УДК 519.173:510.5

А.А. Кобозева, канд. физ.-мат. наук, доц.,
Е.В. Нариманова, магистр, Одес. нац. политехн. ун-т.

МЕТОД ПОСТРОЕНИЯ ТОПОЛОГИЧЕСКОЙ СОРТИРОВКИ ОБЪЕДИНЕНИЯ ГРАФОВ, ИСПОЛЬЗУЕМЫЙ ПРИ РАСПАРАЛЛЕЛИВАНИИ ПОСЛЕДОВАТЕЛЬНЫХ АЛГОРИТМОВ

А.А. Кобозєва, О.В. Наріманова. **Метод побудови топологічного сортування об'єднання графів, що використовується при розпаралелюванні послідовних алгоритмів.** Пропонується метод побудови топологічного сортування графа за наявними топологічними сортуваннями його підграфів, що використовується при дослідженні послідовних алгоритмів за допомогою теорії графів з метою їх розпаралелювання; приводяться результати обчислювального експерименту.

A.A. Kobozeva, E.V. Narimanova. **Method of construction of graph association topological sorting used at parallelizing of consecutive algorithms.** The method of constructing topological sorting of the graph by available topological sortings of its subgraphs is offered, which is used studying the consecutive algorithms through the graph theory with the purpose of their parallelizing; the results of computing experiment are presented.

Появление параллельных вычислительных систем и внедрение их в практику решения больших прикладных задач привело к возникновению целого ряда проблем в области численного программного обеспечения. Построение новых численных методов для параллельных вычислительных систем процесс долговременный и сложный. Попытки разработать специальные параллельные методы не всегда оказывались состоятельными [1]. Кроме того, непростительно просто отбросить при работе на параллельных ЭВМ огромный багаж численных методов и программ, накопленный в процессе длительного использования последовательных компьютеров. Актуальны не только исследования "классических" последовательных алгоритмов для определения возможности их использования в параллельных вычислительных системах, но и сведения о параллельных свойствах алгоритмов, а также знания, позволяющие эти сведения получать.

Эффективное использование различного типа параллельных вычислительных систем возможно за счет одновременного выполнения ими ветвей вычислений, не связанных между собой информационно. Поэтому целью исследования последовательного алгоритма является поиск и выделение таких ветвей. Если они найдены, говорят, что алгоритму присущ внутренний параллелизм, и его принципиально возможно реализовать на параллельной вычислительной системе, в противном случае его использование в ней нецелесообразно.

Одной из возможных форм записи алгоритмов является их представление в виде графов [2]. Граф алгоритма $G(V, E)$, где V — множество вершин графа, а E — множество его ребер, [3] определяет все множество возможных реализаций, показывая, как при этих реализациях происходит распространение информации.

Реализация алгоритма порождает определенную разбивку его операций на группы, которые выполняются последовательно, а операции внутри каждой группы могут выполняться параллельно (одновременно). Разбивка операций алгоритма порождает соответствующую разбивку вершин графа алгоритма, и наоборот.

Если задан ориентированный ациклический граф, имеющий n вершин, то существует такое целое число $s \le n$, в соответствии с которым все вершины графа можно

так пометить одним из индексов 1, 2, …, *s*, что если ребро идет из вершины с индексом *l* в вершину с индексом *p*, то $l < p$ [3]. Такое помечивание вершин графа называется топологической сортировкой. Следует отметить, что у одного и того же ориентированного ациклического графа может быть несколько различных топологических сортировок, каждая из которых разделяет его вершины и соответствующие им операции алгоритма на группы, содержащие вершины с одинаковыми топологическими индексами. Такие группы называются топологическими уровнями. Операции, соответствующие вершинам графа одного уровня топологической сортировки, являются информационно независимыми, а, значит, могут выполняться параллельно. Группы операций, соответствующие различным топологическим уровням, выполняются последовательно в порядке возрастания номеров вершин графа, входящих в них.

Как правило, чем сложнее устроен граф, чем больше его размерность, тем труднее построить его топологическую сортировку, тем больше времени занимает процесс его предобработки. Кроме того, однозначно граф алгоритма определяется только для конкретных входных данных. Если же алгоритм содержит условные операторы, то он определяет целое семейство “похожих” графов [3]. Если результаты условных операций определяются результатами промежуточных вычислений, то можно даже не знать точно, какой граф семейства будет отвечать конкретному набору входных данных. В таком случае имеет смысл рассмотреть объединение графов семейства в один граф и использовать его топологическую сортировку, при этом каждый из графов рассмотренного семейства становится подграфом построенного графа. Для решения указанных проблем необходимо иметь возможность не только восстанавливать топологические сортировки подграфов по имеющейся топологической сортировке графа [3], но и по сортировкам подграфов строить топологическую сортировку последнего.

Предлагается алгоритм построения топологической сортировки графа алгоритма по сортировкам подграфов его разбиения; приводятся результаты вычислительного эксперимента, дающие возможность сравнить предлагаемый алгоритм с используемым ранее непосредственным построением топологической сортировки графа большой размерности [3].

Пусть граф $G(V, E)$ является сложным для исследования и обработки в силу своей размерности или других причин, некоторые из которых уже указаны. Разложим граф на подграфы. Построим разбиение $\{V_1, \cdots, V_k\}$ множества вершин V: $\bigcup_{i=1}^{k} V_i = V$, $V_i \cap V_j = \varnothing$, для $i \neq j$.Каждое подмножество V_i, $i = \overline{1,k}$, определяет порожденный подграф $G_i(V_i, E_i)$ графа $G(V, E)$. Для любого множества $V_i \subseteq V$ порожденным подграфом называется максимальный подгаф графа $G(V, E)$, множеством вершин которого является V_i. Построение сортировки графа $G(V, E)$ проведем на основе сортировок подграфов $G_i(V_i, E_i)$. Заметим, что $\bigcup_{i=1}^{k} G_i(V_i, E_i) \neq G(V, E)$, поскольку при выделении подграфов $G_i(V_i, E_i)$, происходит потеря ребер, инцидентных вершинам из разных подмножеств V_i. Вследствие такой потери разрываются связи между вершинами графа $G(V, E)$, проход по которым может определить топологический индекс вершины. Отметим, что связный граф невозможно разбить на непересекающиеся подграфы, объединение которых давало бы исходный. Например, $G_1(V_1, E_1) \cup G_2(V_2, E_2) \neq G(V, E)$ (рис.1), т.к. объединение не содержит ребро $\langle 1, 5 \rangle$, а значит при построении топологической сортировки графа $G(V, E)$ по топологическим сортировкам $G_1(V_1, E_1)$,

$G_2(V_2, E_2)$ не учитывается возможность получения вершинами исходного графа с индексами 5 и 6 топологических номеров по цепочке 1, 5, 6.

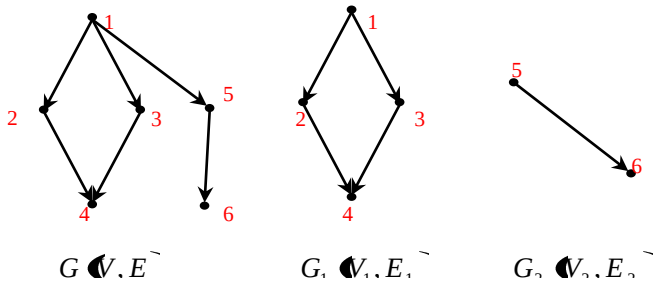


Рис.1.Разбиение графа на подграфы

Возникновение подобных ситуаций делает принципиально невозможным получение топологической сортировки графа $G(V, E)$, по сортировкам его подграфов, порождаемых разбиением множества вершин V.

Определение 1. Множество $E_T = E - \bigcup_{i=1}^{k} E_i$ называется множеством теряемых ребер графа $G(V, E)$ при разбиении $\{V_1, \cdots, V_k\}$ множества V.

Определение 2. Вершины графа $G(V, E)$, являющиеся концевыми вершинами теряемых ребер, называются граничными вершинами.

Во избежание потери ребер при разбиении графа $G(V, E)$ заменим каждый его порожденный подграф $G_i(V_i, E_i)$ на другой порожденный подграф $\overline{G_i}(\overline{V_i}, \overline{E_i})$, $i = \overline{1,k}$, где множество $\overline{V_i}$ содержит вершины, входящие в V_i, и дополнительно граничные вершины графа $G(V, E)$, каждая из которых смежна хотя бы с одной вершиной из множества V_i. Полученная новая система порожденных подграфов, очевидно, обладает свойством: $\bigcup_{i=1}^{k} \overline{G_i}(\overline{V_i}, \overline{E_i}) = G(V, E)$, но пересечение этих подграфов не обязательно равно Ø. Например, для графа $G(V, E)$,(см. рисунок 1) можно построить систему подграфов $\overline{G_i}(\overline{V_i}, \overline{E_i})$ $i = 1,2$ (рис. 2, пунктиром обозначено теряемое ребро). Построим сортировки наименьшей длины для каждого подграфа в отдельности, при этом каким-либо образом пометив граничные вершины. Без ограничения общности рассуждений, принимается, что запись каждого топологического уровня сортировки содержит сначала индексы всех вершин, не являющихся граничными, а если присутствуют граничные вершины, их индексы завершают запись. Такие топологические сортировки будут иметь минимальное количество уровней, равное длине критического пути подграфа плюс единица [3].

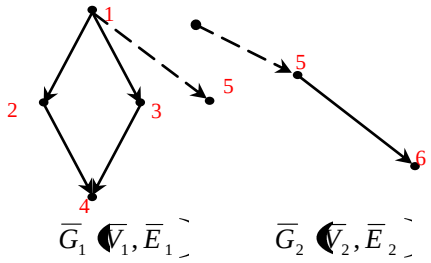


Рис.2.Восстановление теряемого ребра

При построении сортировки произвольного графа $G(V, E)$ для простоты изложения предположим, что количество определенных выше порожденных подграфов равно двум: $\overline{G_1}(\overline{V_1}, \overline{E_1}), \overline{G_2}(\overline{V_2}, \overline{E_2})$, а G_1, G_2 — их топологические сортировки, представляющие массивы, в которых последовательно выписаны топологические уровни, начиная с первого. Обозначим $t_1(\cdot)$, $t_2(\cdot)$ — топологические номера вершины в G_1, G_2 соответственно, $t_r(\cdot,\bullet)$ — топологический номер граничной вершины в результирующей сортировке графа $G(V, E)$, $t_{k(\cdot)}$, $i = 1,2$, — корректировочные разности для $\overline{G_i}(\overline{V_i}, \overline{E_i})$, $i = 1,2$.

Корректировочной разностью подграфа $\overline{G}_i(V_i, \overline{E}_i)$, $i = 1,2$, назовем разность между топологическим индексом вершины в сортировке $\overline{G}_i(V_i, \overline{E}_i)$ и ее индексом в сортировке $G(V, E)$.

Предлагаемый алгоритм содержит следующие основные этапы:

— инициализуются значения корректировочных разностей для подграфов $\overline{G}_1(V_1, \overline{E}_1)$, $\overline{G}_2(V_2, \overline{E}_2)$ и определяется порядок просмотра топологических сортировок: $t_{k(\cdot)} = 0$; $t_{k(\cdot)} = 0$; $i = 1$, $j = 2$;

— последовательный просмотр топологической сортировки $\overline{G}_i(V_i, \overline{E}_i)$.

Пусть v — индекс очередной вершины в сортировке $\overline{G}_i(V_i, \overline{E}_i)$.

Если индекс v соответствует неграничной вершине,

то

$t_i(v) = t_i(v) + t_{k(\cdot)}$, возврат в начало этапа;

иначе —

переход на следующий этап;

— последовательный просмотр топологической сортировки $\overline{G}_j(V_j, \overline{E}_j)$.

Пусть w — индекс очередной вершины в сортировке $\overline{G}_j(V_j, \overline{E}_j)$.

Если индекс w соответствует неграничной вершине,

то

$t_j(w) = t_j(w) + t_{k(\cdot)}$, возврат в начало этапа;

иначе —

M: *если* $v = w$,

то

$t_r(v, w) = \max(t_i(v), t_j(w))$;обновление корректировочных разностей:

$t_{k(\cdot)} = t_r(v, w) - t_i(v)$; $t_{k(\cdot)} = t_r(v, w) - t_j(w)$; переход на предыдущий этап;

если $v \neq w$,

то

если в пределах рассматриваемого топологического уровня $\overline{G}_j(V_j, \overline{E}_j)$ существует такая граничная вершина с индексом u, что $v = u$,

то

вывести вершину с индексом u на первое место в списке граничных вершин данного топологического уровня; переход на метку M;

иначе —

$t_{k(\cdot)} = (\max(t_i(v), t_j(w)) + 1) - t_i(v)$,

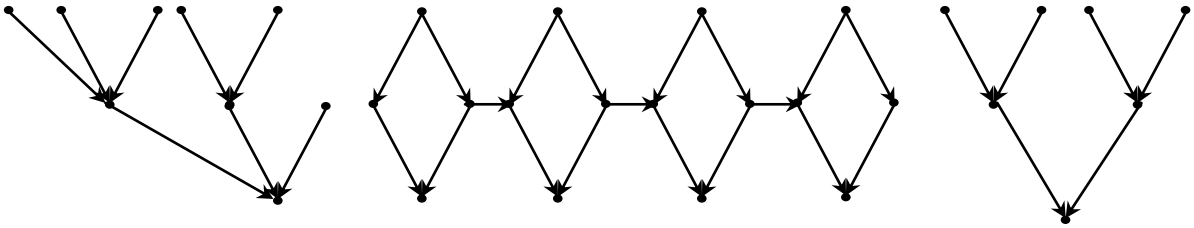
$t_i(v) = \max(t_i(v), t_j(w)) + 1$; $c = i, i = j, j = c$; переход на предыдущий этап.

Обобщение предложенного алгоритма на граф с большим числом подграфов не вызывает труда: для присвоения окончательного топологического номера каждой граничной вершине необходимо будет исследовать не два, а более подграфов, что потребует и большего общего количества арифметических операций. Таким образом, можно

ожидать, что эффективность предложенного алгоритма будет зависеть от числа подграфов, а наибольший выигрыш во времени по сравнению с непосредственным построением топологической сортировки исходного графа, очевидно, будет наблюдаться, когда количество подграфов минимально (два, три), что и подтверждается результатами проведенного вычислительного эксперимента.

В вычислительном эксперименте исследовалась эффективность с точки зрения временных затрат предложенного алгоритма построения топологической сортировки графа по топологическим сортировкам его подграфов по сравнению с непосредственным построением сортировки графа. Эксперимент проводился на графах различных по структуре последовательных алгоритмов. При этом изменялась размерность и количество подграфов, на которые разбивался граф. При проведении эксперимента фиксировалась размерность графа, начиная с которой новый алгоритм работал быстрее. Представлены примеры некоторых графов небольшой размерности (рис.3).

Из полученных результатов вычислительного эксперимента, проведенного в среде MATLAB (см. таблицу), очевидно, что даже при сравнительно небольшом количестве вершин графа построение его топологической сортировки по топологическим сортировкам его подграфов происходит быстрее, чем непосредственное построение топологической сортировки всего графа. Это количество вершин фиксировалось, как только преимущество во времени достигало порядка 10^{-2} с. С увеличением размерности графа преимущество возрастает. Например, для графа 1, количество вершин которого равно 250, а количество подграфов равно двум, время построения сортировки по предложенному алгоритму уже на 5 с меньше, чем при использовании алгоритма непосредственного построения топологической сортировки всего графа в целом. Кроме того, как следует из результатов эксперимента, чем больше количество подграфов разбиения графа, тем при большей его размерности достигается преимущество во времени по сравнению с построением топологической сортировки целого графа, как и предполагалось ранее.



1 *2* *3*

Рис.3. Примеры используемых графов

При проведении вычислительного эксперимента отмечено, что эффективность построения топологической сортировки графа по топологическим сортировкам его подграфов в значительной степени зависит от способа его разбиения на подграфы. При разбиении следует стремиться к минимальному количеству теряемых ребер, а не равному количеству вершин в подграфах.

Результаты вычислительного эксперимента

Граф №	Кол-во подграфов	Кол-во вершин в подграфах	Количество вершин графа, начиная с которого достигается преимущество предложенного алгоритма по времени
1	2	96 / 105	200
1	3	97 / 92 / 101	288
1	4	122 / 76 / 62 / 60	323
2	2	256 / 257	512
2	4	161 / 161 / 161 / 160	640

3	2	176 / 175	350
3	4	139 / 138 / 139 / 138	550
4	2	105 / 296	400
4	3	150 / 173 / 177	500
5	2	288 / 289	576
5	3	217 / 218 / 216	648
5	4	177 / 177 / 178 / 178	704
6	2	98 / 139	235
6	4	102 / 128 / 96 / 133	456

Предложенный алгоритм построения топологической сортировки графов позволяет проводить это построение на графах большой размерности быстрее, чем алгоритм, основывающийся на непосредственной обработке графа, поэтому исследование графов реальных последовательных алгоритмов, размерность которых, как правило, велика, с целью их распараллеливания имеет смысл проводить с помощью предложенного нового алгоритма.

Литература

1. Воеводин В.В. Параллельные вычисления. Воеводин В.В., Воеводин Вл. В. – СПб.: БХВ-Петербург, 2002. –608 с.
2. Харари Ф. Теория графов. – М.: Мир, 1973.
3. Воеводин В.В. Математические основы параллельных вычислений. – М.: Изд-во Моск. ун-та, 1991. –345 с.