ПРИКЛАДНЫЕ ЗАДАЧИ В ПАРАЛЛЕЛЬНОЙ И РАСПРЕДЕЛЕННОЙ СРЕДЕ

Программный комплекс для решения задач дискретной оптимизации на распределенных вычислительных системах*

А. П. Афанасьев, В. В. Волошинов, М. А. Посыпкин, И. Х. Сигал, Д. А. Хуторной

Аннотация

Задачи дискретной оптимизации являются математическими моделями многих проблем, встречающихся в экономике, управлении системами, технических приложениях, военном деле и других областях. С ростом размерности решаемых задач возникают принципиальные вычислительные трудности при нахождении точного или приближенного решения. Традиционно применяемые последовательные вычислительные технологии, как правило, не позволяют найти решение за приемлемое время. Поэтому представляется естественной разработка технологий решения таких задач, основанных на применении методов параллельных и распределенных вычислений. В данной работе рассматриваются разработанные авторами методы и инструментальные средства для решения задач дискретной оптимизации большой размерности на распределенных системах, объединяющих несколько разнородных по производительности и архитектуре многопроцессорных вычислительных комплексов (МВК), связанных между собой с помощью локальной сети или через Интернет.

1. Введение

К задачам дискретного программирования (или оптимизации) относятся задачи нахождения экстремума функции на некотором конечном множестве. Общая постановка задачи дискретного программирования имеет следующий вид. Пусть задана функция f, такая что $f : G \to \Re$, $0 < |G| < \infty$.

* Работа выполнена при поддержке РФФИ, проект 06–07–89079-а.

Требуется найти элемент x^0 из множества G, на котором функция f принимает экстремальное (максимальное или минимальное — в зависимости от постановки) значение:

$$f(x^0) = \max f(x), \quad x \in G.$$

Задачи дискретного программирования являются математическими моделями многих проблем, встречающихся в экономике, управлении системами, технических приложениях, военном деле и других областях.

Перечислим некоторые прикладные задачи, математические модели которых — задачи дискретного программирования:

- задачи о расписаниях;
- размещение объектов на неоднородной территории;
- задачи трассирования;
- маршрутизация;
- распределение ограниченных ресурсов в различных постановках;
- задачи транспортного типа и др.

Для решения этих задач применяются, как правило, комбинаторные методы. При этом метод ветвей и границ [1] и различные его модификации применяются наиболее часто и относятся к числу основных при решении задач дискретного программирования. Решение задач большой размерности [2] на последовательных вычислительных машинах может потребовать существенных вычислительных ресурсов (время, память), что связано с перебором большого числа вариантов. Возможным путем ускорения процесса решения таких задач является применение методов параллельных [3] и распределенных [4] вычислений.

В данной работе рассматривается программный комплекс BNB-Grid, предназначенный для решения задач дискретной оптимизации методом ветвей и границ в распределенной среде. Основные характеристики и отличительные особенности распределенной среды рассматриваются в разделе 2. Раздел 3 посвящен описанию метода ветвей и границ. В разделе 4 представлена программная архитектура системы, дано описание основных ее компонент. Раздел 5 содержит результаты вычислительного эксперимента.

2. Характеристики распределенной среды

Под распределенной вычислительной средой будем понимать совокупность вычислительных машин, связанных между собой с помощью локальной сети или через Интернет. Вычислительные машины, входящие в распределенную систему будем называть *узлами* этой системы. Узлы могут быть как обычными рабочими станциями с одним процессором, так и *многопроцессорными вычислительными комплексами* (МВК).

Заметим, что *распределенные системы* и *параллельные системы* имеют ряд общих черт, но при этом отличаются характеристиками, перечисленными в табл. 1. Параллельная система может быть узлом распределенной системы, но не наоборот.

Таблица 1

Характеристика	Параллельная система	Распределенная система
Степень связности	Высокая степень связности: высокопроизводительная сеть (интерконнект) либо общая память	Слабая связность: каналы с относительно низкими показателями пропускной способности и задержки
Узлы	Как правило, однородные по производительности и архитектуре процессорные устройства	Неоднородные по производительности и архитектуре вычислительные машины
Готовность выполнять вычисления	Выделенные процессоры доступны в течение всего времени вычислений	Узлы могут выходить из состава распределенной системы или наоборот добавляться к ней в процессе вычислений

Перечисленные в таблице различия между параллельным и распределенными системами служат причиной различий в подходах к численному решению задач на этих системах. В частности, при решении задач на распределенной системе следует распределять нагрузку с учетом неоднородной производительности вычислительных узлов и относительно низкой скорости передачи данных.

Другим существенным аспектом, который необходимо учитывать при организации распределенных вычислений, является возможность изменения состава среды в процессе работы приложения: вычислительные узлы могут добавляться к распределенной системе или наоборот, выходить из нее. Рассмотрим в качестве примера ситуацию, когда в состав распределенной вычислительной системы входит несколько МВК, на каждом из которых установлена система управления потоком заданий (СУПЗ). В этом случае пользователь имеет возможность «заказать» выполнение задания, которое будет помещено в очередь и начнет выполняться только через некоторое время, определяемое степенью загрузки МВК. Вследствие этого задачи, входящие в состав распределенного приложения, начинают работать в разное время. Система управления заданиями обычно позволяет поставить задачу на счет только в течение определенного времени, по истечении которого приложение принудительно завершается, и вычислительный узел выходит из состава распределенной среды. Вычислительный узел может также покинуть распределенную среду в связи с разрывом соединения или сбоем аппаратуры.

Вследствие перечисленных причин при организации распределенных вычислений необходимо применять методы динамической балансировки нагрузки и предусматривать возможность изменения состава распределенной среды в процессе вычислений. Кроме этого, алгоритмы, предназначенные для использования в среде распределенных вычислений, должны

допускать декомпозицию на совокупность слабо связанных между собой заданий, между которым не происходит интенсивного обмена данными.

Перечисленным требованиям удовлетворяет метод ветвей и границ [1], который подробно рассматривается в следующем пункте. Этот метод является одним из основных, применяемых для решения задач дискретной оптимизации. Он состоит в последовательной декомпозиции исходной задачи на подзадачи и отсеве подзадач, заведомо не содержащих оптимального решения. Этот метод обладает большой привлекательностью с точки зрения применения технологий параллельных и распределенных вычислений, так как разные подзадачи могут обрабатываться одновременно и независимо друг от друга.

3. Метод ветвей и границ

Метод ветвей и границ является одним из основных методов, применяемых при численном решении задач дискретной оптимизации. Суть метода заключается в последовательном разбиении множества допустимых решений на подмножества с последующим отсевом подмножеств, не содержащих решения.

Рассмотрим задачу дискретного программирования $f(x) \to \max$, $x \in G$. Будем говорить, что функция $\phi : 2^G \to \Re$, является *верхней оценкой* для функции f, если для любого подмножества G' множества G справедливо следующее соотношение: $\forall x \in G' : \phi(G') \geqslant f(x)$. *Правило отсева* подмножеств, не содержащих оптимальных решений, состоит в том, что если для некоторого $x_0 \in G$ и $G' \subseteq G$ известно, что $\phi(G') < f(x_0)$, то подмножество G' может быть исключено из дальнейшего поиска, так как согласно определению верхней оценки, оно не содержит оптимальных решений. При применении данного правила в качестве величины $f(x_0)$ берется наиболее близкое к экстремуму значение, найденное к текущему моменту. Это значение принято называть *рекордом*.

Для получения оценки решается *оценочная задача*. На основании результатов решения оценочной задачи подмножество S исключается из дальнейшего процесса поиска, если для него выполнено хотя бы одно из следующих свойств:

1. S не содержит допустимых решений.
2. При решении оценочной задачи на подмножестве S получено допустимое решение исходной задачи.
3. S может быть отсеяно по правилу отсева.

В процессе решения поддерживается *список задач-кандидатов*, который в начальный момент содержит только один элемент, соответствующий всему множеству допустимых решений, определяемому исходной задачей. На каждом шаге в методе ветвей и границ производится разбиение (ветвление) одного из элементов списка задач-кандидатов на несколько новых задач, для каждой из которых решается оценочная задача. В этом списке

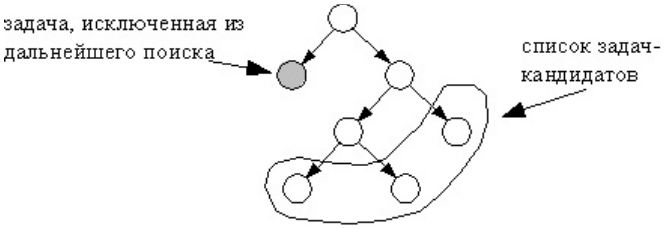

Рис. 1. Дерево ветвления в методе ветвей и границ

задача, подвергнутая ветвлению, заменяется задачами, полученными в результате этого ветвления. При этом, какая-то из этих задач может быть отсеяна по правилам 1–3, перечисленным выше.

Графически процесс решения задачи методом ветвей и границ можно представить в виде дерева, называемого *деревом ветвлений*. Вершинам дерева соответствуют задачи, получаемые в результате ветвления, а дуги соединяют данную задачу с задачами, полученными из нее в результате ветвления. Ветвлению подвергаются задачи, соответствующие концевым вершинам дерева. Будем обозначать затемненным кружком задачи, исключенных из дальнейшего процесса поиска по правилам 1–3, а белым кружком — все прочие задачи. Тогда текущий список задач-кандидатов соответствует всем белым концевым вершинам дерева ветвления (рис. 1).

Общая идея параллельной либо распределенной реализации метода ветвей и границ заключается в том, что каждый узел поддерживает свой список задач-кандидатов и производит ветвление. При этом различные узлы взаимодействуют между собой с целью передачи задач-кандидатов или обмена значениями рекордов.

4. Архитектура программного комплекса для распределенного решения задач методом ветвей и границ

4.1. Общая организация

Программный комплекс BNB-Grid предназначен для решения задач методом ветвей и границ на распределенных вычислительных системах. Его структура представлена на рис. 2. Вычисления на узлах, входящих в распределенную систему, выполняются библиотекой BNB-Solver [5–7], которая предназначена для решения задач методом ветвей и границ на последовательных и параллельных вычислительных системах. Управление процессом вычислений и обмен информацией между различными вычислительным узлами осуществляется с помощью среды IARnet [8].

Среда IARnet позволяет организовывать взаимодействие разнородных географически удаленных информационно-вычислительных ресурсов. Для доступа к ресурсу используются так называемые *агенты*. С точки зрения

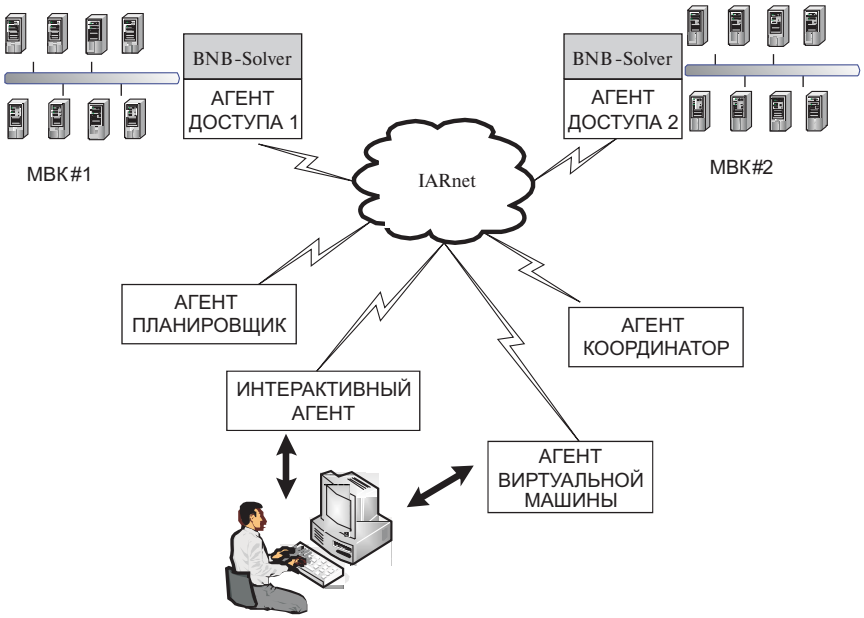

Рис. 2. Архитектура программного комплекса BNB-Grid

программной реализации агент представляет собой Java-процесс, совокупность которых образует распределенное приложение. В состав BNB-Grid входят следующие агенты:

- **Агент-координатор.** Является центральным агентом во всей системе, принимает и обрабатывает запросы от остальных агентов.
- **Управляющий агент виртуальной машины.** Считывает конфигурацию виртуальной машины из файла и загружает ее на агент-координатор.
- **Интерактивный агент.** Осуществляет взаимодействие с пользователем, предоставляет ему информацию о ходе процесса вычислений и возможность управлять этим процессом.
- **Агент-планировщик.** Автоматизирует процесс управления процессом вычислений.
- **Агент доступа к МВК.** Предоставляет интерфейс для доступа к высокопроизводительному вычислительному ресурсу, который позволяет инициировать выполнение приложения на удаленном ресурсе, осуществлять взаимодействие с работающим приложением и диагностировать сбои в работе оборудования или канала связи.

4.2. Инициализация процесса вычислений

Перед началом процесса вычислений создается *виртуальная распределенная машина*, на которой будет выполняться приложение. На первом

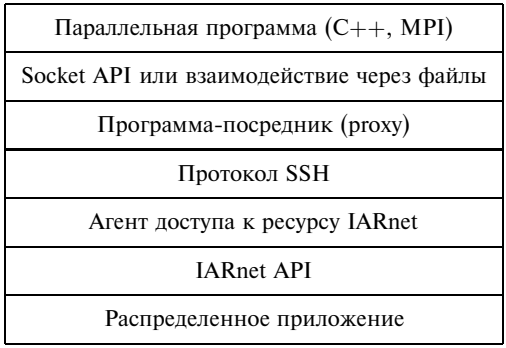

Рис. 3. Организация доступа к МВК

этапе запускаются агенты доступа для тех МВК, которые планируется использовать в процессе распределенных вычислений и агент-координатор. Далее производится запуск менеджера виртуальной машины, который считывает из файла и загружает на агент-координатор список идентификаторов агентов доступа к МВК, которые будут использованы агентом-координатором для взаимодействия с ними. Предусмотрен также графический интерфейс, который позволяет редактировать состав вычислительных ресурсов: выбирать из списка те вычислительные ресурсы, на которых планируется проводить расчеты.

4.3. Доступ к МВК

Организация распределенных вычислений в среде IARnet предполагает создание и координированное использование типизированных информационно-алгоритмических ресурсов (ИАР) [8]. На основе описанного ниже агента доступа к МВК для распределенной реализации метода ветвей и границ в среде IARnet создан специализированный агент доступа к параллельной библиотеке BNB-Solver, позволяющий использовать ее в качестве ресурса IARnet.

Доступ пользователей к многопроцессорным вычислительным комплексам осуществляется через управляющую машину, предназначенную для управления очередью пользовательских задач, запуска и завершения их на многопроцессорном вычислителе. Эти функции выполняет установленная на управляющей машине СУПЗ, например, PBS [13], LoadLeveler [14], Maui [12] или собственная система управления прохождением заданий, используемая на кластерах МСЦ РАН.

Взаимодействие пользователя с управляющей машиной, как правило, возможно только по протоколу SSH. Реализованный в соответствии с требованиями IARnet агент доступа к МВК предполагает лишь наличие у пользователя возможности для установления SSH-соединения и не требует установки на управляющей машине дополнительного ПО.

Текущая реализация агента доступа к библиотеке BNB-Solver основана на взаимодействии агента и задания, запущенного на кластере, со-

гласно своему специфическому протоколу. При этом для передачи команд протокола используется ssh-соединение с управляющей машиной кластера. На управляющей машине работает программа-посредник (proxy), осуществляющая пересылку данных между агентом доступа и заданием, работающим на кластере.

Основные функции, выполняемые описанной связкой агент-proxy, следующие:

- *Обмен данными между распределенным приложением и параллельной программой, работающей на МВК.*
  Обмен между proxy и параллельным заданием может осуществляться с помощью сокетов (socket) или файлов. Первый способ дает возможность управлять параллельной программой в процессе ее выполнения: получать текущие результаты, передавать новые данные для обработки и т. п. Это предполагает, что исходная параллельная программа будет модифицирована для выполнения этих функций.
  Взаимодействие через файлы имеет смысл использовать, например, если требуется использовать готовую параллельную программу без внесения в нее изменений.
- *Отслеживание статуса параллельного приложения.*
  Proxy отсылает сообщения агенту при установлении и разрыве соединения с программой, работающей на МВК.
- *Запуск задания.*
  Фактически, proxy осуществляет постановку задания в очередь СУПЗ.
- *Отмена ранее поставленного в очередь задания.*
- *Остановка задания.*
  Данная функция необходима при рассматриваемом подходе для того, чтобы задание на кластере не простаивало в отсутствие запросов, а могло быть принудительно остановлено клиентом или агентом.
- *Предоставление информации о текущем состоянии МВК —*
  общее число узлов, число свободных узлов, информация о состоянии очереди заданий.
- *Прогнозирование времени запуска задания.*
  Прогноз осуществляется на основе данных, предоставляемых СУПЗ и ресурсов, запрашиваемых заданием.
- *Бронирование ресурсов МВК.*
  Отличается от постановки в очередь тем, что запрашивается ресурс для использования не в ближайшее возможное, а в указанное клиентом время.
  Возможное использование такой функции: если необходима одновременная работа нескольких параллельных заданий, то можно рассчитать ближайшее время одновременного запуска и забронировать необходимые ресурсы.
  Также будет полезна функция отмены бронирования (например, если станет доступным ресурс более предпочтительный, чем ранее забронированный).

В настоящее время реализованы первые две из указанных функций. Возможность реализации таких функций, как прогнозирование времени запуска и бронирование ресурсов, в существенной мере зависит от средств, предоставляемых конкретной СУПЗ. Например, указанные возможности могут быть легко реализованы при использовании СУПЗ Maui [12]. Помимо реализации описанной функциональности, планируется создание адаптеров для различных СУПЗ, что обеспечит переносимость между различными МВК.

Еще одно направление для развития описанного механизма — создание API, предоставляющего авторам параллельных программ средства для их интеграции в распределенную вычислительную среду IARnet.

Существующие подходы к организации вычислений с использованием нескольких МВК можно условно разделить на три типа. Системы первого типа делают акцент на планирование выполнения заданий на объединяемых МВК (scheduling), организации глобальных централизованных или иерархических систем планирования заданий, опирающихся на локальные СУПЗ, не рассматривая при этом возможности взаимодействия различных заданий непосредственно между собой или с внешним управляющим процессом [11]. Ко второму типу можно отнести системы, основанные на организации «виртуальных кластеров», путем объединения нескольких МВК и организации взаимодействия процессов одной параллельной программы с помощью механизма передачи сообщений. Такой подход осуществляют системы на основе реализации стандарта MPI MPICH-G2 [10]. Системы третьего типа сочетают возможности планирования и организации взаимодействия параллельных процессов.

Описанный в данной работе подход к организации вычислений с использованием распределенных МВК позволяет использовать уже запущенное на кластере задание, передавая ему новые данные для счета в процессе выполнения. Таким образом, может быть организовано взаимодействие нескольких вычислительных подзадач (в том числе различных по содержанию) в рамках одной большой задачи. Данный механизм эффективен в случае, когда время передачи данных мало по сравнению со временем, необходимым для проведения вычислений.

4.4. Интерактивное управление процессом вычислений

После инициализации вычислительного процесса запускается интерактивный агент, который предоставляет пользователю графический интерфейс для управления процессом вычислений. Это интерфейс позволяет:

- выбрать тип решаемой задачи;
- установить параметры вычислительного процесса на каждом из узлов виртуальной параллельной машины;
- задать исходные данные задачи;
- запустить процесс решения задачи;
- осуществлять мониторинг загрузки узлов системы;

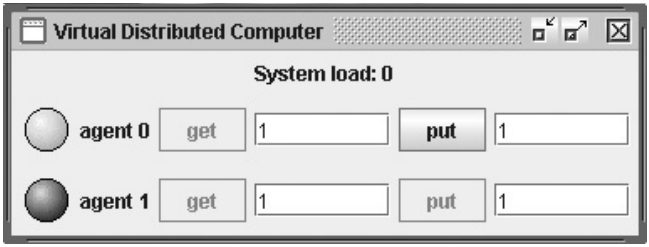

Рис. 4

- перераспределять задания между узлами системы;
- переключать алгоритмы автоматизации управления балансировкой нагрузки.

На данный момент реализована часть из перечисленных функций. На рис. 4 представлен фрагмент графического интерфейса, отвечающий за мониторинг и перераспределение задач-кандидатов между узлами распределенной системы. Индикатор готовности показывает состояние вычислительного узла: серый цвет индикатора означает, что узел не готов к выполнению работы, желтый означает отсутствие загрузки и готовность выполнять работу, зеленый означает, что узел выполняет работу. Кнопки “put” и “get” предоставляют возможность загружать задачи-кандидаты либо запрашивать их. Количество задач-кандидатов на процессе-координаторе отображается в строке “system load”.

4.5. Балансировка нагрузки

Библиотека BNB-Solver поддерживает два типа операций взаимодействия: для взаимодействия между процессами в пределах одного параллельного приложения и для взаимодействия с другими приложениями. Поддержка взаимодействия с другими приложениями реализована на основе протокола TCP/IP с помощью библиотеки sockets. Этот внешний интерфейс позволяет управлять работой параллельного приложения, а именно: загружать исходные данные задачи, запускать и останавливать процесс вычислений и взаимодействовать с приложением в процессе его работы. Внешний интерфейс позволяет получать информацию о текущем значении рекорда (наиболее близком к экстремуму найденном значении целевой функции), количестве необработанных задач-кандидатов на управляющем процессе, числе свободных процессов. Также, используя этот интерфейс, можно запросить некоторое количество необработанных задач-кандидатов, или, наоборот — загрузить задачи-кандидаты на управляющий процесс параллельного или последовательного приложения, работающего на данном узле.

Внешний интерфейс позволяет осуществлять балансировку нагрузки, поддерживая равномерную загрузку вычислительных ресурсов. Балансировка может осуществляться интерактивно оператором, наблюдающим за ходом процесса вычислений при помощи графического интерфейса,

либо автоматически с помощью *агента-планировщика*. Агент-планировщик управляет распределением нагрузки по процессам на основании информации, получаемой через внешний интерфейс. Управление заключается в пересылке данных с одного вычислительного ресурса на другой. Программная организация системы BNB-Grid позволяет подключать различные алгоритмы балансировки и переключаться между ними в процессе работы приложения.

5. Результаты экспериментов

Эксперименты проводились на примере задачи о ранце с одним ограничением, которая формулируется следующим образом:

$$f(x) = \sum_{i=1}^{n} c_i x_i \to \max, \quad \sum_{i=1}^{n} a_i x_i \leqslant R, \quad x_i \in \{0, 1\}, \quad i = 1, 2, \ldots, n.$$

В работе [9] предложен пример, асимптотическая сложность решения которого составляет $\frac{2^{n+3/2}}{\sqrt{\pi(n+1)}}$:

$$\sum_{i=1}^{n} 2x_i \to \max, \quad \sum_{i=1}^{n} 2x_i \leqslant 2\left[\frac{n}{2}\right] + 1, \quad n \geqslant 4, \quad x_i \in \{0, 1\}, \quad i = 1, 2, \ldots, n.$$

Для экспериментов была взята задача размерности $n = 30$. В качестве вычислительной платформы была выбрана распределенная система, в состав которой входили два МВК, расположенных на площадках МСЦ РАН и ВЦ РАН, и персональная рабочая станция, установленная в ИСА РАН (рис. 5). На рабочей станции был запущен интерактивный агент, а на суперкомпьютерах вычислительные модули, выполняющие счет. На каждом МВК при решении задачи были задействованы 8 процессоров. Процесс решения начинался на одном из МВК, после чего интерактивный агент

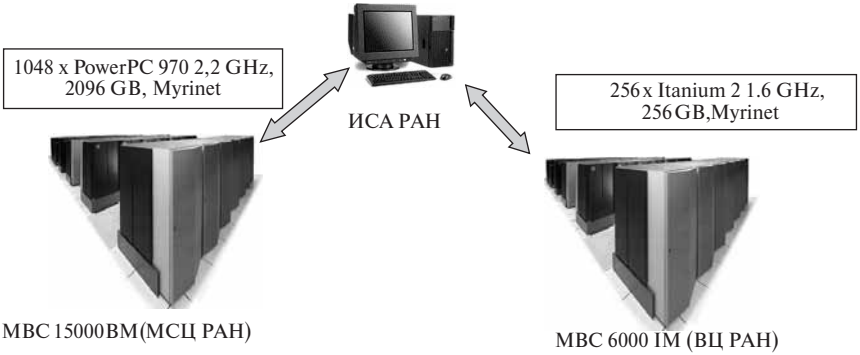

Рис. 5

Таблица 2

8 CPU MVS 15000 BM	5,57 мин
8 CPU MVS 6000 IM	6,03 мин
8 CPU MVS 15000 BM + 8 CPU MVS 6000 IM	3,15 мин

использовался для перераспределения нагрузки между МВК. Времена решения задачи на каждом из многопроцессорных комплексов и на всей системе в целом приведены в табл. 2.

Полученные результаты экспериментов показывают, что на выбранном примере удается приблизительно вдвое увеличить скорость решения задачи о ранце за счет удвоения вычислительной мощности. Этот факт позволяет сделать вывод об эффективности предлагаемого подхода.

6. Заключение

В работе приводится описание программного комплекса для решения задач оптимизации методом ветвей и границ в распределенной вычислительной среде. Рассмотрена его программная организация и приведены результаты вычислительного эксперимента.

Применяемое в данный момент интерактивное управление процессом вычислений позволяет изучать свойства рассматриваемой среды и добиваться высоких показателей эффективности. Вместе с тем, понятно, что такой подход имеет серьезные ограничения, так как пользователь не в состоянии реагировать на изменения в ходе процесса вычислений с требуемой скоростью и на протяжении достаточно длительного периода. Поэтому в дальнейшем планируется разработать автоматизированные планировщики нагрузки, которые будут управлять ходом процесса вычислений в автономном режиме.

Литература

1. *Сигал И. Х.*, *Иванова А. П.* Введение в прикладное дискретное программирование. М.: Физматлит, 2002.
2. *Сигал И. Х.* Параметризация и исследование некоторых задач дискретного программирования большой размерности // Известия РАН. Теория и системы управления. 2001. № 2. С. 83–92.
3. *Воеводин В. В.*, *Воеводин Вл. В.* Параллельные вычисления. СПб.: БХВ-Петербург, 2004.
4. Проблемы вычислений в распределенной среде / Под ред. С. В. Емельянова, А. П. Афанасьева. М.: ИСА РАН, 2003.
5. *Посыпкин М. А.*, *Сигал И. Х.*, *Галимьянова Н. Н.* Алгоритмы параллельных вычислений для решения некоторых классов задач дискретной оптимизации. М.: ВЦ РАН, 2005.

6. *Посыпкин М. А.*, *Сигал И. Х.* Исследование алгоритмов параллельных вычислений в задачах дискретной оптимизации ранцевого типа // ЖВМ и МФ. 2005. Т. 45. № 10. С. 1801–1809.
7. *Посыпкин М. А.* Архитектура и программная реализация библиотеки для решения задач оптимизации методом ветвей и границ. См. статью в этом сборнике.
8. *Емельянов С. В.*, *Афанасьев А. П.*, *Волошинов В. В.*, *Гринберг Я. Р.*, *Кривцов В. Е.*, *Сухорослов О. В.* Реализация Grid-вычислений в среде IARnet // Информационные технологии и вычислительные системы. 2005. № 2. С. 61–75.
9. *Финкельштейн Ю. Ю.* Приближенные методы и прикладные задачи дискретного программирования. М.: Наука, 1976.
10. MPICH-G2 (www3.niu.edu/mpi).
11. Moab Grid Suite (www.clusterresources.com/pages/products/moab-grid-suite.php).
12. Maui Cluster Scheduler (www.clusterresources.com/pages/products/maui-cluster-scheduler.php).
13. Portable Batch System (www.openpbs.org).
14. LoadLeveler (www.mhpcc.edu/training/workshop/loadleveler/MAIN.html).