Формальная постановка задачи оптимизации расписания с использованием алгоритма имитации отжига

Автор: Кирнев Юрий

Формальная постановка задачи

Дано:

- N – количество работ;
- $J = \{j_1, j_2, \ldots, j_N\}$ – множество работ;
- $\tau = \{t_1, t_2, \ldots, t_N\}$ – множество времен выполнения соответствующих заданий j_i, $\forall i \in \overline{1, N}\ t_i > 0$;
- M – количество процессоров;
- $P = \{p_1, p_2, \ldots, p_M\}$ – множество процессоров, на которых выполняются работы.

Расписание:

Расписанием является булева матрица $H_P \in B^{N \times M}$, в которой $h_{pij} \in \{0, 1\}$, где $i \in \overline{1, N}$, $j \in \overline{1, M}$. Значение $h_{pij} = 1$ означает, что работа с номером i выполняется на процессоре с номером j, $h_{pij} = 0$ – что работа с номером i не выполняется на процессоре с номером j. Кроме того, для каждого процессора p_j определяется упорядоченное множество G_j индексов работ, выполняемых на нём в последовательном порядке.

Требуется:

Построить расписание $H_P^{N \times M}$ с порядками G_j, при котором будет минимизирован критерий К1, при этом все задания J будут выполнены на множестве процессоров P без прерываний, с учетом ограниченных ресурсов, и не будет пересечений в использовании процессоров, т.е.

$\forall i \in \overline{1, N}\ \exists! j \in \overline{1, M} : h_{pij} = 1 \Leftrightarrow$

$$\begin{cases} \sum_{i=1}^{N} \sum_{j=1}^{M} h_{pij} = N \\ \forall i \in \overline{1,N} \ \sum_{j=1}^{M} h_{pij} = 1 \end{cases}$$

Минимизируемый критерий:

В зависимости от остатка от деления на 2 контрольной суммы CRC32 от фамилии и инициалов выбирается один из следующих критериев:

- Критерий К1 (разбалансированность расписания)
- Критерий К2 (суммарное время ожидания)

CRC32 = 34370795, следовательно, выбираем критерий К1 для реализации.

Критерий разбалансированности расписания:

Для каждой работы i определяется время завершения C_i в расписании: если $i \in G_j$, то $C_i = \sum_{k \in G_j, \text{порядок } k < \text{порядок } i} t_k + t_i$.

Тогда
$T_{\max} = \max_{i \in \overline{1,N}} C_i$
$T_{\min} = \min_{i \in \overline{1,N}} C_i$
$K1 = T_{\max} - T_{\min}$

Вычислительная методология

Для расчёта К1:

1. Назначить работы процессорам и упорядочьте их в каждом G_j.
2. Для каждого G_j рассчитать кумулятивные завершения: C для k-й работы в G_j = сумма t первых k работ.
3. Собрать все C_i и найти max/min.
4. К1 = разница.

Иллюстративный пример

Рассмотрим четыре работы: A (длительность 3), B (2), G (3), V (8). - Процессор 0: A затем B (завершения: A в 3, B в 5). - Процессор 1: G затем V (завершения: G в 3, V в 11). - Завершения: 3, 5, 3, 11 $\rightarrow T_{\min} = 3$, $T_{\max} = 11$, K1 $= 8$.

Если переупорядочить (B затем A на 0: B в 2, A в 5; V затем G на 1: V в 8, G в 11), завершения: 2, 5, 11, 8 $\rightarrow T_{\min} = 2$, $T_{\max} = 11$, K1 $= 9$.

Значение и применения

K1 способствует справедливости, уменьшая время между первым и последним завершением, полезно в системах с синхронизацией барьеров.

Ограничения

1. Каждый процессор $p_j \in P$ в любой момент времени может выполнять не более одного задания.

2. Во время выполнения задания процессором не возникает прерываний.

3. Процессор может мгновенно (без прерывания) переключаться между заданиями.

4. Время выполнения $t_i \in \tau$ фиксировано.

Результаты исследований последовательного алгоритма

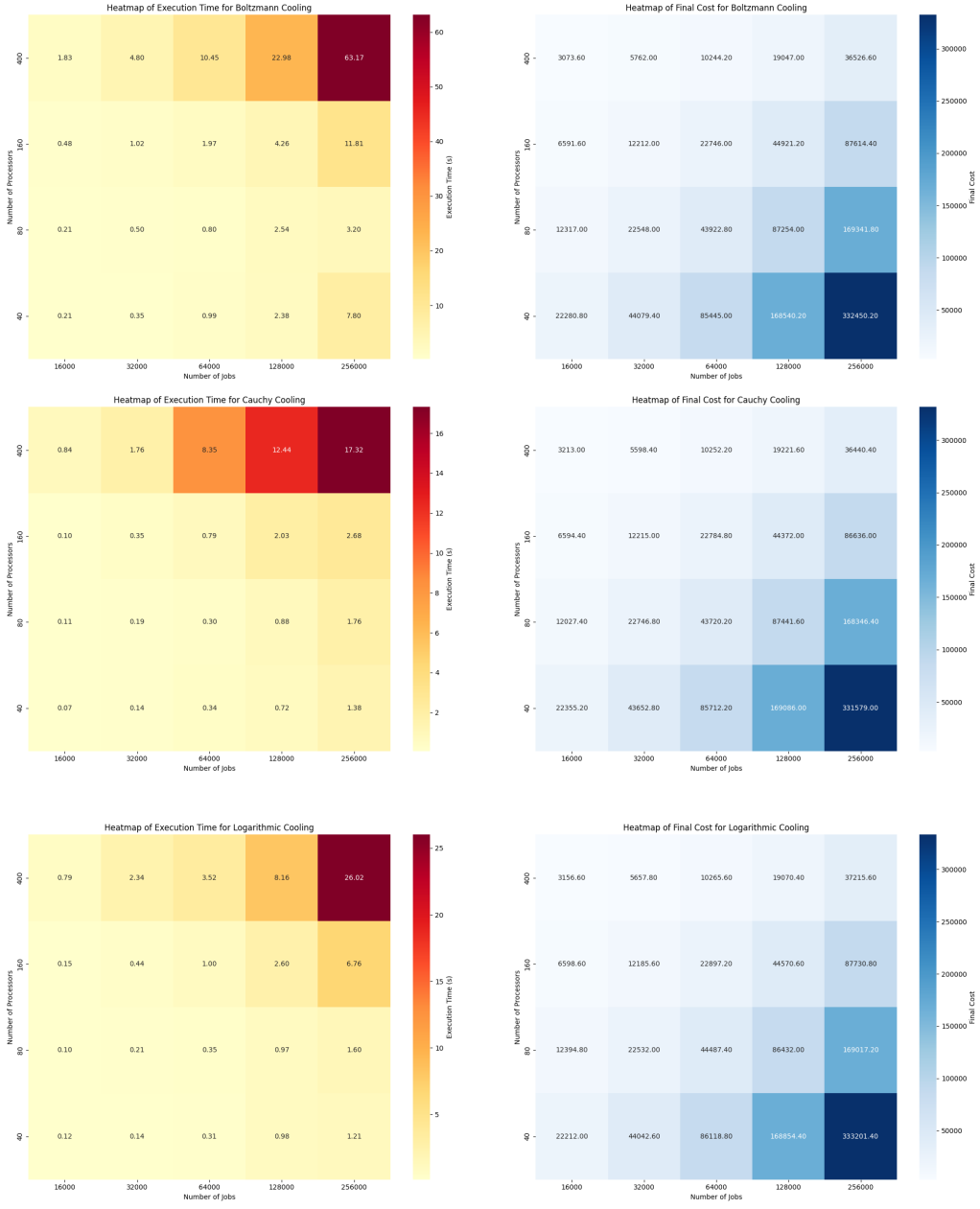

Рис. 1: Тепловые карты времени выполнения и финальной стоимости для различных законов охлаждения.

Тепловые карты представляют собой средние значения, полученные в результате пяти запусков последовательной программы. На основании проведенных исследований можно сделать вывод, что алгоритм понижения температуры на основе модели Больцмана демонстрировал самое длительное время выполнения по сравнению с другими алгоритмами (так как этот закон понижения температуры самый медленный). Тем не менее, это не отразилось на точности вычислений. Все алгоритмы показали сопоставимые результаты.

Результаты исследований параллельного алгоритма

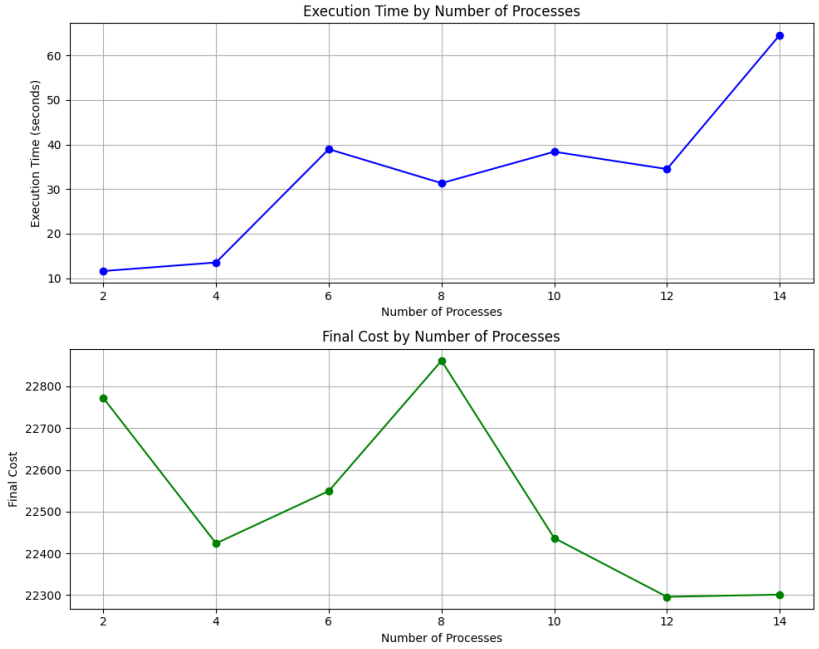

Рис. 2: График зависимости времени работы и стоимости от числа потоков.

Исследование проводилось на компьютере с 8 ядрами процессора. На графике представлены средние значения метрик за 5 запусков. Были взяты данные из 16000 задач и 40 процессоров. На основании полученных

результатов можно сделать вывод, что параллельный алгоритм работает гораздо дольше последовательного, однако имеет примерно такое же качество. Время работы алгоритма растёт с ростом количества нитей.