Построение многопроцессорного расписания с использованием жадных стратегий и ограниченного перебора

Савицкий Илья
Научный руководитель: к.т.н. доцент Костенко Валерий Алексеевич

26 мая 2022 г.

Цели и задачи курсовой

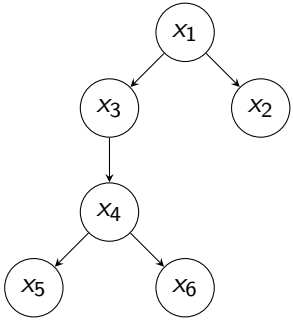
Целью этой курсовой работы является разработка алгоритма построения многопроцессорного расписания с дополнительными ограничениями на основе комбинации жадных стратегий и ограниченного перебора.
Для достижения указанной цели требуется:

1. Провести обзор алгоритмов построения списочных расписаний с целью выявления жадных критериев и схем ограниченного перебора которые могут быть модифицированы для решения данной задачи.
2. Разработать и реализовать алгоритм.
3. Провести исследование свойств алгоритма.

Постановка задачи

Дано:

1. Ориентированный граф работ G без циклов, в котором дуги - зависимости по данным, а вершины - задания. Вершин n, дуг m
2. Вычислительная система, состоящая из p различных процессоров
3. Матрица C_{ij} длительности выполнения работ на процессорах, $i = 1 \dots n, j = 1 \dots p$. Каждая строка этой матрицы - длины выполнения n-й задачи на p процессорах.
4. Матрица D_{kl} передач данных между процессорами, $k = 1 \dots p, l = 1 \dots p, D_{kk} = 0$. D_{ij}-й элемент этой матрицы - время передачи данных между процессорами i и j.



Граф потока данных

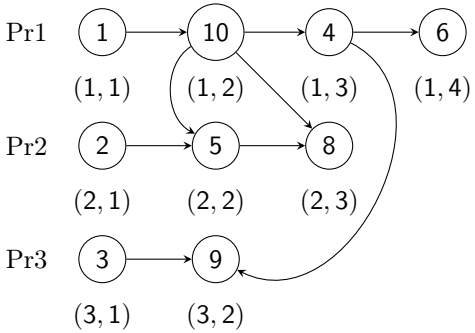
Расписание

Расписание программы определено, если определены

1. Множества процессоров и работ
2. Привязка
3. Порядок

Привязка - всюду определенная на множестве работ функция, которая задает распределение работ по процессорам
Порядок задает ограничения на последовательность выполнения работ и является отношением частичного порядка, удовлетворяющим условиям ацикличности и транзитивности. Отношение порядка на множестве работ, распределенных на один процессор, является отношением полного порядка.

Графическая форма представления расписания



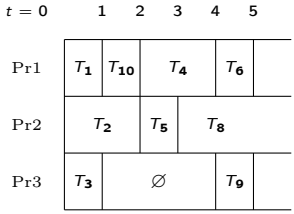
Графическая форма представления расписания

Графическая форма представления расписания $\Leftrightarrow$ Временная диаграмма

Постановка задачи

Требуется:

1. Построить расписание HP, то есть для i-й работы определить время начала ее выполнения s_i и процессор p_i на которм она будет выполняться
2. Минимизируемый критерий: время завершения выполнения расписания



Представление расписания в виде временной диаграммы

Модель расписания

Множество корректных расписаний HP задается набором ограничений:

- В расписании не допустимы прерывания
- Интервалы выполнения заявок не пересекаются
- Каждая работа назначена на процессор
- Любую работу обслуживает один процессор
- Частичный порядок, заданный графом зависимостей G, сохранен в $HP : G \subset G^T_{HP}$, где G^T_{HP} - транзитивное замыкание отношения G_{HP}

Постановки задачи

1. Задача с однородными процессорами (длительность выполнения работы не зависит от того, на каком процессоре она выполняется) и дополнительными ограничениями на количество передач:
   - $CR = \frac{m_{ip}}{m}$, где m_{ip} - количество передач данных между работами на каждый процессор
   - $CR2 = \frac{m_{2edg}}{m}$, где m_{2edg} - количество дуг, начальный и конечный узлы которых назначены на процессоры, не соединенных напрямую
2. Задача с однородными процессорами и дополнительным ограничением сбалансированности распределения работ:
   - $BF = \left(\frac{a_{max} \cdot p}{n}\right) - 1$, где a_{max} - наибольшее, по всем процессорам, количество работ на процессоре
3. Задача с неоднородными процессорами, но без дополнительных ограничений на расписание

Обзор существующих алгоритмов

Название алгоритма	Рандомизированность	Итерационный	Возможность масштабирования
Генетические алгоритмы	Рандомный	Итерационный	+/-
Алгоритм имитации отжига	Рандомный	Итерационный	+
Муравьиные алгоритмы	Рандомный	Итерационный	-
Жадные стратегии и ограниченный перебор	Детерминированный	Конструктивный	+

Дополнительные обозначения

1. $D = (d_1, d_2, \ldots, d_l)$, где l - количество вершин, доступных для добавления.
2. K - вектор длин критических путей от "головной"вершины до каждой вершины графа.
3. (s_i, p_i) - достаточное количество информации для размещения работы в расписании. Установка соотношения между работой t и парой (s_i, p_i) и есть построение расписания

Жадные критерии

1. $GR1$ - критерий, используемый в выборе работы на постановку
2. $GR2$ - критерий, используемый в выборе места постановки работы

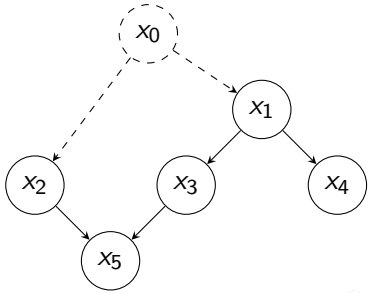
Процедуры ограниченного перебора

1. $H1$ - процедура перебора для создания места для постановки работы
2. $H2$ - процедура перебора для приближения времени старта работы к длине критического пути до нее

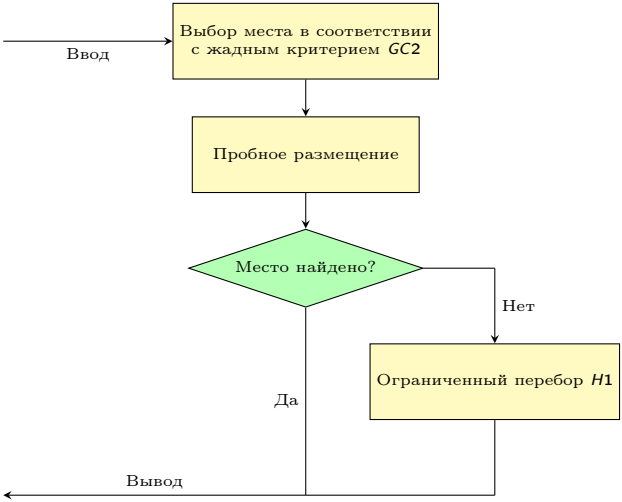
Предподсчет

1. Формируется множество D
2. Вычисляется вектор K. В случае, если такой вершины нет - создается фиктивная вершина с нулевой длительностью. Вектор k заполняется при помощи алгоритма Дейкстры.

Фиктивная вершина

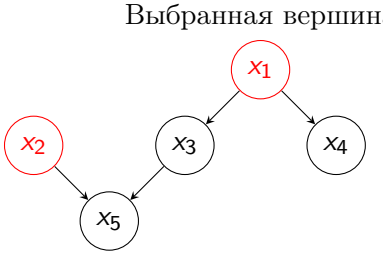


Блок-схема пробного размещения работы



Жадный критерий выбора размещения

Из множества D выбирается работу по критерию $GC1$ максимальности количества потомков у вершины.



Пробное размещение работы

Пробное размещение работы производится с учетом жадного и дополнительных критериев.
Жадный критерий $GC2$ - скорейшее завершение работы в расписании.
Способы выбора места:

1. Подсчет усредненного взвешенного показателя среди критериев

$$crit_{CR} = C_1 \cdot GC2 + C_2 \cdot CR + C_3 \cdot CR2$$
$$crit_{BF} = C_1 \cdot GC2 + C_2 \cdot BF$$

   где C_1, C_2 и C_3 - параметры алгоритма. Работа размещается на место с наибольшим значением параметра $crit$.

2. Допускная система выбора

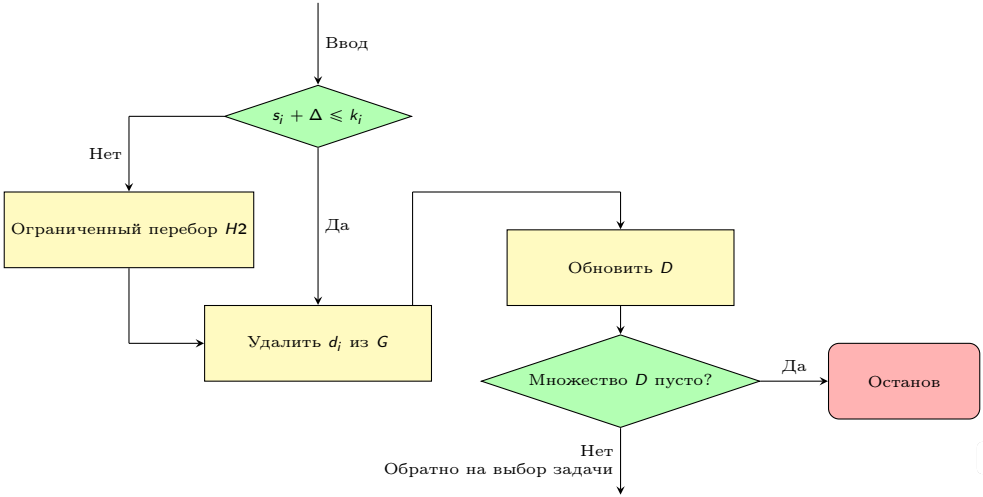
Допускная система выбора

1. Список мест размещения работ ранжируется по $GC2$, после чего отсекаются верхние $n\%$ работ, где n - параметр алгоритма
2. Такие же действия повторяются для каждого дополнительного критерия
3. В конечном списке выбрать место по жадному критерию

Процедура ограниченного перебора

1. После неудачной пробной постановки работы в расписание алгоритм создает набор $T = (t_1, t_2, \ldots, t_s)$, состоящий из s последних добавленных работ (s – параметр алгоритма).
2. Процедурой полного перебора пробуются различные расписания до тех пор, пока не получится расписание, удовлетворяющее заданным критериям.

Блок схема корректировки расписания



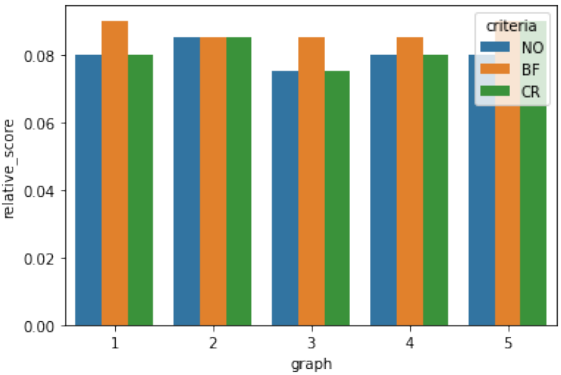
Программная реализация

Алгоритм реализован на языке C++ с помощью фреймворка boost.
Проект обладает следующей структурой:

1. logging - функции настройки логирования для проекта. Реализовано на основе Boost::log
2. schedule - модуль для работы с графом входных данных и матрицами C и D, подаваемыми на вход. Реализован на основе Boost::graph и Boost::uBLAS.
3. time_schedule - модуль для работы с временной диаграммой.
4. main.cpp - main() программы. Основной алгоритм реализован тут. Разбор аргументов основан на Boost::program_options.
5. Doxyfile - файл с настройками Doxygen.

Для сборки проекта используется CMake.

Точность полученного расписания

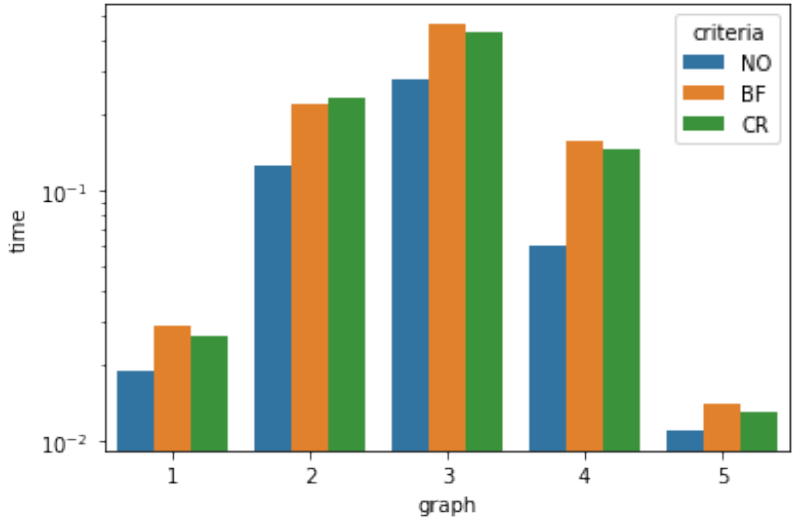


Наборы исходных данных, используемых в тестировании

Граф	Вершин	Процессоров	Передач
1	126	4	716
2	417	8	2367
3	408	8	8763
4	296	8	395
5	93	4	92

$$relative\_score = \frac{\text{время полученного расписания}}{\text{время оптимального расписания}} - 1$$

Время выполнения программы



Текущие результаты

Реализовано:

1. Проведен обзор алгоритмов построения списочных расписаний. Цель обзора: выявление жадных критериев и схем ограниченного перебора, которые могут быть модифицированы для решения данной задачи.
2. Разработан и ревлизован алгоритм, основанный на сочетании жадных стратегий и ограниченного перебора.
3. Подобраны оптимальные параметры алгоритма.
4. Проведено исследование свойств алгоритма, которое показало что алгоритм генерирует расписание, превосходящее оптимальное на 8%. Более 90% задач размещены при помощи жадной стратегии.