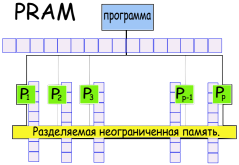
Параллельный вероятностный алгоритм Луби

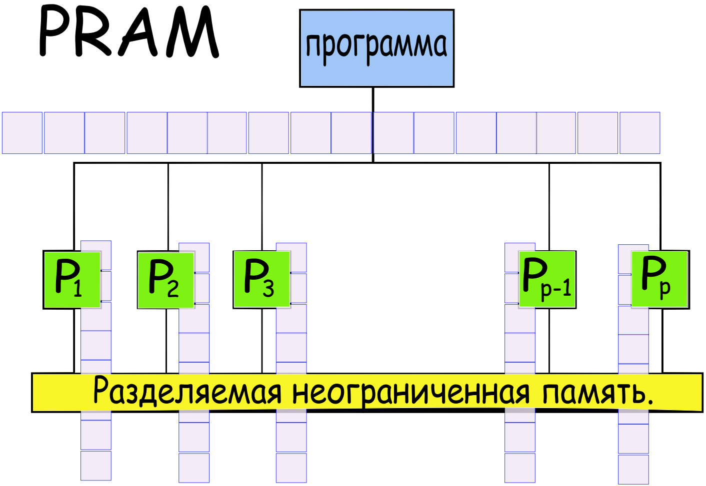
Н. Н. Кузюрин С. А. Фомин

3 декабря 2010 г.



Параллельный алгоритм нахождения максимального по включению независимого множества в графе. Полилогарифмическое время алгоритма «в среднем».

PRAM — Parallel Random Access Machine



Модели параллельных вычислений PRAM

В зависимости от типа, разные процессоры могут одновременно:

EREW PRAM: *exclusive read/exclusive write* — читать и изменять только разные ячейки памяти.

CREW PRAM: *concurrent read/exclusive write* — читать из одной ячейки, но изменять только разные.

CRCW PRAM: *concurrent read/concurrent write* — параллельно читать и изменять любую ячейку.

Типы CRCW PRAM

WEAK CRCW PRAM — несколько процессоров могут записывать в одну ячейку только число 0.

COMMON CRCW PRAM — несколько процессоров могут записывать в одну ячейку только одинаковое число.

ARBITRARY CRCW PRAM — несколько процессоров могут записывать в одну ячейку разные числа, а содержимое ячейки становится одним из этих чисел, но не известно каким.

PRIORITY CRCW PRAM — результатом записи в ячейку является значение, поступившее от процессора с большим номером.

STRONG CRCW PRAM — результатом записи в ячейку является максимальное значение, поступившее от процессоров.

Типы PRAM: примеры алгоритмов

Рассмотрим поясняющий пример: вычислить $x_1 \vee \ldots \vee x_n$.
WEAK CRCW PRAM: n процессоров могут записывать в одну ячейку число 0.

Алгоритм с константным числом параллельных шагов.
1. $y := 1$.
2. write parallel $x_1, \ldots, x_n$ into y.

EREW PRAM: требуется $O(\log n)$ параллельных шагов.
Метод сдваивания: 1) вычислить
$y_1 = x_1 \vee x_2$, $y_2 = x_3 \vee x_4$, …, $y_{n/2} = x_{n-1} \vee x_n$,
2) вычислить
$t_1 = y_1 \vee y_2$, $t_2 = y_3 \vee y_4$, …, $t_{n/4} = y_{n/2-1} \vee y_{n/2}$,
и т. д.

Типы PRAM: примеры алгоритмов

Задача: для матрицы $n \times n$ и вектора x вычислить Ax.
EREW PRAM с n^2 процессорами: требуется $O(\log n)$ параллельных шагов.

Задача: для чисел $x_1, \ldots, x_n$ вычислить все префиксные суммы
$y_1 = x_1$, $y_2 = x_1 + x_2$, $y_3 = x_1 + x_2 + x_3$, …, $y_n = x_1 + x_2 + \ldots + x_n$.
EREW PRAM с n процессорами: требуется $O(\log n)$ параллельных шагов.

Максимальное по включению независимое множество

Определение

Пусть $G = (V, E)$ — неориентированный граф с n вершинами и m ребрами.
Подмножество вершин $I \subseteq V$ называется ***независимым*** *в G, если никакое ребро из E не содержит обе своих конечных вершины в I.*

Определение

Независимое множество I называется ***максимальным по включению***[a]*, если оно не содержится в качестве собственного подмножества в другом независимом подмножестве в G.*

[a] В англоязычной литературе — *Maximal Independent Set*.

Последовательный алгоритм сложности $O(m)$

$\Gamma(v)$ — вершины-соседи для v.
Алгоритм **Greedy MIS**:
Вход: Граф $G = (V, E)$, $|V| = n$, $|E| = m$.
Выход: Максимальное по включению множество $I \subseteq V$.

1. $I := \emptyset$.
2. **for** $v = v_1$ **to** v_n **do**
3. **if** $I \cap \Gamma(v) = \emptyset$ **then** $I := I \cup \{v\}$.

Лемма

*Алгоритм **Greedy MIS** завершает работу после $O(n)$ шагов и выдает максимальное по включению независимое множество при условии, что граф на вход подается в виде списка смежности.*

Идея параллельного алгоритма

Идея — на каждой итерации находится независимое множество S, которое добавляется к уже построенному I, а $S \cup \Gamma(S)$ удаляется из графа.

Чтобы число итераций было небольшим, независимое множество S должно быть таким, чтобы $S \cup \Gamma(S)$ было большим.

Для реализации этой идеи мы выбираем большое случайное множество вершин $R \subseteq V$. Маловероятно, что R будет независимым, но, с другой стороны, имеется немного ребер, оба конца которых принадлежат R.

Для получения независимого множества из R мы рассмотрим такие ребра и удалим концевые вершины с меньшей степенью.

Параллельный алгоритм **Parallel MIS**:

Вход: Граф $G = (V, E)$.

$I \leftarrow \emptyset$
while $V \neq \emptyset$ **do**
  for all $v \in V$ (параллельно) **do**
    if $d(v) = 0$ **then**
      $V \leftarrow V \setminus \{v\}$, $I \leftarrow I \cup \{v\}$
    else
      $marked(v) \leftarrow 1$ с вероятностью $\frac{1}{2d(v)}$
    end if
  end for
  for all $(u, v) \in E$ (параллельно) **do**
    if $marked(v) \wedge marked(u)$ **then**
      $marked(v) \leftarrow 0$ ($d(v) \leq d(u)$)
    end if
  end for
  for all $v \in V$ (параллельно) **do**
    if $marked(v)$ **then**
      $S \leftarrow S \cup \{v\}$
    end if
  end for
  $V \leftarrow V \setminus S$ {и все инцидентные ребра из E}
  $I \leftarrow I \cup S$
end while

Выход: Максимальное-независимое $I \subseteq V$.

Параллельный алгоритм **Parallel MIS**:

Вход: Граф $G = (V, E)$.

$I \leftarrow \emptyset$
while $V \neq \emptyset$ **do**
  for all $v \in V$ (параллельно) **do**
    if $d(v) = 0$ **then**
      $V \leftarrow V \setminus \{v\}, I \leftarrow I \cup \{v\}$
    else
      $marked(v) \leftarrow 1$ с вероятностью $\frac{1}{2d(v)}$
    end if
  end for
  for all $(u, v) \in E$ (параллельно) **do**
    if $marked(v) \wedge marked(u)$ **then**
      $marked(v) \leftarrow 0$ ($d(v) \leq d(u)$)
    end if
  end for
  for all $v \in V$ (параллельно) **do**
    if $marked(v)$ **then**
      $S \leftarrow S \cup \{v\}$
    end if
  end for
  $V \leftarrow V \setminus S$ {и все инцидентные ребра из E}
  $I \leftarrow I \cup S$
end while

Выход: Максимальное-независимое $I \subseteq V$.

Упражнение

Покажите, что каждая итерация алгоритма может быть реализована за время $O(\log n)$ на модели EREW PRAM с $(n + m)$ процессорами.

Определение

$\Gamma^*(v) = \{w \in \Gamma(v) : d(w) \le d(v)\}$

$v \in V$ — ***хорошая*** $\Leftrightarrow |\Gamma^*(v)| \ge \frac{d(v)}{3}$. *Иначе* — ***плохая***.

Определение

$\Gamma^*(v) = \{w \in \Gamma(v) : d(w) \leq d(v)\}$
$v \in V$ — ***хорошая*** $\Leftrightarrow |\Gamma^*(v)| \geq \frac{d(v)}{3}$. *Иначе —* ***плохая****.*

Лемма

Пусть $v \in V$ *— хорошая,* $d(v) > 0$*. Тогда*

$$\mathsf{P}\{\exists w \in \Gamma(v) : \mathit{marked}(w) = 1\} \geq 1 - \exp\left(-\frac{1}{6}\right)$$

Определение

$\Gamma^*(v) = \{w \in \Gamma(v) : d(w) \le d(v)\}$
$v \in V$ — ***хорошая*** $\Leftrightarrow |\Gamma^*(v)| \ge \frac{d(v)}{3}$. *Иначе* — ***плохая***.

Лемма

Пусть $v \in V$ — хорошая, $d(v) > 0$. Тогда

$$\mathrm{P}\{\exists w \in \Gamma(v) : marked(w) = 1\} \ge 1 - \exp\left(-\frac{1}{6}\right)$$

Доказательство.

$$P_{bad} = \mathrm{P}\{\nexists w \in \Gamma(v) : marked(w) = 1\} \le$$

$$\le \prod_{w \in \Gamma^*(v)} \left(1 - \frac{1}{2d(w)}\right) \le \left(1 - \frac{1}{2d(v)}\right)^{d(v)/3} \le e^{-1/6}.$$

$\square$

Построение независимых множеств

Лемма

На каждой итерации помеченная вершина выбирается в S с вероятностью не менее $1/2$.

Доказательство.

$$P_{bad} = \mathrm{P}\left\{\exists v \in \Gamma(w) : d(v) \geq d(w) \wedge marked(v)\right\} \leq$$

$$\leq |\{v \in \Gamma(w) \mid d(v) \geq d(w)\}| \times \frac{1}{2d(w)} \leq$$

$$\leq \sum_{v \in \Gamma(w)} \frac{1}{2d(w)} = d(w) \times \frac{1}{2d(w)} = \frac{1}{2}.$$

$\square$

Комбинируя две предыдущие леммы получаем

Лемма

Вероятность того, что хорошая вершина принадлежит множеству $S \cup \Gamma(S)$ *не меньше, чем* $(1 - e^{-1/6})/2$.

Определение

Ребро называется ***хорошим****, если хотя бы одна из его концевых вершин является хорошей, иначе —* ***плохим*** *(*$\in E_{bad}$*).*

Комбинируя две предыдущие леммы получаем

Лемма

Вероятность того, что хорошая вершина принадлежит множеству $S \cup \Gamma(S)$ не меньше, чем $(1 - e^{-1/6})/2$.

Определение

Ребро называется ***хорошим****, если хотя бы одна из его концевых вершин является хорошей, иначе —* ***плохим*** *($\in E_{bad}$).*

Финальный шаг заключается в оценке числа хороших ребер.

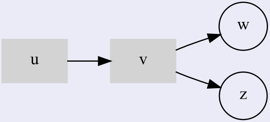
Лемма

В графе $G = (V, E)$ число хороших ребер, не меньше, чем $|E|/2$.

Доказательство

$\binom{E}{2}$ — множество всех неупорядоченных пар из E.
Определим $f: E_{bad} \to \binom{E}{2}$ так, чтобы: $\forall e_1 \neq e_2 \in E_{bad} : f(e_1) \cap f(e_2) = \emptyset$.

Ориентируем каждое ребро графа E от вершины меньшей степени в сторону вершины большей степени (в случае равенства — произвольно).



Пусть $(u \to v) \in E_{bad}$.
Поскольку v плохая, число выходящих из нее ребер, по крайней мере, вдвое превосходит число входящих в нее ребер.

Отсюда следует, что $|E_{bad}| \leq |E|/2$.

Анализ

Поскольку константная доля ребер инцидентна хорошим вершинам, а хорошие вершины удаляются с константной вероятностью, то математическое ожидание числа ребер, удаленных на итерации, составляет константную долю от текущего числа ребер.

Отсюда легко следует, что математическое ожидание числа итераций алгоритма **Parallel MIS** есть $O(\log n)$

докажите это в качестве упражнения

Интернет поддержка курса

```
http://discopal.ispras.ru/
```

Вопросы?