УДК 681.3

Оценка степени параллелизма алгоритма с использованием матрицы отношений

Ватутин Э.И., студент

Курский государственный технический университет

В работе рассматривается алгоритм оценки степени параллелизма алгоритма. Дается его формализованное описание. Оцениваются временная и емкостная сложность. Проводится сравнение с другими известными алгоритмами.

Степень параллелизма алгоритма является важной характеристикой алгоритма и характеризует максимальное количество его попарно-параллельных ветвей. При морфологическом синтезе системы логического управления на основе микроконтроллерной сети данная характеристика определяет минимальное количество микроконтроллеров, требуемых для ее построения. В ряде случаев (например, [2]) возникает необходимость в ее вычислении. Следует отметить, что если при оценке качества разбиения интересует только числовое значение характеристики, то иногда требуется получение еще и подмножества попарно-параллельных вершин (например, нахождение базового сечения в *R*-алгоритме [1]). Второй случай является более общим.

Для получения значения данной характеристики можно воспользоваться следующим алгоритмом.

1. Для каждой операторной и условной вершины $a_i \in A^0$ сформировать множество параллельных с ней вершин $A_\omega(a_i) \subseteq A^0,\ \forall a_j \in A_\omega(a_i): a_i \omega a_j,\ a_i \in A_\omega(a_i)$.
2. В сформированном множестве вершин $A_\omega(a_i)$ выделить все максимальные по включению подмножества $A_\omega^1,\ A_\omega^2,\ ...,\ A_\omega^M$ попарно-параллельных вершин $\forall a_i \in A_\omega^k,\ a_j \in A_\omega^k : a_i \omega a_j,\ k = \overline{1,M},\ i,j = \overline{1,N},\ N = \left|A^0\right|$.

3. Положить оценку степени параллельности вершины равной мощности подмножества, содержащего максимальное количество элементов $\omega(a_i) = \max\limits_{j=1,M} \left| A_\omega^j \right|$.
4. Положить оценку степени параллелизма алгоритма равной максимальной оценке степени параллельности вершин $\omega_{\max}\left(A^0\right) = \max\limits_{i=1,N} \omega(a_i)$.

Рассмотренный алгоритм не только определяет степень параллелизма $\omega_{\max}$, но и выделяет подмножество попарно-параллельных вершин, т.е. решает наиболее общую задачу.

Произведем анализ вычислительной сложности алгоритма [3]. Просмотр всех вершин алгоритма производится за $O(N)$ операций, построение множества параллельных вершин для каждой выбранной вершины производится также за $O(N)$ операций. Выделение подмножеств параллельных вершин производится не более чем за $O(W!)$ действий, где W – максимальная мощность подмножества. Оценка мощности подмножества производится не более чем за $O(W)$ операций. Операция выбора максимального по мощности подмножества производится за время $O(W!)$. Таким образом, совокупная сложность операции определения мощностей подмножеств составляет $O\left(N^2 W \cdot W!\right)$ операций, а операции формирования оценок $\omega(a_i)$ соответственно $O\left(N^2\left(W! \cdot W + W!\right)\right) = O\left(N^2 W \cdot W!\right)$. Операция выбора вершины с максимальной оценкой производится за $O(N)$ операций. Т.о. совокупная сложность алгоритма составляет $O\left(N^2 W \cdot W! + N\right) = O\left(N^2 W \cdot W!\right)$.

Оценим емкостную сложность алгоритма. Сам по себе алгоритм не требует существенных затрат памяти – они не зависят от размера оцениваемого алгоритма и составляют величину $O(1)$. Однако для его функционирования необходима матрица отношений [4], для хранения которой требуется $O\left(N^2\right)$ ячеек памяти.

Сравним предложенный алгоритм с другими известными аналогами. Прежде всего рассмотрим алгоритм полного перебора попарно-

параллельных подмножеств вершин, характеризующийся сложностью $O(N!)$. В общем случае выполняется соотношение $\omega_{\max} \le W < N$, однако для реальных алгоритмов точнее будет записать $\omega_{\max} \approx W \ll N$ (1). Из приведенного соотношения вытекает, что $O(NW) < O((N-1)(N-2))$ (*), кроме того, при $W < N-3$ справедливо соотношение $O(W!) < O((N-3)!)$ (**). При почленном перемножении соотношений (*) и (**) получим $O(NW \cdot W!) < O((N-1)(N-2) \cdot (N-3)!)$ или $O(NW \cdot W!) < O((N-1)!)$. Домножив обе части полученного неравенства на $O(N)$, получим $O(N^2 W \cdot W!) < O(N!)$. Т.е. уже при $W < N-3$ рассматриваемый метод обладает меньшей вычислительной сложностью, чем метод полного перебора.

Попытаемся сравнить предложенный алгоритм с *R*-алгоритмом, характеризующимся сложностью $O\left(\frac{N^3}{\omega_{\max}}\right)$ [1]. При условии, что рассматриваемый метод обладает меньшей сложностью, должно выполняться соотношение $O(N^2 W \cdot W!) < O\left(\frac{N^3}{\omega_{\max}}\right)$ или $O(\omega_{\max} \cdot W \cdot W!) < O(N)$. С учетом (1) можно записать $O(W^2 \cdot W!) < O(N)$. Справедливость данного неравенства на практике представляется сомнительной, т.к. для его соблюдения требуются очень серьезные ограничения. Например, при N=1000 W=4 ($4^2 \cdot 4! = 384$), а при N=100000 W=6 ($6^2 \cdot 6! = 25920$). Для подтверждения сформулированных предположений сравним реализации предложенного алгоритма с *R*-алгоритмом. Для этого воспользуемся программной системой PAE [2], сравнение проведем на выборке случайных алгоритмов. С целью уменьшения временных затрат на проведение сравнения ограничимся пятью алгоритмами для каждого заданного числа вершин, временные затраты на нахождение искомой оценки степени параллелизма усредним.

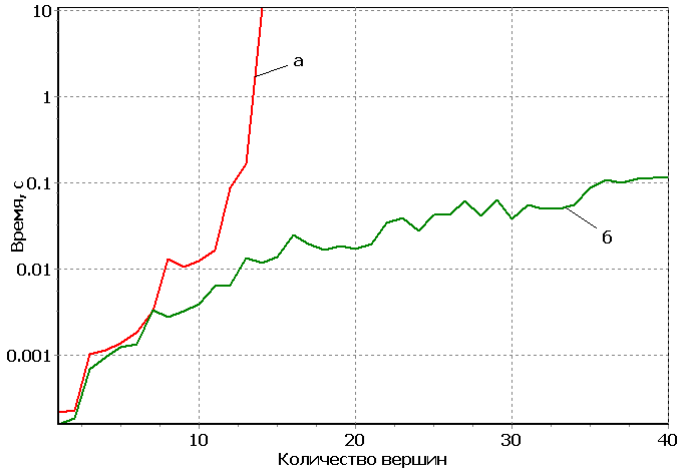

Рис. 1. Сравнение реализаций алгоритмов оценки степени параллелизма алгоритма (а – предложенный в статье переборный метод, б – R-алгоритм)

По результатам сравнения можно сделать вывод о том, что предложенный алгоритм хуже *R*-алгоритма по временным затратам, что подтверждается программной реализацией. Однако следует отметить, что *R*-алгоритм довольно сложен и трудоемок в реализации, в то время как предложенный алгоритм сравнительно прост и может быть использован для оценки степени параллелизма алгоритмов размерности $N \le 14$ (точнее, $W \le 14$). Если же степень параллелизма алгоритма существенно меньше его размерности и не превосходит 14 (приблизительно), то предложенный алгоритм также может быть использован для ее получения.

Библиографический список

1. Зотов И.В., Колосков В.А., Титов В.С., Сапронов К.А., Волков А.П. Организация и синтез микропрограммных мультимикроконтроллеров. Курск: ГУИПП «Курск», 1999. 368 с.
2. Ватутин Э.И., Зотов И.В. Программная система для построения разбиений параллельных управляющих алгоритмов // Идентификация систем и задачи управления (SICPRO’06). М.: ИПУ РАН, 2006. С. 2239–2250.
3. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. Пер. с англ. М.: Мир, 1979. 536 с.

4. Ватутин Э.И., Зотов И.В. Построение матрицы отношений в задаче оптимального разбиения параллельных управляющих алгоритмов // Известия КурскГТУ. Курск, 2004. № 2. С. 85–89.