УДК 004.03

Гибридный алгоритм распределения задач в вычислительной системе

Юрич М.Ю.
Запорожский национальный технический университет
mary@yurich.org.ua

Abstract

Yurich M.Yu. Hybrid algorithm for allocation of tasks in the computer system. *In this article a hybrid algorithm for the distribution of tasks in the computer system using the basic operators of the classical genetic algorithm (GA) is developed. The classical structure of the GA is modified with features the problem of effective distribution of tasks in the computer system.*

Введение

Задача распределения нагрузки в вычислительных системах, представленная модифицированной транспортной задачей (математической модель ее описана в [1]), является труднорешаемой задачей и относится к классу NP–полных задач. Как известно, не существует методики, позволяющей решать такие задачи за полиномиальное время.

Применение для решения такого класса задач метода полного перебора часто оправдано в силу отсутствия других эффективных способов решения конкретной задачи. В данный момент существуют различные методики оптимизации полного перебора, с помощью которых сложные переборные задачи сводятся к более простым, и получают новый, эффективный способ решения [2]. Однако, решение найти за полиномиальное время все равно остается невозможным. Полученное решение либо не является оптимальным, т.е. наилучшим, а не просто эффективным, либо может быть найдено только для задач малой размерности.

В данной статье предложен гибридный алгоритм, позволяющий решить задачу балансировки нагрузки с удовлетворяющим качеством, т.е. найти достаточно эффективное решение, оптимальное при распределении задач в вычислительной системе, но которое, все же, не во всех случаях достигает глобального минимума целевых функций поставленной задачи.

Подход к построению алгоритма

Для решения задачи, описанной в [1], разработан алгоритм, состоящий из двух частей:

1. Первая часть позволяет рассчитать необходимое количество задач для решения на первом компьютере и первоначальную суммарную длительность выполнения задач на каждом из компьютеров;
2. Вторая часть позволяет оптимизировать решение с учетом предложенных целевых функций поставленной задачи.

Первая часть алгоритма использует некоторый набор правил, в то время как вторая часть основана на применении генетического алгоритма. Таким образом, разработанный алгоритм является гибридом методики нахождения решения, основанной на принципах жадных алгоритмов, и эвристического алгоритма, являющегося разновидностью эволюционных вычислений, а именно генетического алгоритма.

При решении задачи предлагаемым алгоритмом в первой части необходимо следовать некоторым правилам, верным на любом этапе выполнения первой части алгоритма и при любом из вариантов следования алгоритма. Эти правила таковы:

1. Все операции проводятся, начиная от наиболее мощного компьютера к менее мощному компьютеру, в порядке убывания.
2. Элементы для использования выбираются, начиная от максимального к минимальному из существующих возможных вариантов.
3. При выборе элементов, составляющих в сумме необходимую величину либо стремящуюся к необходимой величине, следует выбирать значения таким образом, чтобы количество элементов было минимальным.

Для того чтобы оценить, на сколько верным является подход к описанному построению первой части алгоритма решения поставленной задачи, дадим некоторые разъяснения к указанным выше правилам:

1. Поскольку, целью задачи является минимизация, а не максимизация, некоторых функций, то проводить все действия необходимо, начиная с наиболее мощного

компьютера, обладающего большим быстродействием, а значит решающего одну и ту же задачу за время меньшее, чем более медленный компьютер;

2. Нельзя следовать от минимального элемента к максимальному элементу, т.к. тогда сложить результирующую сумму может оказаться невозможным, поскольку все минимальные элементы уже задействованы;

3. Третье правило логически вытекает из второго, поскольку, чем целостнее, не делимей, будет набрана сумма на предшествующем шаге, тем больше вариантов набрать эту же сумму останется на последующих шагах алгоритма, что обеспечивает увеличение вероятности распределить задачи между компьютерами более равномерно.

Вышеуказанные три правила обеспечивают следующее свойство: следуя этим правилам максимально быстро можно найти одну из целевых функций поставленной задачи:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \cdot c_{ij} \to \min \ . \qquad (1)$$

Как уже было отмечено ранее, весь алгоритм условно можно разбить на две части:

– достижения значения (1) – в первой части алгоритма;

– достижения значения (2) во второй части алгоритма, которое, в свою очередь, может изменить первоначально найденное значение (1).

$$\max\left(\sum_{i=1}^{m} x_{ij} \cdot c_{ij}\right) \to \min, \text{при } \mathrm{j} = \overline{1,\mathrm{n}} \ . \qquad (2)$$

Таким образом, находим целевую функцию поставленной задачи:

$$\begin{cases} max\left(\sum\limits_{i=1}^{m} x_{ij} \cdot c_{ij}\right) \to min, \quad j = \overline{1,n}, \\ \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} x_{ij} \cdot c_{ij} \to min \, . \end{cases} \qquad (3)$$

Алгоритм решения задачи

Что касается первой части алгоритма, она не требует особых объяснений и будет пошагово описана далее. Что же касается второй части алгоритма, то на этом стоит остановиться более подробно.

Вторая часть алгоритма построена на основе генетического алгоритма (ГА). Однако, представленная модификация ГА имеет ряд особенностей, связанных со спецификой задачи. Как известно, классический ГА имеет следующие этапы [3]:

1. Создание начальной популяции;
2. Определение (задание) функций приспособленности для особей популяции (оценивание);
3. Выбор индивидов из текущей популяции;
4. Скрещивание и/или мутация;
5. Вычисление функций приспособленности для всех особей;
6. Формирование нового поколения (селекция);
7. Если выполняются условия останова, то решение найдено, иначе возвращаемся к п.3.

Опишем каждый из этапов с точки зрения особенностей, проявляющихся в разработанном алгоритме:

1. Создание начальной популяции. Для этого случайным образом выбираем компьютер, который будет первым для проведения набора требуемой суммы, и выбираем задачи случайным образом для решения на нем, составляющие в сумме требуемое значение длительности выполнения на данном компьютере. Далее случайным образом выбираем следующий компьютер и проделываем для него аналогичные операции, далее следующий компьютер и так, пока не будут задействованы все компьютеры системы. Вся описанная в данном пункте последовательность выполняется m^d раз, обеспечивая, таким образом, m^d индивидов начальной популяции. Параметр d очень сильно влияет на точность полученных результатов и, в то же время, на вычислительную сложность алгоритма; m – количество задач, которые необходимо распределить в вычислительной системе;

2. Определение (задание) функций приспособленности для особей популяции (оценивание). Функция приспособленности в данном случае известна изначально из математической модели поставленной задачи распределения заданий (задач) в вычислительной системе и является также целевой функцией системы, которая рассчитывается только для тех случаев, когда все задачи распределены между компьютерами. Выражается она в виде системы уравнений (3);

3. Выбор индивидов из текущей популяции. В предложенном алгоритме на данном этапе выбирается только одно наилучшее решение, т.е. только один индивид с наилучшей функцией приспособленности.

3а. Затем не переходим к п.4, как это делается в классическом ГА, а выполняем корректировку суммы длительностей задач, распределенных на каждый из компьютеров, к которой стремились в п.1 и переходим к п.3б;

3б. Далее выполняем действия, описанные в п.1 – создаем новую популяцию, но сумма длительностей задач, выполняющихся на каждом из компьютеров, к которой будем стремиться, теперь будет равна значению, найденному в п.3а. Затем выполняем пп.2, 3, 3а,

3б до тех пор, пока такое решение возможно будет найти;

4. Скрещивание и/или мутация. Разделим на 2 части данный пункт: скрещивание и мутацию. Сначала рассмотрим скрещивание. Особенностью предложенного ГА является то, что скрещивание происходит не между разными индивидами популяции, а внутри самого индивида. Поскольку в рассматриваемой задаче речь идет об n компьютерах ($j=\overline{1,n}$) и m задачах ($i=\overline{1,m}$), которые необходимо распределить на эти компьютеры, то в данном случае скрещивание означает изменение номера выполняющего задачи компьютера для задач, «находящихся» между двумя точками кроссовера. Компьютеры, используемые при скрещивании, выбираются такие, у которых сумма длительностей распределенных на них задач минимальна и максимальна. Т.е. выбираем случайным образом две точки кроссовера и для i не равного нулю, меняем между собой значение j. Данную операцию выполняем всего m^d раз, проводя скрещивание каждый раз только для наилучшего полученного решения (т.е. обязательно после каждого раза скрещивания выполняем пункты 5 и 6);

4а. Под мутацией в данном случае понимаем (как и для скрещивания) изменение номера компьютера. Однако, этот номер меняем для задач, выбранных определенным образом (более подробно рассмотрим далее), начиная от наиболее мощного компьютера. Далее выполняем пп.5, 6, 7.

5. Вычисление функций приспособленности для всех особей. Данная операция производится аналогично описанной в п.2;

6. Формирование нового поколения (селекция). В данном случае в виде формирования нового поколения происходит выбор наиболее приспособленного одного индивида, выбранного по результатам предыдущих шагов алгоритма;

6а. Для выбранного в п.6 индивида по итогам п.4 проводится вторая часть п.4 классического ГА – мутация, т.е. переходим к п.4а;

7. Если выполняются условия останова, то решение найдено, иначе возвращаемся к п.3. В данном случае условия останова будут однозначно выполнены, поскольку ими будет служит найденное в процессе следования алгоритма лучшее решение среди наиболее приспособленного индивида до скрещивания, после скрещивания и после мутации. Следовательно, это и будет найденным решением всей задачи.

Таким образом, разработанный ГА значительно отличается от классического, за счет иначе построенной последовательности выполнения операторов классического ГА. Данная модификация позволяет уже на первоначальном этапе формирования выделить лучшее из лучших решение и далее только улучшать это значение за счет применения операторов классического ГА.

Теперь распишем весь разработанный гибридный алгоритм пошагово. Алгоритм состоит из следующих шагов.

1. Сортируем массив длительностей выполнения задач и массив коэффициентов мощностей компьютеров по возрастанию.

2. Рассчитываем количество задач, которое необходимо послать для выполнения на первом компьютере по формуле:

$$r1 = \frac{m}{\sum_{j=1}^{n} \frac{KMK_{\min}}{KMK_j}} * \frac{KMK_{\min}}{KMK_{\max}}, \qquad (4)$$

где m – количество задач, поступающих в систему для распределения между компьютерами;

KMK_j – коэффициент мощности компьютера j;

$KMK_{\min}$, $KMK_{\max}$ – коэффициенты мощности компьютеров с минимальной и максимальной мощностью соответственно.

KMK_j находим как отношение мощности компьютера, на котором была первоначально оценена длительность решения задачи i, и мощности компьютера j вычислительной системы, на который эта задача предположительно будет распределена:

$$KMK_j = \frac{M_i^o}{M_j}. \qquad (5)$$

Мощность компьютера для оценивания длительности решения всех i–тых задач, задается изначально для всех одинаковой.

При этом значение $r1$ округляется по стандартным законам математики.

3. Для нахождения суммы длительностей задач Sum, которые необходимо выполнить на первом компьютере, суммируем длительности $\frac{r1}{3}$ максимальных по длительности задач с $(r1-\frac{r1}{3})$ минимальных по длительности задач для первого компьютера. При этом, рассчитывая значение $\frac{r1}{3}$ берем целую часть от деления $r1$ на 3.

4. Теперь, придерживаясь трех вышеуказанных правил, «набираем» на каждом компьютере задачи, длительность которых в сумме будет снизу стремиться к числу Sum, т.е. будет меньше или равна значению Sum.

5. Считаем сумму длительностей задач, нераспределенных ни на один компьютер. Длительности считаются по первому компьютеру. Получаем значение k.

6. Если сумма в п.5 неравна нулю переходим к п.7, если равна – к п.15.

7. Считаем длительность по каждому из компьютеров, на которую превышено или недобрано значение Sum по следующей формуле:

$$Over_j = Sum_j - Sum . \quad (6)$$

8. Пересчитываем эти значения по первому компьютеру с учетом знака, в сумме получаем значение *f*:

$$f = \sum_{j=1}^{n} Over_j * \frac{KMK_{\max}}{KMK_j} \quad (7)$$

9. Суммируем значение, полученное в п.8, и значение, полученное в п.5, получаем значение $l = k + f$.

10. Проверяем, если значение в п.9 равно нулю, то значение суммы, к которой будем стремиться, увеличиваем на $\frac{k}{2}$ и переходим к п.13.

11. Рассчитываем коэффициент *kf* по формуле:

$$kf = \frac{\sum_{j=1}^{n} KMK_j}{KMK_{\max}} \quad (8)$$

12. Вычисляем откорректированное значение Sum1, к которому будет стремиться значение суммы длительностей на каждом из компьютеров системы. Для этого к значению Sum прибавляем значение, полученное в п.9. (учитывая знак), деленное на *kf*:

$$Sum1 = Sum + \frac{l}{kf} . \quad (9)$$

13. Сохраняем значение Sum1 как Sum, к которому будем стремиться далее.

14. Повторяем п. 4–13 до тех пор, пока сумма длительностей нераспределенных задач (считается в п.5) не будет равна нулю.

15. Проверяем возможность равномерного распределения задач. Если, в первом столбце осталась одна задача, а значение суммарной длительности выполнения задач хотя бы на одном из оставшихся (*n–1*) компьютеров соответствует одному из значений:

– на порядок меньше длительности задачи *m*, распределенной на компьютер 1;

– равно нулю,

то для дальнейшей правильной работы алгоритма с целью соблюдения условий оптимальности распределения задач «вычеркиваем» первый элемент из массива коэффициентов мощностей компьютеров и строку *m* из массива, содержащего в качестве элементов упорядоченные по возрастанию длительности решения задач. Если нет, то переходим к п. 17.

16. Далее выполняем пп. 2–15, но уже с количеством задач и количеством компьютеров равным на единицу меньшим, чем в сформировавшихся ранее массивах.

17. Случайным образом выбираем компьютер, для которого самого первого будем производить набор требуемой суммы Sum.

18. Случайным образом выбираем задачи, составляющие в сумме требуемое значение.

19. Далее выбираем случайным образом следующий компьютер и проделываем для него все операции описанные в п.18.

20. Проделываем п.19 до тех пор, пока не будут задействованы все компьютеры.

21. Генерируем таким образом (п.17–20) m^d решений.

22. Лучшее из полученных решений запоминаем. Лучшее выбирается только для случаев, когда все задачи распределены, основываясь на системе уравнений (3).

23. Проводим расчет корректирующего элемента *f* по формуле (7), с учетом (6).

24. Проводим корректировку суммы, к которой будем стремиться далее, используя запомненный в предыдущем пункте результат:

$$Sum1 = Sum + \frac{f}{kf} . \quad (10)$$

25. Сохраняем значение Sum1 как Sum, к которому будем стремиться далее.

26. Выполняем пп.17 – 25 до тех пор, пока в течение m^d раз генерации решений не будет возможным нахождение результата для п.22, т.е. не будет хотя бы одного варианта решения, когда все задачи распределены.

27. Для выбранного лучшего решения выбираем компьютер с наименьшим и наибольшим значением суммы длительностей выполнения задач.

28. Произвольным образом выбираем 2 точки скрещивания.

29. Между этими точками производим «скрещивание». В данном случае скрещивание, как это было более подробно описано выше, означает использование задействованных задач на каждом из компьютеров, но при этом меняется их хозяин (выполняющий их компьютер), т.е. для *i* не равного нулю, меняем между собой значение *j*.

30. Запоминаем лучшее решение из полученного в п.22 и решения, полученного

после скрещивания в п.29 (принцип выбора лучшего решения описан в п.22).

31. Если решение в п.29 лучше (новое значение лучше старого), то дальше выполняем пп. 27-29 для решения, полученного в п.29, если нет, то выполняем пп. 27-29 для решения, полученного в п.22.

32. Опять выполняем выбор наилучшего решения по принципу, описанному в п.22.

33. Выполняем генерацию точек скрещивания всего m^d раз.

34. Для выбранного лучшего варианта после этапа скрещивания производим мутацию. Для этого выбираем компьютер j с наименьшей суммой длительностей выполнения задач и добавляем к нему наиболее «быструю» i–ю задачу из ближайшего компьютера с индексом большим, чем у данного ($j+1$).

35. Если верно неравенство (11), то задачу добавляем, если нет, то переходим к следующему компьютеру ($j+2$) и т.д., пока не будет выполнено неравенство (11):

$$Sum_{\max} - Sum_j \le a_{ij}, \qquad (11)$$

где $Sum_{\max}$ – максимальная сумма длительностей задач по одному из компьютеров;

Sum_j – минимальная сумма длительностей задач по компьютеру j;

a_{ij} – длительность выполнения i–й задачи на компьютере, для которого сумма длительностей выполнения задач равна Sum_j.

36. Выполняем выбор наилучшего решения по принципу, описанному в п.22.

37. Лучший из вариантов запоминаем и считаем его найденным решением.

Следует также остановится на некоторых деталях описанного выше алгоритма. Как уже говорилось ранее, параметр m^d имеет не малое влияние на получаемые результаты. В данном случае стоит уточнить, что d может принимать любое положительное значение, как целое, так и дробное. При этом значение d на разных этапах выполнения алгоритма не обязательно должно быть одинаковым. Выбор наилучшего значения зависит от конкретной задачи и размерности используемых данных – количества компьютеров (n) и задач (m). Естественно, с ростом значения d, увеличивается время выполнения алгоритма, и соответственно, время решения поставленной задачи, но при этом точность полученного результата будет только улучшаться.

Результаты работы алгоритма

Рассмотрим на примере то, как работает приведенный алгоритм.

Пусть для распределения в вычислительную систему из 5 компьютеров поступило 15 задач. Имеем массив коэффициентов мощностей компьютера:

0,86 3,63 5,00 5,58 5,59

Также имеем массив длительностей выполнения задач:

2 3 5 5 6
6 7 7 7 8
8 8 9 9 10.

Пересчитав длительности решения задач с учетом мощностей компьютеров, получим:

1,72	7,26	10,00	11,16	11,18
2,58	10,89	15,00	16,74	16,77
4,30	18,15	25,00	27,90	27,95
4,30	18,15	25,00	27,90	27,95
5,16	21,78	30,00	33,48	33,54
5,16	21,78	30,00	33,48	33,54
6,02	25,41	35,00	39,06	39,13
6,02	25,41	35,00	39,06	39,13
6,02	25,41	35,00	39,06	39,13
6,88	29,04	40,00	44,64	44,72
6,88	29,04	40,00	44,64	44,72
6,88	29,04	40,00	44,64	44,72
7,74	32,67	45,00	50,22	50,31
7,74	32,67	45,00	50,22	50,31
8,60	36,30	50,00	55,80	55,90
0,86	3,63	5,00	5,58	5,59

Здесь номер столбца соответствует номеру компьютера, а номер строки – номеру задачи. Нижняя строка (под чертой) указывает на коэффициенты мощности компьютеров.

После выполнения «первой» части алгоритма, получим следующее лучшее решение:

Максимальное значение: 50,82
Сумма: 246,51
Распределение задач по компьютерам:

--------	--------	--------	11,16	--------
--------	--------	15,00	--------	--------
4,30	--------	--------	--------	--------
4,30	--------	--------	--------	--------
--------	--------	--------	33,48	--------
5,16	--------	--------	--------	--------
--------	--------	35,00	--------	--------
--------	25,41	--------	--------	--------
--------	25,41	--------	--------	--------
6,88	--------	--------	--------	--------
6,88	--------	--------	--------	--------
6,88	--------	--------	--------	--------
7,74	--------	--------	--------	--------
--------	--------	--------	--------	50,31
8,60	--------	--------	--------	--------
Сумма: 50,74	50,82	50,00	44,64	50,31

После проведения операции «скрещивание» решение будет следующим:

Максимальное значение: 50,74
Сумма: 248,46
Распределение задач по компьютерам:

--------	--------	--------	11,16	--------
--------	--------	15,00	--------	--------
4,30	--------	--------	--------	--------
4,30	--------	--------	--------	--------
--------	21,78	--------	--------	--------
5,16	--------	--------	--------	--------
--------	--------	35,00	--------	--------
--------	--------	--------	39,06	--------
--------	25,41	--------	--------	--------
6,88	--------	--------	--------	--------
6,88	--------	--------	--------	--------
6,88	--------	--------	--------	--------
7,74	--------	--------	--------	--------
--------	--------	--------	--------	50,31
8,60	--------	--------	--------	--------
Сумма: 50,74	47,19	50,00	50,22	50,31

После проведения операции «мутация» в данном случае решение будет имеет такой же вид, как и после проведения операции «скрещивание». Однако, такой результат объясняется тем, что количество компьютеров и задач невелико, а значение d для расчета m^d было взято равным 3, что обеспечило достаточную точность получения результата еще на этапе скрещивания.

Таким образом, выбрав лучшее из представленных лучших решений, получим решение идентичное решению после этапа скрещивания.

На выполнение расчета для данного примера гибридным алгоритмом потребовалось менее 10 мс, при использовании полного перебора, потребовалось времени 9ч 30мин. Полученный результат при использовании полного перебора совпал с полученным результатом при использовании гибридного алгоритма. Верхняя оценка вычислительной сложности алгоритма не превышает $O(m^2 \cdot n)$, где m – количество задач, n – количество компьютеров, участвующих в распределении, что говорит об эффективности предложенного метода.

Выводы

Таким образом, разработанный гибридный алгоритм распределения задач в вычислительной системе с использованием основных операторов классического генетического алгоритма эффективно решает поставленную задачу.

Литература

1. Юрич М.Ю. Математическая модель системы диспетчеризации задач // Науковий вісник Чернівецького національного університету. Серія: Комп'ютерні системи та компоненти", 2009, Випуск 479. – с. 135-139.

2. Мосеев А.В. Применение методов искусственного интеллекта в переборных алгоритмах [Электронный ресурс] // Режим доступа: http://underwood.narod.ru/ as/diplom/ chapter1.html

3. Материал из Википедии — свободной энциклопедии [Электронный ресурс] // Режим доступа: http://ru.wikipedia.org/ wiki/Генетический_алгоритм

Поступила в редакцию 29.03.10