Генетический алгоритм для задачи планирования работ на многоядерном процессоре с учетом их взаимного влияния

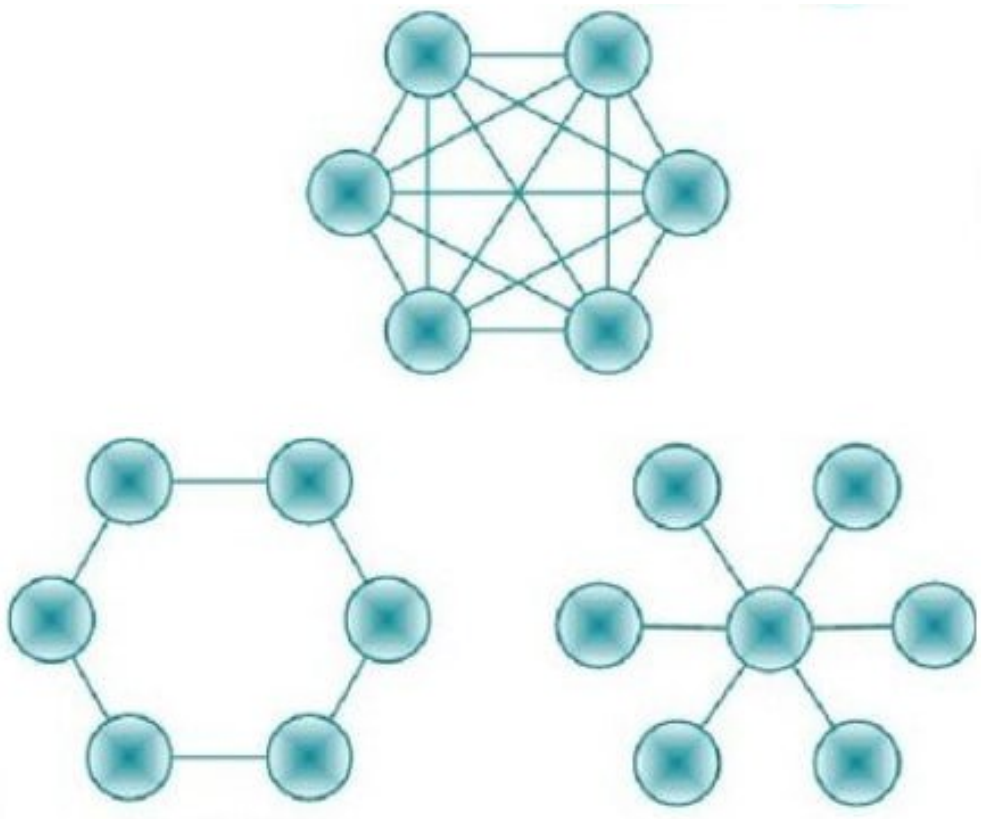
М.Ю. Сахно[1], Ю.В. Захарова[1]

[1] Омский филиал Института математики им. С.Л. Соболева СО РАН

Исследование выполнено за счет гранта Российского научного фонда № 22-71-10015, https://rscf.ru/project/22-71-10015.

Ресурсы: шина данных

Шина данных — это канал передачи информации между различными компонентами вычислительной системы. Она обеспечивает транспортировку данных между центральным процессором, оперативной памятью, устройствами ввода-вывода и другими элементами системы.



Постановка задачи

- Имеется n работ, m ядер процессора.

Постановка задачи

- Имеется n работ, m ядер процессора.
- Для каждой работы j, $j = 1, ..., n$, известно количество единиц времени s_j необходимое ей для выполнения в идеальных условиях.

Постановка задачи

- Имеется n работ, m ядер процессора.
- Для каждой работы j, $j = 1, ..., n$, известно количество единиц времени s_j необходимое ей для выполнения в идеальных условиях.
- Для каждой работы j задан процент потребления шины данных b_j при выполнении в идеальных условиях. Также задано некоторое правило, согласно которому можно рассчитать замедления работ при совместном выполнении.

Постановка задачи

- Имеется n работ, m ядер процессора.
- Для каждой работы j, $j = 1, ..., n$, известно количество единиц времени s_j необходимое ей для выполнения в идеальных условиях.
- Для каждой работы j задан процент потребления шины данных b_j при выполнении в идеальных условиях. Также задано некоторое правило, согласно которому можно рассчитать замедления работ при совместном выполнении.
- Необходимо составить расписание выполнения работ на ядрах процессора с минимальным общим временем завершения.

Предыдущие исследования

- Быстрые онлайн эвристики (Merkel et al, 2010; Zhuravlev et al, 2010).
- Известна работа, в которой предполагается, что количество ресурсов, выделяемых каждой работе, определяется планировщиком (Jozefowska, Weglarz, 2004).
- Также в литературе рассматривается случай, когда ресурсы назначаются ядрам, а распределение работ по ядрам процессора и порядок выполнения работ фиксированы (Althaus et al 2018).
- Известны и другие модели и методы для случая, когда *прерывания* допустимы (Shioura et al, 2018).
- Известен адаптивный генетический алгоритм с оптимальной рекомбинацией для задач с учетом энергии (Zakharova, Sakhno, 2024)
- Просчитываются все возможные конфигурации работ (Eremeev at el, 2020)

Алгоритм 1 Правило расчета замедления работы j в конфигурации c

Шаг 1: Положим процент свободной шины данных $freePercent := 100\%$ и текущее количество работ $jobsCount$ равно количеству работ в конфигурации c.

Шаг 2: Пока $jobsCount$ не равно 0, выполнять:

2.1: Вычислить процент шины данных, который может быть выделен каждой работе: $percent := freePercent/jobsCount$

2.2: Если в конфигурации c присутствует такая работа j, для которой $b_j \leq percent$, положить:

$z_{jc} := b_j$,
$freePercent := freePercent - b_j$,
$jobsCount := jobsCount - 1$.

Шаг 3: Скорость выполнения каждой работы j в конфигурации c пропорциональна отношению z_{jc}/b_j.

Алгоритм 2 Жадный алгоритм для задачи планирования работ на многоядерном процессоре с учетом их взаимного влияния.

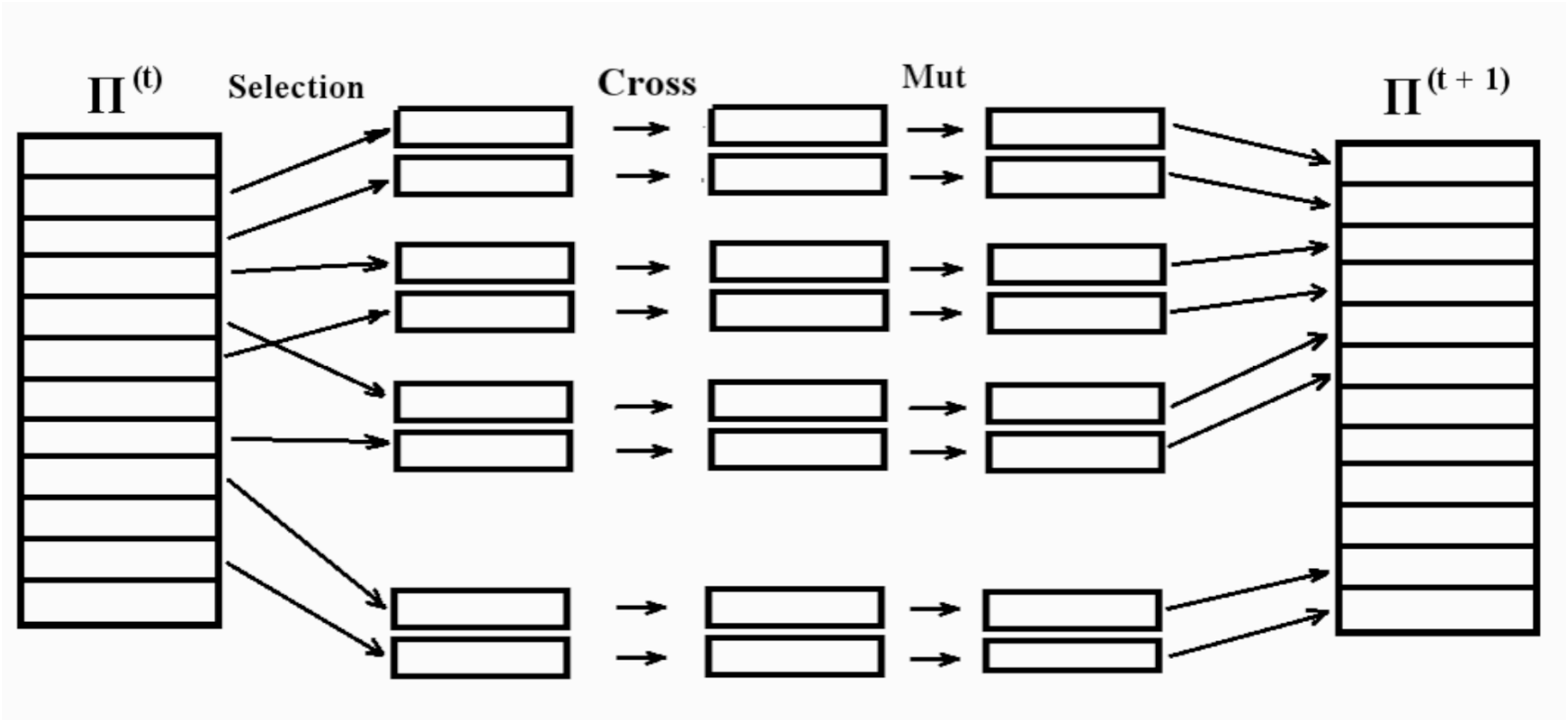
Пока присутствуют незавершенные работы, повторять:

1: Выбрать конфигурацию для выполнения.

2: Вычислить скорости выполнения работ в выбранной конфигурации.

3: Определить, какая из работ завершится первой.

Схема генетического алгоритма



Реализация генетического алгоритма

- Решения кодируются как перестановки на множестве работ.

Реализация генетического алгоритма

- Решения кодируются как перестановки на множестве работ.
- Условие завершения работы: достижение заданного количество вычислений значений целевой функции n_{max} или отведенного времени работы.

Реализация генетического алгоритма

- Решения кодируются как перестановки на множестве работ.
- Условие завершения работы: достижение заданного количество вычислений значений целевой функции n_{max} или отведенного времени работы.
- Операторы кроссинговера: 1PX, CX, OX, PMX.

Реализация генетического алгоритма

- Решения кодируются как перестановки на множестве работ.
- Условие завершения работы: достижение заданного количество вычислений значений целевой функции n_{max} или отведенного времени работы.
- Операторы кроссинговера: 1PX, CX, OX, PMX.
- Операторы мутации: сдвига, обмена.

Реализация генетического алгоритма

- Решения кодируются как перестановки на множестве работ.
- Условие завершения работы: достижение заданного количество вычислений значений целевой функции n_{max} или отведенного времени работы.
- Операторы кроссинговера: 1PX, CX, OX, PMX.
- Операторы мутации: сдвига, обмена.
- Операторы селекции: ранговая и турнирная.

Реализация генетического алгоритма

- Решения кодируются как перестановки на множестве работ.
- Условие завершения работы: достижение заданного количество вычислений значений целевой функции n_{max} или отведенного времени работы.
- Операторы кроссинговера: 1PX, CX, OX, PMX.
- Операторы мутации: сдвига, обмена.
- Операторы селекции: ранговая и турнирная.
- Для автоматической настройки параметров используется пакет IRACE.

Теорема 1. Предложенный генетический алгоритм в среднем не более чем за $O\left(n\left(\frac{k^2n^2}{P_m(1-P_c)}\right)^n\right)$ итераций впервые достигает оптимум, если в качестве оператора селекции выбрана ранговая селекция.

Теорема 2. Предложенный генетический алгоритм в среднем не более чем за $O\left(n\left(\frac{k^sn^2}{P_m(1-P_c)}\right)^n\right)$ итераций впервые достигает оптимум, если в качестве оператора селекции выбрана s-турнирная селекция.

Алгоритм 3 Адаптивная схема выбора оператора кроссинговера

1: Выбрать оператор кроссинговера a. Вероятность выбора каждого оператора пропорциональна его весу.

2: Применить оператора кроссинговера a к родительским особям и вычислить его оценку:

$$\phi_a = \begin{cases} w_1, & \text{если найденное решение является новым лучшим найденным решением (среди всех перезапусков),} \\ w_2, & \text{если найденное решение лучше, чем лучшее найденное решение с момента последнего перезапуска,} \\ w_3, & \text{если найденное решение лучше как минимум одного из родителей,} \\ 0 & \text{иначе.} \end{cases}$$

3: Вес оператора a обновляется следующим образом:

$$\rho_a := \lambda\rho_a + (1-\lambda)\phi_a.$$

Тестовые примеры

- В качестве работ использовались процедуры из библиотеки Intel MKL.
- В реальном эксперименте замерялось время выполнения и процент потребления шины данных каждой работы в идеальных условиях.

Значения параметров, найденных пакетом IRACE

Значение	Описание
202	размер популяции
0	количество элит
ранговая	оператор селекции
0.68	вероятность применения оператора кроссинговера
0.28	вероятность применения оператора мутации
shift	оператор мутации

Результаты эксперимента

$b_j \in \{4, 7, 8, 9, 10, 11\}$

	min	avg	max
50 работ, 8 ядер	1.97%	6.1%	12.97%
100 работ, 24 ядра	13.77%	19.44%	27.83%

Таблица: Минимальное, среднее и максимальное отклонение решений жадного алгоритма от решений генетического алгоритма

	min	avg	max
50 работ, 8 ядер	0.13%	0.18%	0.26%
100 работ, 24 ядра	45.03%	53.17%	61.83%

Таблица: Минимальное, среднее и максимальное отклонение решений генетического алгоритма от нижней границы

Нижняя граница вычисляется как $max_{i \in \mathcal{J}}(s_i, \frac{\sum_{j \in \mathcal{J}} s_j}{m})$

Заключение

- Предложен генетический алгоритм для задачи планирования работ на многоядерном процессоре с учетом их взаимного влияния.
- Доказана сходимость и определена оценка среднего числа итераций предложенного генетического алгоритма до первого достижения оптимума.
- Проведен вычислительный эксперимент, результаты которого показали преимущество предложенного генетического алгоритма над уже известным жадным алгоритмом.

Спасибо за внимание!