G.I. Borzunov, M.A. Kupriyashin

Distribution of Computational Load between Processors in an Exact Solution of the Knapsack Problem with Exhaustive Search

Keywords: knapsack problem, distributed calculations, computational workload balance.
In this paper, we propose a new approach to the definition of basic problems in a constructive enumeration of combinations of knapsack vector elements. The proposed approach provides a more even distribution of computational load on multiprocessor systems when trying to find the solution of the knapsack problem that enhances the efficiency of distributed computing.

Г.И. Борзунов, М.А. Куприяшин

РАСПРЕДЕЛЕНИЕ ВЫЧИСЛИТЕЛЬНОЙ НАГРУЗКИ МЕЖДУ ПРОЦЕССОРАМИ ПРИ ТОЧНОМ РЕШЕНИИ ЗАДАЧИ О РЮКЗАКЕ МЕТОДОМ ПОЛНОГО ПЕРЕБОРА

Наличие устойчивых к ранее разработанным атакам рюкзачных систем защиты информации в сочетании с высоким быстродействием этих систем делает актуальными теоретические и экспериментальные исследования в этой области [1–3]. Так, в работе [3] предложена модель преобразования числовой последовательности с динамически изменяющимся рюкзачным вектором. Выбор определённых значений для параметров данной модели приводит к невозможности использования «непереборных» методов анализа, в связи с чем требуется развитие эффективных распределённых методов, предназначенных для анализа стойкости рюкзачных систем шифрования и основанных на конструктивном перечислении векторов укладок рюкзака. При этом основной целью применения распределённых вычислений является не просто уменьшение времени выполнения вычислений, но нахождение решения задач о рюкзаке в таких постановках, при которых получение решения на однопроцессорных вычислительных системах за приемлемое время невозможно. Эта цель требует разработки параллельных алгоритмов, обеспечивающих эффективное масштабирование, на основе которого реализуется равномерное распределение вычислительной нагрузки между процессорами [4]. В статье [5] показано, что разделение лексикографической последовательности векторов укладок на равные отрезки приводит к неравномерному распределению вычислительной нагрузки процессоров. В данной работе приводятся оценки указанной неравномерности, полученные на основе анализа алгоритма конструктивного перечисления векторов укладок рюкзака в лексикографическом порядке. Для обеспечения эффективной реализации параллельного алгоритма точного решения задачи о рюкзаке, основанного на использовании конструктивного перебора, предлагается метод распределения вычислительной нагрузки, учитывающий различия во времени проверки векторов укладки.

При проектировании параллельного алгоритма с использованием перебора векторов укладок в лексикографическом порядке в качестве базовой задачи (операция 1) принимается проверка точного равенства заданному значению суммарного веса элементов очередной укладки. Распределение базовых подзадач по процессорам выполняется главным процессором по следующему алгоритму, представленному в виде псевдокода с использованием элементов языка СИ (Алгоритм 1):

1. Определяется общее число задач (число проверяемых укладок): $dN = 2^n - 1$. Здесь n – размерность рюкзачного вектора.

2. Вычисляется среднее число задач, приходящееся на каждый процессор, по формуле: $ddN = \left\lfloor \frac{dN}{p} \right\rfloor$, где p – число доступных процессоров (вычислительных узлов).

3. Вычисляется остаток $ddNp$, и устанавливается начальное значение рабочей переменной $ddNpj$ для отслеживания размещения этого остатка: $ddNp = dN - ddN * p$; $ddNpj = 0$.

4. Устанавливаются для первого процессора (нумерация с нуля) номер первого характеристического вектора проверяемой укладки $dB[0]$ и верхняя граница для последнего вектора проверяемой укладки $dE[0]$:

$dB[0]=0$; $dE[0]=ddN$;

if($ddNpj < ddNp$){$ddNpj$+=1.0;$dE[0]$+=1.0;}.

5. Определяются параметры вычислительной нагрузки $dB[i]$, $dE[i]$, которые передаются процессорам-исполнителям для поиска решения задачи о рюкзаке вместе с заданной сумой рюкзачных элементов и рюкзачным вектором:

for (i =1; $i < p$; i ++){$dB[i] = dE[i$-1]; $dE[i] = dE[i$-1] + ddN;

if ($ddNpj < ddNp$){$ddNpj$ += 1.0; $dE[i]$+= 1.0;};

if (dN-$dE[i] < ddN$) $dE[i] = dN$;

Передать $dB[i]$, $dE[i]$ i-му процессору-исполнителю}

Далее каждым i-м процессором-исполнителем выполняются действия:

1. $ii = dB[i]$;

2. По номеру подмножества рюкзачных элементов (укладки) ii определяется характеристический вектор $V[]$ этого подмножества по алгоритму:

jj =; for(j = 0; $j < n$; j++){$V[j]$= jj %2; jj = jj/2;}, где «%» – операция получения остатка от деления.

3. Если $V[]$ – решение, то отправить его главному процессору, стоп.

4. Генерируется очередной характеристический вектор укладки $V[]$:

for(j = 0; $j < n$; j++){$if(V[j]$==0){$V[j]$=1;перейти к 5;}else{V[j]=0;}

Решение не найдено, стоп;}

5. ii++; $if(ii < dE[i])$ перейти к 3;

6. Решение не найдено, стоп.

Анализ приведённого выше алгоритма показывает, что по количеству базовых операций (проверок точного равенства суммарного веса элементов укладки заданному значению) вычислительная нагрузка процессоров различается не более чем на 1. Однако очевидно, что сами проверки точного равенства суммарного веса элементов заданному значению имеют разную временную сложность из-за разного числа скалярных операций, выполняемых при реализации этих проверок. Это число скалярных операций равняется количеству рюкзачных элементов в укладке и может меняться в пределах от 1 до n, где n – размерность рюкзачного вектора. Для оценки фактической неравномерности вычислительной нагрузки при её распределении между процессорами с использованием алгоритма 1 необходимо в качестве базовой задачи (базовая операция 2) принять скалярную операцию сложения веса рюкзачного элемента с текущей суммой весов элементов укладки. Тогда величина указанной неравномерности вычислительной нагрузки i-го процессора будет равна относительному отклонению числа базовых операций 2, отнесенных к этому процессору, от среднего значения количества таких операций, приходящихся на 1 процессор: $d(T)$. Вычислительный эксперимент подтвердил результаты анализа и показал, что такое относительное отклонение от среднего значения зависит от числа доступных процессоров и размерности рюкзачного вектора и достигает значительной величины в худшем из рассмотренных случаев. Так, при измене-

нии числа доступных процессоров от 10 до 50 и увеличении размерности рюкзачного вектора от 24 до 30 относительное отклонение от среднего значения вычислительной нагрузки $d(T)$ изменяется от 0,12 до 0,25.

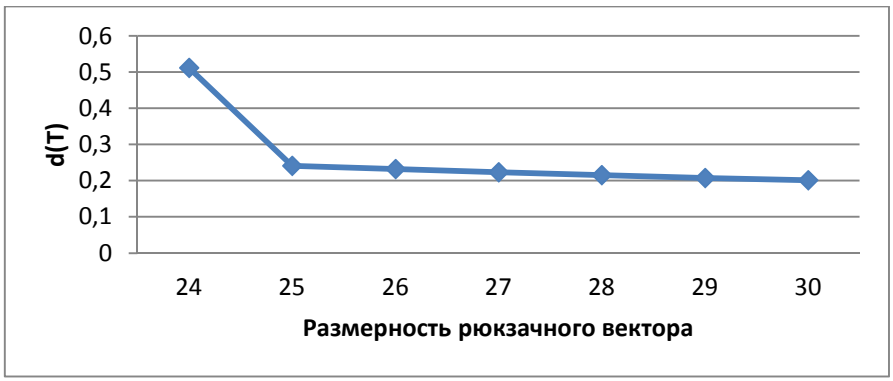

Рис. 1. Зависимость неравномерности вычислительной нагрузки $d(T)$ от размерности рюкзачного вектора при фиксированном числе доступных процессоров ($p = 50$)

График на рис. 1 иллюстрирует зависимость неравномерности вычислительной нагрузки (относительного отклонения числа базовых операций 2 от среднего значения количества таких операций, приходящихся на один процессор: $d(T)$) при увеличении размерности рюкзачного вектора и при фиксированном числе доступных процессоров. График, представленный на рис. 2, иллюстрирует сравнительно быстрое увеличение неравномерности вычислительной нагрузки при увеличении количества используемых процессоров и при фиксированной размерности рюкзачного вектора.

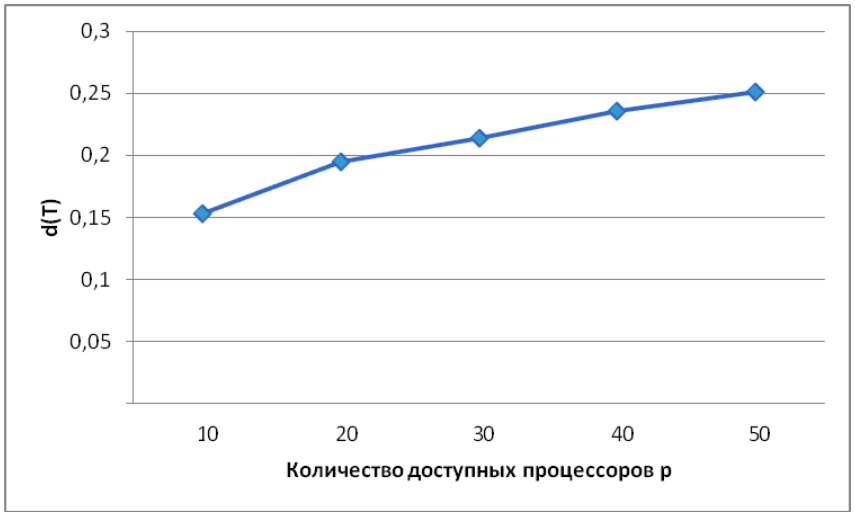

Рис. 2. Зависимость неравномерности вычислительной нагрузки от числа доступных процессоров при фиксированной размерности рюкзачного вектора ($n = 24$)

Для исключения неравномерной загрузки процессоров предлагается рассматривать в качестве базовой операции скалярную операцию сложения веса рюкзачного элемента с текущей суммой весов элементов укладки и проверки совпадения полученной суммы с заданным числом (базовая операция 2). В этом случае распределение базовых

подзадач по процессорам выполняется главным процессором по следующему алгоритму (Алгоритм 2).

1. Предварительно выполняется быстрое вычисление вектора $Cmn[i]$, каждая координата которого равна числу сочетаний из n по i элементов:
for(i=1;i<= n; i++){i1=i+1; for(j=1;j<=i1;j++)C_m_n[i][j]=C_m_n[i-1][j-1]+C_m_n[i-1] [j];}
for(i=1;i<= n; i++)Cmn[i]=C_m_n[n][i+1].

2. Вычисляется общее число базовых подзадач $SdNStart$:$SdNStart = 0$;

for(j=1; j<=m; j++){$S\_C\_m\_n$+=Cmn[j]; $SdNStart$+= (j*Cmn[j]);}.

3. Определяется среднее число базовых операций ddN, приходящихся на каждый процессор-исполнитель, и число дополнительных операций $ddNp$, подлежащих распределению меду процессорами-исполнителями:

$ddN = floor$l($SdNStart/p$); $ddNp = SdNStart-ddN*p$.

4. Устанавливается верхняя граница числа базовых операций, выполняемых процессором-исполнителем:

$(ddNddNp) : ddNddNp = ddN + ddNp$.

5. Для каждого i-го процессора-исполнителя определяется вычислительная нагрузка в виде первого проверяемого характеристического вектора укладки (сочетания) и последнего первого проверяемого характеристического вектора укладки (сочетания). Первый характеристический вектор укладки задаётся индексом класса сочетания (число единиц в характеристическом векторе) $dB1[i]$ и номером сочетания в классе сочетаний $dB2[i]$, последний характеристический вектор укладки аналогично задаётся индексом класса сочетания $dE1[i]$ и номером сочетания в классе сочетаний $dE2[i]$. Это действие выполняется в два шага. На первом шаге выполняется быстрое увеличение вычислительной нагрузки процессора-исполнителя:

while($SdNp$[i]<$ddNddNp$){$SdNp$[i] += j*Cmn[j];j++;}.

При этом определяются $dB1[i]$, $dB2[i]$, $dE1[i]$, $dE2[i]$.

На втором шаге выполняется балансировка путём увеличения $dE2[i]$:

while($SdNp$[i]+ j <=$ddNddNp$){$SdNp$[i] += j; $dE2$[i] += j ;}$ddNp$= $ddNddNp$ - $SdNp$[i].

Далее каждым i-м процессором-исполнителем выполняются действия:

1. Устанавливаются индекс класса сочетания равным $dB1[i]$ и номер в классе равным $dB2[i]$; выполняется $ii = dB1[i]$.

2. По индексу класса сочетания и по его номеру в классе генерируется характеристический вектор сочетания (укладки) $V[]$ с помощью известного алгоритма [6].

3. Если $V[]$ – решение, то отправить его главному процессору, стоп.

4. Если не все сочетания перечислены в данном классе, то генерируется очередной характеристический вектор сочетания (укладки) $V[]$ по алгоритму перечисления сочетаний в лексикографическом порядке и выполняется переход к 3.

5. ii++; if(ii<= $dE1[i]$) {Устанавливаются индекс класса сочетания равным ii и номер в классе; выполняется переход к 3}.

6. Решение не найдено, стоп.

В результате неравномерность (отклонение от средней вычислительной нагрузки) не превосходит остатка от деления n на p (здесь, как и выше, n – размерность рюкзачного вектора, а p – число доступных процессоров), в то время как при использовании алгоритма 1 она может равняться произведению $(n-1)*(dN - ddN*p)$ в худшем случае. При использовании распределённых вычислений алгоритм 2 обеспечивает ускорение точного решения задачи о рюкзаке по сравнению с решением этой задачи с использованием алгоритма 1 (табл. 1).

Таблица 1. Зависимость коэффициента ускорения S от количества доступных процессоров p и размерности рюкзачного вектора n: $S(p,n)$

p	N					
-	24	25	26	27	28	29
10	1,152777	1,146666	1,141026	1,135802	1,130952	1,126437
20	1,194442	1,186666	1,179487	1,172839	1,166666	1,160919
30	1,213886	1,205332	1,197435	1,190123	1,183333	1,177011
40	1,236106	1,226664	1,217948	1,209876	1,202381	1,195402
50	1,251177	1,24113	1,231858	1,223272	1,215298	1,207874

Конкретные значения коэффициента ускорения в вычислительном эксперименте определялись как отношение числа базовых операций 2 ($T1$), приходящихся на процессор с максимальной вычислительной нагрузкой в результате применения алгоритма 1, к числу базовых операций 2 ($T2$), приходящихся на тот же процессор в результате применения алгоритма 2. На рис. 3 приводится график зависимости этого коэффициента ускорения от размерности рюкзака при постоянном количестве доступных процессоров. Характер этой зависимости в основном совпадает с зависимостью, представленной на рис. 1.

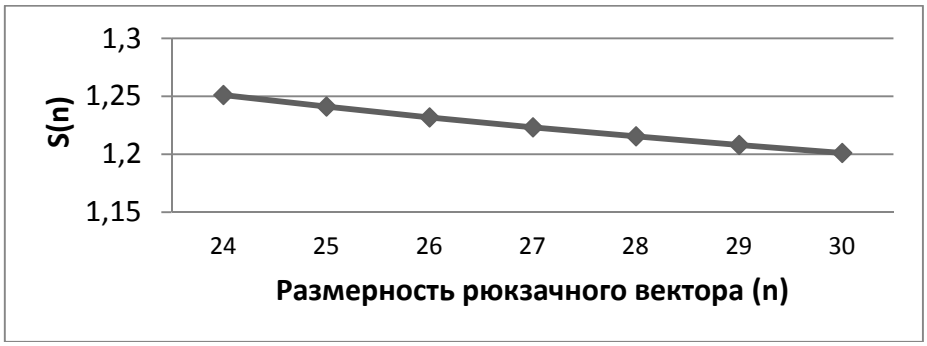

Рис. 3. Зависимость коэффициента ускорения S от размерности рюкзачного вектора n при постоянном количестве доступных процессоров ($p = 50$)

Рис. 4. Зависимость коэффициента ускорения S от количества доступных процессоров p при постоянной размерности рюкзачного вектора ($n = 50$)

График на рис. 4 иллюстрирует увеличение коэффициента ускорения с увеличением числа доступных процессоров при постоянной размерности рюкзачного вектора.

Характер зависимости коэффициента ускорения от числа доступных процессоров при постоянной размерности рюкзачного вектора также в основном совпадает с аналогичной зависимостью неравномерности вычислительной нагрузки, представленной на рис. 2. При анализе стойкости рюкзачных систем защиты информации чаще всего размерность рюкзачного вектора является заданной величиной. Поэтому приоритетное значение имеет зависимость, представленная на рис. 4. Таким образом, для эффективной реализации параллельного алгоритма точного решения задачи о рюкзаке, следует использовать алгоритм 2, основанный на предложенном в данной работе методе распределения вычислительной нагрузки.

СПИСОК ЛИТЕРАТУРЫ:

1. Куприяшин М.А., Борзунов Г.И. Эволюция рюкзачных систем шифрования// Безопасность информационных технологий. 2015. № 1. С. 102–103.
2. Kate A. Generalizing cryptosystems based on the subset sum problem / A. Kate, I. Goldberg // International Journal of Information Security. 2011. Т. 10. № 3. P. 189–199.
3. Подколзин В.В. Моделирование систем на основе односторонних рюкзачных отображений: автореф. дис. канд. техн. наук. Краснодар: Кубанский государственный университет. 2011.
4. Борзунов Г.И., Петрова Т.В., Сучкова Е.А. Повышение эффективности масштабирования набора задач при поиске экстремальных разбиений (статья) // Безопасность информационных технологий. 2011, № 3. С. 116–120.
5. Куприяшин М.А., Борзунов Г.И. Визуализация и анализ алгоритма точного решения задачи о рюкзаке, основанного на использовании конструктивного перебора // Научная визуализация. 2015. № 4.
6. Тимашевская Н.Е. О нумерации перестановок и сочетаний для организации параллельных вычислений в задачах проектирования управляющих систем. / Известия Томского политехнического университета. 2004. Т. 307. № 6. С.18–20.

REFERENCES:

1. Kupriyashin M.A., Borzunov G.I. Evolucia ruykzachnyh sistem shifrovania (in Russian) // Bezopasnost Informatsionnykh Tekhnology. 2015. № 1. P. 102–103.
2. Kate A. Generalizing cryptosystems based on the subset sum problem / A. Kate, I. Goldberg // International Journal of Information Security. 2011. T. 10, № 3. P. 189–199.
3. Podkolzin V.V. Modelirovanie sistem na osnove odnostoronnikh ruykzachnyh otobrazheniy (Ph.D. thesis summary; in Russian) . Krasnodar: Kuban State Unversity, 2011.
4. Borzunov G.I., Petrova T.V., Suchkova E.A. Povyshenie effectivnosti mashtabirovania nabora zadach pri poiske extremalnyh razbienii (paper; in Russian) // Bezopasnost Informatsionnykh Tekhnology. 2011, № 3. P. 116–120.
5. Kupriyashin M.A., Borzunov G.I. Visualization and Analysis of the Exact Algorithm for Knapsack Problem based on Exhaustive Search // Scientific Visualization. 2015. №4.
6. Timashevskaya N.E. O numeracii perestanovok i sochetaniy dlya organizacii parallelnyh vychisleniy v zadachah proektirovania upravlyauschih sistem (in Russian) / Izvestya Tomskogo politechnicheskogo universiteta. 2004. Vol. 307. №6. P. 18–20.