РЕШЕНИЕ ЗАДАЧ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ БОЛЬШОЙ РАЗМЕРНОСТИ НА МНОГОПРОЦЕССОРНЫХ КОМПЛЕКСАХ И ГРИД-СИСТЕМАХ[1]

А.П. Афанасьев, Ю.Г. Евтушенко, М.А. Посыпкин, И.Х. Сигал

ВВЕДЕНИЕ

Задачи поиска глобального экстремума функции часто настолько сложны, что ресурсов однопроцессорной рабочей станции оказывается недостаточно. В этом случае целесообразен переход к расчетам на многопроцессорных и Грид-системах. В данной работе рассматриваются принципы организации решения задач оптимизации на таких системах и программный комплекс, позволяющий решать задачи глобальной дискретной и непрерывной оптимизации в средах параллельных и распределенных вычислений.

РАССМАТРИВАЕМЫЕ ЗАДАЧИ И МЕТОДЫ РЕШЕНИЯ

Задача поиска глобального минимума (максимума) функции на допустимом множестве X состоит в отыскании такой точки $x_* \in X$, что $f(x_*) \le f(x)$ ($f(x_*) \ge f(x)$) для всех $x \in X$. Далее будем предполагать, что решается задач минимизации функции *f*. Ограничения, связанные с вычислительной погрешностью или недостатком ресурсов, часто не позволяют найти точное решение данной задачи. В этом случае переходят к поиску приближенного решения, т.е. точки из множества ε -оптимальных решений $X_*^\varepsilon = \{x \in X : f(x) \le f(x_*) + \varepsilon\}$. Поиск точного решения можно рассматривать как частный случай поиска приближенного решения с $\varepsilon = 0$.

В работе рассматриваются *методы покрытия*[1] для поиска глобального экстремума. К этому классу можно отнести различные варианты метода ветвей и границ, отсечений и др. Методы покрытий имеют две основные составляющие: построение покрытия и нахождении рекорда. Вычисление рекорда заключается в построении последовательности точек $x_1, \ldots, x_k$ допустимого множества X и вычисление рекордной точки x_r по формулам: $f_r = f(x_r) = \min\limits_{1 \le i \le k} f(x_i)$ после добавления каждой новой точки. *Покрытие* является последовательностью $Z_1, \ldots, Z_m$ подмножеств допустимого множества, заведомо не содержащих точек, которые бы улучшали найденный рекорд более чем на заданную величину ε. Обычно для проверки этого факта используется нижняя оценка g_i на множестве Z_i: $f(x) \ge g_i$ при $x \in Z_i$. Тогда если известно, что $g_i \ge f_r - \varepsilon$, то множество Z_i может быть добавлено к покрытым множествам и исключено из дальнейшего рассмотрения. Применяются и другие условия добавления, называемые также *условиями (правилами) отсева.* Вычисления останавливаются, как только последовательность $\{Z_i\}$ покрывает все допустимое множество: $X \subseteq \bigcup\limits_{i=1}^{n} Z_i$. В этом случае рекордная точка x_r является ε -оптимальным решением.

В данной работе ограничимся рассмотрением *методов ветвей и границ* (МВГ), в которых покрытие получается с помощью разбиения допустимого множества. Приведем общую схему метода. На протяжении всего времени работы поддерживается список $\{X_i\}$ подмножеств допустимого множества. Первоначально он состоит из одного элемента – допустимого множества X. Далее выбирается один из элементов списка – подмножество X_i. Если X_i удовлетворяет условию отсева, то выбранный элемент удаляется из списка (становится элементом покрытия $\{Z_i\}$). В противном случае он подвергается дроблению на более мелкие подмножества, которые замещают в списке выбранный элемент. Алгоритм завершает свою работу, когда в последовательности $\{X_i\}$ не остается ни одного элемента. В этом случае множества $\{Z_i\}$ образуют покрытие допустимого множества.

Часто в процессе нахождения рекорда последовательность точек $x_1, \ldots, x_k$, в которых вычисляется значения целевой функции, формируется на основе разбиваемых множеств. Например, если множества $\{X_i\}$ являются

[1] Работа выполнена при финансовой поддержке программы №14 фундаментальных исследований Президиума РАН «Раздел II: Высокопроизводительные вычисления и многопроцессорные системы» 2008 г., а также программы №15 Президиума РАН «Исследование и создание научных приложений в распределенной среде суперкомпьютерных приложений на базе ВЦ РАН» 2008 г.

параллелепипедами, то в качестве элементов последовательности $x_1,\ldots,x_k$ берутся их центры. Также возможен подход, при котором последовательность точек $x_1,\ldots,x_k$ формируется независимым от построения покрытия способом.

Быстрое нахождение значения рекорда, близкого к оптимуму, может сделать отсев подмножеств более эффективным и тем самым ускорить решение задачи. Для нахождения рекорда применяют различные *эвристические* алгоритмы, которые позволяют эффективно находить хорошие приближения к оптимуму, но не гарантируют оптимальности. В качестве эвристик могут применяться различные локальные алгоритмы (метод градиентного спуска, Ньютона, перебор в некоторой окрестности и т.п.), а также более сложные алгоритмы, например стохастические или генетические подходы. Таким образом, в решении задачи участвуют как алгоритмы, выполняющие разбиение, так и эвристические алгоритмы. Они взаимодействуют в соответствии со следующей схемой (рис. 1): в процессе разбиения допустимого множества генерируются точки, предоставляемые для обработки эвристическому алгоритму. Эвристический алгоритм обрабатывает предоставленные решения, полученный в результате улучшенный рекорд используется для усиления отсева при выполнении разбиений.

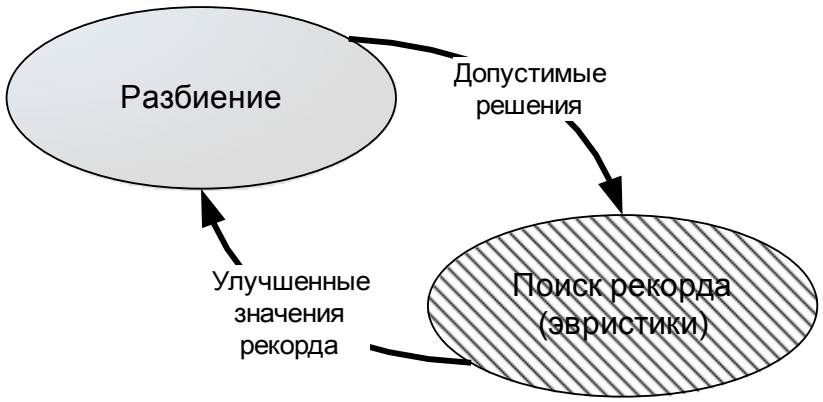


Рис.1. Взаимодействие разбиений и эвристических алгоритмов.

ПРИНЦИПЫ РЕАЛИЗАЦИИ МЕТОДА ВЕТВЕЙ И ГРАНИЦ НА СИСТЕМАХ С РАСПРЕДЕЛЕННОЙ ПАМЯТЬЮ

Вычислительная система с распределенной памятью[2] состоит из некоторого числа узлов (модулей), обладающих локальной памятью, взаимодействующих с помощью коммуникационной сети. Общая идея реализации разбиений на системах с распределенной памятью заключается в том, что каждый процессор работает со своим списком подмножеств, сохраняемым в локальной памяти. Если в результате выполнения разбиений локальный список становится пустым, то для предотвращения простоев дополнительные множества пересылаются с другого процессора. Задача считается решенной, когда на всех процессорах списки не содержат подмножеств.В данной работе рассматриваются *централизованные* схемы организации вычислений: выделенный управляющий процессор отправляет рабочим процессорам области поиска и последний рекорд. Эти процессоры выполняют разбиения полученного подмножества, при необходимости пересылая на управляющий процессор часть подмножеств. Конкретный пример алгоритма балансировки нагрузки приведен в разделе, посвященном реализации.

Рассмотрим два возможных способа параллельной реализации нахождения рекорда в методе ветвей и границ. При первом подходе разбиение и нахождение рекорда выполняются на каждом процессоре, который периодически переключается между этими действиями (рис. 2). Данный подход был успешно применен для распараллеливания процесса поиска глобального экстремума непрерывной функции многих переменных[3]. Такой способ особенно удобен, когда для построения рекорда используются точки, полученные в процессе разбиения, к которым применяется один из методов локальной оптимизации. В тоже время в рамках данной схемы сложно создать параллельную реализацию алгоритма, работающего со всей совокупностью точек $x_1,\ldots,x_k$. В этом случае предпочти-

тельнее схема, при которой одна группа процессоров выполняет разбиения, а другая задействована в поиске рекорда (рис. 3).

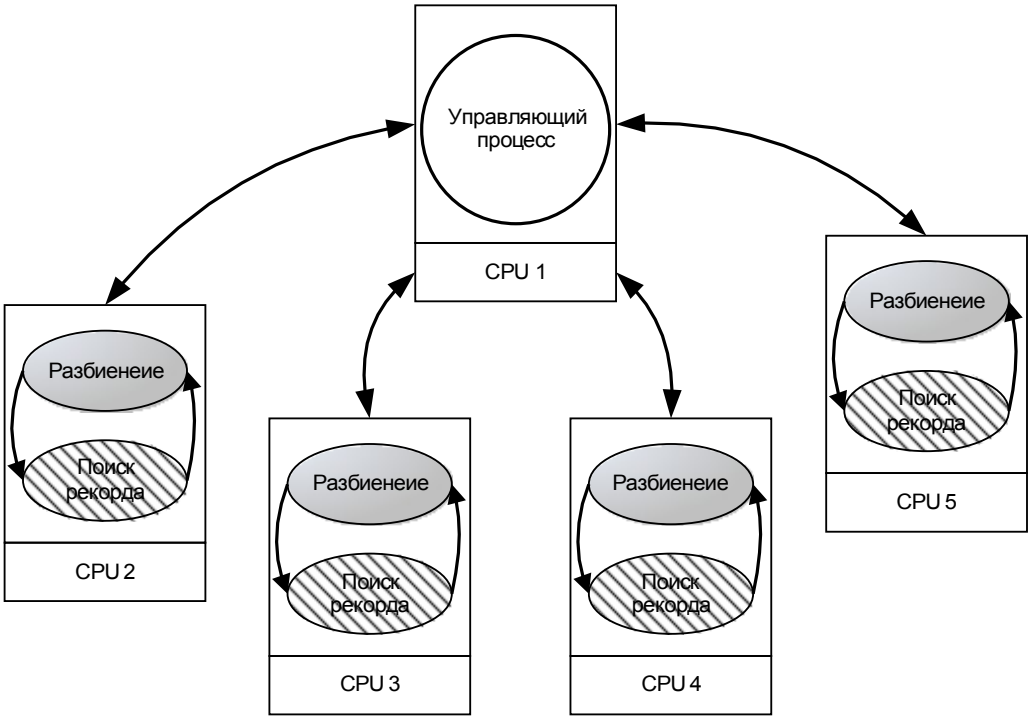


Рис. 2. Разбиение и построение рекорда выполняются на одном процессоре

Рассмотрим базовую схему, подходящую для большинства алгоритмов, применяемых при нахождении рекорда. Выделенный управляющий процессор поддерживает список точек $x_1,\ldots,x_k$. Некоторая точка удаляется из списка и рассылается всем либо некоторым из рабочих процессоров, которые подвергают ее процедуре оптимизации (как правило, одному из вариантов локального поиска). В результате на рабочем процессоре образуется список точек, который пересылается управляющему процессору и объединяется с его списком. Максимальная длина списка точек, правила выбора очередной точки для обработки, распределение точек по рабочим процессорам, способ добавления точек, полученных от рабочих процессоров к общему списку, локальный алгоритм являются параметрами общей схемы и определяют конкретный эвристический метод поиска рекорда. Список точек на управляющем процессоре также может пополняться точками, полученными в результате разбиений.

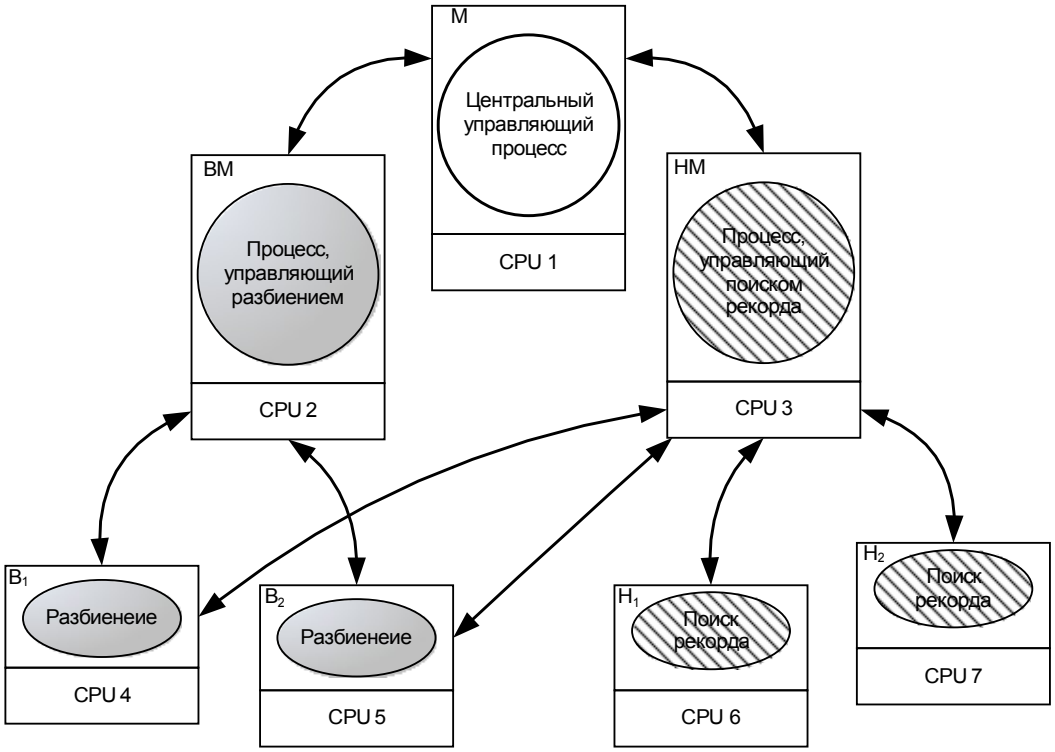


Рис. 3. Разбиение и построение рекорда выполняются на разных процессорах

РЕАЛИЗАЦИЯ ДЛЯ МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ КОМПЛЕКСОВ

Вычисления в рассматриваемых алгоритмах проводятся по общей схеме, а различия наблюдаются в способах вычисления верхних и нижних оценок, способе разбиения допустимого множества, эвристических процедурах. Поэтому представляется целесообразным разработать единую параметризованную реализацию общей схемы. Реализации конкретных методов при этом получаются за счет подстановки в нее требуемых алгоритмов в качестве параметров. Такой подход особенно эффективен на многопроцессорных вычислительных комплексах (МВК), так как в этом случае параллельная реализация также делается на уровне общей схемы, и при добавлении очередного алгоритма или класса задач работу по распараллеливанию повторно проводить не требуется. Рассмотренный подход поддерживается библиотекой BNB-Solver[4], предназначенной для поиска глобального экстремума на МВК. Библиотека написана на языке Си++ с использованием механизма шаблонов (templates), которые позволяют эффективно реализовать параметрический подход. Для распараллеливания используется MPI.

На рис. 3 процессы помечены следующим образом: M – процесс, управляющий работой всего приложения; $\{B_i\}$ - процессы, которые выполняют операции разбиения, BM – процесс, координирующий работу процессов $\{B_i\}$, $\{H_i\}$ – процессы, которые выполняют эвристический поиск, HM координирует работу этих процессов. На начальном этапе процесс M рассылает исходные данные задачи остальным процессам. Разбиение выполняется группой, состоящей из управляющего процесса BM и рабочих процессов B_i. Процесс HM получает точки от процессов H_i и B_i, обновляет значение рекорда и рассылает его по всем процессам B_i, реализующим разбиения. Также HM направляет полученную точку всем (или некоторым, в зависимости от выбранного алгоритма) процессам H_i, выполняющим эвристический поиск. Если на процессе H_i удается улучшить рекорд, то соответствующая рекордная точка посылается процессу HM.

Библиотека BNB-Solver предоставляет возможность подключать различные алгоритмы распределения работы между процессами $\{B_i\}$, выполняющими разбиение. На данный момент применяется следующая схема. Процесс B_i управляется двумя параметрами – T и S, посылаемыми ему управляющим процессом. Получив подмножество для обработки, B_i выполняет T разбиений, после чего пересылает S сгенерированных подмножеств на управляющий процесс BM. По окончании разбиения своего подмножества процесс B_i получает от BM новое подмножество. Для предотвращения переполнения памяти на управляющем процессе было введено два пороговых параметра m_0 и M_0, $m_0 < M_0$. Если число подмножеств на управляющем процессе становится больше M_0, то управляющий процесс рассылает всем рабочим процесса новое значения T, равное -1, и процессы B_i перестают пересылать подмножества управляющему процессу BM. Когда число подмножеств на управляющем процессе становится меньшим m_0,

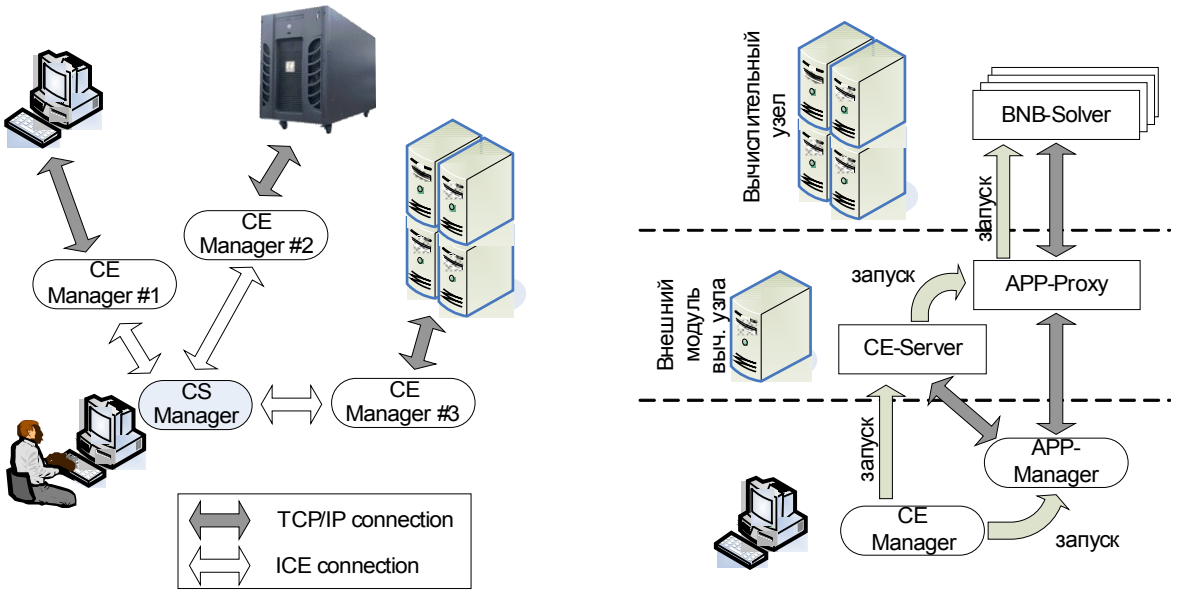
он рассылает рабочим процессам первоначальные значения T , M и передача подмножеств с рабочих процессов управляющему возобновляется.

Библиотека BNB-Solver применялась для решения классических задач дискретной оптимизации о коммивояжере[5] и ранце[6], а также для поиска глобального экстремума функции многих переменных[7].

ПЕРЕХОД К РАСПРЕДЕЛЕННЫМ СИСТЕМАМ: ОБЪЕДИНЕНИЕ НЕСКОЛЬКИХ РАЗНОРОДНЫХ ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСОВ

Для решения многих практически важных задач оптимизации одного МВК оказывается недостаточно и приходится задействовать в расчетах сразу несколько вычислительных ресурсов, т.е. переходить к расчетам в распределенной вычислительной средех[8]. В настоящее время в распоряжении специалиста, как правило, имеются ресурсы следующих типов: рабочие станции, небольшие многопроцессорные комплексы, доступные в монопольном режиме, суперкомпьютеры общего пользования. Программный комплекс BNB-Grid предназначен для решения задач оптимизации на распределенных системах, состоящих из узлов перечисленного типа. BNB-Grid запускает и организует взаимодействие параллельных приложений, решающих задачу с помощью библиотеки BNB-Solver на различных вычислительных узлах. В результате формируется иерархическая распределенная система: на верхнем уровне работа (подмножества или точки в методе покрытий) распределяются между параллельными приложениями, а далее они распределяются по процессорам средствами библиотеки BNB-Solver.

Ядро системы реализовано на языке программирования Java с использованием промежуточного программного обеспечения Internet Communication Engine (ICE)[9] – аналога CORBA. Каждый вычислительный узел представлен в системе распределенным экземпляром объекта типа CE-Manager – Computing Element Manager (рис. 4(а)). Этот объект предоставляет интерфейс для запуска и остановки приложений на вычислительном узле. Координация работы объектов типа CE-Manager осуществляется с помощью другого распределенного объекта типа CS-Manager - Computing Space Manager.



(а) (б)

Рис. 4. Схемы организации вычислений (а) и запуска приложения на вычислительном узле (б)

Доступ к удаленному вычислительному узлу осуществляется по следующей схеме (рис. 4 (б)). Объект CE-Manager устанавливает SSH-соединение с управляющим модулем узла, запускает на нем процесс CE-Server и устанавливает с ним TCP/IP соединение. Если удаленный узел защищен сетевыми экранами, то применяется механизм туннелирования сетевого соединения через SSH-канал. Процесс CE-Server информирует CE-Manager о состоянии вычислительного узла. При обрыве соединения или перезагрузке узла CE-Server перезапускается. При получении запроса на запуск параллельного приложения CE-Manager создает объект APP-Manager, CE-Server создает процесс APP-Proxy, который запускает параллельное приложение BNB-Solver на узле. При этом CS-Manager взаимодействует с APP-Manager средствами ICE, а APP-Manager, в свою очередь, устанавливает TCP/IP соединение с BNB-Solver через APP-Proxy. Процесс APP-Proxy необходим, так как на суперкомпьютерах общего доступа непосредственный доступ из внешней сети к вычислительным модулям, как правило, невозможен.

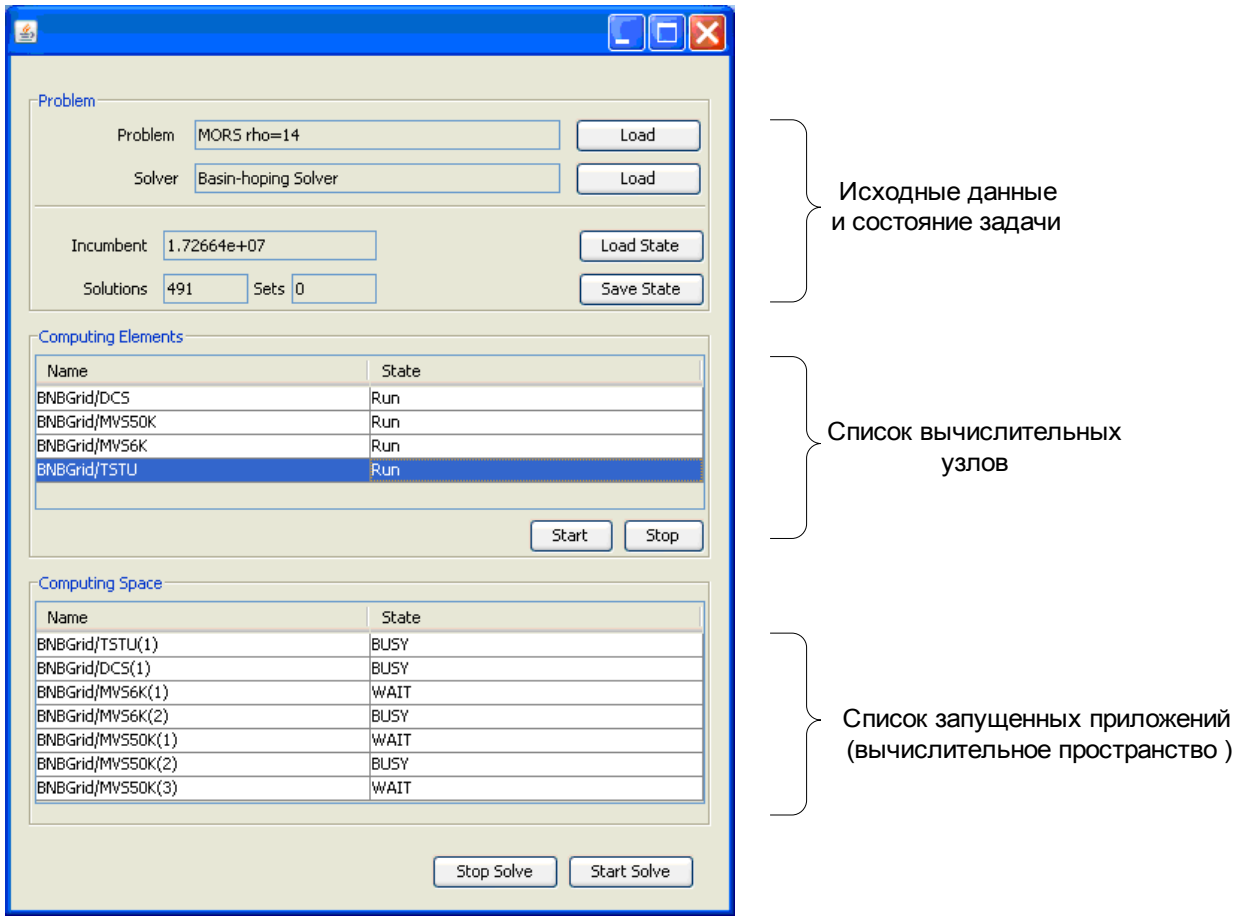


Рис. 5. Главное окно графического интерфейса пользователя системы BNB-Grid.

Запуск параллельных приложений производится в соответствии с соглашениями, принятыми для вычислительного узла. На рабочих станциях и суперкомпьютерах монопольного доступа приложение стартует практически сразу после поступления команды. На кластерах общего доступа с системой пакетной обработки приложение сначала помещается в очередь и запускается через некоторое время, когда запрошенные ресурсы освобождаются. По истечении запрошенного временного интервала приложение завершается. Поэтому в системе BNB-Grid сохраняются резервные копии заданий, переданных параллельным приложениям. В случае аварийного завершения эти резервные копии передаются для обработки другим приложениям.

Загрузка исходных данных, создание вычислительного пространства и управление процессом вычислений контролируются пользователем с помощью графического интерфейса (рис. 5). В главном окне отображаются характеристики исходной задачи, список вычислительных узлов, список запущенных приложений и их состояния.

РЕЗУЛЬТАТЫ ВЫЧИСЛИТЕЛЬНЫХ ЭКСПЕРИМЕНТОВ

Система BNB-Grid была применена для решения задачи о ранце методом ветвей и границ[10] и задачи поиска конфигурации молекулы с минимальной потенциальной энергии взаимодействия, которая формулируется следующим образом. Пусть заданы координаты $x = \left\{x^{(1)}, \ldots, x^{(n)}\right\}$ расположения n атомов в молекуле. Тогда потенциальная энергия молекулы задается функцией $F(x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} v\left(\left\|x^{(i)} - x^{(j)}\right\|\right)$, где $\left\|x^{(i)} - x^{(j)}\right\|$ - евклидово расстояние между атомами i и j, а функция $v(r)$ задает потенциал Морзе парных взаимодействий атомов: $v(r,\rho) = e^{\rho(1-r)}(e^{\rho(1-r)} - 2)$. Требуется найти расположение атомов в пространстве, при котором значение функции $F(x)$ минимально. Эта задача является одной из наиболее сложных задач современной оптимизации. При числе атомов, превосходящем 10, известные методы, доказывающие оптимальность, не позволяют найти минимум за приемлемое время. Поэтому применяют эвристические алгоритмы. Одним из наиболее известных алгоритмов является

Monotonic Basin Hoping (MBH)[11], основанный на комбинации случайного поиска в окрестности и методов локальной оптимизации.

В экспериментах использовались вычислительные узлы, перечисленные в табл. 1. На рабочей станции DCS и кластере TSTU были задействованы все доступные процессоры. На суперкомпьютерах MVS100K и MVS6K запрашивался запуск трех приложений, в каждом по 5 рабочих процессов с лимитом времени 2 часа. Перед началом расчетов компонент CS-Manager загружал исходные данные задачи и 512 случайных стартовых точек для MBH. Далее эти точки распределялись по работающим приложениям BNB-Solver, а те, в свою очередь, распределяли их внутри по своим процессам. После окончания обработки выделенного количества точек на узел передавалась новая порция работы.

Таблица 1

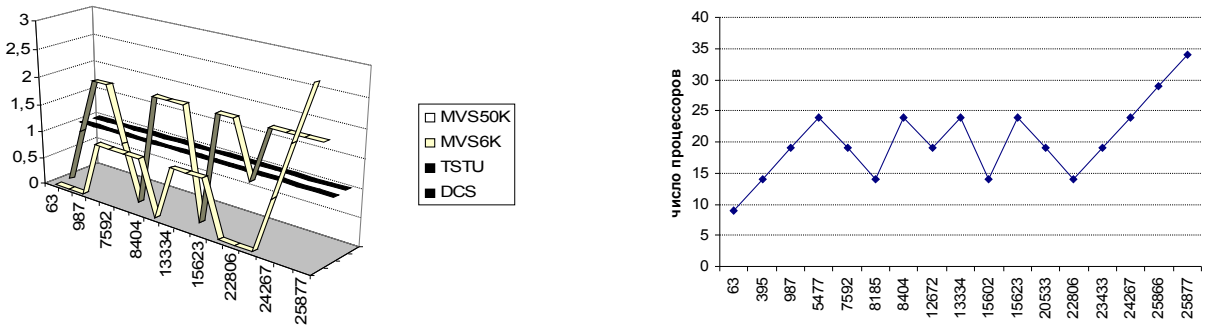
Название	Архитектура процессора	Число процессоров	Местоположение	Наличие системы пакетной обработки
MVS50K	Clovertown (4 core), 3 GHz	940	Межведомственный суперкомпьютерный центр (Москва)	+
MVS6K	Itanium II, 2.2 GHz	256	Вычислительный центр РАН (Москва)	+
TSTU	Pentium IV, 3.2 GHz	8	Тамбовский Государственный Университет (Тамбов)	-
DCS	Pentium IV, 3.2GHz	1	Институт системного анализа РАН (Москва)	-

Для экспериментов были выбраны четыре конфигурации с потенциалом Морзе ($\rho = 14$) и количеством атомов 50, 60, 70, 80. Эти конфигурации являются наиболее сложными и согласно [12] их не удавалось найти без применения геометрически-обоснованных эвристик. В приведенном эксперименте это удалось сделать за счет вовлечения в расчет значительного вычислительного ресурса. Расчеты для каждой конфигурации проводились в течение 10 часов. В табл. 2 приведены времена расчетов, затраченные на нахождение известных минимумов.

Таблица 2

Количество атомов	Время нахождения минимума	Значение
50	19 мин.	-198.456
60	26 мин.	-244.579
70	2 часа 11 мин.	-292.463
80	4 часа 32 мин.	-340.811

На протяжении вычислений количество запущенных приложений и, соответственно, общее число процессоров изменялось. На рис. 6 представлены графики изменения вычислительного пространства за примерно 7 часов работы приложения для конфигурации из 70 атомов. График (а) показывает как изменялось количество экземпляров приложения BNB-Solver на каждом из вычислительных узлов. На компьютерах, доступных в монопольном режиме DCS и TSTU, это количество равняется 1 в течение всего времени измерений. На суперкомпьютерах с системой пакетной обработки это число изменяется в диапазоне от 0 до 3. График (б) показывает общее число задействованных процессоров, как функцию времени.



(а) (б)

Рис.6. Изменение вычислительного пространства по ходу вычислений – число приложений (а) и общее число процессоров (б) как функция времени (в секундах)

Результаты экспериментов показывают, что предложенный подход и разработанный на его основе комплекс программ позволяет решать сложные вычислительные задачи оптимизации на многопроцессорных вычислительных комплексах и Грид-системах. В дальнейшем планируется расширить спектр решаемых задач и провести эксперименты на Грид-системах, состоящих из 10-100 вычислительных узлов.

ЛИТЕРАТУРА:

1. Ю.Г. Евтушенко "Численный метод поиска глобального экстремума функций (перебор на неравномерной сетке) " // ЖВМ и МФ, 1971, Т. 11, № 6, С. 1390-1403.
2. В.В. Воеводин, Вл.В. Воеводин "Параллельные вычисления". СПб.: БХВ-Петербург, 2002. 608 с.
3. Ю.Г. Евтушенко, В.У. Малкова, А.А. Станевичюс "Распараллеливание процесса поиска глобального экстремума" // Автоматика и телемеханика, 2007, № 5, С. 56-58.
4. М.А. Посыпкин "Архитектура и программная организация библиотеки для решения задач дискретной оптимизации методом ветвей и границ на многопроцессорных вычислительных комплексах" // Труды ИСА РАН, 2006, Т. 25, С. 18-25.
5. И.Х. Сигал, Я.Л. Бабинская, М.А. Посыпкин "Параллельная реализация метода ветвей и границ в задаче коммивояжера на базе библиотеки BNB-Solver" // Труды ИСА РАН, 2006, Т. 25, С. 26-36.
6. М.А. Посыпкин, И.Х. Сигал "Применение параллельных эвристических алгоритмов для ускорения параллельного метода ветвей и границ" // ЖВМиМФ, том 47, № 9, Сентябрь 2007 , С. 1524-1537.
7. Ю.Г. Евтушенко, М.А. Посыпкин "Параллельные методы решения задач глобальной оптимизации" // принята к публикации в сборнике трудов IV Международная конференции «Параллельные вычисления и задачи управления», 2008.
8. А.П. Афанасьев, В.В. Волошинов, С.В. Рогов, О.В. Сухорослов "Развитие концепции распределенных вычислительных сред" // Проблемы вычислений в распределенной среде: организация вычислений в глобальных сетях. Труды ИСА РАН. - М.: РОХОС, 2004, с.6-105.
9. M. Henning "A New Approach to Object-Oriented Middleware" // IEEE Internet Computing, Jan 2004.
10. А.П. Афанасьев, В.В. Волошинов, М.А. Посыпкин, И.Х. Сигал, Д.А. Хуторной "Программный комплекс для решения задач оптимизации методом ветвей и границ на распределенных вычислительных системах" // Труды ИСА РАН, 2006, Т. 25, с. 5-17.
11. R. H. Leary "Global Optimization on Funneling Landscapes" // J. of Global Optimization 18, 4 (Dec. 2000), 367-383.
12. A. Grosso, M. Locatelli, F. Schoen "A population based approach for hard global optimization problems based on dissimilarity measures" // Mathematical Programming, 2007, 110(2):373-404.