Исследование возможностей использования деревьев в ходе решения задачи о рюкзаке при помощи распределённых вычислений

М.А. Куприяшин

Национальный исследовательский ядерный университет МИФИ

Алгоритмы решения задачи о рюкзаке представляют интерес из-за большого числа связанных с ней прикладных задач. Рюкзаки являются важным криптографическим примитивом. Среди небольшого числа алгоритмов точного решения задачи о рюкзаке, заметное место занимают алгоритмы, основанные на методе обхода дерева вариантов укладки. В данной статье отмечается, что проблемы с разделением вычислительной нагрузки препятствуют созданию их эффективной параллельной реализации. Одним из путей решения этой проблемы является построение приемлемого на практике метода линеаризации, позволяющего представить множество потенциальных решений в виде нумерованной последовательности и разбить его на требуемое число равных частей.

1. Введение

Задача о рюкзаке представляет интерес для исследователей, так как с её решением связано большое количество прикладных задач в экономике, логистике и других областях (напр. [**Ошибка! Источник ссылки не найден.**]). В последние годы возрастает интерес к применению рюкзаков как криптографических примитивов при разработке новых моделей систем шифрования[**Ошибка! Источник ссылки не найден.**]. Стойкость таких систем шифрования к анализу определяется сложностью нахождения точного решения задачи о рюкзаке, в результате чего совершенствование точных алгоритмов приобретает актуальность. Для повышения эффективности этих алгоритмов целесообразно применение распределённых и параллельных вычислений.

Общая формулировка задачи о рюкзаке варьируется в зависимости от конкретной прикладной области. В рамках данной статьи предполагается, что задача о рюкзаке сформулирована следующим образом. Дан упорядоченный набор предметов, каждому из которых сопоставлена неотрицательная, целочисленная величина веса. Задача состоит в том, чтобы отыскать все возможные наборы из этих предметов, вес которых в точности равен заданному значению. Каждый из таких наборов далее называется «укладка рюкзака», суммарный вес предметов этого набора — вес укладки.

2. Основные алгоритмы точного решения задачи о рюкзаке

Как уже было отмечено, существует большое число алгоритмов решения задачи о рюкзаке, но лишь несколько из них предназначены для поиска точного решения[**Ошибка! Источник ссылки не найден.**]. Это связано с тем, что в значительном числе практических задач величины, фигурирующие в задаче о рюкзаке, обладают физическим смыслом. Например, при планировании загрузки транспортного корабля предметам в рюкзаке могут быть охарактеризованы реальным значением веса или объёма. Решение, приближенное по суммарному весу/объёму при этом может быть вполне приемлемым. Это делает актуальным изучение алгоритмов, которые способны быстро отыскать какое-либо решение, приближенное к оптимуму[**Ошибка! Источник ссылки не найден.**].

В рамках криптографического анализа описанный подход неэффективен. Веса предметов в данном случае представляют собой абстрактные значения, лишённые какого-либо смысла. Блоки исходного текста представляются как набор предметов, а веса этих наборов становятся шифрованным текстом. Приближённое решение даёт нам блок текста, шифрованное представление которого численно близко к весу искомого блока. Очевидной связи между укладкой, вос-

становленной таким образом, и реальной искомой укладкой, нет, вплоть до 100% несовпадения. Так как необходимо восстановить укладку, вес которой в точности соответствует заданному значению, целесообразно использовать для этого алгоритмы точного решения задачи о рюкзаке.

Такие алгоритмы, в общем случае, имеют экспоненциальную сложность[**Ошибка! Источник ссылки не найден.**,**Ошибка! Источник ссылки не найден.**], а значит, даже для небольшого числа предметов нахождение точного решения задачи о рюкзаке потребует значительных затрат времени. Это делает целесообразным применение параллельных и распределённых вычислений.

Существует несколько известных подходов к нахождению точного решения задачи о рюкзаке. Базовый подход подразумевает перебор всех возможных укладок и проверку получаемых значений веса для каждой из них. Высокая ($O(2^n)$) сложность этого алгоритма компенсируется возможностью создания распределённой реализации, обеспечивающей близкую к 1 эффективность каждого из вычислительных узлов. Тем не менее, суммарный объём вычислений очень велик: необходимо проверить 2^n укладок, для каждой из которых необходимо осуществить подсчёт суммарного веса и сравнение с заданным значением.

Переход от данного алгоритма к более эффективным возможен путём сокращения (отсечения) некоторого количества укладок, которые заведомо не могут содержать решений. Например, если определённый набор предметов превысил по весу целевое значение, то рассмотрение всех укладок, содержащих этот набор в качестве поднабора, заведомо бессмысленно. На этом принципе базируются ещё два алгоритма точного решения задачи о рюкзаке: алгоритм обхода дерева вариантов укладки[**Ошибка! Источник ссылки не найден.**] и алгоритм слияния списков Хоровица-Сани[**Ошибка! Источник ссылки не найден.**].

Алгоритм слияния списков основан на идее разделения исходного набора предметов на два поднабора. Для каждого из поднаборов рассчитываются веса всех возможных укладок (полным перебором), после чего списки весов сортируются, и элементы из этих списков попарно суммируются. Так как оба списка рассчитанных весов отсортированы, часть пар (сумма которых превышает заданное значение веса) удастся исключить из рассмотрения. Платой за это является высокая сложность по памяти: необходимо хранить в памяти все промежуточные значения весов. Это делает алгоритм непригодным для целей криптографического анализа. Более подробно алгоритм рассмотрен в [**Ошибка! Источник ссылки не найден.**].

Алгоритм обхода дерева вариантов укладки основан на представлении множества укладок в виде вершин связного графа. Путём обхода графа при помощи известных алгоритмов можно проверить все возможные укладки по аналогии с алгоритмом полного перебора. Выигрыш от использования данного алгоритма заключается в возможности отсечения ветвей дерева, заведомо не содержащих решений. В частности, в [**Ошибка! Источник ссылки не найден.**] предлагается соединять вершины рёбрами, если соответствующие укладки отличаются на один предмет с наибольшим номером из набора. Показано, что такой подход к построению графа позволяет всегда получать дерево с предсказуемой структурой. Очевидно, что если суммарный вес укладки в одном из узлов дерева превышает заданное значение, рассматривать поддерево с корнем в этой вершине бессмысленно. В отличие от алгоритма слияния списков, данный алгоритм обладает полиномиальной сложностью по памяти. Более подробное описание можно найти в [**Ошибка! Источник ссылки не найден.**,**Ошибка! Источник ссылки не найден.**].

Отдельно следует упомянуть известный алгоритм динамического программирования [**Ошибка! Источник ссылки не найден.**]. В отличие от всех описанных выше алгоритмов, при динамическом программировании решение заданного экземпляра задачи строится на основе решения задачи, имеющей размерность на единицу меньше, которое, в свою очередь, также строится на решении задачи меньшей размерности. Таким образом, за n шагов рекурсии можно свести задачу к тривиальной. В работе [**Ошибка! Источник ссылки не найден.**] рассматривается подход к оптимизации этого алгоритма для параллельных вычислений на общей памяти.

3. Проблема создания параллельной реализации алгоритма обхода дерева вариантов укладки

Как видно из изложенного выше, возможность ускорить решение задачи о рюкзаке за счёт сокращения множества рассматриваемых вершин требует введения некой структуры на множестве укладок. В рамках алгоритма прямого конструктивного перебора множество возможных укладок представляет собой пул равнозначных невзаимосвязанных укладок, который можно равномерно разделить между большим числом процессоров (с точностью до остатка от деления). Но в случае с анализом дерева вариантов укладок каждая укладка связана с одной или несколькими другими. Соответственно, разделять дерево произвольным образом крайне нецелесообразно. Для этого необходимо применение методов параллельного обхода деревьев.

В [**Ошибка! Источник ссылки не найден.**] рассматривается применение метода назначаемых поддеревьев и метода выделяемых поддеревьев при анализе методом обхода дерева вариантов укладки. Показано, что оба упомянутых подхода не позволяют добиться эффективной параллельной реализации алгоритма. Это связано с тем, что приведённые методы функционируют в оптимальном режиме, если размеры поддеревьев приблизительно одинаковы.

Рассмотрим подробнее структуру деревьев, с которыми работает алгоритм. Множество вершин отождествим с множеством всех возможных укладок. Распределим вершины по ярусам таким образом, чтобы номер яруса для укладки совпадал с количеством предметов в ней. Зададим рёбра таким образом, чтобы две укладки, отличающиеся на один предмет с наибольшим номером, были соседними. Например, вершина, соответствующая укладке 1, 2 и 4ого предмета находится на третьем ярусе, так как содержит три предмета. Смежные с ней укладки четвёртого яруса содержат дополнительно 5й, либо 6й, либо 7й и т.д. Ребро к укладке четвёртого яруса, содержащей 1, 2, 3 и 4й предметы, отсутствует. На втором ярусе с данной вершиной связана укладка, содержащая предметы 1 и 2. Укладки второго яруса, содержащие 1, 4 и 2, 4 не связаны с рассматриваемой вершиной. Более подробно данный подход к построению рёбер описан в [**Ошибка! Источник ссылки не найден.**]. На **Рис. Ошибка! Источник ссылки не найден.** приведён внешний вид графов, получаемых при решении задач размерностей 3, 4 и 5.

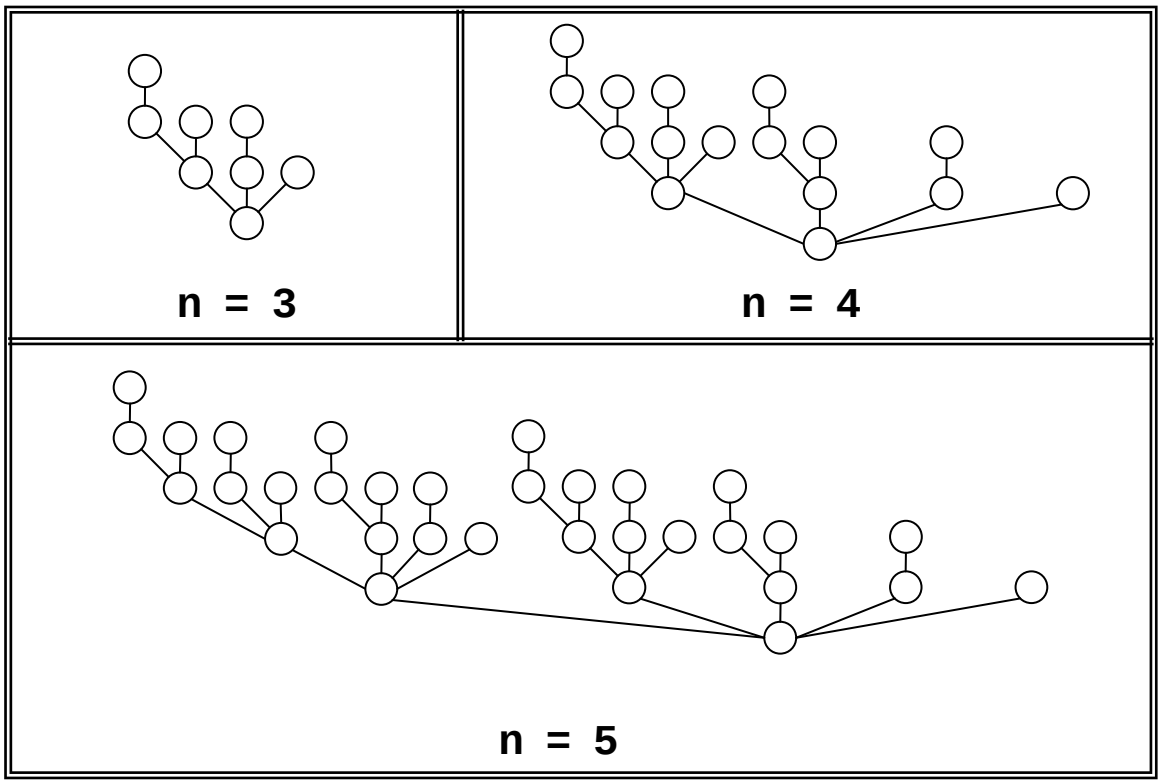

Рис. 1. Структура дерева вариантов укладки для задач размерности 3, 4 и 5

Видно, что по мере роста размерности задачи в графе появляются поддеревья с резко возрастающим числом вершин. Например, для размерности 5, самое большое поддерево с корнем в вершине первого яруса содержит половину всех вершин графа, тогда как самое маленькое — только одну. Более детальный анализ показывает, что размеры каждого следующего поддерева с корнем на том же ярусе в два раза превосходят размеры предыдущего. Так, для задачи размерности $n = 5$ на первом ярусе расположены поддеревья с размерами 1, 2, 4, 8 и 16.

Разумеется, можно выделять небольшие поддеревья, близкие по размеру. Но такие деревья включают вершины расположенные на верхних ярусах, а прирост эффективности алгоритма достигается за счёт отсечения верхних ярусов путём анализа нижних. Таким образом, создание параллельной реализации на основе алгоритма обхода дерева вариантов укладки представляется затруднительным.

4. Оптимизация параллельного алгоритма при помощи линеаризация множества вершин

Простота эффективного разделения вычислительной нагрузки при прямом переборе обусловлена однородностью и отсутствием специфической структуры множества укладок. Диспетчер распределённых вычислений может легко передать на рабочий процессор фрагмент желаемого размера. Более того, доступны алгоритмы, позволяющие установить биективное соответствие между номером укладки в пуле и её записью. Например, если множество всех возможных укладок перечислено в лексикографическом порядке, то вид укладки соответствует двоичному представлению её номера. Это означает, что диспетчерский процесс может интерпретировать множество укладок как однородное множество нумерованных элементов. Для выделения подзадачи рабочему процессору диспетчеру достаточно назвать номер первой вершины, а также количество анализируемых вершин. Алгоритмы, позволяющие перейти от порядкового номера элемента в порядке обхода к его представлению и обратно, известны как алгоритмы линеаризации (тж. «нумерация»[**Ошибка! Источник ссылки не найден.**]; “linear arrangement”[**Ошибка! Источник ссылки не найден.**]).

Для алгоритма обхода дерева вариантов укладки алгоритм линеаризации неочевиден. Известен лишь метод построения окрестности вершины по её представлению (он напрямую следует из алгоритма построения рёбер в используемом графе).

Однако, необходимо отметить, что структура дерева вариантов укладки зависит исключительно от размерности задачи, но не от параметров конкретного экземпляра задачи. Порядок обхода дерева для размерности $n = 3$ приведён на **Рис. Ошибка! Источник ссылки не найден.** (также см. **Рис. Ошибка! Источник ссылки не найден.**).

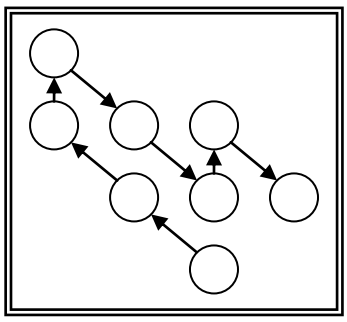

Рис. 2. Порядок обхода дерева вариантов укладки в глубину для задачи размерности 3

Необходимо отметить, что по мере обхода существует вероятность, что ряд вершин будет отсечен, но порядок обхода сохранится. Более того, можно предвидеть количество вершин, исключённых из последовательности обхода (количество вершин в любом поддереве данного графа легко подсчитывается). Это даёт основания предполагать, что можно построить и эффективно применить алгоритм линеаризации для задачи обхода данного дерева.

Использование алгоритма линеаризации позволит представить дерево вариантов укладки как последовательность вершин, причём, зная порядковый номер вершины, всегда можно по-

строить её представление (соответствующий набор предметов) и наоборот. Благодаря этому, диспетчерский процесс может разделить последовательность обхода на фрагменты желаемой длины и сообщить рабочим процессорам их номера. Более того, диспетчерский процесс и рабочие процессы смогут легко отслеживать предстоящий объём вычислений для каждой подзадачи. При необходимости можно произвести динамическую балансировку: для этого диспетчерскому процессору необходимо передать лишь номер начальной вершины для новой подзадачи и её длину.

В данный момент ведётся работа по разработке, реализации и тестированию алгоритма линеаризации. Как показано выше, данный алгоритм позволит создать более эффективную распределённую реализацию алгоритма обхода дерева вариантов укладки.

Крайне важно отметить, что применение данного метода негативно скажется на эффективности исходного алгоритма. При разделении нагрузки, процесс-диспетчер может выделить подзадачу, начало которой лежит на отсекаемом поддереве. Установив это, рабочий процесс быстро выйдет из данной области, но суммарное количество анализируемых алгоритмом вершин возрастёт, а как следствие — сократится эффективность (по сравнению с последовательной реализацией алгоритма). Так, если число вершин в графе совпадает с числом доступных вычислительных узлов, эффект от отсечения ветвей дерева сводится к нулю, независимо от конкретного экземпляра задачи. Алгоритм обхода дерева, таким образом, сводится по эффективности к алгоритму полного перебора. Однако, при решении практических задач можно ожидать, что количество доступных вычислительных узлов будет несравнимо меньше количества возможных укладок.

Заключение

Внедрение алгоритма линеаризации в распределённую реализацию алгоритма обхода дерева поиска позволит решить проблему разделения нагрузки между процессорами. Данное решение будет не идеальным, так как эффективность алгоритма будет сокращаться по мере роста числа используемых вычислительных узлов. Более того, отсечение поддеревьев по ходу анализа может приводить к непредсказуемому сокращению вычислительной нагрузки; как следствие, требуется обеспечивать динамическую балансировку нагрузки, что дополнительно усложнит алгоритм и снизит его эффективность.

Литература

1. Li K., Liu J., Wan L., Yin S., Li K. A Cost-Optimal Parallel Algorithm for the 0-1 Knapsack Problem and Its Performance on Multicore CPU and GPU Implementations (Abstract) // Parallel Computing – 2015.
2. Куприяшин М.А., Борзунов Г.И. Эволюция рюкзачных систем шифрования // Безопасность информационных технологий – 2015. – № 1. – С.101-103
3. Woeginger G.J. Exact algorithms for NP-hard problems: A survey // Springer, 2003. – P.185–207
4. Pisinger D. Algorithms for Knapsack Problems – 1995. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.9780&rep=rep1&type=pdf (дата обращения: 29.07.2015)
5. Horowitz E., Sahni S Computing partitions with applications to the knapsack problem // Journal of the ACM (JACM) – 1974. – Vol. 21 – No. 2 – P.277–292
6. Куприяшин М.А., Борзунов Г.И. Алгоритм решения задачи о рюкзаке, основанный на обходе дерева вариантов укладки // Безопасность информационных технологий – 2014. – № 2 – С.45–48
7. Куприяшин М.А., Борзунов Г.И. Анализ и сравнение алгоритмов нахождения точного решения задачи о рюкзаке // Тр. VI междунар. интернет-конференции молодых ученых, аспирантов и студентов «Инновационные технологии: теория, инструменты, практика» (InnoTech 2014). –Пермь: Пермский Национальный Исследовательский Политехнический

Университет, 2014. – 9с..
URL: http://www.conference.msa.pstu.ru/public/TK/Kuprijashin.pdf (дата обращения: 27.05.2015).

8. Тимошевская Н.Е. Распараллеливание обхода дерева поиска для решения задачи о рюкзаке на кластерной системе / под ред. Р.Г. Стронгин. Нижегородский государственный университет имени Н.И. Лобачевского: Издательство Нижегородского госуниверситета, 2002. – С.16–20
9. Тимошевская Н.Е. Разработка и исследование параллельных комбинаторных алгоритмов // Прикладная дискретная математика – 2009. – № 2. – С.96-103.

Research on using trees to solve the knapsack problem by means of parallel computation

Mikhail Kupriyashin

Keywords: parallel algorithms, knapsack problem, packing tree

Algorithms for the knapsack problem are of interest due to extensive amount of various applications. In particular, knapsacks are used as a cryptographic primitive. The algorithm based on searching the knapsack packing tree is notable among the few exact algorithms for the knapsack problem. This paper states that the problems with load balancing make parallel implementations of the algorithm inefficient. One of the ways to solve this problem is to devise a feasible linear arrangement that represents the set of potential solutions as a numbered sequence and provides for splitting it into subsequences of equal lengths.