УДК 519.854.2

РОЕВОЙ АЛГОРИТМ ПЛАНИРОВАНИЯ ЗАДАНИЙ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ

А.М. Грузликов

АО «Концерн «ЦНИИ «Электроприбор»

Россия, 197046, Санкт-Петербург, Малая Посадская ул., 30

E-mail: agruzlikov@yandex.ru

Ключевые слова: теория расписаний, поточное планирование, роевой алгоритм, распределенные системы, система реального времени.

Аннотация: В работе рассмотрена задача поточного планирования заданий в распределенных вычислительных системах реального времени. Доказано, что задача является NP-трудной, поэтому на практике применяются различные эвристические и метаэвристические алгоритмы. В работе предлагается решение с использованием алгоритма роя частиц для задачи дискретной оптимизации. Рассматривается расширение алгоритма путем выбора начального размещения частиц с использованием набора вычислительно простых методов. Приводится математическая постановка задачи планирования. Исследуется оценка эффективности решения задачи по критерию минимума выполнениях всех заданий для различных архитектур систем - при отсутствии и при наличии ограничений на обмен между машинами.

1. Введение

Проблема планирования возникает на практике в различных прикладных областях, а именно, при планировании производства, составлении расписаний движения транспорта, планировании вычислений и многих других. В современной научной литературе проблеме планирования уделяется большое внимание. При этом разнообразие рассматриваемых задач весьма велико [1, 2]. При решении проблемы планирования производства, можно выделить вопросы планирования для поточных линий (англ. permutation flow shop, PFS), для которых характерна одинаковая последовательность выполнения заданий на каждом из этапов обработки [2], с этим направлением тесно связаны вопросы планирования вычислений в распределенных системах реального времени [3, 4].

В классической постановке проблемы PFS рассматривается архитектура системы, которая представляет собой конвейер с линейным графом информационных связей, в настоящем исследовании это граф – ациклический, что соответствует более широкому классу организации распределенных систем используемых на практике. Дополнительной особенностью обсуждаемого ниже материала является то, что постановка задачи предполагает отсутствие или наличие одного из двух вариантов ограничений: первое ограничение, отсутствие у машины буферов для

хранения промежуточных результатов обработки; второе ограничение, обработка без задержки на этапах передачи от одного этапа обработки, следующему. На практике примерами таких систем, является бортовая система технического зрения, с ограничением по памяти для обмена видеокадра; прокатный стан, где не допускается простой (охлаждение) обрабатываемого элемента.

Доказано, что проблема PFS является NP-трудной [1], поэтому для ее решения используются различные эвристики и метаэвристики. Под метаэвристиками принято понимать общие схемы построения эвристических алгоритмов для решения различных классов задач, например для задач упаковки, коммивояжера, составление расписаний и т.д. Примером метаэвристики является предложенный в 1995 году Дж. Кеннеди и Р. Эрберхартом алгоритм оптимизации непрерывных нелинейных функций – роевой алгоритм (англ. Particle Swarm Optimization) (PSO). Поскольку предложенный метод PSO оперирует движением частиц в непрерывном пространстве целевой функции, для задач дискретной оптимизации метод напрямую не применим ввиду неопределенности понятия вектора скорости движения частиц в дискретном пространстве.

В докладе исследуется PSO алгоритм применительно к проблеме дискретной оптимизации. Проведено исследование при начальном размещении частиц с использованием вычислительно простого алгоритма. Получены оценки эффективности на эталонном наборе тестовых заданий, у которых выполнение операций на машинах задается как простой последовательностью, т.е. для архитектур конвейерного типа, так и на заданиях, назначенных на архитектуру типа ациклический граф. Каждый тип архитектуры исследовался при отсутствии и при наличии ограничений.

2. Постановка проблемы поточного планирования

Опишем рассматриваемую далее постановку проблемы PFS-планирования для распределенных вычислительных систем при отсутствии и при наличии ограничений.

Введем следующие обозначения: $\tau = \{\tau^{(j)} | j = \overline{1,n}\}$ – множество заданий; $M = \{M_i | i = \overline{1,m}\}$ – множество машин. Считаем, что задания τ выполняются на множестве машин с сохранением последовательности, т.е. порядок выполнения заданий на машине 1 совпадает с порядком выполнения заданий на машине 2 и т.д. Это означает, что задана матрица $R_{i,k} \in \{0,1\}$, которая задает отношение предшествования задач, при этом 1 – задачи должны быть предварительно завершены на i-машине до запуска задач на k-машине, 0 – иначе; очевидно, что $R_{i,k} \cdot R_{k,i} \neq 1$.

Пусть $e_i^{(j)}$ – длительность выполнения задачи j-го задания на i-й машине, значение которого известно точно. С практической точки зрения это означает использование средних значений или верхних границы этих длительностей.

Пусть $\pi = (\pi_1, \pi_2, \cdots, \pi_n)$ – перестановка, которая определяет порядок выполнения заданий τ на множестве машин M; $C(\pi_{[j]}, k)$ - время завершения j-го задания на k-й машине при последовательности выполнения π.

Тогда для первых машин (т.е. для таких машин k, для которых $R_{i,k} = 0$ при

любых i , обозначим удовлетваряющие данному условию машины как 1^*):

$$C(\pi_{[1]}, 1^*) = e_{1^*}^{(\pi_{[1]})};$$
$$C(\pi_{[j]}, 1^*) = C(\pi_{[j-1]}, 1^*) + e_{1^*}^{(\pi_{[j]})}, j = \overline{2, n},$$

для последующих машин (т.е. для таких машин i, k, для которых $R_{i,k} = 1$):

$$C(\pi_{[1]}, k) = \max_{\forall i: R_{i,k}=1} (C(\pi_{[1]}, i)) + e_k^{(\pi_{[1]})};$$
$$C(\pi_{[j]}, k) = \max_{\forall i: R_{i,k}=1} (C(\pi_{[j-1]}, k), C(\pi_{[j]}, i)) + e_k^{(\pi_{[j]})}, j = \overline{2, n},$$

и время завершения всех заданий: $C_{max}(\pi) = \max_{j,k}(C(\pi_{[j]}, k), j = \overline{1, n}, k = \overline{1, m}$.

Цель планирования состоит в поиске такой перестановки π^* из множества Π всех возможных перетановок заданий τ, при которых: $C_{max}(\pi^*) \leqslant C_{max}(\pi), \forall \pi \in \Pi$..

3. Метод роя частиц

Роевой алгоритм. Алгоритм PSO был изначально разработан для моделирования социального поведения и основа на поведении стаи птиц. Модель, описывающая принятие решений частицами оказаласть простым методом оптимизации, который допускает эффективное распараллеливание при поиске решения. Рой состоит из определенного количества частиц. Индивидуальная память о прошлом опыте сохраняет знания о том, где в поисковой области он проявил себя лучше всего. На каждой итерации все частицы перемещаются в пространстве целевой функции в соответствии с правилами взаимодействия между частицами с учетом индивидуального опыта и опыта группы, чтобы найти глобальный оптимум. Скорость и положение i-частицы на $k + 1$ шаге определяется как:

$$v_{k+1}^i = w \cdot v_k^i + c_1 \cdot r_1 \cdot (p_k^i - x_k^i) + c_2 \cdot r_2 \cdot (p_k^g - x_k^i)$$
$$x_{k+1}^i = x_k^i + v_{k+1}^i,$$

где v, x – скорость и положение частицы; $r_1, r_2 \in [0, 1]$ – случайные значения, определяющие вектор случайного поиска; $c_1, c_2 \in \mathbb{R}$ – настроечные коэффициенты когнитивного (индивидуального) и социального (группового) поведения частицы; w – коэфициент инерции; p_k^i – локальное наилучшее положение частицы и p_k^g – лучшая позиция по информации всех частиц за все время решения задачи.

Метод роя частиц для задачи дискретной оптимизации. Пусть X – множество всех возможных положений частиц в области определения целевой функции, тогда $\forall x^i \in X$, $x^i = (x^i(1), x^i(2), \cdots, x^i(n))$ – которая соответствует некоторой перестановке $\pi \in \Pi$.

Определим функцию расстояния $h(x^i, x^j)$, которая определяет число перестановок элементов для перемещения из точки x^i в точку x^j. Тогда, для $\forall x^i \in X$ можно определить множество «соседей», которое находится на расстоянии не более чем s перестановок от x^i: $N(x^i, s) = \{x^j | x^j \in X, h_{i,j} \leqslant s\}$.

Тогда, положение частицы для задачи дискретной оптимизации можно определить как $x_{k+1}^i = x_k^i \otimes SS$, где $SS = (SO_1(x_k^i, p_k^i), SO_2(x_k^i, p_k^g))$ – определяет применение перестановки для положения частицы x_k^i в соответствии с

индивидуальным и групповым опытом: $SO_1, SO_2 : x^i \to x^j$, $x^j \in N(x_i, s)$ – оператор изменения текущего положения частицы x^i в заданном направлении с учетом заданной дистанции s.

Настройка роевого алгоритма. Эффективность алгоритма PSO зависит от различных исходных и настроечных параметров, включая число частиц, число итераций, варианты топологии общения между частицами для обмена опытом, коэффициенты инерции, когнитивного и социального поведения. Одним из исходных параметров алгоритма является начально размещение частиц.

Для повышения эффективности планирования, алгоритм начального размещения частиц должен удовлетворять следующим условиям: первое, сложность алгоритма размещения должна быть не выше полиномиальной; второе, позволять генерировать различные варианты перестановок (частицы не должны группироваться в одной окрестности). Одним из алгоритмов, который удовлетворяет заданным условиям, является алгоритм РКС (алгоритм разрешимых классов систем) [3].

Алгоритм разрешимых классов. Приведем краткое описание теоретических основ предлагаемого алгоритма начального размещения частиц – РКС-алгоритма. Этот алгоритм основан на использовании понятия разрешимого класса системы. Важным следствием принадлежности системы к разрешимому классу является существование для нее оптимального алгоритма для задачи PFS полиномиальной сложности [3].

В рассматриваемой распределенной системе, каждая входная машина связаны с выходной машиной хотя бы одним путем. Назовем путь критическим, если время его выполнения является максимальным среди всех основных путей системы.

Для определения разрешимых классов предварительно введем на множестве машин отношение доминирования.

Определение 1. *Машина M_q доминирует над машиной M_r, если:*

$$M_q \geqslant M_r \leftrightarrow \min_j e_q^{(j)} \geqslant \max_j e_r^{(j)}, j = \overline{1,n}.$$

Общее свойство рассматриваемых ниже разрешимых классов состоит в следующем: для любого задания, критический путь единственен и проходит по одним и тем же машинам: $M_{[1]}, M_{[2]}, \cdots, M_{[m^*]}$, где $[i]$ – порядковый номер машины вдоль пути и m^* – длина данного пути.

Определение 2. ***(класс 1–4).*** *Задача PFS-планирования $\{M, \tau\}$ является:*

- *системой класса 1, если множество машин критического пути представляет собой последовательность: $M_{[1]} \geqslant M_{[2]} \geqslant \cdots \geqslant M_{[m^*]}$, убывающую по отношению доминирования;*
- *системой класса 2, если множество машин критического пути представляет собой последовательность: $M_{[1]} < M_{[2]} < \cdots < M_{[m^*]}$;*
- *системой класса 3, если множество машин критического пути представляет собой последовательность: $M_{[1]} < M_{[2]} < \cdots < M_{[h^*]} \geqslant M_{[m^*-1]} \geqslant M_{m^*}$ первая, из которых возрастает, а вторая убывает по отношению доминирования (h^* - номер машины стыковки двух последовательностей);*
- *системой класса 4, если множество машин критического пути представляет собой последовательность $M_{[1]} \geqslant M_{[2]} \geqslant \cdots \geqslant M_{[h^*]} < M_{[m^*-1]} < M_{m^*}$.*

Теорема 1. *Если задача PFS-планирования соответствует 1–3 классу, тогда существует полиномиальный алгоритм реализующий оптимальное планирование. В случае, если задача планирования соответствует 4-му классу, тогда существует полиномиальный алгоритм, реализующий приближенное решение.*

Поскольку для начального размещения частиц необходимо сформировать различные варианты планирования, воспользуемся неоднозначностью соотнесения задачи PFS $\{M, \tau\}$ с одним из классов, а именно, будем выбирать случайный класс 1–4 и выполнять планирование в соответствии с теоремой. В итоге, у нас будет 4^n вариантов начального размещения частиц.

Оценка эффективности. Выбор тестовых примеров для проверки NP-трудных задача, как правило, осуществляется субъективно. Традиционно для алгоритмов задачи PFS-планирования используются тесты, предложенные Тейлардом в 1989 г. для архитектур конвейерного типа [5]. Предложенный Тейлардом подход был обобщен на структуры более широкого типа - ациклические графы. Проведено исследование эффективности для систем при отсутствии и при наличии ограничений.

4. Заключение

По результату моделирования, показано, что предлагаемый роевой алгоритм с начальным размещением частиц с использованием алгоритма разрешенных классов в среднем проигрывает оценке оптимального планирования (на тестах Тейларда):

- 3-5% для систем без ограничений;
- 7-10% для систем с ограничением по памяти при обмене;
- 8-17% для систем с требованием по отсутствию задержек.

Для распределенных вычислительных систем заданных ациклическим графом:

- 3-9% для систем без ограничений;
- 5-19% для систем с ограничением по памяти при обмене;
- 7-24% для систем с требованием по отсутствию задержек.

Исследование выполнено за счет гранта Российского научного фонда № 22-29-00339, https://rscf.ru/project/22-29-00339.

Список литературы

1. Brucker P. Scheduling Algorithms. Springer, 2007.
2. Лазарев А.А. Теория расписаний. Методы и алгоритмы. М.: ИПУ РАН, 2019. 408 с.
3. Грузликов А.М., Колесов Н.В., Скородумов Ю.М., Толмачева М.В. Планирование заданий в распределенных системах реального времени // Известия РАН. Теория и системы управления. 2017. № 2. С. 67-76.
4. Cottet F.,Kaiser J., Mammeri Z. Scheduling in Real-Time Systems. John Wiley & Sons Ltd, 2002.
5. Taillard E. Benchmarks for Basic Scheduling Problems // Europ. J. Operational Research. 1993. Vol. 64, No. 2. P. 278–285.