УДК 656.021.052:004

Сравнительный анализ эффективности различных вариантов метода динамического программирования для решения оптимизационных задач на этапе размещения элементов микросхем

М.А. Посыпкин[1], Си Ту Тант Син[2]

[1]Институт проблем передачи информации им. А.А. Харкевича Российской академии наук, Москва

[2]Национальный исследовательский университет “МИЭТ”, Москва,

sithuthantsin86@gmail.com

***Аннотация* — Одним из этапов проектирования интегральной схемы является этап размещения ее элементов на кристалле. Формально проблема размещения может быть рассмотрена как задача дискретной оптимизации ранцевого типа. Одним из базовых методов решения подобных задач является метод динамического программирования. Работа посвящена сравнительному исследованию различных последовательных вариантов метода динамического программирования для задачи о ранце. Приведена постановка задачи, описаны основные варианты метода динамического программирования, проведено экспериментальное сравнение на случайно-сгенерированных исходных данных, сделаны выводы об эффективности различных вариантов динамического программирования.**

***Ключевые слова* — задача о ранце, динамическое программирование, дискретная оптимизация.**

I. Введение

Оптимизация является неотъемлемой частью разработки электронных схем [1] и широко применяется при разработке различных электронных систем [2]. В частности, на этапе размещения элементов интегральной схемы на плате решается оптимизационная задача, которая может быть сведена к задачам ранцевого типа. Современные микросхемы могут содержать миллион и более базовых элементов, что приводит к оптимизационным постановкам сверхбольшой размерности. В результате возникает необходимость применять методы параллельных вычислений.

В данной работе изучается один из наиболее известных методов решения задач оптимизации – метод динамического программирования. В качестве модельной задачи взята задача о ранце [3], которая является одной из наиболее известных задач комбинаторной оптимизации. Для задачи о ранце разработаны различные алгоритмы [4-6], сложностные свойства которых достаточно хорошо изучены [7].

Одним из наиболее популярных подходов к нахождению точного решения задачи о ранце является метод динамического программирования, основанный на принципе оптимальности Беллмана [4]. В дальнейшем были предложены различные варианты этого метода [5,6]. В данной работе проводится численное сравнение эффективности этих вариантов на различных наборах исходных данных. Табличный вариант метода динамического программирования исследуется в последовательном и параллельном вариантах, для остальных методов исследовался только последовательный вариант.

Задача о ранце формально записывается следующим образом:

$$\begin{cases} f(x) = \sum_{j=1}^{n} p_j x_j \to \max, \\ \sum_{j=1}^{n} w_j x_j \le C, \\ x_j \in \{0,1\}, 1 \le j \le n. \end{cases}$$

В данной постановке p_j и w_j - это стоимость и вес предмета с номером j, помещаемого в ранец грузоподъемностью C. Необходимо определить набор предметов максимальной суммарной стоимости, который можно разместить в ранце. Естественно предположить, что $p_j > 0$, $0 < w_j < C$, $j = 1,2,3,\ldots,n$. Множество допустимых решений этой задачи — это множество n-мерных булевых векторов

$x = (x_1, x_2, x_3, \ldots, x_n)$, удовлетворяющих условию $\sum_{j=1}^{n} w_j x_j \le C$.

Решением задачи является вектор из нулей и единиц, задающий значения переменных.

II. Табличный вариант метода динамического программирования решения задачи о ранце

Методы динамического программирования основаны на принципе оптимальности Беллмана.

$$f_j(i) = \max(p_j + f_{j-1}(i - w_j), f_{j-1}(i))$$

$$j = 1,2,3,\ldots,n; i = 1,2,3,\ldots,C;$$

Здесь через $f_j(i)$ обозначено значение максимума в задаче

$$\begin{cases} f(x) = \sum_{k=1}^{j} p_k x_k \to \max, \\ \sum_{j=k}^{j} w_k x_k \le i, \\ x_j \in \{0,1\}, 1 \le k \le j, \end{cases}$$

где i – текущая грузоподъемность.

Табличный алгоритм динамического программирования реализует принцип оптимальности с помощью двумерной матрицы размерности $n \times C$. Необходимым условием применимости этого алгоритма является целочисленность коэффициентов задачи. В таблице вес меняется от нуля до максимального значения (т.е. грузоподъемности) вдоль строк, а вдоль столбцов меняется количество предметов. Прямой ход алгоритма динамического программирования решения задачи о ранце использует уравнение Беллмана для вычисления очередного значения в ячейке таблицы. Строки заполняются сверху вниз, каждая строка – слева направо, используя правило (1).

Рассмотрим пример задачи о ранце

$$3x_1 + 2x_2 + 2x_3 + 7x_4 \to \max,$$
$$2x_1 + 3x_2 + 2x_3 + 3x_4 \le 6.$$

Результаты работы метода динамического программирования даны в табл. 1. Максимальное значение целевой функции оказывается в правом нижнем углу. Далее выполняется обратный ход алгоритма, на котором происходит вычисление значений всех переменных в решении.

Табличный вариант метода динамического программирования был реализован в последовательном и параллельном вариантах на языке C++.

Таблица 1

Таблица метода динамического программирования

j – предметы								
	1	0	0	3	3	3	3	3
	2	0	0	3	3	3	5	5
	3	0	0	3	3	5	5	5
	4	0	0	3	7	7	10	10 (Результат)
		0	**1**	**2**	**3**	**4**	**5**	**6**
		i – грузоподъемность						

III. Вариант метода динамического программирования со списком

Алгоритм динамического программирования со списком [3,5] использует список пар, первый элемент которых содержит суммарную стоимость положенных в ранец предметов, а второй – оставшийся лимит по весу, т.е. разность грузоподъемности ранца и суммарного веса, положенных в него предметов. Пара (a,b) соответствует подзадаче

$$\begin{cases} f(x) = a + \sum_{k=j+1}^{n} p_k x_k \to \max, \\ \sum_{k=j+1}^{n} w_k x_k \le b, \\ x_j \in \{0,1\}, 1 \le k \le j. \end{cases}$$

Концепция доминирования позволяет сокращать перебор в методе динамического программирования за счет удаления неперспективных пар. Пара (a,b) доминирует пару (c,d) , если $a \ge c$ и $b \ge d$. Очевидно, что при условии, что мы можем докладывать в ранец одинаковые предметы, пара (c,d) никогда не приведет к лучшим результатам по сравнению с парой (a,b) . Поэтому пару (c,d) можно удалить из списка.

При $j = 0$ имеем список $L_0 = \{(0,C)\}$, состоящий из одной пары с суммарной стоимостью положенных предметов, равной нулю, и остаточной грузоподъемностью, равной грузоподъемности ранца. Последующий L_j список получается из списка L_{j-1} в два этапа. На первом этапе («добавление») вес w_j вычитается, а ценность p_j добавляется к весу и ценности всех пар, содержащихся в списке L_{j-1} . Получаем новый список L'_{j-1} . Это покомпонентное дополнение будем обозначать через

$$L'_{j-1} = L_{j-1} \oplus (p_j, -w_j) .$$

Все пары вида (a,b) , где $b < 0$, удаляются из списка, т.к. они не удовлетворяют ограничению по весу.

На втором этапе («слияние») два списка L_{j-1} и L'_{j-1} объединяются для получения L_j . При этом происходит исключение доминируемых пар.

Алгоритм динамического программирования со списком, таким образом, имеет следующий вид:

$L_0 := (0, C)$

Для $j = 1$ до n

$L'_{j-1} := L_{j-1} \oplus (p_j, -w_j)$ // «добавление»

Удалить все пары $(\overline{w}, \overline{p}) \in L'_{j-1}$ с условием $\overline{w} > C$.

$L_j := \text{MergeLists}(L'_{j-1}, L_{j-1})$ // «слияние»

Рассмотрим пример

$$3x_1 + 2x_2 + 2x_3 + 7x_4 \to \max,$$
$$2x_1 + 3x_2 + 2x_3 + 3x_4 \le 6.$$

Таблица 2

Таблица метода динамического программирования со списком

Условие начало					
Шаг-1	**Список–1**	0,6			
Добавление (3,–2)					
Шаг-2	**Список–1**	0,6			
	Список–2	3,4			
	Список–3	0,6	3,4		
Добавление (2,–3)					
Шаг-3	**Список–1**	0,6	3,4		
	Список–2	2,3	5,1		
	Список–3	0,6	3,4	5,1	
Добавление (2,–2)					
Шаг-4	**Список–1**	0,6	3,4	5,1	
	Список-2	2,4	5,2		
	Список-3	0,6	3,4	5,2	
Добавление (7,–3)					
Шаг-5	**Список-1**	0,6	3,4	5,2	
	Список-2	7,3	10,1		
	Список-3	0,6	3,4	7,3	10,1

Использование доминирования позволяет существенно сократить перебор по сравнению с табличным вариантом метода.

IV. Ускорение метода динамического программирования с помощью нижних и верхних границ

В качестве нижней оценки используется наибольшее найденное на данном шаге алгоритма значение целевой функции, также называемое рекордом. Верхняя оценка может вычисляться различными способами, рассматриваемыми далее. Основа подхода состоит в исключении из рассмотрения подзадач (пар), верхняя граница которых не превосходит значения рекорда, на том основании, что их решение не улучшит рекорд

Предположим, что элементы сортируются в соответствии с понижением эффективности:

$$\frac{p_1}{w_1} \ge \frac{p_2}{w_2} \ge \ldots . \ge \frac{p_n}{w_n} .$$

Первая верхняя оценка для пары (p, r) получается следующим образом:

$$UB_1 = p + \left(\frac{p_{j+1}}{w_{j+1}} \times r \right),$$

где j – номер первой свободной (которой не присвоено значение) переменной.

Более точную верхнюю оценку можно получить с помощью решения задачи линейной релаксации, получаемой путем замены условия целочисленности интервальными ограничениями. В случае задачи о ранце это означает, что ограничение $x_j \in \{0,1\}$ заменяется на $0 \le x_j \le 1$ для $j = 1,2,3,..n$. Таким образом, получаем следующее условие:

$$f(x) = \sum_{j=1}^{n} p_j x_j \to \max,$$
$$\sum_{j=1}^{n} w_j x_j \le C,$$
$$0 \le x_j \le 1, 1 \le j \le n.$$

Для задачи максимизации значение оптимального решения проблемы линейной релаксации не меньше чем значение оптимума в исходной задаче. Задача линейной релаксации с одним ограничением решается за линейное от n время методом Данцига [2]. Определяется номер дробной переменной s из неравенств

$$\sum_{j=1}^{s-1} w_j \le C \text{ and } \sum_{j-1}^{s} w_j > C .$$

Соответствующее значение верхней оценки вычисляется по следующей формуле

$$UB_2 = \sum_{j=1}^{s-1} p_j + \left(C - \sum_{j=1}^{s-1} w_j \right) \frac{p_s}{w_s} .$$

Вычисление нижних оценок производится с помощью обновления рекорда на каждом шаге алгоритма. Верхние оценки вычисляются для каждого кортежа в списке одним из описанных выше способов.

V. Экспериментальное сравнение эффективности различных алгоритмов

Эксперименты проводили на компьютере с двумя процессорами IntelCore 2 Quad E5335 2,00 ГГц и 8 Гб памяти.

Таблица 3

Время выполнения

	Номер алгоритма			
№ задачи	**1**	**2**	**3**	**4**
1	11,75	0,17	0,02	0,17
2	12,42	0,11	0,02	0,18
3	32,43	0,33	0,09	1,02
4	32,49	0,33	0,1	1,06
5	69,35	0,56	0,29	4,59
6	66,44	0,48	0,25	3,62
7	104,64	0,57	0,46	7,81
8	103,09	0,54	0,55	7,71
9	124,39	0,58	0,53	5,37
10	129,31	0,6	0,59	5,56
Средние значения	**68,631 сек**	**0,427 сек**	**0,29 сек**	**3,7 сек**

Таблица 4

Число шагов метода

	Номер алгоритма			
№ задачи	**1**	**2**	**3**	**4**
1	124183500	4020370	443984	389158
2	130084500	2631118	454002	403906
3	314434500	7469538	1724668	1527713
4	316546500	7458493	1819872	1575112
5	649049500	12818457	5367178	4172996
6	616601500	10840034	4592442	3660256
7	948435000	12763222	8245818	5292275
8	936984000	12327515	7915072	5225552
9	1118873000	13160550	9418132	2975564
10	1157832500	13528633	10394322	3172020
Средние значения	**631302450**	**9701793**	**5037549**	**2839455**

Сравнивались следующие алгоритмы:

1. Табличный метод.
2. Алгоритм со списком.
3. Алгоритм со списком с оценкой UB_1.
4. Алгоритм со списком с оценкой UB_2.

Для сравнения использовались задачи о ранце с 500 переменными, веса и цены которых были сгенерированы случайным образом в диапазонах [1,10000], соответственно. При этом вместимость ранца вычислялась следующим образом:

$$C = \alpha \times \sum_{j=1}^{n} w_j ,$$

где параметр α брался равным $0.1,0.25,0.5,0.75,0.9$. Всего было сгенерировано 10 примеров задачи о ранце по два для каждого значения α.

Времена работы алгоритмов приведены в табл. 3, а сравнительное число шагов – в табл. 4.

VI. Заключение

Результаты проведенных экспериментов показывают, что наилучший эффект дает сочетание учета доминирования и верхней оценки UB_1. Применение оценки UB_2 хотя и сокращает перебор, но приводит к более длительному времени работы из-за больших временных потерь на решение задачи линейной релаксации.

Поддержка

Работа выполнена при поддержке проекта РФФИ 13-07-00291 А.

Литература

[1] Гаврилов С.В., Глебов А.Л., Стемпковский А.Л. Структурная оптимизация цифровых КМОП схем // Информационные технологии и вычислительные системы. 2002. №4. С. 40-47.

[2] Лупин С.А. и др. К вопросу оценки точности алгоритмов дискретной оптимизации // V Всероссийская научно-техническая конференция «Проблемы разработки перспективных микро- и наноэлектронных систем-2012». М.: ИППМ РАН. 2012. С. 553-556.

[3] Kellerer H., Pfershy U., Pisinger D. Knapsack Problems. Springer Verlag. 2004. 546 p.

[4] Беллман Р. Динамическое программирование. М.: Изд-во иностранной литературы, 1960.

[5] Лазарев А.А. Графический подход к решению задач комбинаторной оптимизации // Автоматика и телемеханика. 2007. № 4. С. 13-23.

[6] Сигал И.Х., Иванова А.П. Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы. М.: Физматлит, 2002.

[7] Колпаков Р.М., Посыпкин М.А. Асимптотическая оценка сложности метода ветвей и границ с ветвлением по дробной переменной для задачи о ранце // Дискретн. анализ и исслед. опер. 2008. № 15. С. 58-81.