не является критичной.

Идентификация параметров обычно является очень хорошо масштабируемой задачей. Размер и размерность области параметров D могут быть большими, а изменение либо точности, либо размера сетки может сделать процесс поиска еще более требовательным и ресурсоемким. В то же время промежуточные результаты, полученные на ранних этапах поиска, могут значительно способствовать процессу из-за новых знаний об области параметров или обоснования самой модели. Это мотивирует использование сложных алгоритмов поиска, а не слепого поиска.

В [5] мы описали решение задачи идентификации параметров модели разложения гидрида алюминия методом сканирования пространства параметров. Мы нашли гораздо больше решений, чем ожидалось, присутствующих во всех частях области. Однако сам процесс поиска может быть оптимизирован.

Статья имеет следующую структуру. Сначала мы описываем модель и обратную задачу, которую нужно решить. Затем мы обсуждаем Desktop Grid на базе BOINC в качестве инструмента для решения задачи и показываем, что политика запроса заданий может

быть оптимизирована. Мы сравниваем три стратегии планирования, предназначенные для быстрого получения хороших результатов. Одна из них — игра между вычислительными узлами, которые выбирают подмножества области D. Повышение эффективности достигается без централизованного управления, что хорошо для архитектуры клиент-сервер, так как не требует дополнительной нагрузки на сервер.

2. Разложение гидридов металлов

Гидриды металлов могут запасать значительное количество водорода при менее экстремальных по сравнению с криосистемами, баллонами высокого давления и т.п. системами условиях и потому перспективны с точки зрения проблемы хранения водорода. Сравнительная безопасность гидридных аккумуляторов водорода особенно важна для мобильных систем. Недостатком гидридов является относительно низкое массовое содержание водорода. Гидриды легких металлов (например, алюминий и магний) с этой точки зрения наиболее перспективны. Помимо количества, важной характеристикой является скорость выделения водорода. Поэтому важно детальное понимание процессов разложения гидрида и выделения водорода и оценка параметров этого процесса для различных материалов и различных условий эксперимента, методов активации, используемых катализаторов и т.д.

В этой работе мы не касаемся вопросов моделирования разложения металлогидридов; см [6, 7]. Отметим только, что требования высокой скорости и большого содержания водорода ставят под вопрос относительно простые подходы типа Аврами-Ерофеева, выделение для анализа отдельных участков кинетических кривых (например, приблизительно линейного), рассмотрение единственного лимитирующего процесса и т.п. Сложность модели должна соответствовать информативности экспериментальных результатов. Модель, как минимум, должна удовлетворять законам сохранения и описывать конкуренцию реакции разложения гидрида и десорбции водорода.

Модель, используемая в этой работе, описана в [8], см. также ссылки в этой статье. Моделируется одна частица порошка металлогидрида единичного радиуса. Дегидрирование проходит по сценарию сжимающегося ядра: новая фаза металла формирует корку на поверхности частицы и затем продвигается к центру. Модель легко обобщается на случай других форм с подходящей группой симметрий (в том числе длинный тонкий цилиндр, тонкая чешуйка и несколько экзотических форм) [9], но показано, что влияние формы на кинетику несущественно.

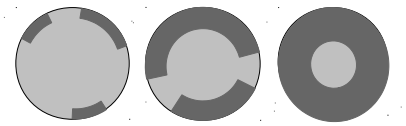


Рисунок 1. Формирование корки и сценарий сжимающегося ядра; светло-серым выделена фаза гидрида.

Обозначим толщину сферического слоя корки через $h = 1 - \rho(t)$, где ρ — радиус ядра старой (гидридной) фазы, см. рис. 1. Формирующаяся модельная корка состоит из симметричных зародышей, полностью описываемых полной площадью на поверхности и толщиной h; взаимное распределение зародышей в таких предположениях не играет роли.

Физические предположения:

- диффузия растворенного в решетке металла водорода быстрая (в микроскопических частицах порошка), поэтому градиентами концентрации водорода можно пренебречь;
- отсутствует ре-адсорбция водорода: десорбция в вакуум;
- десорбция из металлической фазы идет через поры и трещины материала;
- концентрация водорода в металле мала по сравнению с принятой за единицу стехиометрической концентрацией в фазе гидрида; однако поток десорбции с фазы металла не равен нулю;
- размер частицы порошка не меняется в ходе реакции;
- доля ν потока водорода из фазы металла обусловлена сжатием ядра, остальное — за счет расширения корки.

Обозначая плотности потоков водорода для фаз гидрида и металла через J' и J, соответственно, запишем выражение для полного потока водорода:

$$J'(1 - S(t)) \cdot 4\pi + JS(t)4\pi\rho^2.$$

Локальный баланс вещества дает уравнение для размера гидридного ядра:

$$\frac{d\rho}{dt} = -\nu(S)J, \quad \rho > 0. \tag{2}$$

Здесь $\nu(S) = \nu$ при $S < 1$, т.е. если корка не занимает еще всей поверхности; $\nu(1) = 1$.

Глобальный закон сохранения вещества дает второе уравнение:

$$\frac{dS}{dt}\frac{1-\rho^3}{3} = J'(1-S(t)) + (1-\nu)JS(t)\rho^2. \tag{3}$$

Измеряется относительный объем прореагировавшего вещества, т.е. доля объема новой фазы:

$$f(t) = \big(1-\rho^3(t)\big)S(t).$$

Модель (2)–(3) содержит пять параметров:

(1) плотности потоков водорода с поверхности двух фаз J' и J;

(2) начальное состояние корки: S_0 и ρ_0 (активация способна порождать активные центра десорбции, формально создавая кору новой фазы);

(3) фактор ν, характеризующий рост корки («вглубь» или «вширь»).

Все параметры неотрицательны; также $S_0 \le 1$, $\rho_0 \le 1$, $\nu \le 1$. Плотности потоков не могут быть слишком велики и потому «мягко» ограничены сверху: точная верхняя грань неизвестна.

Таким образом, мы имеем параллелепипед в пятимерном пространстве R^5, на котором будем искать оптимумы среднеквадратичной (или иной) невязки двух кривых: модельной зависимости $f(t)$ и экспериментальных измерений. нормируем эту невязку на среднеквадратичную норму экспериментальной кривой, чтобы оптимизируемая функция была безразмерной (мы опускаем некоторые технические подробности).

Несмотря на простоту модели и аналитическую разрешимость уравнений, численное решение выглядит более предпочтительным по ряду причин: оно эффективнее, иногда даже численно точнее, является гораздо более гибким: требует минимальных изменений в коде при модификации модели.

Простейший метод типа предиктор-корректор (метод Хейна) реализован на стандартном Фортране-90 и может быть запущен на любой платформе.

3. Вычислительная среда и численные эксперименты

Первоначально порог для выбора хороших результатов (1) был установлен в 6%. Мы вручную нашли набор параметров, обеспечивающих хорошую подгонку (6%) в качестве эталонного решения: $\bar{J} = \bar{J}' = 10$, $\bar{\rho}_0 = 88$, $\bar{S}_0 = 2$ и $\bar{\nu} = 50$ (значения потока выражены в удобных абстрактных единицах, другие параметры выражены в %). Сетка была однородной вдоль каждой оси с 5 узлами для S_0 (0–4) и 10 узлами для каждого из остальных параметров: 2–20 для

плотности потока, 80–98 для R_0 и 0–90 для ν. Таким образом, сетка состояла из 50 тыс. точек, что составило 250 тыс. заданий для пяти экспериментальных кривых.

Сначала будем рассматривать разные кривые как отдельные эксперименты и не будем отдавать предпочтение наборам параметров, которые схожи для разных кривых; сначала мы хотим увидеть, существует ли множество решений для выбора среди них.

Численное сканирование области параметров выполнялось в локальной Desktop Grid на базе BOINC [10] Карельского научного центра [1]. Наша цель состояла не только в нахождении кинетических параметров для разложения гидрида алюминия, но и в изучении возможности использования Desktop Grid для решения таких задач и поиске способов улучшения процесса поиска, чтобы решать задачи с большими размерностями быстрее и при меньших затратах.

BOINC является одним из самых популярных промежуточных ПО с открытым исходным кодом для систем Desktop Grid, как для добровольных вычислений, так и для Enterprise Desktop Grid [11]. Такая система использует простаивающие возможности настольных компьютеров и других неспециализированных вычислительных ресурсов (таких как серверы и даже мобильные устройства [12]).

Несмотря на относительно низкую среднюю производительность каждого вычислительного узла, Desktop Grid могут собирать много вычислительной мощности на единицу стоимости за счет использования множества компьютеров без каких-либо вообще или небольших дополнительных затрат. Очевидным недостатком является относительно медленная сетевая связь, так что обмен данными, зависимости заданий и синхронизация ухудшают производительность. Однако набор независимых заданий (пакет заданий), как и в нашем случае, идеально подходит для Enterprise Desktop Grid. Платформа BOINC позволяет решать пакеты заданий в гетерогенных средах. Приложения не должны быть специально разработаны для платформы и не обязательно должны иметь открытый исходный код.

BOINC имеет двухуровневую архитектуру клиент-сервер: сервер создает задания; клиентские узлы, имеющие свободные ресурсы, запрашивают задания и получают их. После выполнения задания узел возвращает ответ на сервер.

[1] http://cluster.krc.karelia.ru/

Таблица 1. Распределение решений на сетке параметров: (а) — *хорошие* (%); (б) — *наилучшие* (%).

J	(а)	(б)	J'	(а)	(б)	ρ_0	(а)	(б)	ν	(а)	(б)	S_0	(а)	(б)
2	17.5	92.6	2	7.4	7.49	80	9.4	14.0	10	13.8	29.4	0	19.6	22.8
4	13.7	7.25	4	3.8	2.64	82	9.6	13.6	20	13.5	26.0	1	19.8	25.0
6	11.0	0.08	6	3.5	2.72	84	9.9	12.7	30	13.2	21.0	2	20.1	21.0
8	9.2	0.04	8	8.9	4.02	86	10.2	11.8	40	12.4	14.2	3	20.2	19.0
10	8.4	0	10	11.7	7.81	88	10.5	10.9	50	10.6	6.76	4	20.3	12.2
12	8.3	0	12	12.9	11.4	90	10.7	9.66	60	8.0	1.02			
14	8.2	0	14	13.4	14.9	92	10.8	8.52	70	6.1	0.51			
16	8.1	0	16	13.2	16.2	94	10.6	7.69	80	6.7	0.39			
18	8.0	0	18	12.9	16.9	96	9.9	6.46	90	7.8	0.44			
20	7.6	0	20	12.3	15.9	98	8.4	4.73	100	7.9	0.47			

В ходе экспериментов, описанных в [5], мы использовали один настольный компьютер, два сервера, контрольный узел и три вычислительных узла кластера Карельского научного центра (каждый узел кластера имеет два 4-ядерных процессора).

Общее время сканирования составило около часа, что было более чем в 23 раза быстрее по сравнению с одним персональным компьютером. Сканирование обнаружило гораздо больше решений, чем ожидалось. 85 988 результатов были лучше предопределенного порога 6%. Поэтому мы уменьшили порог до 1.6% и получили более 2 537 решений.

Заметим, что дальнейшее снижение порога качества невозможно на данном наборе данных, поскольку у некоторых экспериментальных кривых отсутствуют решения, хотя у других их много. Например, уменьшение порога до 1% дает 220 решений, все — для одной и той же экспериментальной кривой. Это критическое значение неизвестно заранее, поэтому пост-обработка результатов очень важна, и может потребоваться повторить сканирование. Также заметим, что локально сходящиеся методы могут вообще не найти лучшего решения, обнаруживая локально минимальное 6%-ное решение.

Далее в статье будем называть результаты, выбранные при пороге 6%, *хорошими*, а результаты, выбранные при пороге 1.6%, *наилучшими*. Будем использовать эти два понятия для оценки и обсуждения политик планирования заданий.

Таблица 1 иллюстрирует распределение физически различимых решений на сетке параметров.

4. Методы поиска

Критерии оптимизации сканирования области параметров могут быть разными. Мы использовали фиксированную сетку и должны были проверять каждую точку; сканирование всей области (или даже набора всех возможных значений параметров) представляется более информативным, но его трудно реализовать. Однако можно рассмотреть сетку с переменным шагом. Идеи, представленные в этой статье, также могут быть использованы с этой целью, но прежде всего сосредоточимся на поиске по сетке с фиксированным шагом. Каждый метод должен проверить все точки сетки к концу вычислений. Поскольку можно выбрать только порядок проверки точек (заданий), оптимизация не изменяет расчеты; поэтому мы можем рассматривать только линейный порядок точек, принимая время выполнения одного задания за единицу.

Чтобы сравнить различные политики упорядочения точек, оценим время, необходимое для получения всех *хороших* (или *наилучших*) результатов или некоторой фиксированной части из них.

Простое планирование заданий. Мы использовали простое остаточное кодирование сетки параметров для генерации заданий. Каждый набор (и, следовательно, каждое задание) был связан с натуральным числом $n \leq 250\,000$, которое было получено из номера экспериментальной кривой и одномерных индексов отдельных сеток каждого параметра как остатки (плюс 1) деления на число кривых (5) и размеры отдельных сеток (10 четыре раза, а затем 5). Этот подход меняет J быстрее, чем другие параметры, а табл. 1 показывает, что большая часть работы была расточительной, потому что все *наилучшие* решения относятся к области низких значений J.

Аналогичная ситуация с ν, хотя этот параметр изменяется медленнее, чем другие (кроме S_0). С другой стороны, *наилучшие* параметры распределены достаточно однородно (табл. 1), поэтому порядок поиска не имеет значения. Наибольшие порядковые номера *хороших* и *наилучших* решений были соответственно 249 970 и 242 535 из 250 000. Поэтому перед тем, как собрать все *хорошие* и *наилучшие* результаты, необходимо проверить почти все точки сетки. 90% *хороших* и *наилучших* результатов были получены после выполнения заданий 246 531 и 96 534 соответственно, что составляет 98% и 46%: к счастью, многие из наилучших результатов относятся к блоку $J = 2$ сетки.

Случайный поиск. Другая возможность — случайный поиск с некоторым вероятностным распределением на области D. Выбор

этого распределения — интересный вопрос. Самый простой выбор — это равномерное распределение. Для его корректировки можно использовать некоторые априорные предположения: например, можно было бы предположить, что граничные значения с меньшей вероятностью обеспечивают хорошую подгонку (это предположение оказалось неверным). Другое предположение, также неверное, заключалось в том, что хорошие решения маловероятны при низких ν и нулевом S_0. Третье предположение заключалось в том, что вероятность выше в середине области D (в соседней окрестности эталонного решения); это тоже оказалось неверным. Таким образом, полагаться на априорные догадки рискованно с точки зрения производительности: в случае неправильного предположения случайный поиск может быть еще медленнее, чем поиск в фиксированном порядке. Оценим среднее время, необходимое для сбора всех хороших результатов.

Для того чтобы получить $R\%$ хороших результатов, необходимо проверить, в среднем, $R\%$ всех заданий.

Эвристическое планирование заданий. Третий способ повысить эффективность поиска — использование эвристических алгоритмов распределения заданий с использованием накопленной информации. Проблема заключается в отсутствии таких эвристик и их уязвимости к изменениям в модели и экспериментальных данных. Наши неправильные догадки были основаны на опыте численного решения многих подобных обратных задач, но в рассматриваемом случае они не сработали. Другим недостатком такого централизованного подхода является необходимость управления процессом на стороне сервера, который может быть перегружен даже без этой дополнительной работы.

Теоретико-игровое планирование заданий. Далее в этой статье мы исследуем возможность применения теоретико-игровой оптимизации поиска. Узлы не просто запрашивают задания, но также сообщают серверу о распределении своих предпочтений по D, стремясь максимально повысить свой выигрыш. Решения, принимаемые узлами, взаимно влияют друг на друга, так что функции полезности зависят от всех узлов. Если функции полезности (другими словами, правила игры) выбраны хорошо, равновесие будет оптимальным или, по крайней мере, лучше, чем другие политики поиска.

5. Теоретико-игровая модель

Рассмотрим вычислительную систему, состоящую из m вычислительных узлов (которые являются игроками) $C_1, \ldots, C_m$, которая

решает набор заданий B. Каждый узел C_n характеризуется производительностью e_n: средним числом операций в единицу времени. Множество заданий B разобьем на n непересекающихся блоков $B = B_1 \cup \ldots \cup B_n$ с размерами $N_1, \ldots, N_n$.

Чтобы определить приоритет блока, необходимо оценить качество решений, которые можно в нем, предположительно, обнаружить. Выберем порог ϵ и определим хорошие решения как удовлетворяющие неравенству $F(s) \leq \epsilon$, где $s \in D$ — точка сетки.

Тогда для каждого блока B_i можно определить его качество p_i как

$$p_i = \sum_{\substack{s \in B_i \\ s \text{ is computed}}} \left(1 - F(s)/\epsilon\right)^+, \quad (x)^+ = \max(x, 0), \quad 0 \leq i \leq n. \tag{4}$$

Если ни одна точка из B_i еще не посчитана, $p_i = 0$ по определению. Приоритет блока B_i положим равным

$$\sigma_i = \frac{\frac{1}{n} + p_i}{1 + p_1 + \ldots + p_n}, \quad 0 < \sigma_i \leq 1.$$

Предположим, что все задания в данном блоке B_i имеют одинаковую среднюю сложность θ_i, т.е. одно и то же фиксированное число операций требуется для расчета задания. Узлы принимают решения в дискретные моменты времени $0, \tau, 2\tau, \ldots$. На каждом шаге каждый узел выбирает ровно один блок заданий. Когда узел завершает выделенный ему набор заданий, он возвращает результат на сервер и готов к принятию новой порции заданий. Пусть выигрыш узла C_i на шаге τ есть количество полезной работы, выполненной за этот шаг. Оно зависит от числа выполненных заданий из выбранного блока, сложности этих заданий, приоритета блока и числа узлов, выбравших этот блок. Чем меньше узлов исследуют блок B_i одновременно, тем ценнее их работа. Пусть n_i — число узлов, выбравших блок B_i на заданном шаге. Определим коэффициент заполненности $\delta(n_i)$ блока как

$$\delta(n_i) = \frac{m + 1 - n_i}{m} N_i^{\text{av}},$$

где N_i^{av} — число доступных заданий в блоке.

Выигрыш узла C_k, выбравшего блок B_i, определим как

$$U_{ki} = \sigma_i \delta(k_i) \frac{e_k}{\theta_i}.$$

Таблица 2. Истинные доли «хороших» и «лучших» результатов в блоках; H и L обозначают область «высоких» или «низких» значений.

	HH··	HL··	LH··	LL··
··HH	0.047, 0.000	0.025, 0.000	0.070, 0.008	0.038, 0.003
··HL	0.083, 0.000	0.045, 0.000	0.125, 0.269	0.067, 0.090
··LH	0.047, 0.000	0.025, 0.000	0.070, 0.014	0.038, 0.005
··LL	0.083, 0.000	0.045, 0.000	0.125, 0.458	0.067, 0.153

Отметим, что

$$U_{kr} = \frac{e_k}{e_i} U_{ir} \quad \forall r,\ 1 \le r \le n. \tag{5}$$

Итак, на каждом шаге мы имеем игру заполнения (singleton congestion game) $G = (C, B, U)$, где C — множество игроков (вычислительных узлов), B — множество блоков, из которых каждый игрок выбирает ровно один, а U — множество функций выигрыша. Стратегии — это векторы $s = (s_1, \ldots, s_m)$, где компонента $s_i = j$ задает номер блока B_j, выбранного игроком C_i.

Такие игры хорошо изучены. Существование по крайней мере одного равновесия по Нэшу в чистых стратегиях доказано для случая идентичных игроков в [13] и для случая идентичных блоков [14]. Равновесие означает, что ни один игрок не может увеличить свой выигрыш, в одиночку отклонившись от распределения. Предложены схемы, сходящиеся к равновесию за полиномиальное время [14, 15].

Неоднородность системы усложняет модель: выигрыш игрока зависит не только от его производительности, но и сложности заданий в блоке. Однако при выбранном виде функций выигрыша и вполнении условия (5) игра G имеет хотя бы одно равновесие в чистых стратегиях [16].

6. Приложение к обратной задаче

Разобьем диапазон изменения каждого параметра, кроме S_0, на две равные части: высокихи низких значений. Так, например, значения $J < 11$ низкие, а $J > 11$ — высокие. Диапазон S_0 не разбивается. Тогда получается 16 блоков $B_1, \ldots, B_{16}$, отвечающих всевозможным комбинациям высоких и низких областей четырех параметров.

Для простоты предположим, что все узлы имеют одинаковую вычислительную мощность, принимаемую за единицу: $e_i = 1$, а все

задания имеют одинаковую сложность $\theta_i = 1$. Тогда функции выигрыша $U_{ki} = U_i$ зависят только от блока.

Упомянутые выше схемы, сходящиеся к равновесию [14, 15], могут применяться блоком принятия решений клиента для запроса заданий из выбранного блока от сервера. Рассмотрим другой подход, основанный на том, что выигрыши узлов в каждом блоке в равновесной ситуации совпадают: в противном случае узлы перейдут в более выгодный блок, выравнивая нагрузку. Выигрыши всех игроков U_i все равны величине C/m; тогда

$$n_i = m + 1 - \frac{C}{\sigma_i N_i^{\text{av}}}.$$

Сумма всех n_i равна m, поэтому

$$C = \frac{(m+1)n - m}{\sum\limits_{j=1}^{n} \frac{1}{\sigma_i N_i^{\text{av}}}}.$$

При этом некоторые n_i, возможно, окажутся отрицательны; это означает, что соотвествующие блоки настолько плохи, что требуемый выигрыш на них недостижим даже при минимальной нагрузке $n_i = 0$. Выбирать эти блоки не имеет смысла, поэтому их исключаем из рассмотрения, уменьшаем n и пересчитываем n_i. Алгоритм вычисляет неотрицательные, возможно, дробные загрузки блоков n_i, которые могут использоваться сервером для распределения узлов по блокам. Также округленные значения могут служить приближениями решения. Мы тестировали упрощенное приближение: приравняв отрицательные n_i нулю, положительные перенормируются:

$$\frac{m}{\sum\limits_{i,\ n_i>0} n_i}.$$

7. Результаты имитационного моделирования

Имитационное моделирование проводилось на основе статистики поиска, произведенного в работе [5].

В в таблице 3 представлены величины p_i, оцененные после обработки всех узлов сетки и нормализованных по размеру блоков, далее обозначаемые $\bar{p}_i$. Очевидно, что некоторые блоки лучше других; также отметим, что есть блоки, которые хороши в целом, но имеют относительно низкое количество лучших результатов (LHLH). Это дополнительная сложность для работы алгоритма, потому что хороший

Таблица 3. Средние значения качества блоков $\bar{p}_i$.

	HH $\cdot\cdot$	HL $\cdot\cdot$	LH $\cdot\cdot$	LL $\cdot\cdot$
$\cdot\cdot$ HH	0.85	0.65	0.66	0.68
$\cdot\cdot$ HL	0.77	0.54	0.78	0.61
$\cdot\cdot$ LH	0.74	0.84	0.93	0.85
$\cdot\cdot$ LL	0.85	0.77	0.92	0.70

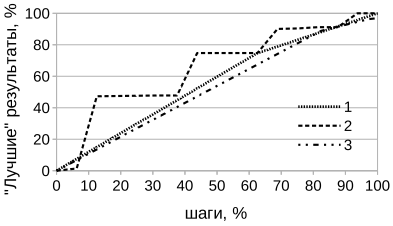
блок «привлечет внимание», хотя в итоге не даст много наилучших результатов.

Мы оцениваем эффективность алгоритма в среднем, т. е. в предположении, что p_i, полученные на основе завершенных заданий, соответствует статистическому среднему: $p_i = \bar{p}_i N_i^{\text{done}}$, где N^{done} — число выполняемых заданий в блоке i.

Используя предлагаемый алгоритм планирования заданий, 90% лучших результатов находятся с помощью Desktop Grid после выполнения 85% шагов; 75% — 63% шагов; 50% — менее 42% шагов; наконец, 25% результатов найдены после выполнения 21% шагов. Поток наилучших результатов показан на рисунке 2. На рисунке предложенный алгоритм сравнивается с двумя другими политиками планирования: случайный поиск на основе распределения качества и «лучший блок выбирается первым».

Показателем качества изображенных кривых является их относительная близость к верхнему левому углу диаграммы. Результаты показывают, что начиная с самых ранних этапов теоретико-игровой алгоритм работает лучше, чем основанный исключительно на приоритетах блоков. Третий рассмотренный алгоритм зависит от доступной информации об истинных долях хороших результатов в блоках. Представленные результаты показывают, что это является серьезным недостатком на ранних этапах вычислений, когда имеющейся информации недостаточно.

Моделирование показало, что начальное равномерное распределение вычислительных узлов над блоками заданий быстро сходится к стабильной конфигурации. Некоторые блоки остаются неисследованными, а более перспективные становятся пустыми. Лучшие блоки изучаются интенсивнее, так что скоро график становится равномерным на лучших узлах и нулевым на худших. Когда в большинстве изученных блоков больше нет заданий, график меняется и сходится к другому, исследуя менее перспективные блоки. Градиентный скачок на рисунке 2



1 – теоретико-игровой подход;
2 – число узлов на блок пропорционально приоритету блока;
3 – выбирается блок с высшим приоритетом.

Рисунок 2. Число хороших результатов, полученных тремя алгоритмами

Таблица 4. Стабильные конфигурации. Номера блоков слева направо, сверху вниз по таблице 2.

Блок	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2-ой шаг	5.0	0.0	0.0	0.0	2.6	0.0	2.9	0.0	1.6	4.7	6.9	5.0	5.0	2.6	6.7	0.2
Равновесие 1	4.3	0.0	0.0	0.0	4.3	0.0	4.3	0.0	4.3	4.3	4.3	4.3	4.3	4.3	4.3	0.0
Равновесие 2	0.0	7.8	7.8	7.8	0.0	7.7	0.0	7.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.2

отмечает этот момент: после него интенсивность получения хороших результатов уменьшается.

Обе стабильные конфигурации показаны в таблице 4. Ячейки таблицы содержат дробные количества вычислительных узлов, которые выбирают соответствующий блок.

Предлагаемый алгоритм планирования заданий является гибким: он может самостоятельно регулировать свою работу «на лету», когда поступает новая информация, вычислительные узлы покидают или присоединяются к Desktop Grid или меняется конфигурация блоков.

Существует несколько возможностей для дальнейшего улучшения описанного подхода к сканированию пространства параметров. Во-первых, мы планируем увеличить количество блоков. Тогда высокое качество одного блока сделает его соседние блоки более привлекательными. Кроме того, конфигурация блоков может быть переменной; перспективные блоки можно разделить на более мелкие, а наихудшие блоки можно сгруппировать.

Выбор функции полезности является важным вопросом. В изученном примере худшие блоки почти не изучались до тех пор, пока не были закончены лучшие блоки. Кажется более безопасным направить хотя бы несколько узлов на изучение плохие блоков, чтобы накапливать и обновлять статистику. Тем не менее, несколько шагов можно выполнить

в соответствии с равномерным расписанием для сбора информации о качестве блоков.

Наконец, мы планируем протестировать алгоритм для более сложных задач поиска: более плотные сетки, непрерывные блоки, задачи с большим количеством параметров, гетерогенные Desktop Grid большего масштаба. Если вычисления требуют недель или даже месяцев, быстрое получение хороших результатов становится чрезвычайно важным.

Заключение

В работе мы рассматриваем задачу идентификации параметров и решаем ее методом параллельного сканирования пространства параметров. Чтобы изучить методы эффективного поиска, мы фокусируем внимание на простом, но полезном примере: относительно небольшая сетка в пространстве параметров была проверена в среде Enterprise Desktop Grid небольшого размера.

Мы предлагаем теоретико-игровой алгоритм планирования заданий над подмножествами пространства параметров, называемых блоками, и изучаем его производительность по сравнению с двумя эвристиками. Моделирование показывает, что алгоритм позволяет повысить производительность с точки зрения скорости обнаружения хороших результатов.

Список литературы

[1] R. C. Aster, B. Borchers, C. H. Thurber. *Parameter estimation and inverse problems*, Elsevier, 2005, 301 pp. ↑36

[2] А. Н. Тихонов, В. Я. Арсенин. *Методы решения некорректных задач*, Наука, М., 1979, 228 с. ↑36

[3] I. Foster, C. Kesselman. *The grid: Blueprint for a new computing infrastructure*, Morgan Kaufmann Inc., San Francisco, CA, USA, 1999. ↑37

[4] I. Foster, C. Kesselman, S. Tuecke. “The anatomy of the grid: Enabling scalable virtual organizations”, *International J. Supercomputer Applications*, **15**:3 (2001), pp. 200–222. ↑37

[5] И. А. Чернов, Е. Е. Ивашко, Н. Н. Никитина, И. Е. Габис. «Численная идентификация модели дегидрирования в грид-системе на базе BOINC», *Компьютерные исследования и моделирование*, **5**:1 (2013), с. 37–45. ↑37,42,47

[6] I. A. Chernov, I. E. Gabis. “Mathematical modelling of hydride formation”, *Mathematical Modeling, Clustering Algorithms and Applications*, Mathematics

Research Developments, ed. C. L. Wilson, Nova Science Publishers, 2011, pp. 203–246. ↑38

[7] I. E. Gabis, I. A. Chernov, *Kinetics of decomposition of binary metal hydrides*, Chemistry Research and Applications, Nova Science Publishers, 2017, 137 pp. ↑38

[8] I. E. Gabis, A. P. Voyt, I. A. Chernov, V. G. Kuznetsov, A. P. Baraban, D. I. Elets, M. A. Dobrotvorsky. "Ultraviolet activation of thermal decomposition of α-alane", *International Journal of Hydrogen Energy*, **37**:19 (2012), pp. 14405–14412. ↑38

[9] С. В. Маничева, И. А. Чернов. «Математическая модель гидридного фазового перехода в частице порошка симметричной формы», *Компьютерные исследования и моделирование*, **4**:3 (2012), с. 569–584 (in Russian). ↑38

[10] D. P. Anderson. "BOINC: A system for public-resource computing and storage", *GRID '04 Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, IEEE Computer Society, 2004, pp. 4–10. ↑41

[11] E. Ivashko. "Enterprise desktop grids", CEUR Workshop Proceedings, vol. **1502**, Proceedings of the Second International Conference BOINC-based High Performance Computing: Fundamental Research and Development (BOINC:FAST 2015), 2015, pp. 16–21. ↑41

[12] M. A. Khan. "A survey of computation offloading strategies for performance improvement of applications running on mobile devices", *Journal of Network and Computer Applications*, **56** (2015), pp. 28–40. ↑41

[13] R. W. Rosenthal. "A class of games possessing pure-strategy Nash equilibria", *International Journal of Game Theory*, **2**:1 (1973), pp. 65–67. ↑46

[14] I. Milchtaich. "Congestion games with player-specific payoff functions", *Games and Economic Behavior*, **13**:1 (1996), pp. 111–124. ↑46,47

[15] S. Ieong, R. McGrew, E. Nudelman, Y. Shoham, Q. Sun. "Fast and compact: A simple class of congestion games", *AAAI'05 Proceedings of the 20th national conference on Artificial intelligence*. V. 2, 2005, pp. 489–494. ↑46,47

[16] M. Gairing, M. Klimm. "Congestion games with player-specific costs revisited", *Algorithmic Game Theory*, SAGT 2013, Lecture Notes in Computer Science, vol. **8146**, ed. B. Vöcking, Springer, Berlin–Heidelberg, 2013, pp. 98–109. ↑46

Поступила в редакцию 29.12.2017
Переработана 23.10.2018
Опубликована 01.11.2018

Рекомендовал к публикации *Программный комитет*
Шестого национального суперкомпьютерного форума ***НСКФ-2017***

Пример ссылки на эту публикацию:

И. А. Чернов, Н. Н. Никитина. «Эффективное сканирование пространства параметров в Desktop Grid для идентификации модели разложения гидридов металлов». *Программные системы: теория и приложения*, 2018, **9**:4(39), с. 35–52. 10.25209/2079-3316-2018-9-4-35-52

http://psta.psiras.ru/read/psta2018_4_35-52.pdf

Эта же статья по-английски: 10.25209/2079-3316-2018-9-4-53-68

Об авторах:

Илья Александрович Чернов

Кандидат физико-математических наук, старший научный сотрудник ИПМИ КарНЦ РАН. Доцент Кафедры математического анализа ПетрГУ. Эксперт в области математического моделирования физических процессов и краевых задач математической физики. Область научных интересов: вычислительная математика, моделирование разложения гидридов металлов, численное моделирование крупномасштабной динамики моря.

0000-0001-7479-9079

e-mail: chernov@krc.karelia.ru

Наталия Николаевна Никитина

Кандидат технических наук, младший научный сотрудник ИПМИ КарНЦ РАН. Область научных интересов: высокопроизводительные и распределенные вычисления, планирование заданий, Desktop Grid, приложения математической теории игр.

0000-0002-0538-2939

e-mail: nikitina@krc.karelia.ru

Sample citation of this publication:

Ilya Chernov, Natalia Nikitina. “Effective scanning of parameter space in a Desktop Grid for identification of a hydride decomposition model”. *Program Systems: Theory and Applications*, 2018, **9**:4(39), pp. 35–52. (*In Russian*).

10.25209/2079-3316-2018-9-4-35-52

http://psta.psiras.ru/read/psta2018_4_35-52.pdf

The same article in English: 10.25209/2079-3316-2018-9-4-53-68