Задача вершинного покрытия

Сергей Воронов

Московский физико-технический институт (государственный университет)
Факультет управления и прикладной математики

11 декабря 2013 г.

Где используются графы

Примеры использования графов:

- Задачи календарного планирования
- Логистика, распределение ресурсов
- Компьютерная химия
- Схемотехника (граф соединения элементов)
- Социальные графы (анализ и предсказание)

Реальные графы

Разреженный граф — граф, в котором количество ребер растет линейно при увеличении количества вершин.

Естественно возникающие разреженные графы:

- Социальные
- Коммуникационные
- Биологические
- Графы цитирований / ссылок в Интернете

Хранение графов

Способы хранения неориентированных невзвешенных графов:

- Матрица смежности ($O(V^2)$, слишком большой размер памяти)
- Список ребер ($O(E)$, сложно выбрать вершины смежные с данной)
- Списки смежности ($O(V + E)$)

Для каждой задачи нужен свой метод хранения.

Вершинное покрытие

Расматриваются только неориентированные графы.

Вершинное покрытие — это множество его вершин S, такое что, у каждого ребра графа хотя бы один из концов входит в S.

Задача о вершинном покрытии

Найти минимальное по размеру вершинное покрытие (или указать одно из них).

Если найти минимальное вершинное покрытие, то можно сократить число записей в списках смежности.

Оптимизационная задача

Возможен взвешенный вариант задачи о вершинном покрытии: каждой вершине сопоставляется вес, и нужно найти вершинное покрытие с минимальным весом (количество вершин может быть не минимальным).

Задача поиска минимального вершинного покрытия для взвешенного графа $G = (V, E)$ с весами $c(v)$ эквивалентна оптимизационной задаче:

$$\begin{cases} \sum_{v \in V} c(v) x_v \to \min \\ x_u + x_v \geq 1, & \forall uv \in E \\ x_v \in \{0, 1\}, & \forall v \in V \end{cases}$$

Связанные задачи

Множество вершин графа называется независимым, если никакие две вершины этого множества не соединены ребром.

Задача о независимом множестве

В заданном графе G требуется найти независимое множество максимального размера.

Множество независимо тогда и только тогда, когда его дополнение является вершинным покрытием.

Напоминание

Класс P — это множество задач разрешимости, для которых существуют алгоритмы решения, время работы которых полиномиально зависит от размера входных данных.

Класс NP — это множество задач разрешимости, решение которых при наличии некоторых дополнительных сведений (т. н. сертификата решения, размер которого полиномиален от размера входных данных) можно проверить за время, полиномиальное от размера входных данных.

NP-полная задача — это задача из класса NP, к которой можно свести любую другую задачу из класса NP за полиномиальное время.

Задача о вершиннном покрытии является NP-полной.

Приближенные алгоритмы

Оценки приближенных алгоритмов:

- Жадный алгоритм: для любой константы c найдется граф, для которого этот алгоритм выдаст покрытие, которое по размеру будет более чем в c раз хуже оптимального
- Gavril and Yannakakis [1]: ошибается не более чем в 2 раза
- Bar-Yehuda and Even[2]: $2 - \Theta\left(\frac{\log\log n}{\log n}\right)$
- Halperin[3]: $2 - (1 - o(1))\frac{\ln\ln\Delta}{\ln\Delta}$, Δ – максимальная степень вершины
- Karakostas[4]: $2 - \Theta\left(\frac{1}{\sqrt{\log n}}\right)$

[1] F. Gavril и M. Yannakakis, 1974.

[2] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem, 1985.

[3] E. Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs, 2000.

[4] G. Karakostas. A better approximation ratio for the Vertex Cover problem, 2004.

Жадный алгоритм

Описание алгоритма:

1. $S := \{\emptyset\}$
2. Выбираем вершину с максимальной степенью
3. Добавляем в решение S эту вершину
4. Удаляем из графа все ребра, инцидентные выбранной вершине
5. Если остались ребра, возвращаемся к шагу 2

Контрпример к жадному алгоритму

Пример графа:

- Рассматриваем двудольный граф на долях U и V
- $|U| = n!$
- $V = V_1 \cup V_2 \cup \cdots \cup V_n$, $|V_i| = \frac{n!}{i}$
- Каждая вершина $u \in U$ имеет ровно одного соседа в группе V_i
- $\forall v \in V_i \;\; deg(v) = i$

Действие алгоритма:

- U — вершинное покрытие
- Жадный алгоритм может последовательно выбрать сначала все вершины из V_n, потом все вершины из V_{n-1}, ...
- Полученное покрытие: $n!\ (\frac{1}{n} + \frac{1}{n-1} + \ldots + 1)$ вершин

Простой алгоритм

Описание алгоритма:

1. $S := \{\emptyset\}$
2. Выбираем случайное ребро графа
3. Добавляем в решение S оба конца ребра
4. Удаляем из графа все ребра, инцидентные концам ребра
5. Если остались ребра, возвращаемся к шагу 2

Алгоритм строит покрытие, превосходящее минимальное не более чем в два раза[5].

[5] При необходимости, доказательство можно узнать тут: `http://habrahabr.ru/post/120328/`

Сложности существующих алгоритмов

Алгоритм	Сложность
Полный перебор	$2^k n^{O(1)}$
Buss and Goldsmith, 1993	$O(kn + 2^k k^{2k+2})$
Chen and Kanj, 2001	$O(kn + 1.286^k)$
Chandran and Grandoni, 2004	$O(kn + 1.2745^k k^4)$
Chen and Kanj, 2006	$O(kn + 1.2378^k)$
Теоретический предел[6]	$c^k n^{O(1)}$, для $c > 1$

[6] При условии, что 3-SAT не может быть решена за субэкспоненциальное время.

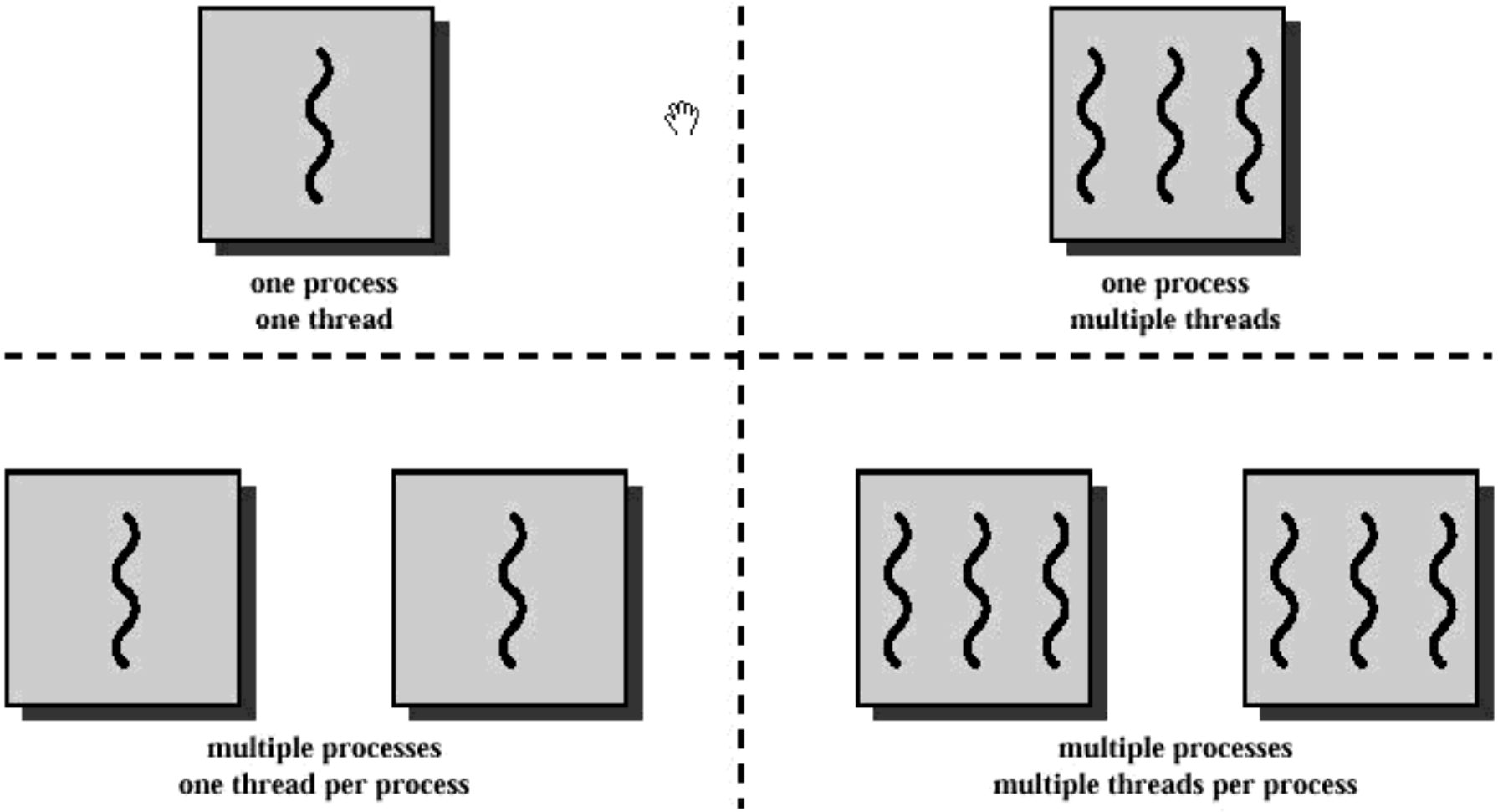
Статьи с точными алгоритмами

- J. Buss and J. Goldsmith. Nondeterminism within P, 1993.
- J. Chen, I. Kanj, and W. Jia. Vertex cover: further observations and further improvements, 2001.
- L. Chandran and F. Grandoni. Refined memorisation for vertex cover, 2004.
- J. Chen, I. Kanj, and G. Xia. Improved Parameterized Upper Bounds for Vertex Cover, 2006.

Обзор технологий распараллеливания

- Системы с распределенной памятью:
  - Явное задание коммуникаций между процессами
  - Библиотеки для передачи сообщений:
    - MPI (“Message Passing Interface”)
    - PVM (“Parallel Virtual Machine”)
    - MPT (Cray)
- Shmem
- Shared memory системы
  - Программирование, основанное на “Thread” (pthread)
  - Директивы компиляторов (OpenMP, . . .)

Типы программ



Преимущества и недостатки

Распределенная память (MPI-like):

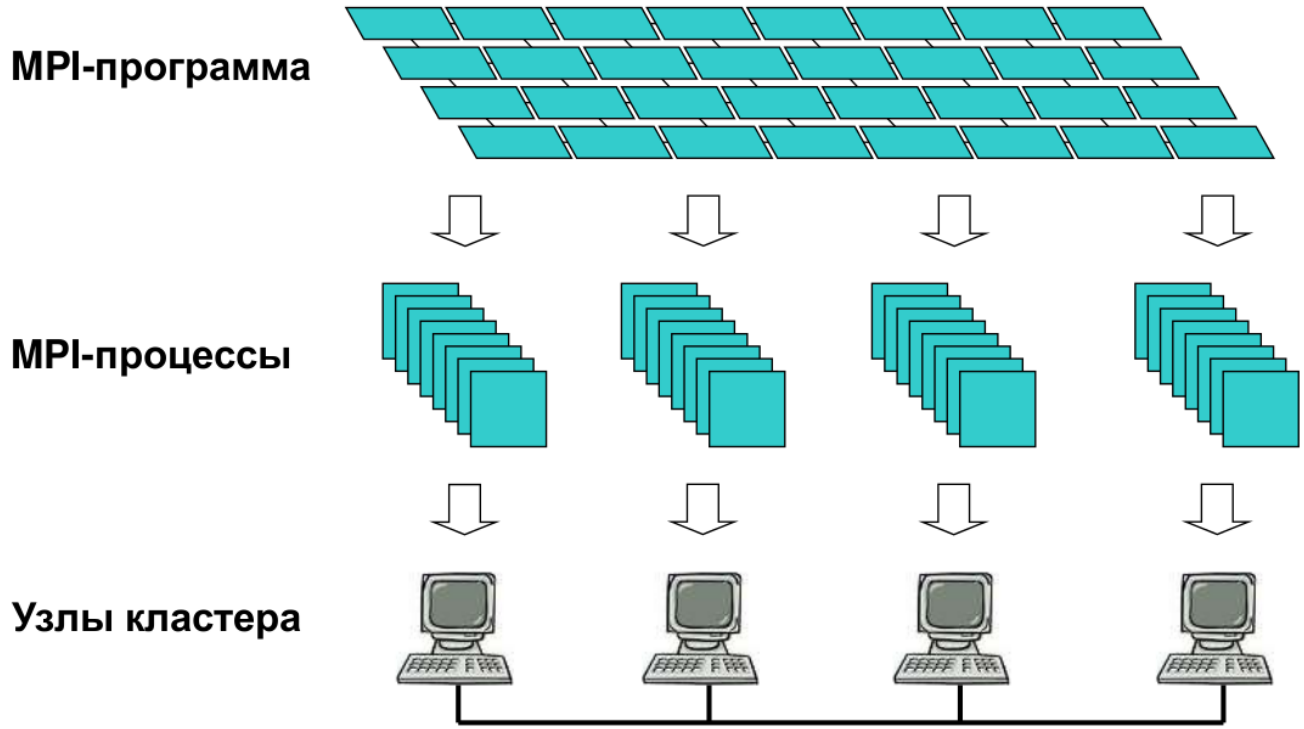
+ Сильная масштабируемость
+ Меньшие требования к аппаратуре
– Ниже эффективность
– Сложность в написании

Общая память (OpenMP-like):

+ Проще в написании
+ Выше эффективность
– Слабая масштабируемость
– Дорогое оборудование (при масштабировании)

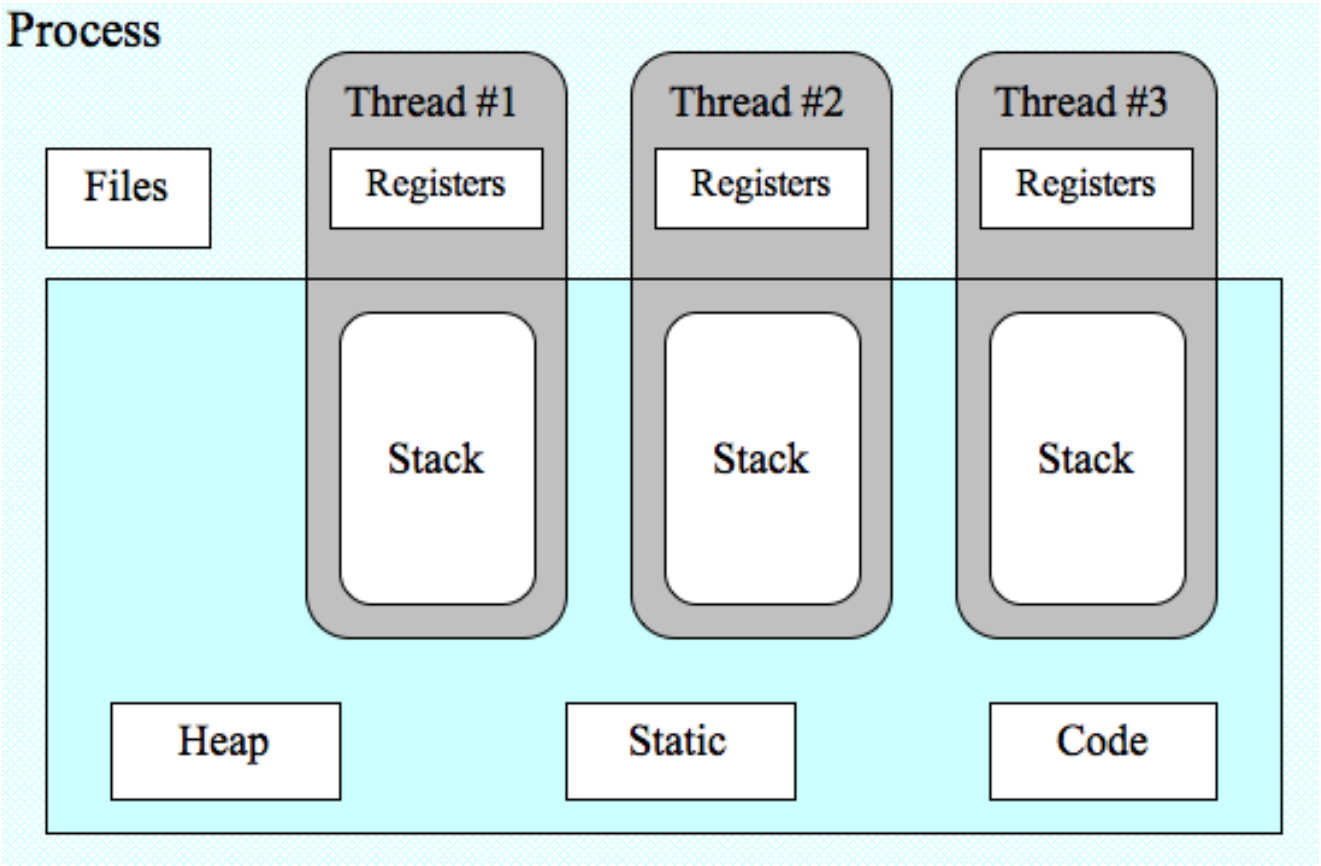
MPI

Message Passing Interface (MPI) — программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу.



Threads

POSIX Threads (Pthreads) — стандарт POSIX реализации потоков (нитей) выполнения; определяет API для управления потоками, их синхронизации и планирования.



Максимально достижимое ускорение

Закон Амдала

Пусть доля α общего объема вычислений может быть получена только последовательными расчетами, а $1 - \alpha$ может быть распараллелена так, что время работы обратно пропорционально количеству использованных вычислителей. Тогда ускорение на системе из p вычислителей не может превышать:

$$S_p = \frac{1}{\alpha + \frac{1 - \alpha}{p}}$$

Закон Густафсона-Барсиса

При тех же условиях ускорение не превышает:

$$S_p = \alpha + (1 - \alpha)p$$

Структура представлямого решения

1. Инициализация:
   - В очереди 1 набор: пустой набор вершин
   - Лучшее решение: все вершины
2. Выполнять пока очередь не пуста:
   - Достать набор из очереди
   - Проверить набор на покрытие всех ребер
   - Если данный набор решение, и он лучше, чем текущее решение:
     - Переписать лучшее решение
   - Иначе
     - **Сгенерировать новые решения** по набору, положить их в очередь

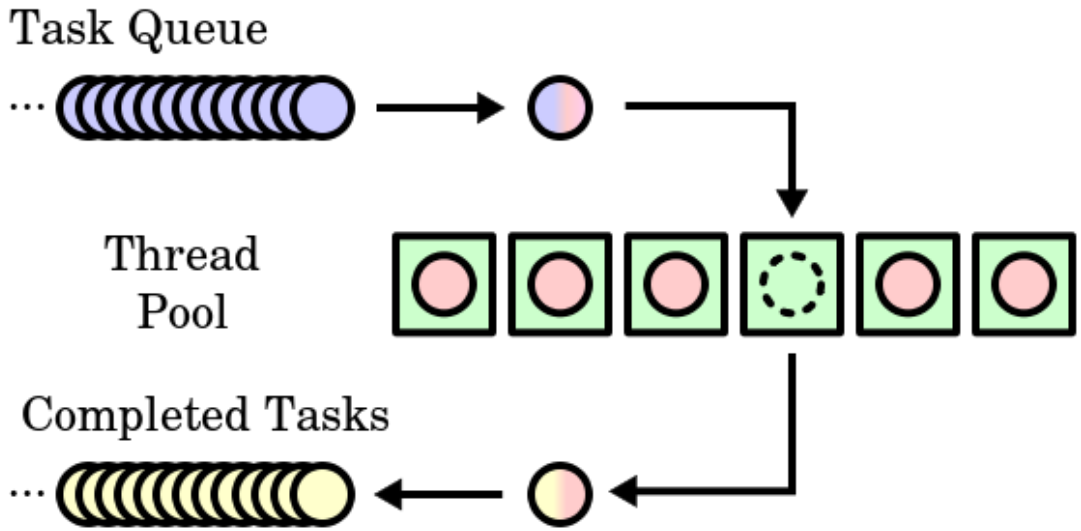
Генерация новых наборов

Функции **сгенерировать новые решения** по набору $\{a_1, \ldots, a_s\}$:

1. Добавить все возможные наборы, получаемые добавлением одной вершины
2. Добавить все возможные наборы, получаемые добавлением одной вершины, если s (длина набора) меньше, чем длина лучшего решения.
3. Добавить все возможные наборы, получаемые добавлением одной вершины с номером $c > \max(a_1, \ldots, a_s)$, если s меньше, чем длина лучшего решения.
4. Если s меньше, чем длина лучшего решения, выбрать наименьшее (лексикографически) непокрываемое ребро, и добавить обе вершины по отдельности, получив два набора.

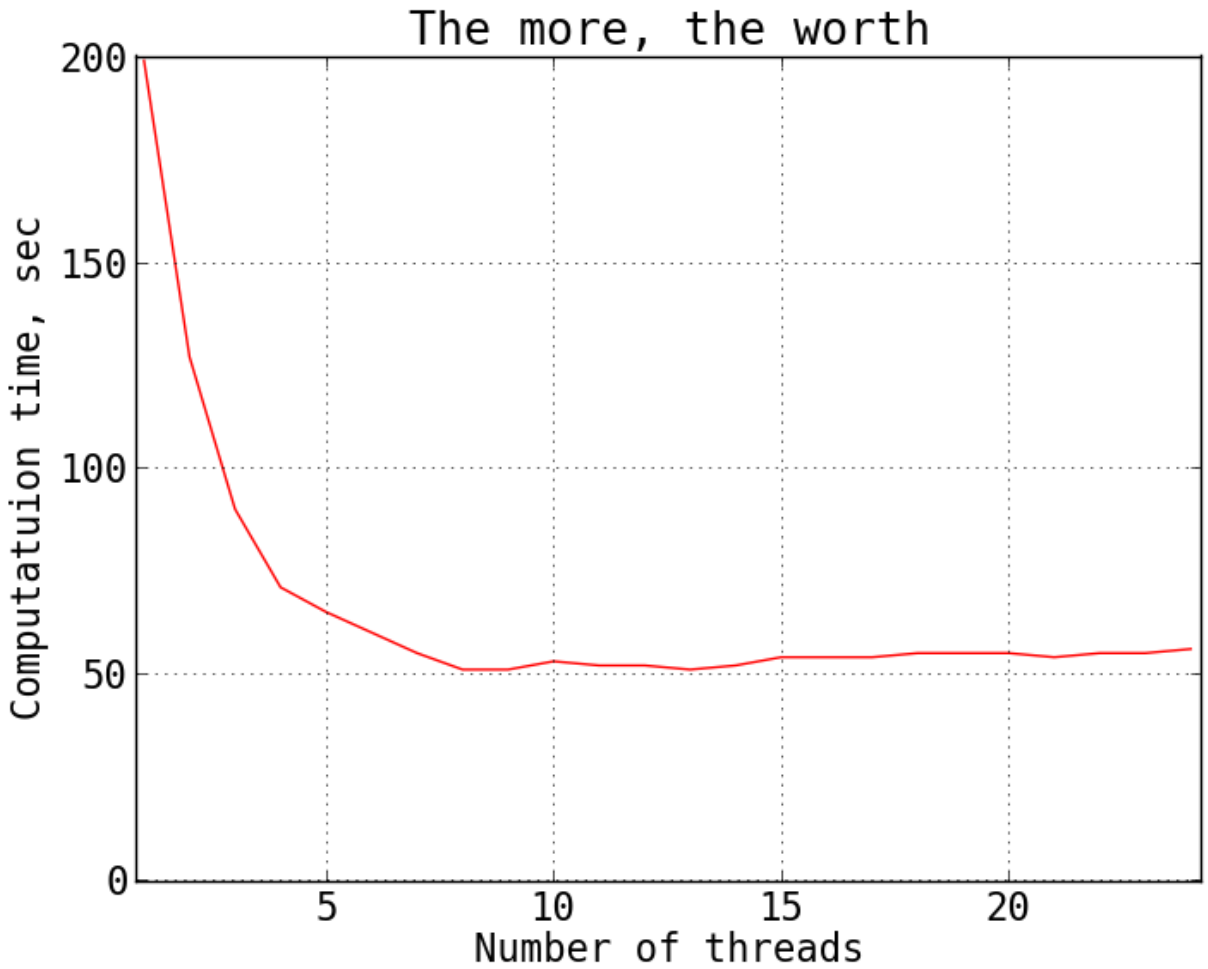
Технические особенности

Использована Shared-memory технология pthread, являющаяся реализацией POSIX-Threads.
Был написан ThreadPool:



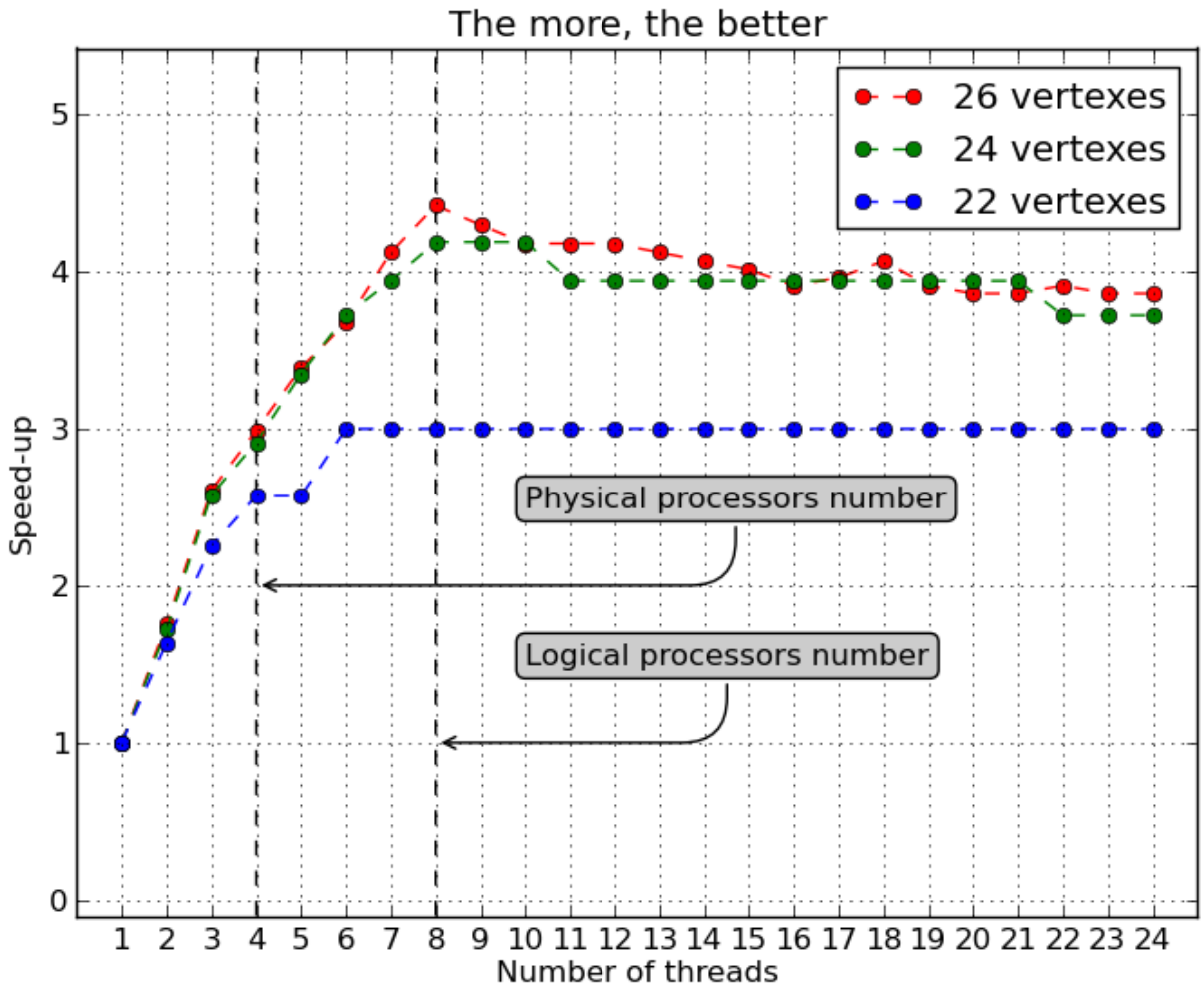
Заданием являлся набор вершин (std::tr1::unordered_set).

Зависимость времени работы от количества потоков



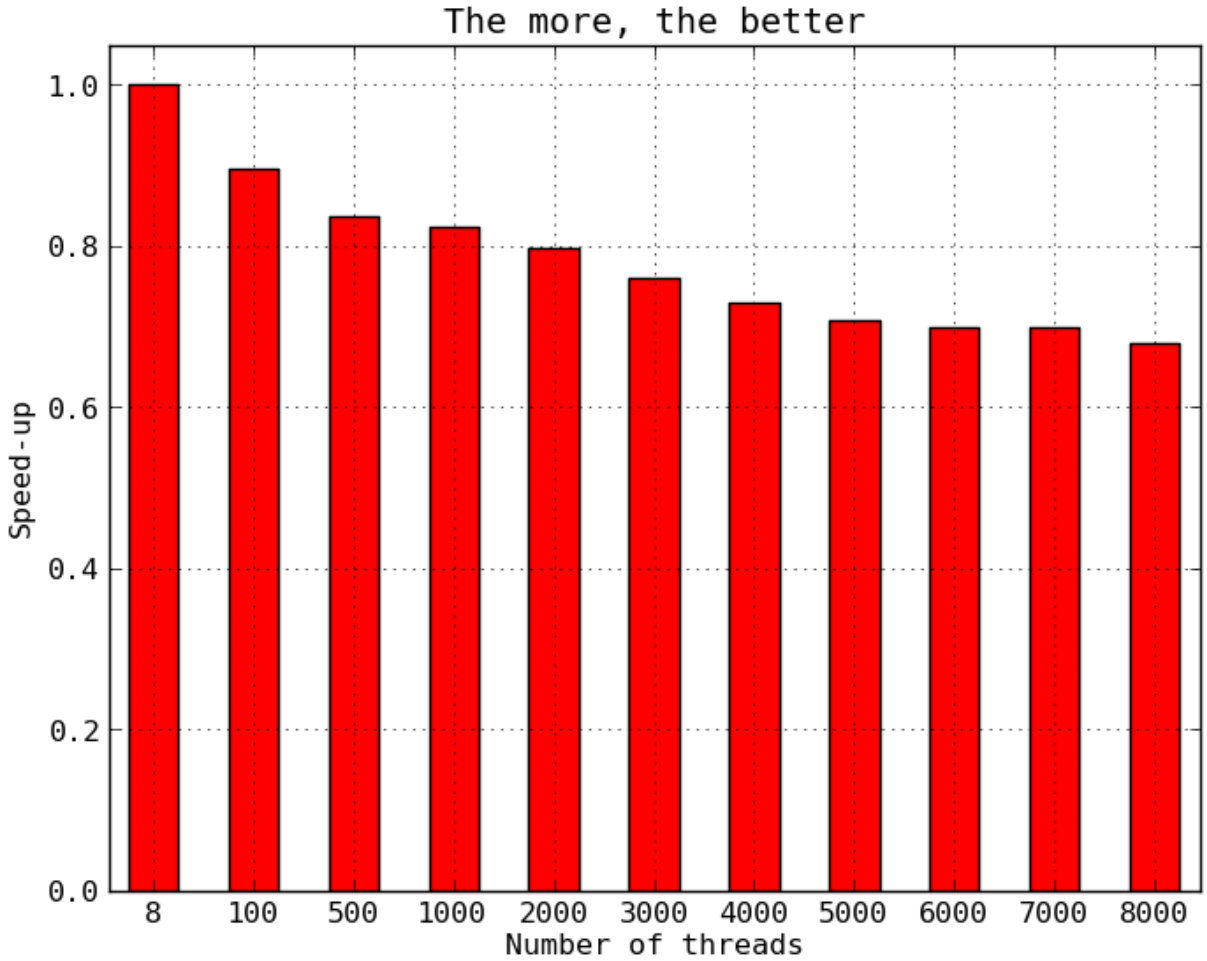
Использован разреженный 100-вершинный граф ($\sim 1\%$ ребер)

Зависимость производительности от количества потоков



Использованы полные 22-, 24- и 26-вершинные графы

Зависимость производительности от количества потоков



Использован разреженный 300-вершинный граф ($\sim 0.016\%$ ребер)

Репозиторий

Адрес репозитория с кодом: bitbucket.org/RDkL/vertex-cover.