Реализация генетического алгоритма поиска глобального минимума на многопроцессорном кластере

С.Г. Сидоров, И.Ф. Ясинский, Ф.Н. Ясинский
Ивановский государственный энергетический университет

Параллельная реализация алгоритма

Поиск глобального минимума сложных многомерных функций при использовании генетического алгоритма связан с выбором большого числа точек в исходном поколении. Вычислительная эффективность однопроцессорного варианта алгоритма генетического поиска при этом снижается. Применение многопроцессорной высокопроизводительной вычислительной техники позволяет вести одновременный поиск глобального минимума различными исходными поколениями точек. Это позволяет расширить возможности поиска, рамки которого находятся в прямой зависимости от числа одновременно запущенных параллельных процессов.

При разработке параллельного варианта алгоритма актуальным является выбор топологии межпроцессорных связей. В проведенных исследованиях использовалась топология "звезда", при которой каждый из процессоров ("работников") связан с центральным процессором, исполняющем роль диспетчера (см. рис. 1).

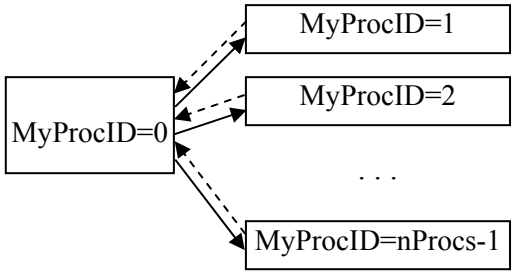

Рисунок 1. Топология звезда. Пунктирными стрелками помечены пересылки лучших решений процессоров, сплошными – пересылка лучшего поколения.

Каждый из рабочих процессоров выполняет однопроцессорный вариант алгоритма, но со своим исходным поколением точек. В процессе поиска каждый из процессоров находит лучшее решение, которое он отправляет центральному процессору.

На центральном процессоре также может исполняться поиск, но как дополнительная функция. Основной его задачей является сбор лучших решений от рабочих процессоров и формирование нового поколения точек. По мере формирования нового поколения оно рассылается по рабочим процессорам.

Получив новые поколения точек, рабочие процессоры повторяют генетический поиск, но уже из лучших условий. При этом существует вероятность получения одинаковых решений на различных рабочих процессорах. Избежать такого исхода позволяет применение случайных процессов при выборе касты и родительских точек. Генератор случайных чисел на каждом из процессоров должен быть при этом инициализирован индивидуально, например, в зависимости от номера процессора.

Дополнительным средством является использование в качестве исходного поколения не только лучших решений полученных от центрального процессора, но и части собственных лучших решений, оставшихся от предыдущей итерации поиска.

Остановку вычислений можно произвести либо по выполнении определенного числа смен лучших поколений (отсылаемых от диспетчера), либо, при получении новых решений, незначительно (в рамках заданной погрешности) отличающихся от решений полученных на предыдущих итерациях поиска.

Описание алгоритма

В предлагаемом алгоритме глобальной минимизации можно выделить несколько основных этапов.

1. Выбираем M случайных точек в пределах N-мерного "ящика". Выбранные точки являются исходным поколением для генетического поиска (родители).

$$x_i^j = XL_i + Z \cdot (XP_i - XL_i); \quad Z \in [0,+1];$$

где i = 1, 2, …, N – координата точки, j = 1, 2, …, M – номер точки, XL_i – левая граница "ящика" для i-ой координаты, XP_i – правая граница "ящика" для i-ой координаты, Z – случайное число.

В каждой из полученных точек вычисляем целевую функцию Q_j (зависит от решаемой задачи).

2. Среди полученных M значений целевых функций осуществляем поиск минимальной (Q_{min}) и максимальной (Q_{max}) по величинам целевых функций. По вычисленным значениям определяем интервал отклонения целевых функций.

$$\Delta Q = Q_{max} - Q_{min};$$

3. Для каждой из точек исходного поколения вводится функция предпочтения, на основе которой в дальнейшем будут формироваться касты.

$$F_j = \frac{(Q_{max} - Q_j)}{\Delta Q};$$

$$F\big|_{Q_{max}} = 0; \qquad F\big|_{Q_{min}} = 1; \qquad F \in [0,+1];$$

Максимальное предпочтение отдается родительским точкам, имеющим минимальную по величине целевую функцию.

4. Интервал от "0" до "1" разбивается на заданное число каст. Количество каст должно определяться в зависимости от количества исходных родительских точек таким образом, чтобы в касты попадало более одной точки. В противном случае возникнут проблемы формирования нового поколения. Размеры каст в простейшем случае могут быть взяты одинаковыми. Но, для улучшения селекции, шансы каст с точками, имеющими более высокую функцию предпочтения, должны быть выше. В связи с этим предпочтительнее выбирать размер касты в прямой зависимости от ее порядкового номера (величины функции предпочтения). Номер касты KS меняется от 1 до MKS (максимального числа каст). Для фиксации границ каст вводятся массивы FL[KS] и FP[KS], определяющие левую и правую границу касты KS соответственно.

5. Производим "разброс" точек по кастам. Вводим дополнительные массивы MMKS[KS] и SP[KS,jKS]. Массив MMKS[KS] определяет число точек в касте KS. Массив SP[KS,jKS] определяет списки номеров точек в касте KS с порядковым номером элемента jKS.

Изначально производим очистку указанных массивов, т.е. считаем, что количество элементов в каждой касте равно нулю и списки с номерами точек не определены (равны нулю).

Заполнение каст производим на основе значений функции предпочтения. Условием попадания точки в касту является попадание значения ее функции предпочтения в интервал между левой и правой границами касты.

$$FL[KS] \le F[j] \le FP[KS];$$

В этом случае номер точки j попадает в список касты KS, а число элементов в касте увеличивается на единицу.

$$MMKS[KS] = MMKS[KS] + 1;$$

$$SP[KS, MMKS[KS]] = j;$$

6. Производим случайный выбор касты. Если величина интервала касты зависит от порядкового номера касты (значения функции предпочтения), шансов быть выбранной становится больше у каст содержащих точки с лучшими целевыми функциями.

Определяем случайное число $Z \in [0,+1]$. Условием выбора касты является попадание случайного числа Z в границы касты.

$$FL[KS] \le Z \le FP[KS];$$

При выборе касты также следует убедиться, что каста не пуста и содержит, по меньшей мере, две точки исходного поколения. В противном случае выбор касты следует повторить, т.к. пустая каста не может дать "потомство".

Далее производится отбор пары родителей. Для получения наилучшего потомства родители должны иметь близкие целевые функции, но находиться на максимальном расстоянии друг от друга.

Номер первого родителя rod1 в касте KS определяется с помощью генератора случайных чисел.

$$rod1 = random(MMKS[KS]);$$

Для определения номера второго родителя rod2 производится вычисление расстояний между всеми точками касты и первой родительской точкой.

$$rast2 = \sum_{i=1}^{N} (x[rod1,i] - x[j,i])^2;$$

Точка, для которой вычисленное расстояние окажется максимальным, выбирается в качестве второго родителя.

$$rod2 = max(rast2_j);$$

7. По прямой, между выбранными родительскими точками, производим глобальный поиск минимума целевой функции. Для этого вводим смещение от первого родителя ETA, которое может изменяться с шагом HETA. Получаем новые точки (потомки), которые расположены между родительскими точками (см. рис. 2).

$$xx[i] = x[rod1,i] + ETA \times$$
$$\times (x[rod2,i] - x[rod1,i]);$$

$$ETA = ETA + HETA;$$

В каждой из полученных точек вычисляем значение целевой функции Q_{lin}. Среди полученных целевых функций определяем оптимальную целевую функцию Q_{opt} имеющую минимальное значение. Точка, целевая функция которой оказалась оптимальной, и которая проходит условие отбора, попадает в новое поколение. Условием отбора является значение целевой функции меньшее по сравнению с родительскими.

$$Q_{opt} < Q_{rod1}; \quad Q_{opt} < Q_{rod2};$$

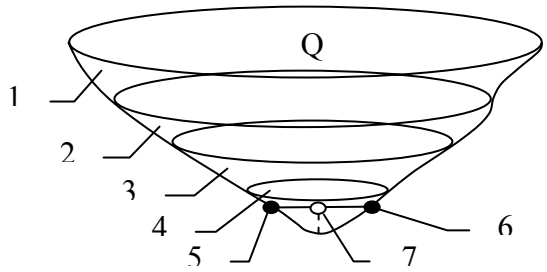

Рисунок 2. Генетический поиск. 1,2,3,4 – номера каст;
5,6 – родительские точки; 7 – точка-потомок.

Когда в новом поколении накопится M точек, то происходит смена поколений. В этом случае поиск продолжается, начиная со второго этапа.

Поиск производится до смены NPokol поколений. По окончании поиска производится выдача наилучших результатов.

Результаты использования

Предложенный алгоритм был апробирован в однопроцессорном варианте и в многопроцессорном варианте на различных многомерных функциях, имеющих как глобальные, так и локальные минимумы.

В большинстве случаев однопроцессорный поиск давал точные (в рамках заданной погрешности) решения (≈80-90%). В остальных случаях выдавались значения ближайшего (к глобальному минимуму) локального минимума.

При использовании параллельного варианта алгоритма на суперЭВМ МВС-100 (НИИММ ИГТА) и многопроцессорной кластерной системе "Энергет" (ИГЭУ) попаданий в локальные минимумы не наблюдалось.

Теоретически такие попадания возможны, но их вероятность значительно снижена за счет одновременного поиска из различных исходных условий. Т.е. если существует хотя бы одно глобальное решение и несколько локальных решений, приоритет будет отдан глобальному решению, как имеющему лучшее значение целевой функции.

Литература

1. Сидоров С.Г, Ясинский Ф.Н. Распараллеливание генетического алгоритма, прогнозирующего течение сложных технологических процессов. Вестник ИГЭУ. – Иваново, 2001. – Вып.2.