САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Кафедра системного программирования

**Сервис для моделей оптимизации на основе рекуррентных алгоритмов**

Дипломная работа студента 544 группы

Вахитова Александра Тимуровича

Научный руководитель,

профессор………………………………………………………/О.Н. Граничин/

Рецензент,

аспирант……………………………………………………../В.Е. Краснощеков/

“Допустить к защите”

Заведующий кафедрой,

профессор………………………………………………………./А.Н. Терехов/

Санкт-Петербург  
2007 г.

# 

# Оглавление

[Оглавление 2](#_Toc169458063)

[Аннотация 3](#_Toc169458064)

[Введение 3](#_Toc169458065)

[Сравнение с близкими продуктами 5](#_Toc169458066)

[Глава 1. Рандомизированные алгоритмы стохастической аппроксимации 9](#_Toc169458067)

[Основные предположения. Постановка задачи 14](#_Toc169458068)

[Глава 2. Сервис рекуррентных алгоритмов RecService 15](#_Toc169458069)

[Язык описания компонентов и задач 18](#_Toc169458070)

[Глава 3. Применения 19](#_Toc169458071)

[Вводный пример 19](#_Toc169458072)

[Интерактивный определитель в Интернет 20](#_Toc169458073)

[Рис. 2 Общая модель определителя с компонентом для расчета по методу Лобанова 22](#_Toc169458074)

[Метод адаптивной балансировки 22](#_Toc169458075)

[Задача балансировки 25](#_Toc169458076)

[Алгоритм балансировки 26](#_Toc169458077)

[Метод подстройки пользовательских приоритетов при поиске по коллекциям изображений 28](#_Toc169458078)

[Задача подстройки приоритетов 30](#_Toc169458079)

[Адаптивный метод средних рандомизированных показателей 32](#_Toc169458080)

[Заключение. Результаты 36](#_Toc169458081)

[Литература 38](#_Toc169458082)

# 

# Аннотация

Дипломная работа содержит определение языка для описания систем, использующих рекуррентные алгоритмы стохастической оптимизации. Описывается разработанный автором прототип сервиса задания исполняемых моделей, способных взаимодействовать через веб-службы с внешними системами и хранить свое состояние в реляционной СУБД. Приведены примеры, иллюстрирующие использование сервиса в различных задачах информатики и управления.

Модели состоят из компонентов. Каждый компонент решает определенную задачу или несколько задач. Модель задается на уровне задач, то есть имеется возможность замены одного компонента на другой, если оба решают одинаковую задачу.

Сервис является легко расширяемым, так как основан на открытых стандартах описания данных и информационных потоков. Планируемые расширения связаны с построением удобного графического интерфейса для управления хранилищем компонентов и построения моделей, созданием новых компонентов, возможностью запуска вычислительных заданий в GRID-системы, что существенно расширит круг решаемых сервисом задач.

# Введение

Современное программное обеспечение зачастую используется для взаимодействия со сложными инженерными объектами. Примерами совместного использования программного обеспечения и инженерных конструкций может служить управление полетом спутника, где сначала компьютер рассчитывает управляющие воздействия, затем посылает сигнал и спутник реагирует, изменяя характеристики своего движения или просто сообщая о собственном состоянии. Другим примером является ледокол, управляемый автоматизированной системой, контролирующей величину и скорость движения судна в зависимости от давления льда, определяемого по показаниям датчиков.

Такие системы являются многокомпонентными и гибридными, то есть демонстрирующими как непрерывные, так и дискретные аспекты поведения (см. [1]). При этом оказывается, что непрерывные компоненты в иерархии управления системой являются более низкоуровневыми, в дискретных же содержится высокоуровневая логика (см. [1,2]).

В дипломе описывается разработанное автором средство (сервис) **RecService** для задания исполняемых моделей, состоящих из определенного набора компонентов, связанных по управлению и по данным. Подразумевается возможность задания компонентов модели как интерфейсов к внешним системам. Хотя **RecService** может использоваться для широкого класса задач, основное внимание в работе уделено его использованию в задачах оптимизации с помощью рекуррентных алгоритмов.

Задача поиска экстремума может быть решена аналитически либо численно. В ряде случаев «в силу сложности или невозможности построения аналитического решения задача … может считаться решенной, если указан алгоритм, позволяющий численно построить приближенное решение с требуемой точностью» [3]. *Рекуррентный алгоритм оптимизации* имеет набор параметров, пересчитываемых на каждом шаге работы дискретной системы, на основе сделанных на этом шаге измерений (наблюдений) для оптимизации некоторой (неизвестной, не полностью заданной) функции стоимости (минимизации штрафной функции). При таком определении, в класс рекуррентных алгоритмов оптимизации попадает широкое множество алгоритмов для решения прикладных задач информатики и управления, примеры которых будут разобраны более детально в главе 3.

Тема дипломной работы затрагивает несколько областей знания, хорошо известных программистам и системным аналитикам: архитектуру, ориентированную на использование сервисов, средства описания моделей динамических систем, стохастическую оптимизацию.

Практическое значение разработанного автором сервиса **RecService** может состоять во внедрении его в практику студенческих и аспирантских работ, проводимых на кафедре системного программирования СПбГУ. Переиспользование кода основных алгоритмов позволит при решении новой прикладной задачи сконцентрироваться на разработке модели для нее и возможного сведения ее решения к использованию имеющихся алгоритмов, настройке их параметров.

Глава 1 посвящена рекуррентным алгоритмам стохастической оптимизации. Автору совместно с О.Н. Граничиным и С.С. Сысоевым принадлежит результат о расширении класса допустимых помех для сходимости алгоритмов с пробным возмущением. Глава 2 посвящена описанию сервиса **RecService**, предназначенного для реализации моделей на основе рекуррентных алгоритмов стохастической оптимизации и им подобных.

В главе 3 описываются несколько моделей сервисов применения алгоритмов стохастической оптимизации: интерактивный определитель Webkey-X, реализующий субоптимальные алгоритмы идентификации для построения вопросника минимальной длины, балансировка загрузки параллельных систем, адаптивный поиск по изображениям, в создании которых принимал участие автор.

В Заключении перечислены результаты работы и обсуждаются перспективы развития сервиса и расширения его функциональности.

### Сравнение с близкими продуктами

Согласно классификации [1], пакеты математического моделирования, к которым можно отнести и описываемый в данной работе сервис, бывают *специализированные*, ориентированные на конкретные области применения, и *универсальные*, описывающие модели определенного класса и применимые в любой области, для которой возможно задание соответствующей модели.

Универсальные пакеты по сфере применения могут быть либо *математическими,* либо относиться к классу *пакетов* *компонентного моделирования*. Математические пакеты предполагают полное математическое задание систем (примером такого пакета является MATLAB). Пакеты компонентного моделирования предполагают отдельное описание каждого компонента системы, которое не обязательно полностью математически характеризует объект. Для пакетов компонентного моделирования характерна способность к проведению численных экспериментов над моделями. В [1] пакеты компонентного моделирования разделяются на *промышленные* и *исследовательские*.

**RecService** следует отнести к универсальным пакетам компонентного моделирования, так как пакет может сочетать как функциональность для помощи исследователю в проведении несложных численных экспериментов, так и средства совместной работы для создания библиотек компонентов, разделения труда по работе с различными фрагментами системы между участниками команды (это видно из списка особенностей описываемого сервиса).

Основными чертами **RecService** являются:

* простота описания моделей на основе XML с возможностями прямого указания параметров компонент и связей между компонентами, которая позволяет быстро прототипировать системы и изменять их. Поверх сервиса сравнительно легко создается полнофункцинальный пользовательский интерфейс для задания моделей;
* ориентация на веб-службы позволяет достаточно легко наблюдать работу систем и интегрировать их с приложениями, работающими в Интернете;
* сервис способен инициировать процесс вычислений в рамках модели и продолжать его произвольно длительное время;
* простота стандарта для реализации алгоритмов, в том числе декларативность описания данных для инициализации и хранения состояния компонента, обеспечивает легкость внедрения и изменения алгоритмов;
* на основе сервиса можно организовывать сложные схемы совместного решения задач (метод классной доски, [4]). Сервис подразумевает реализацию надстроек для задания таких схем.

Необходимость в создании сервиса такого рода проистекает из сложности моделирования и реального применения алгоритмов, использующих нетривиальные методы оценивания и оптимизации. Алгоритмы сложны в настройке параметров и реализации; в ходе исследований применимости алгоритмов необходима возможность их сравнения и подключения к реальной системе. Существующие средства моделирования математически сложно описываемых систем не являются в полной мере гибкими для этого. Они либо являются лишь средствами описания, либо математическими пакетами (MATLAB), либо предоставляют средства компонентного моделирования, но не позволяют в полной степени использовать себя в реальных системах из-за сложности задания систем с хранимым состоянием и сложности интерфейсов взаимодействия.

В первую очередь в составе сервиса реализуются алгоритмы стохастической оптимизации [5,6], которая является одной из областей прикладной математики, хорошо себя зарекомендовавшей при использовании в динамических и многопараметрических системах с существенной неопределенностью. О разнообразном опыте применения методов можно узнать на страницах сайта Applied Physics Laboratory John Hopkins University, США [7] и в публикациях [8].

Важной чертой **RecService** является использование шаблона внедрение зависимости (dependency injection) [9] и технологии веб-служб [10], а также средств хранения данных и сопоставления объектной модели реляционных отношений базы данных [11]. Частичным аналогом разрабатываемого сервиса является продукт Simulink компании MathWorks, средство для построения моделей динамических систем в рамках MATLAB, описанное в [12].

# Глава 1. Рандомизированные алгоритмы стохастической аппроксимации

Развитие методов стохастической аппроксимации (СА) началось в пятидесятые годы ХХ в. с алгоритма Роббинса-Монро [13] и процедуры Кифера-Вольфовица (КВ) [14].

Рассмотрим для примера задачу о нахождении стационарной точки *θ* некоторой функции  (точки локального минимума или максимума) при условии, что для каждого значения - входа алгоритма (управляемой переменной) - наблюдается случайная величина *Y(x)*, являющаяся зашумленным значением функции  в точке *x*



Дж. Кифер и Дж. Вольфовиц для решения этой задачи при некоторых дополнительных ограничениях обосновали сходимость к точке *θ* рекуррентной последовательности, определяемой по правилу (алгоритму)

,

где *{αn}* и *{βn}* - некоторые заданные убывающие числовые последовательности с определенными свойствами.

Основное условие-ограничение на свойства помех наблюдения, которое обычно предполагается выполненным, - это условная центрированность помех наблюдения. Его можно сформулировать таким образом. Для *статистики* (функции, выборочные значения которой точно наблюдаются или вычисляются)



математическое ожидание при малом *β* близко к значению производной функции



Поведение последовательности оценок, доставляемых алгоритмом стохастической аппроксимации, зависит от выбора наблюдаемых функций-статистик *g(x,β)*. В ряде практических приложений бывает недостаточно информации относительно статистических свойств ошибок измерения, или они могут просто задаваться неизвестной экспериментатору детерминированной функцией. Это ведет к существенным трудностям в обосновании применимости обычной процедуры Кифера--Вольфовица, и часто ее оценки не сходятся к искомой точке. Hо это не означает, что, решая такие проблемы, надо вообще отказаться от использования достаточно простых в представлении алгоритмов стохастической аппроксимации. Выше в примере уже предлагалось использовать ``другие'' наблюдения. Если удачно заменить статистику *g(x,β)* на другую, которая ``в среднем'' лучше аппроксимирует производную функции , то можно надеяться на получение лучшего качества поведения последовательности оценок.

Предположим, что функция  дважды непрерывно дифференцируема и задана наблюдаемая реализация некоторой бернуллиевской последовательности независимых случайных величин {*Δn*}, равных с одинаковой вероятностью плюс/минус единице, некоррелированных с ошибками наблюдения на шаге n. Тогда процедуру Кифера-Вольфовица можно модифицировать, используя рандомизированную статистику



Разложив функцию  по формуле Тейлора и, воспользовавшись некоррелированностью *Δn* и помех наблюдения, для этой новой статистики имеем



Если значения числовой последовательности {*βn*} в алгоритме стремятся к нулю, то в пределе эта статистика ``в среднем'' совпадает со значением производной функции .

Такими же свойствами обладает и более простая статистика



использующая только одно наблюдение на каждой итерации (шаге). Добавление в алгоритм и канал наблюдения нового случайного процесса {*Δn*}, называемого *пробным одновременным возмущением*, приводит к обогащению последовательности наблюдений.

Пробное возмущение по своей сути является возбуждающим воздействием, так как главная цель его использования заключается в стремлении добиться невырожденности получаемых наблюдений.

В многомерном случае, когда , для построения последовательности оценок обычная процедура Кифера-Вольфовица, основанная на конечно-разностных аппроксимациях вектора-градиента функции, использует *2q* наблюдений на каждой итерации (по два наблюдения для каждой компоненты *q*-мерного вектора-градиента). Рандомизированные статистики и  допускают более простой с вычислительной точки зрения способ обобщения на многомерный случай, использующий всего два или одно измерение функции на каждой итерации. Пусть {*Δn*} - бернуллиевский *q*-мерный случайный процесс. Тогда



а вид формулы для  такой же, как и в скалярном случае. Использовать статистику  было предложено Дж. Спалом в статье [15]. В его работе, в частности, показано, что для больших *n* вероятностное распределение соответствующим образом отмасштабированных ошибок оценивания является приблизительно нормальным. Полученная формула для асимптотической дисперсии ошибки вместе с подобной характеристикой обыкновенной процедуры Кифера-Вольфовица была им использована для сравнения двух алгоритмов. Выяснилось, что при прочих равных условиях новый алгоритм имеет ту же скорость сходимости, что и обычная процедура Кифера-Вольфовица, несмотря на то, что в многомерном случае использует существенно меньше наблюдений (в *q* раз меньше при ). Статистика  впервые была предложена в работах [16,17]. В статье [16] она использовалась для построения последовательности оценок, состоятельных при почти произвольных помехах в наблюдении.

Вопрос о скорости сходимости оценок алгоритмов стохастической аппроксимации был, наверное, основным, стимулирующим модификации первоначальных алгоритмов. Свойства оценок обычной процедуры Кифера-Вольфовица и некоторых ее обобщений детально изучены в работах [3,18, 19, 20,21] и во многих других.

Скорость сходимости зависит от гладкости функции. Если она дважды дифференцируема, то среднеквадратичная ошибка обыкновенного алгоритма КВ убывает как O(n-1/2), если трижды дифференцируема - как *O(n-2/3)*[19]. В. Фабиан [22] модифицировал процедуру Кифера-Вольфовица, предложив использовать кроме аппроксимации первой производной конечно-разностные аппроксимации производных высших порядков с определенными весами. Если функция  имеет *l* непрерывных производных, тогда алгоритм Фабиана обеспечивает среднеквадратичную скорость сходимости порядка *O(n-(l-1)/ l)* для нечетных *l*. С вычислительной точки зрения алгоритм Фабиана очень усложнен, число наблюдений на одной итерации быстро увеличивается с ростом гладкости и размерности, кроме этого, на каждом шаге приходится обращать некоторую матрицу.

При использовании рандомизированных алгоритмов СА в задачах с достаточно гладкими исследуемыми функциями  добиться увеличения асимптотической среднеквадратичной скорости сходимости удается без увеличения количества измерений функции на каждой итерации.

В том случае, когда некоторый обобщенный показатель гладкости функции  равен *γ* (*γ=l+1*, если все частные производные функции порядка до *l* включительно удовлетворяют условию Липшица), в [17] было предложено использовать статистики вида



и , где  - некоторая вектор-функция с конечным носителем (дифференцирующее ядро), определяемая с помощью ортогональных многочленов Лежандра степени меньшей *γ*. Два соответствующих рандомизрованных алгоритма дают среднеквадратичную скорость сходимости последовательности оценок, равную O (n-(*γ*-1)/ *γ*). В той же работе было показано, что для широкого класса алгоритмов эта скорость сходимости оптимальна в некотором асимптотически минимаксном смысле, т. е. не может быть улучшена ни для какого-либо другого алгоритма, ни для любого другого допустимого правила выбора точек измерения.

Сходимость рандомизированных алгоритмов стохастической аппроксимации в многомерном случае при почти произвольных возмущениях обоснована в работах [23,24] и др. В статье [17] отмечается, что алгоритм с одним измерением в асимптотическом смысле ведет себя хуже, чем алгоритм с двумя измерениями. В [25] показано, что это не совсем так, если при сравнении алгоритмов рассматривать количество итераций, умноженное на количество измерений. Кроме того, во многих практических применениях, в частности, при оптимизации работы систем реального времени, лежащие в основе математической модели динамические процессы могут изменяться слишком быстро, не позволяя успеть получить два последовательных измерения. В некоторых задачах просто невозможно для одного шага алгоритма сделать два измерения таких, чтобы помехи наблюдения в обеих точках  и  были некоррелированны с . (Это одно из основных условий применимости алгоритма!) Избежать последнего недостатка позволяет предложенный в [26] алгоритм с двумя последовательными наблюдениями в точках  и .

### Основные предположения. Постановка задачи

Пусть  - дифференцируемая по первому аргументу функция, *x1,x2*,… - выбираемая экспериментатором последовательность точек измерения (план наблюдения), в которых в которых в каждый момент времени *n=1,2*,… доступно наблюдению с аддитивными помехами *vn* значение функции 

,

где *wn* - неконтролируемая последовательность случайных величин из , имеющих одинаковое, вообще говоря, неизвестное распределение .

*Постановка задачи*. Требуется по наблюдениям *y1,y2*,… построить последовательность оценок  неизвестного вектора *θ*, минимизирующего функцию



(1)

типа функционала среднего риска.

Обычно рассматривается задача минимизации функции  при более простой модели наблюдений

,

которая легко укладывается в общую схему. Сделанное обобщение в постановке задачи диктуется стремлением учесть случай мультипликативных помех в наблюдениях

,

который входит в общую схему с функцией .

В том случае, когда функция распределения  неизвестна, рассматриваемая задача выходит за рамки классической теории оптимизации.

Если измерения значений функции  фактически делаются с некоторой аддитивной случайной центрированной независимой ошибкой , то, в силу общности поставленной задачи, это усложнение непринципиально. Расширив вектор *w* дополнительной компонентой *v* и, обозначив

,

можно рассматривать вместо  новую функцию



со схемой наблюдения без дополнительных возмущений и новое совместное неизвестное распределение,  вместо , которое все равно и ранее предполагалось неизвестным. Если ошибки измерения не обладают хорошими статистическими свойствами, то упрощать задачу нельзя. Надо рассматривать модель наблюдений с дополнительными возмущениями *vn*.

В [27,28,29] описаны результаты, полученные при участии автора, о сходимости и скорости сходимости рекуррентных алгоритмов стохастической оптимизации.

# Глава 2. Сервис рекуррентных алгоритмов RecService

Сервис состоит из модулей коммуникации, хранения данных и набора компонентов моделей для расчета, а также информационной системы.

Начнем с модуля набора компонентов как основного. Модель для **RecService** – это набор компонентов, связанных в замкнутую систему. Направленная связь одного компонента с другим представляется на более низком уровне вызовом объектом метода другого объекта. Компонент может быть применен для решения одной или более задач. Пример: генератор случайных чисел с распределением Бернулли, задача – получение случайных чисел распределения Бернулли, метод – выдать следующее случайное число. Более сложный пример: алгоритм стохастической аппроксимации с одним измерением, задача – рекуррентная оптимизация с возможностью проведения активного эксперимента, методы – получить текущую оценку, сообщить данные измерения.

Таким образом, компонент может иметь свойства и самостоятельно вызывать другие компоненты для решения возникающих в ходе вычисления подзадач. Иначе говоря, компоненты соединяются направленными каналами передачи информации, обычно – в векторной либо скалярной форме. Абстракции компонента и канала ничем не отличаются принципиально от принятых в π-calculus [30] понятий канала и агента. Сложные многокомпонентные модели, прежде чем быть реализованными в данном сервисе, могут верифицироваться средствами π-calculus. Для задания многоагентных моделей в рамках сервиса c участием нескольких внешних агентов необходимо интеграция с каким-нибудь средством исполнения бизнес-процессов.

Вычисление в рамках замкнутой системы невозможно без инициатора, который контролирует переход от шага *n* к шагу *n+1*. Иногда такой компонент называют *delay*, задержка, так как между последовательными шагами процесса должно пройти время. Он важен для полностью автономных систем. Если к процессу подключена внешняя система, то ее обращения к веб-службе и являются индикатором перехода к следующему шагу. В данной версии **RecService** можно осуществить подключение системы, участвующей в процессе (не мониторинг), только в рамках одного компонента. Поэтому не возникает необходимости поддержания сложного потока работ. В будущем, если с помощью **RecService** будут решаться действительно сложные задачи, возможно использование внешнего средства для выполнения потока работ.

Специальный компонент мониторинга, присоединяемый к каналу передачи информации, может сообщать дополнительные сведения о проходящих данных, наблюдать за процессом на основе данных выбранного канала.

Для реализации выбран язык программирования Java. В реализации каждой задаче соответствует интерфейс, каждому компоненту - класс. Связи между компонентами реализуются через внедрение ссылок на объекты, реализующие интерфейсы необходимых компоненту подзадач, с помощью Spring Framework [31].

Хранение данных реализовано на основе Hibernate 3.0 Core [11], используется база данных HSQL DB. Для каждого компонента задается дескриптор (mapping) в формате Hibernate, определяющий сопоставление свойств объекта полям таблицы реляционной базы данных. В дескрипторах свойства-ссылки на внешние компоненты помечены атрибутом lazy=false, что означает загрузку и сохранение данных о связях в рамках одной транзакции базы данных. При необходимости выполнения метода определенного компонента, он извлекается из базы данных, затем сессия закрывается и производятся необходимые вычисления. После этого объект сохраняется в новой сессии. Разделение транзакций взятия компонента и его сохранения позволяет производить расчеты значительное время. Впоследствии развитием RecService может стать система, отсылающая вычислительно сложные задачи в грид.

Координирующим средством для всего сервиса является фреймворк Spring. Основной чертой Spring признана удобная реализация dependency injection-паттерна[9], используемого в RecService при инициализации процесса и обмене сообщениями с внешними участниками процесса. В приложении 1 приведен xml-файл с описанием задач и компонентов (их свойств и связей) для модели вводного примера из главы 3. Таким синтаксисом описывается инициализация процесса, а также изменения свойств компонентов в силу реакции внешней системы.

За общение с внешними участниками отвечает модуль коммуникации. Он реализован на основе Spring-WS, удобном средстве генерации contract-first веб-служб, что означает создание службы на основе wsdl-дескриптора, что позволяет сделать его как можно более простым и обеспечить высокую независимость интерфейса и реализации. Сообщения сервису могут быть трех типов: инициировать процесс по передаваемой модели, продолжить процесс, остановленный на стадии обмена данными с внешней системой, закончить процесс и удалить сведения о нем из базы данных сервиса.

Информационная система сервиса предоставляет доступ к иерархии задач и семантическим описаниям компонентов. С ее помощью возможно построение удобного визуального редактора моделей. На данном этапе развития сервиса, существует только прототип информационной системы.

### Язык описания компонентов и задач

Компоненты для создания моделей описываются прежде всего в терминах решаемых ими задач (выполняемых функций). Для решения задачи может сосуществовать несколько компонент, реализующих существенно разные решения. RecService предлагает свой базовый язык описания задач и компонентов, на котором можно задавать исполняемые модели.

Задача – это декларативное точное описание функциональности, необходимой для исполнения модели. Задача задается семантически описанием (пример задачи: построение оценок для (1), удовлетворяющих ).

Задача задается парой <*S,M*>, где *S* – это семантическое описание, хранящееся в информационной системе и понятное человеку, а *M* - формальное описание. *M* является конечным набором методов вида   
{*(ik->ok)*} где *ik,ok* – (упорядоченные) списки входных и выходных параметров. Естественной аналогией и средством реализации задачи для коллекции компонентов RecService является интерфейс.

Компонент – это средство решения задачи, то есть реализация функциональности, необходимой для исполнения модели. Компонент является исполняемым модулем, написанным на языке программирования, имеет открытые методы, обладает состоянием и поведением. Задается компонент четверкой <*I,T,F,P*>, где *I* – набор упорядоченных списков параметров для инициализации компонента, *T* – набор задач, решаемых компонентом, вместе с сопоставлением {*(ikl->i’kl,okl->o’kl)*} для *ikl,okl* – параметров метода *k* задачи *l* и *i’k,o’k* – параметров метода *k* компонента, соответствующего *k*-му методу задачи *l*.

С помощью описания задач можно построить модель системы. Для того, чтобы эта модель стала исполняемой, необходимо для каждой задачи выбрать компонент.

Модель задается сильно связным направленным графом *G=<V,B>* где каждому  - множества вершин – соответствует задача RecService, каждому  - множества ребер – компонент RecService.

# Глава 3. Применения

### Вводный пример

В [29] описан алгоритм стохастической аппроксимации с рандомизированным пробным возмущением с проектированием. Для моделирования его сходимости в случае нерегулярных помех, описанного в [29], была построена модель (см рис. 1). В ней участвуют упомянутый алгоритм (компонент SPSA) и управляемый объект, а также проектор P для проецирования точек наблюдения, генерируемых алгоритмом, источник случайных возмущений Δ, источник нерегулярной помехи v, внутреннего зашумления объекта *w*.

С точки зрения реализации, были заданы задачи, решаемые компонентами, в виде интерфейсов, и сами компоненты – в виде набора классов.



Рис. 1

### Интерактивный определитель в Интернет

Пусть некоторая задача решается в ходе серии экспериментов, каждый из которых можно представить в виде вопроса и последующего ответа. Каждый новый эксперимент выявляет ранее неизвестные свойства некоторого объекта. Классическая задача определения фальшивой монеты за минимальное число взвешиваний является хорошим примером задач этого класса [32].

Много позднее появились задачи диагностики и идентификации, формулируемые в виде подобного сложного эксперимента. Для их решения стали использовать *экспертные системы*, для которых основным инструментом задания следующего вопроса являлись механизмы логического вывода на множестве фактов. Их отличает довольно разнообразный круг решаемых проблем, из областей медицины и биологии, управления, распознавания.

Теория подобных систем, определяющих принадлежность объекта к какому-либо множеству с уникальным присущим этому множеству набором свойств через серию экспериментов - вопросов известна как теория вопросников. Определяется множество допустимых вопросов *T* и множество событий *E*. Событие заключается в определении (класса, уникального свойства) некоторого объекта. Вопросом является опыт, нетривиально разделяющий какое-либо (не единичной мощности) подмножество *E*. План сложного эксперимента, то есть совокупность вопросов и последовательности, в которой они должны быть поставлены для идентификации событий множества *E*, называется вопросником для *E*. Множество *T* допустимых вопросов является подмножеством множества всех возможных вопросов. Естественным требованием к *T* является возможность отделить каждый элемент *E* от любого другого элемента *E* вопросами из *T*. Множество допустимых вопросов, совпадающее с множеством всех вопросов, называется полным, иначе – неполным. Подробно теория вопросников освещена в [32].

В случае полного множества вопросов, для построения вопросника, задающего в среднем минимальное число вопросов для идентификации события, используется метод выбора на каждом шаге вопроса с максимальной энтропией. Аналогичные методы могут использоваться и в случае неполного множества вопросов, однако если ранее они могли считаться оптимальными, то теперь – нет. Для достаточно большого количества вопросов алгоритм поиска оптимального вопросника вычислительно нереализуем, так как задача принадлежит классу NP[33]. Усложним задачу, позволив после каждого задания вопроса корректировать вопросник, на основании приобретенной информации. Целью является минимизация в среднем числа необходимых вопросов.

Субоптимальные методы решения задачи изложены в [34,35,36]. Среди них есть и адаптивные, основанные на рекуррентном пересчете оценок [36].



Рис. 2 Общая модель определителя с компонентом  
 для расчета по методу Лобанова

На рис. 2 показана декомпозиция процесса работы с ключом Webkey-X. Модель состоит из трех компонентов: базы данных ключа, алгоритма сортировки признаков, определяющего вопросник на каждом шаге, и пользователя, обеспечивающего ответ на один из предложенных вопросов. Инициирует каждый следующий шаг вычисления пользователь, посылая пару *(C(k),s(k))* - выбранный вопрос и ответ на него.

В рамках работы по гранту Российского фонда фундаментальных исследований разработано несколько алгоритмов сортировки признаков для построения вопросника, близкого к оптимальному, которые в рамках этой модели можно заменять, так как они представляют собой разные компоненты, решающие одну задачу.

## Метод адаптивной балансировки[[1]](#footnote-2)

В этом разделе излагается решение методами стохастической аппроксимации типичной для распределенных вычислений задачи балансировки загрузки. Применимость предлагаемого подхода растет с числом параллельных вычислителей, как будет видно из обоснований предложенного ниже алгоритма. Таким образом, не только постановка задачи, но и сам метод решения дают право говорить об актуальности подхода для GRID-вычислений. Современные исследователи считают очень важным при определении GRID понятие виртуальной организации. «*Виртуальной организацией GRID* называется сообщество пользователей GRID с динамически меняющимся составом, необходимое для совместного использования … ресурсов в рамках общих задач»[38]. GRID – это «согласованная, открытая и стандартизованная среда, обеспечивающая гибкое, безопасное, раздельное использование разнообразных компьютерных ресурсов виртуальными организациями»[38].

Широкое внедрение GRID-систем для научных вычислений в последнее время привело к быстрому росту числа компьютеров, включенных в активно развивающиеся GRID, например для EGEE это 20 000, для Nordugrid 6000 процессоров [38].

В то же время, в силу технической сложности интеграции компьютеров для решения общих задач, не теряет актуальности проблема эффективности использования ресурсов GRID, что означает минимизацию простаивания определенных ресурсов, в то время как другие загружены работой.

Для выполнения в GRID характерны программы со слабой связью по данным и управлению между выполняющимися параллельно участками кода. В этой работе рассматривается случай Single Program Multi Data (SPMD) вычислений. SPMD – это тип параллельных программ, для которых характерно выполнение одного и того же программного кода на разных фрагментах данных (*параллелизм по данным*).

Для подобных программ актуальной является балансировка вычислений. Целью балансировки является минимизация общего времени выполнения параллельной программы в GRID. Балансировка означает перераспределение заданий в ходе работы программы, с учетом накапливаемой информации о дисбалансе. В силу того, что зачастую передача задания с одного процессора на другой требует переноса существенных объемов данных, ставится задача балансировки с целью минимизировать число пересылок для достижения требуемого уровня балансировки.

GRID система состоит из координирующих элементов и рабочих узлов. Рабочий узел выполняет вычисления. Координирующие элементы (в данной работе обозначим их общим понятием брокер) распределяют вычислительные задания между рабочими узлами и управляют передачей данных, необходимых для выполнения заданий.

Отправка заданий осуществляется через клиентский интерфейс. Затем задание поступает на брокер, который выделяет ресурсы для выполнения. После этого задание вместе с необходимыми данными передается на рабочий узел (отдельный компьютер или кластер) и выполняется.

Каждому пользователю GRID выдается определенные вычислительные ресурсы и время для расчета на них его заданий. Типичное выполнение пользовательского задания состоит из трех этапов: загрузки данных, расчетов и выгрузки результатов. Для произведения расчетов обычно на каждом из рабочих узлов создается процесс, будем также называть его рабочим, который последовательно обсчитывает выделенную ему очередь заданий. В процессе расчетов пользователь не изолирован от взаимодействия с данными процессами, имеет определенный доступ к ресурсам, на которых идут вычисления. Возможностей у пользователя достаточно, чтобы перераспределять задания между рабочими узлами в ходе работы рабочих процессов.

Предварительную загрузку данных иногда разрешается выполнять до начала выделенного времени, поэтому первоначальное распределение заданий следует производить как можно более точным образом, а балансировку производить лишь в случае неприемлемого для системы дисбаланса.

Таким образом, вычисление SPMD-задания в GRID начинается с распределения атомарных заданий между рабочими узлами. Затем на каждом из рабочих узлов запускается процесс, обрабатывающий атомарные задания. В тот момент, когда задания на рабочем узле заканчиваются, процесс переходит в режим ожидания до тех пор, пока не получит новых заданий либо сигнала о завершении работы.

Далее уделим внимание задаче балансировки наборов атомарных заданий на рабочих узлах.

### Задача балансировки

Пусть имеется *r* рабочих узлов, общее число атомарных заданий *W >> r*. Между узлами изначально распределены атомарные задачи, на *i*-м узле *Wi(0)* заданий. Пусть, далее, задан временной интервал *Δ* такой, что каждые *Δ* секунд инициируется процесс балансировки. Распределение невыполненных заданий между рабочими узлами в момент *t: Δk <t ≤ Δ(k+1)* задается вектором *Wi(k)*. Общее число невыполненных заданий обозначим как *W(k)= Σi Wi(k)*.

Будем считать, что с использованием средств мониторинга GRID есть возможность определить текущее распределение выполненных и не выполненных заданий на рабочих узлах. Обозначим число выполненных заданий на *i*-м рабочем узле в момент *t* как *Wi(t)*, *Wi(0)* =0, *Wi(t)*≥ *Wi(t-1)* для *k≥1*. Пусть имеется возможность выполнения высокоуровневой процедуры с векторным параметром по переносу невыполненных заданий так, чтобы в итоге они оказались распределены в пропорции, так что. Такая операция может быть оптимизирована с целью минимизации времени переноса заданий, что выходит за рамки основной задачи балансировки.

При каждом запуске процессу балансировки необходимо определить, нужно ли вызывать процедуру *C*, и если да, то с каким аргументом. Переформулируя эту задачу, можно сказать, что необходимо определить вектор-оценку оптимальной пропорции разделения оставшихся *Wk* заданий между рабочими узлами. Критерием запуска процедуры может быть превышение некоторого предельного значения нормой разности вектора-оценки и вектора фактического распределения выполненных заданий ,:.

Выполнение одинаковых заданий на одном рабочем узле на практике никогда не занимает одинаковое время. В то же время оборудование рабочего узла является неизменным, так что в среднем выполнение одинаковых заданий должно занимать одинаковое время. В этом случае естественной для минимизации числа запусков процедуры является задача минимизации в среднем нормы разности оценки оптимального распределения заданий и фактической доли выполненных заданий:

.

### Алгоритм балансировки

Учитывая описанную вероятностную природу процесса выполнения заданий, нельзя забывать о постоянстве (в рамках вычислительного процесса) характеристик оборудования рабочего узла, откуда следует наличие некоторой постоянной средней полезной производительности для него. Эта производительность может быть использована не полностью рабочим процессом, так как часть ресурсов узла может уходить на выполнение локальных задач, не относящихся к GRID.

Вводя в рассмотрение величину *pi* как производительность (число обрабатываемых заданий за определенный промежуток времени) *i*-го рабочего узла, мы можем проанализировать предлагаемый ниже алгоритм и обосновать его сходимость для данной задачи.

Предложим алгоритм вида стохастической оптимизации для пересчета оценки, минимизирующей указанную выше функцию:



Алгоритмы такого вида подробно рассмотрены в [6]. Определим особенности его применения для решения описанной задачи балансировки в GRID, а затем определим условия на используемую последовательность *εn*∈R.

Величина *yi(k)* с использованием *pi* выражается как:, где *vi* является величиной отклонения фактической производительности узла от его средней производительности на временном интервале *(0, Δk)*. Эта величина является случайной, предположим, что в среднем она равна 0 для всех *i*. Разделив на *r* числитель и знаменатель выражения, приходим к формуле



что верно в силу усиленного закона больших чисел для достаточно больших *r*.

Используя полученные выражения, получим следующее выражение для разности



Для последнего слагаемого, используя оценку для *y(k)i*, полученную выше, выводим . Вслед за [6] введем обозначение  .

Естественно предположить независимость *vi*  от *vi-1*,…,*v1*, называя , вслед за [6], мартингальными разностями:. Использование этого дополнительного предположения обосновывает постепенное исчезание слагаемых вида : при выборе малого *d*, такого что получается



где дисперсия отклонения будет выражаться как *O(d)εn*. Представленные выражения обосновывают, что изменение параметра в сторону оптимального значения  для больших *n* и малых *d* много более значительны, чем отклонения. Тогда можно сделать вывод, что асимптотическое поведение алгоритма может быть аппроксимировано асимптотикой решения обыкновенного дифференциального уравнения



В [6] для сходимости алгоритма предлагаются условия



Реализация предложенного алгоритма опирается на использование алгоритма стохастической оптимизации из семейства методов, подробно описанных в [6]. Рассмотрим модель описанного алгоритма. Здесь активно участвует как инициатор элемент delay. В то же время, если **RecService** работает с GRID как с внешней системой, delay может быть исключен из компонентов модели сервиса и рассмотрен как часть GRID-контейнера.

 Рис. 3

### Метод подстройки пользовательских приоритетов при поиске по коллекциям изображений[[2]](#footnote-3)

Информационный поиск является хорошо известной задачей современной информатики. Методы поиска хорошо разработаны для текстовой информации, в то время как поиск по более сложно организованной информации не имеет устоявшихся классических подходов. Последнее утверждение косвенно подтверждается в [39]: авторы статьи выражают неуверенность по поводу введения единой метрики для поиска по коллекции неаннотированных текстовыми метками изображений, обосновывая это тем, что поиск подходящего изображения сложно представить в виде уменьшения некоего расстояния. Ставятся под сомнение рефлексивность (зачем искать то, что и так имеется), симметричность (если *A* хочется найти при подаче запроса *B*, не факт, что хочется найти *B* при запросе *A*), и, тем более, неравенство треугольника.

Тем не менее, если рассматривать в некотором смысле однородное пространство характеристик изображения (текстурных, цветовых, формы), то для него задание метрики более обоснованно: поиск приобретает более конкретный смысл, и, например, в смысле поиска по цветовым характеристикам легче поверить в реальность свойств метрики.

Предлагается метод поиска, основанных на изложенных соображениях. Ставится в последнее время популярная задача поиска с обратной связью от пользователя. Строится функция суммарной оценки, определяющая близость изображения к запросу, которая учитывает пользовательские приоритеты по отношению к поиску в разных метризованных пространствах. Приоритеты выявляются с помощью рандомизированного алгоритма (модификации метода рандомизированных сводных показателей). Приоритеты формализуются на основе неравенств между весами, поставленными в соответствие каждому пространству. Итеративный поиск с учетом обратной связи последовательно выявляет приоритеты, затем они сохраняются и при задании схожего запроса снова используются. Устаревшие приоритеты постепенно теряют актуальность, в то время как повторяемые приоритеты используются постоянно.

### Задача подстройки приоритетов

Пусть заданы *N* пространств , каждому из которых соответствует класс характеристик изображения. Так, одно пространство соответствует характеристикам цвета, другое - формы, третье - текстуры и пр. На каждом из пространств задана своя метрика , где , отражающая объективную схожесть двух изображений на основе характеристик пространства с номером *i*, и , - объединенные вектора характеристик изображения во всех пространствах. Метрике  однозначно соответствует метод поиска по *i*-му пространству. Метрика пространства удовлетворяет в полной мере определению метрики (3 свойства). Она не обязательно является евклидовой.

Для поиска в базе данных изображений пользователь предоставляет изображение-запрос (*q*). На основании образца производится поиск. Механизм поиска может быть итеративным, на каждом шаге выявляя новую информацию о желании пользователя и корректируя результаты с ее учетом.

Пусть поиск производится на основе общей функции схожести, или суммарной оценки схожести, определяющей для каждого изображения его схожесть с тем, что хочет получить пользователь. Требуется построить такую функцию  суммарной оценки от изображения *x* и пользовательского запроса *q*, что она определяет наиболее подходящие пользователю изображения. Ограничений на функцию *σ* не так много. Исходя из общих соображений, при равных координатах во всех пространствах, кроме одного, близость изображений в этом пространстве должна определять итоговую их близость, то есть должна соблюдаться монотонность функции *σ* по отношению к метрикам пространств.

Пусть есть возможность разделить изображения на непересекающиеся категории. Будем предполагать, что внутри каждой категории существует некая наилучшая суммарная оценка, которую и призвана определить система поиска. Иными словами, считаем приоритеты пользователя постоянными для запросов определенной категории. За рамками пока остается вопрос наилучшего разделения на категории множества всех возможных изображений, принадлежащих декартову произведению имеющихся пространств . Это может быть сделано известными методами кластеризации, см., напр., [40]. Каким образом выражается в таком случае суммарная оценка?

Основным требованием к оценке является то, что она призвана отражать отношение к изображениям пользователя в контексте данного запроса. Более детально, упорядоченные по возрастанию суммарной оценки схожести изображения должны соответствовать порядку, в котором их расположил бы пользователь, просмотрев все изображения в базе данных.

Таким образом, единственно важным является не собственно значение суммарной оценки на каждом изображении, а лишь порядок, в котором мы упорядочиваем изображения. Для задания порядка достаточно качественных предпочтений, являющихся результатами гипотетических сравнений пользователем важности вклада различных метрик в суммарную оценку. Такими предпочтениями могут быть, например, утверждения вида в данной категории при поиске цветовые характеристики важнее текстурных. На выявление подобных знаний и должна быть направлена интегрированная система поиска по различным пространствам характеристик изображений.

Под *задачей адаптивного поиска* в этой работе понимается выявление закона упорядочивания изображений, соответствующего желанию пользователя, в рамках категории.

Первой подзадачей этой общей задачи является создание модели представления приоритетов в виде суммарной оценки.

Второй подзадачей является построение адаптивной схемы выявления приоритетов по результатам итеративной работы пользователя с поисковой системой - указания подходящих изображений среди всех предлагаемых. Система должна предусматривать механизмы выхода из тупиков, когда пользователь ни одну из представленных картинок подходящей не считает.

Третья подзадача связана с оценкой истинного желания пользователя в контексте поиска, то есть вектора характеристик того изображения, которое действительно его интересует (идеального изображения). Это означает, что часто пользователь не имеет удовлетворяющего его изображения, которым до сего момента мы считали его запрос. Он не может выразить свое желание посредством готового изображения, берет лишь то, что оказалось в досягаемости, когда поиск стал для него актуальным. Требуется найти вектор характеристик идеального изображения.

### Адаптивный метод рандомизированных сводных показателей

Вначале рассмотрим метод для решения первой задачи - построения суммарной оценки по выявленным каким-то образом приоритетам. Допустим, некий эксперт обозначил все необходимые приоритеты в категории, к которой относится запрос пользователя.

Н.И. Хованов [41] предлагает использовать метод, названный им Методом рандомизированных сводных показателей для решения данной задачи. Метод основан на построении большого числа случайных примеров и принятии среднего значения показателя по этим примерам как искомой суммарной оценки. В данном случае, предлагается ввести коэффициенты *αi*, выражающие вес (значимость) метрики *ρi* в суммарной оценке. Из указанные величин составлены векторы *α* и *ρ(q,x)*. Кроме того, для масштабирования шкал друг относительно друга следует также ввести коэффициенты , которых нет в исходном методе рандомизированных показателей потому, что показатели берутся из отрезка [0,1]. На основе введенных величин составляются помеченные упорядоченными парами *(i,j)* неравенства вида



выражающие приоритеты. С помощью *αi* можно получить скалярный показатель



где символ ‘’ обозначает скалярное произведение. Далеко не единственный набор коэффициентов {*αi*} удовлетворяет указанным неравенствам. Предлагается провести опыты по расчету показателя : будут сгенерированы случайные наборы *α(k)* с равномерным распределением на интервале *U(0,1)*, отброшены те из них, что не удовлетворяют неравенствам, а затем суммарная оценка будет положена равной среднему значению показателя  для всех векторов показателей. Таким образом, если некоторый эксперт (или сам пользователь) перед началом поиска задал приоритеты, то с их использованием можно получить результат поиска. Подстройку *sij* следует производить однажды алгоритмом типа стохастической аппроксимации на основании наблюдений за реакцией пользователей. Однажды нормированные с помощью корректно настроенных *sij*, шкалы остаются далее неизменными друг относительно друга.

Точность предварительного задания приоритетов обычно нельзя считать высокой. Субъективность восприятия, вероятностное поведение пользователя делают актуальной итеративную подстройку приоритетов. На каждом шаге пользователь помечает понравившиеся изображения. На основе получаемой отсюда информации можно выявлять приоритеты.

Схема подстройки приоритетов задается с помощью введения параметров важности учета приоритета для каждой пары пространств. Параметр важности в вычислительной схеме может интерпретироваться как вероятность применения неравенства. Отсюда можно выявлять новые приоритеты, в явном виде пользователем не обозначенные, корректировать важность применения имеющихся приоритетов, если они противоречат соотношениям между фактически выделенными пользователем изображениями. Предлагается использовать вероятностный параметр , определяющий вероятность того, что ограничение *(i,j)* будет проверено для очередного сгенерированного набора *α(k)*. Корректировку параметра можно производить после каждого шага работы с пользователем по следующей схеме:



Здесь  - значение параметра на шаге *n*, *Nij+* - число изображений, для которых данный приоритет соблюден, среди всех помеченных (как подходящие) изображений, *Nseld* - число всех помеченных изображений, ** - дисконтирующий коэффициент. Для упрощения понимания схемы можно принять условие , однако это не обязательно: можно считать, что одновременная проверка противоречащих неравенств будет означать отсутствие ограничения.

Данная вероятностная схема позволяет динамически подстраивать приоритеты, доверять различным приоритетам по-разному, постепенно избавляться от учета приоритетов, потерявших свою актуальность.

Метод, названный нами «Адаптивным методом рандомизированных сводных показателей», по природе итеративный и соответствует общей направленности реализуемых в **RecService** алгоритмов. Рассмотрим модель алгоритма в терминах **RecService**. Для краткости ограничимся наглядной схемой с отмеченными задачами, компонентами и типами передаваемых между ними данных, а также направлениями этих связей (рис. 4).

Рис. 4

# Заключение. Результаты

Автором разработаны прототип сервиса **RecService** и модели алгоритмов для описанных в третьей главе задач. Сервис демонстрирует свою применимость в задачах с одной подключенной внешней системой и набором связанных заменяемых в рамках одной решаемой задачи компонентов. Рекуррентные алгоритмы типа решающих описанные задачи могут быть применены и в других системах, например [42].

Сервис может быть расширен несколькими новыми модулями. Поддержка моделей на основе задания бизнес-процессов с несколькими участвующими системами позволит полноправно интегрировать **RecService** с другими системами, реализующими стандарты *WSDL* и *UDDI*.

# Язык описания моделей, с которым работает сервис, не обеспечивает поддержки асинхронных процессов. Такая поддержка является существенным для реализации более сложных, чем описанные здесь, моделей в задачах искусственного интеллекта, например см. [43]. Тогда с помощью сервиса станет возможным интегрирование многоагентных систем.ПРИЛОЖЕНИЕ 1.

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

<bean id="sphericOracle" class="com.spbsu.recservice.basicmodel.bounds.SphericBoundOracle">

<!-- rho-->

<property name="deg" value="1.5"/>

<property name="diameter" value="1.31"/>

<property name="center">

<list>

<value>0.0</value>

<value>0.0</value>

</list>

</property>

</bean>

<bean id="bernoulli2DimGen" class="com.spbsu.recservice.basicmodel.randgen.BernoulliVector">

<property name="dimension" value="2"/>

</bean>

<bean id="spsaController" class="com.spbsu.recservice.basicmodel.problems.SPSAProjectOneStep">

<property name="boundOracle">

<ref local="sphericOracle"/>

</property>

<property name="c\_deltav" value="0.0"/>

<property name="init\_value">

<list>

<value>0.8</value>

<value>-0.7</value>

</list>

</property>

<property name="a\_denum" value="0"/>

<property name="a\_num" value="1.0"/>

<property name="alpha" value="0.5"/>

<property name="b\_denum" value="0"/>

<property name="b\_num" value="1.0"/>

<property name="beta" value="0.17"/>

<property name="rand\_dist\_generator">

<ref local="bernoulli2DimGen"/>

</property>

</bean>

</beans>

# Литература

1. Колесов Ю.Б. *Объектно-ориентированное моделирование сложных динамических систем.* СПб. Изд-во СПбГПУ. 2004.
2. Звягин П.Н., Нечаев Ю.И. *Нейросетевое управление морским динамическим объектом.* Научная сессия МИФИ-2006. Нейроинформатика - 2006. Часть 2 Модели адаптивного поведения. Применение нейронных сетей. Применение нейронных сетей. Теория нейронных сетей. Изд-во МИФИ. 2006. С. 81-87
3. Поляк Б.Т. *Введение в оптимизацию.* М.: Наука. 1983.
4. Corkill D.D. *Blackboard Systems*. AI Expert, 6(9), 1991, pp. 40-47
5. Граничин О.Н., Поляк Б.Т. *Рандомизированные алгоритмы оценивания и оптимизации при почти произвольных помехах*. М., Наука. 2003.
6. Kushner H., Yin G. *Stochastic Approximation and Recursive Algorithms and Applications,* 2nd ed.*,* Springer, 2003.
7. *SPSA: A Method For System Optimization,* http://www.jhuapl.edu/spsa
8. *Стохастическая оптимизация в информатике*, сб. под ред. проф. Граничина О.Н. Т. 1. 2005., т. 2. 2006.
9. Fowler M., *Inversion of Control Containers and the Dependency Injection pattern,* http://www.martinfowler.com/articles/injection.html
10. Poutsma A., *Spring Web Services - Reference Documentation,* http://static.springframework.org/spring-ws/site/reference/html/index.html
11. *Hibernate - Relational Persistence for Idiomatic Java. Hibernate Reference Documentation* http://www.hibernate.org/hib\_docs/v3/reference/en/html/
12. Черных И.В., *Simulink: Инструмент моделирования динамических систем,* <http://www.nsu.ru/matlab/MatLab_RU/simulink/>book1/index.asp.htm
13. Robbins H., Monro S., *A stochastic approximation method*, Ann. Math. Statist. V. 22, 1951, pp. 400-407
14. Kiefer J., Wolfowitz J. *Statistical estimation on the maximum of a regression function*. Ann. Math. Statist., V. 23, 1952, pp. 462-466.
15. Spall J. C. *Multivariate stochastic approximation using a simultaneous perturbation gradient approximation*, IEEE Transactions on Automatic Control, V. 37, 1992, pp. 332-341.
16. Граничин О.Н. *Об одной стохастической рекуррентной процедуре при зависимых помехах в наблюдении, использующей на входе пробные возмущения*, Вестник Ленингр. ун-та, сер. 1, в. 1, 1989, с. 92-93
17. Поляк Б.Т., Цыбаков А.Б. *Оптимальные порядки точности поисковых алгоритмов стохастической аппроксимации*, Проблемы передачи информации, 2, 1990, с. 45-53.
18. Blum J. R. *Multidimensional stochastic approximation,* Ann. Math. Statist., V. 9, 1954, pp. 737-744.
19. Вазан М., *Стохастическая аппроксимация*, М.: Мир, 1972.
20. Невельсон М.Б., Хасьминский Р.З. *Стохастическая аппроксимация и рекуррентное оценивание*, М.: Наука, 1972.
21. Михалевич В.С., Гупал А.М., Норкин В.И. *Методы невыпуклой оптимизации* , М.: Наука, 1987.
22. Fabian V. *Stochastic approximation of minima with improved asymptotic speed,* Ann. Math. Statist., v. 38., 1967. pp. 191-200.
23. Граничин О.Н., *Процедура стохастической аппроксимации с возмущением на входе,* Автоматика и телемеханика, 2, 1992, с. 97-104
24. Граничин О.Н. *Оценивание точки минимума неизвестной функции, наблюдаемой на фоне зависимых помех*, Проблемы передачи информации, 2, 1992, с. 16-20.
25. Граничин О.Н., *Оптимальная скорость сходимости рандомизированных алгоритмов стохастической аппроксимации при почти произвольных помехах,* Автоматика и телемеханика, 2, 2003, с. 88-99.
26. Chen H. F., Duncan T. E., Pasik-Duncan B. *A Kiefer-Wolfowitz algorithm with randomized differences*, IEEE Transactions on Automatic Control, V. 44, N 3, 1999. pp. 442-453.
27. Granichin O.N., Vakhitov A.T.  [*Accuracy for the SPSA algorithm with two measurements*](http://www.math.spbu.ru/user/gran/papers/wseas06.pdf). WSEAS Transactions on Systems. № 5.  v. 5. May 2006. pp. 953-957.
28. Вахитов А.Т., Граничин О.Н., Сысоев С.С. [*Точность оценивания рандомизированного алгоритма стохастической оптимизации*](http://www.math.spbu.ru/user/gran/papers/ait06_1.pdf). Автоматика и телемеханика, 2006, № 4 , с.86-96.
29. Вахитов А.Т., Граничин О.Н.(СПбГУ) [Рандомизированные алгоритмы оценивания при нерегулярных помехах](http://www.math.spbu.ru/user/gran/sb2/granichin.pdf), в сб. под ред. проф. Граничина О.Н., т. 2, 2006.
30. Milner R., *The Polyadic pi-Calculus: A Tutorial*, in Logic and Algebra of Specification. Springer-Verlag, 1993.
31. Johnson R. etc. *The Spring Framework - Reference Documentation*, 2.0.5, http://static.springframework.org/spring/docs/2.0.x/reference/index.html
32. Пархоменко П.П. *Теория вопросников (обзор)*. Автоматика и телемеханика, №4 (1970)
33. Valdes-Perez R., Pericliev V., Pereira F. *Concise, Intelligible, and Approximate Profiling of Multiple Classes*. [International Journal of Human-Computer Studies](http://www.ingentaconnect.com/content/ap/hc;jsessionid=14t51qcnhoeif.henrietta), Volume 53, Number 3, September 2000, pp. 411-436(26)
34. Свиридов А.В. *Ключи в биологической систематике: теория и практика.* М: Изд-во МГУ, 1994. 224 с.
35. Лобанов А.Л. *Оценка диагностической ценности рядов признаков в многовходовых определителях, рассчитанных на использование ЭВМ.* Тезисы докладов VI Коми республиканской молодежной научной конференции. 1974. Сыктывкар. c. 125-126.
36. Вахитов А.Т., Граничина О.А. *Алгоритмы классификации за минимальное число шагов* в сб. Стохастическая оптимизация в информатике под ред. Граничина О.Н., вып. 2, 2006, с. 167-174
37. Владимирович А.Г.(СПбГУ) [*Субоптимальный*](http://www.math.spbu.ru/user/gran/sb1/vladim.pdf)[*алгоритм распознавания образов в дискретном случае*](http://www.math.spbu.ru/user/gran/sb1/vladim.pdf). в сб. Стохастическая оптимизация в информатике под ред. Граничина О.Н., вып. 1, 2005, с. 8-16
38. Шошмина И.В. и др. *Использование Grid-технологий для крупномасштабных научных экспериментов. Часть 1. Введение в Grid-технологии с примерами практических занятий на базе ARC Nordugrid.* СПб:Копи-Сервис, 2006. – 49 с.
39. Squire D., Müller W. and Müller H. Relevance feedback and term weighting schemes for content-based image retrieval, Visual Information and Information Systems. 1999. pp. 549-556.
40. Измакова О.А., Сысоев С.С. *Алгоритм стохастической оптимизации с возмущением на входе в задаче самообучения.* Труды Международной школы-семинара Адаптивные роботы 2004. М.-СПб, 2004, с. 49-52
41. Хованов Н. И. Анализ и синтез показателей при информационном дефиците. СПб., СПбГУ, 1996.
42. Гуревич Л.С., *Адаптивный метод выбора каналов новостей*. Дипломная работа, каф. системного программирования математико-механического факультета СПбГУ, 2007.
43. Граничин О.Н., Жувикина И.А. [*Новая модель процесса вычислений: обобщение концепции машины Тьюринга*](http://www.math.spbu.ru/user/gran/papers/granzhuv.doc)*.* Нейрокомпьютеры: разработка, применение, 2006, № 7 , с.24-31.

1. Разработан в рамках студенческого проекта по GRID вычислениям, проводимого в лаборатории Intel на математико-механическом факультете СПбГУ [↑](#footnote-ref-2)
2. Выполняется в рамках гранта «Интернет-математика 2007» от компании Яндекс [↑](#footnote-ref-3)