Реализация крупноблочной схемы параллельного метода ветвей и границ для задач дискретной оптимизации средствами платформы Everest

С. А. Смирнов[a], В. В. Волошинов, О. В. Сухорослов

Институт проблем передачи информации им. А.А. Харкевича Российской академии наук,
127051, г. Москва, Большой Каретный переулок, д.19 стр. 1

E-mail: [a] sasmir@gmail.com

В работе исследуется вариант программной реализации т. н. крупнозернистой параллельной схемы алгоритма ветвей-и-границ. В ней исходная задача предварительно разбивается на набор подзадач, где часть целочисленных переменных фиксируется одним из своих возможных значений. Затем набор солверов, реализующих метод ветвей-и-границ (МВГ), начинает одновременно решать эти подзадачи. Если солвер находит рекордное («наилучшее» среди уже обнаруженных) значение целевой функции на допустимом решении, то оно рассылается другим солверам, что, вообще говоря, ускоряет работу МВГ. Общая организация вычислений (регистрация солверов, запуск и отправка им подзадач) обеспечивается средствами облачной платформы Everest. Для рассылки значений рекордов в среде приложений Everest была разработана специальная подсистема обмена сообщениями. Созданная распределенная система показала заметное ускорение в ходе решения тестовых задач линейного программирования с частично-булевыми переменными.

Ключевые слова: метод ветвей и границ, крупнозернистый параллелизм.

Работа выполнена при частичной финансовой поддержке Российского научного фонда (проект № 16-11-10352).

Введение

Для решения трудоемких задач дискретной оптимизации широкое распространение получили комбинаторные методы, среди которых наиболее часто встречаются методы ветвей и границ (МВГ). МВГ можно представить в виде процесса последовательного разбиения множества допустимых решений на подмножества с последующим отсечением не содержащих оптимальное решение подмножеств. Его реализация требует обхода некоторого дерева поиска. Каждому узлу дерева соответствует оценочная задача, полученная ослаблением части ограничений (обычно — условий целочисленности переменных) исходной задачи.

В работе речь идет об относительно редко применяемой, т.н. «крупноблочной» («крупнозернистой») схеме распараллеливания [Попов, 2007]. В ее основе — параллельное решение подзадач с обменом информацией о найденных (в ходе решения подзадач) значениях целевой функции на допустимых решениях (т.н. рекордов). При этом несколько подзадач могут обрабатываться одновременно, каждая в своем, отдельном, процессе. Если один из процессов находит допустимое решение, то соответствующее рекордное значение целевой функции может быть разослано остальным процессам, позволяя им, в принципе, существенно ускорить работу, за счет отбрасывания заведомо «неоптимальных» ветвей дерева обхода. Подобный подход обсуждается в литературе [Попов, 2007; Valente, Mitra, 2008; Bussieck, Ferris, Meeraus, 2009], но широкого применения пока не получил.

Роль исследователя при этом фактически сводится к поиску подходящего способа первоначальной декомпозиции исходной задачи (предпочтительно, в форме программы на языке оптимизационного моделирования) и настройке параметров алгоритма МВГ для набора пакетов дискретной оптимизации, подключенных к распределенной системе. Программная реализация обмена значениями рекордов может оказаться значительно проще, чем для «мелкозернистой» схемы МВГ. В настоящее время существует ряд пакетов оптимизации, применяющих МВГ для решения частично-целочисленных задач оптимизации: LPsolve, GLPK, CBC, SCIP, Cplex, KNITRO, Gurobi и т.д. Первые четыре из них имеют открытый исходный код и, таким образом, позволяют реализовать на их основе крупноблочную схему МВГ.

В данной работе представлена реализация крупноблочной схемы распараллеливания метода ветвей и границ для частично-целочисленных задач оптимизации. В прошлых работах [Смирнов, Волошинов, 2016; Смирнов, Волошинов, 2015] была описана система, где коммуникация между процессами-решателями осуществлялась системой на языке Erlang. В данной работе Erlang был заменен платформой Everest [Sukhoroslov, Volkov, Afanasiev, 2015]. Это позволяет опираться на существующую развитую систему организации распределенных вычислений. Данная реализация обеспечивает запуск подзадач на некотором предварительно заданном множестве хостов средствами платформы Everest и обмен рекордами между процессами, решающими подзадачи. В основе реализации лежит использование существующих пакетов оптимизации с открытым исходным кодом, а именно CBC и SCIP. Представленная система требует предварительного разбиения исходной задачи на подзадачи. Пользователь может самостоятельно, основываясь на собственных знаниях о задаче, реализовать алгоритм разбиения, например, средствами языка оптимизационного моделирования AMPL.

Указанная система была апробирована на классической задаче коммивояжера, сформулированной в виде задачи линейного программирования с частично-булевыми переменными (MILP).

Пакет оптимизации CBC

Пакет CBC (COIN-OR branch and cut) [Forrest, Lougee-Heimer, 2005] имеет открытый исходный код на языке C++, разрабатывается в рамках проекта COIN-OR (Computational

Infrastructure for Operations Research) и предназначен для численного решения частично-целочисленных задач линейного программирования методами отсечений и ветвей и границ. Использование CBC возможно как через автономное приложение, принимающее данные в форматах AMPL [Fourer, Gay, Kernighan, 2002], MPS и др., так и в виде встраиваемой библиотеки. CBC может использовать многопоточность для ускорения вычислений.

В CBC предусмотрены средства, позволяющие получить или установить текущее значение рекорда. У объектов CbcModel присутствует метод setBestSolution, позволяющий передать пакету оптимизации лучшее на данный момент допустимое решение или только значение целевой функции на этом решении. При этом CBC не будет проверять, является ли это решение на самом деле допустимым. Это свойство важно для обмена значениями рекордов, поступающих от решаемых одновременно подзадач, так как подзадачи часто имеют непересекающиеся множества допустимых решений. Контроль за процессом решения задачи возможен, если установить в CbcModel объект функций обратного вызова, реализующий интерфейс CbcCompare. С его помощью можно узнать об обнаружении нового рекорда, а также подбросить новое значение рекорда, не нарушив при этом работу CBC.

Пакет оптимизации SCIP

Пакет SCIP (Solving Constraint Integer Programs) [Gamrath, et al., 2016] — это один из самых быстрых некоммерческих солверов для задач частично-целочисленного линейного и нелинейного программирования [Mittelmann, [Electronic resource] 2016]. Солвер представляет собой гибкую и расширяемую систему на языке C. Использование SCIP возможно через консольное приложение scip, поддерживающее множество форматов описания задач. Также присутствует отдельное приложение scipampl, предназначенное для работы с задачами в формате языка моделирования AMPL [Fourer, Gay, Kernighan, 2002].

Для интеграции в распределенную среду было разработано приложение-адаптер. Адаптер решает две основные задачи: передавать солверу значения рекордов, пришедшие извне, и отслеживать нахождение солвером новых рекордов. Установить новое значение рекорда в солвере можно вызовом SCIPsetObjlimit. Вызов приходится делать внутри обратного вызова обработчика событий, так как солвер не предусматривает безопасных вызовов из нескольких различных потоков. Таким образом, важно установить обработчик на все события солвера. Это позволяет минимизировать задержку между получением нового значения рекорда потоком, получающим данные от распределенной системы, и передачей значения непосредственно солверу. О нахождении нового рекорда солвер сообщает событием SCIP_EVENTTYPE_BESTSOLFOUND. Далее необходимо получить само решение вызовом SCIPeventGetSol, а затем получить значение целевой функции вызовом SCIPgetSolOrigObj. Важно использовать именно эти вызовы, так как солвер может трансформировать исходную задачу и тогда значение целевой функции, с которой солвер работает в настоящий момент, может оказаться непригодным для передачи другим солверам.

Реализация крупноблочной схемы

Подготовив набор подзадач, пользователь запускает процесс решения с помощью приложения batch_solve.py (рисунок 1). Данное приложение формирует архив с начальными данными для сервиса (приложения) Coarse-grained B&B на сервере Everest. Этот сервис имеет заданный по умолчанию набор вычислительных ресурсов, на которых установлены солверы CBC и SCIP, а также адаптеры к ним. В результате вызова данного сервиса создается задание (Job), состоящее из множества задач (Task). Каждой подзадаче, сформированной пользователем, соответствует по одной задаче внутри задания. Сервер Everest обеспечивает распределение задач по вычислительным ресурсам (хост на рисунке 1). При запуске задачи, в числе исходных данных,

передается скрипт на языке Python, который и запускается агентом в качестве задания. Данный скрипт (task.py) обеспечивает запуск адаптера солвера (scip_port), а также взаимодействие с адаптером (через пайпы) и с агентом посредством сокета. Взаимодействие с агентом необходимо для обмена рекордами с другими задачами внутри задания.

Как только солвер находит новый рекорд (допустимое решение, где значение целевой функции меньше уже найденных), об этом узнает адаптер через обратный вызов. Далее, адаптер через пайп передает новый рекорд скрипту task.py. Этот скрипт сообщает серверу Everest, что нужно установить новое значение переменной рекорда, если передаваемое значение меньше текущего. Для этой цели в Everest был реализован специальный протокол. Если значение действительно оказалось меньше текущего, сервер Everest транслирует новое значение всем выполняющимся в данный момент задачам. Они, в свою очередь, передают новое значение адаптерам солвером, а те уже солверам.

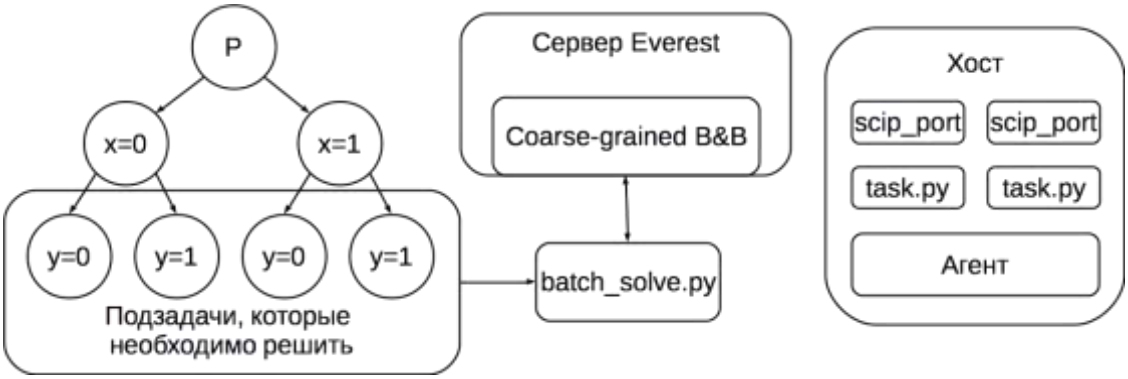

Рисунок 1. Архитектура системы

Вычислительный эксперимент

Вычислительные эксперименты проводились на трех вычислительных узлах: 4 потока на 2 x Intel Xeon E5620 @ 2.40GHz и 2 x 6 потоков на Intel Xeon E5-2620 @ 2.00GHz. Всего системе было доступно 16 потоков.

Тесты проводились на задаче о коммивояжере с числом городов N, равным 80, 90, 100 и 110. Расстояния между городами сгенерированы псевдо-случайным образом. В качестве ориентира было произведено по одному запуску каждой из исходных задач с SCIP 3.2.1 без какого-либо распараллеливания (время T(SCIP) в таблице 1).

Далее было проведено по одному «распределенному» запуску на указанных выше вычислительных ресурсах, результаты перечислены в столбце T(dSCIP) таблицы 1. Фиксировались переменные, соответствующие ребрам наименьшей длины в графе расстояний между городами. Для генерации подзадач использовались средства языка AMPL.

N	Число зафикс. xij	Число подзадач	T(SCIP), мин.	T(dSCIP), мин.
80	4	16	21	2.2
90	5	32	4.0	2.2
100	6	64	7.7	3.4
110	7	128	>2600	117

Таблица 1. Сравнение времени работы SCIP и крупноблочного алгоритма

Выводы

В статье описывается программная реализация параллельной крупнозернистой схемы метода ветвей и границ для решения задач дискретной оптимизации в среде веб-сервисов на основе платформы Everest, http://everest.distcomp.org. Для обмена рекордными значениями целевой функции между параллельно работающими солверами, реализующими алгоритм МВГ, используется подсистема обмена сообщениями в среде приложений Everest.

Для задачи коммивояжера различных размерностей представленная в работе схема распараллеливания метода ветвей и границ продемонстрировала заметное ускорение по сравнению со временем решения тех же задач однопоточным экземпляром МВГ-солвера. Для разбиения исходной задачи на подзадачи применялось эвристический метод, реализованный на языке оптимизационного моделирования AMPL.

Исходные коды разработанной системы доступны по адресу https://github.com/ssmir/dcbc

Список литературы

Попов Л.Д. Опыт многоуровневого распараллеливания метода ветвей и границ в задачах дискретной оптимизации // Автоматика и телемеханика. — 2007. — № 5. — С. 171–181.

Popov L.D. Experience of multilevel parallelizing of the branch and bound method in discrete optimization problems // Autom. Remote Control. — 2007. — Vol. 68, Issue 5. — P. 901–911. (Original Russian paper: Popov L.D. Opyt mnogourovnevogo rasparallelivaniya metoda vetvey i granits v zadachah diskretnoy optimizatsii // Avtomatika i telemehanika. — 2007. — No. 5. — P. 171–181).

Смирнов С.А., Волошинов В.В. Предварительная декомпозиция задач дискретной оптимизации для ускорения алгоритма ветвей и границ в распределенной вычислительной среде // Компьютерные исследования и моделирование. — 2015. — Т. 7, № 3. — С. 719–725.

Smirnov S.A., Voloshinov V.V. Predvaritelnaya dekompozitsiya zadach diskretnoy optimizatsii dla uskoreniya algoritma vetvey i granits v raspredelennoy vichislitelnoy srede // Komputerniye issledovaniya i modelirovaniye [Computer Research and Modeling]. — 2015. — Vol. 7, No 3. — P. 719–725 (in Russian).

Смирнов С.А., Волошинов В.В. Эффективное применение пакетов дискретной оптимизации в облачной инфраструктуре на основе эвристической декомпозиции исходной задачи в системе оптимизационного моделирования AMPL // Программные системы: теория и приложения. — 2016. — Т. 7, № 1. — С. 29–46.

Smirnov S.A., Voloshinov V.V. Effektivnoe primenenie paketov diskretnoy optimizatsii v oblachnoy infrastrukture na osnove evristicheskoy dekompozitsii ishodnoy zadachi v sisteme optimizatsionnogo modelirovaniya AMPL // Programmniye sistemy: teoriya i prilojeniya. — 2016. — Vol.7, No. 1. — P. 29–46 (in Russian).

Bussieck M.R., Ferris M.C., Meeraus A. Grid Enabled Optimization with GAMS // INFORMS Journal on Computing. — 2009. — Vol. 21, No. 3. — P. 349–362.

Forrest J., Lougee-Heimer R. CBC user guide // INFORMS Tutorials in Operations Research. — 2005. — P. 257–277.

Fourer R., Gay D.M., Kernighan B.W. AMPL: A Modeling Language for Mathematical Programming, second edition // Duxbury Press / Brooks/Cole Publishing Company. — 2002.

Gamrath et al. The SCIP Optimization Suite 3.2 // ZIB-Report. — 2016. — No. 15-60.

Mittelmann H. Mixed Integer Linear Programming Benchmark [Electronic resource]: http://plato.asu.edu/ftp/milpc.html.

Sukhoroslov O., Volkov S., Afanasiev A.A. Web-Based Platform for Publication and Distributed Execution of Computing Applications // 14th International Symposium on Parallel and Distributed Computing (ISPDC) // IEEE. — 2015. — P. 175–184.

Valente P., Mitra G. A grid-aware MIP solver: Implementation and case studies // Future Generation Computer Systems. — 2008. — Vol. 24, Issue 2. — P. 133–141.

Implementation of Coarse-Grained Parallel Scheme of Branch-and-Bound Algorithm for Discrete Optimization in Everest Platform

S. A. Smirnov[a], V. V. Voloshinov, O. V. Sukhoroslov

Institute for Information Transmission Problems of the Russian Academy of Sciences,
19, build.1, Bolshoy Karetny per., Moscow, 127051, Russia

E-mail: [a] sasmir@gmail.com

In this study we examine the coarse-grained approach to parallelization of the branch-and-bound algorithm. Our approach is that we split a mixed-integer programming problem into a predefined set of subproblems by fixing some of the integer variables. Then, these subproblems are being sent to a pool of open-source MIP-solvers running in parallel. When solver finds an incumbent it is broadcasted to other solvers which can use this objective value during solution process. Solver life cycle is managed by the Everest Web-based platform for distributed computing. The platform was also modified to al-low solvers exchange messages with incumbents and other data. The system was tested on several mixed-integer programming problems and a noticeable speedup was shown.

Keywords: branch and bound algorithm, coarse grained parallelism.

The work was partially supported by Russian Science Foundation (project #16-11-10352).