АНАЛИЗ ПАРАЛЛЕЛЬНОЙ СТРУКТУРЫ ПОПУЛЯЦИОННЫХ АЛГОРИТМОВ ОПТИМИЗАЦИИ

Ершов Н.М. [1, a], Полуян С.В. [2, b]

[1] *Московский государственный университет им. М. В. Ломоносова, факультет ВМК*
119234, Москва, ГСП-1, Ленинские горы, д. 1, стр. 52, факультет ВМК

[2] *ГБОУ ВО Московской области "Университет "Дубна"*
141982, Московская область, г. Дубна, ул. Университетская, 19

E-mail: [a] ershov@cs.msu.ru, [b] svpoluyan@gmail.com

Работа посвящена вопросам крупно-блочной параллельной реализации методов эволюционной и роевой оптимизации на примере решения задачи минимизации функций действительного аргумента. Рассматривается субпопуляционная и островная схемы распараллеливания. Предлагается классификация паттернов параллельного взаимодействия субпопуляций, выполненная на основе анализа ряда методов рассматриваемого класса. Описывается программная реализация в форме библиотеки шаблонных функций, реализующих данные паттерны, предлагающая пользователю высокоуровневое средство описания популяционных алгоритмов оптимизации. Рассматриваются вопросы параметризации реализуемых алгоритмов с целью исследования эффективности их распараллеливания. Работа выполнена при финансовой поддержке РФФИ (грант № 17-07-01562 А).

Ключевые слова: оптимизация, роевые алгоритмы, генетические алгоритмы, параллельные вычисления

1. Введение

Актуальным подходом к решению многомерных задач непрерывной оптимизации является применение популяционных методов оптимизации [1], к которым относятся эволюционные алгоритмы – генетические, метод дифференциальной эволюции, и роевые – метод роя частиц, алгоритм бактериального поиска и т.д. Практически все алгоритмы данного класса устроены по одной общей схеме: имеется популяция фиксированного размера отдельных особей, представляющих возможные решения оптимизационной задачи, которые выполняют коллективный поиск искомого глобального экстремума. Ключевым моментом такого поиска оказывается взаимодействие особей популяции, при этом разные алгоритмы реализуют разные схемы взаимодействия, эмулируя тот или иной аспект коллективного поведения соответствующего биологического или физического прототипа. Несмотря на существующее разнообразие алгоритмов указанного класса, при организации взаимодействия внутри популяции все они оперируют достаточно ограниченным числом паттернов взаимодействий, комбинируя их тем или иным способом.

Отличительной особенностью популяционных алгоритмов является их высокая вычислительная сложность, которая во многих важных приложениях еще усугубляется сложностью вычисления целевой функции, подлежащей оптимизации [2]. С другой стороны все эти алгоритмы в силу своей природы обладают высокой степенью встроенного параллелизма, т.к. большая часть операций в них выполняется параллельно или над отдельными особями (оператор мутации в генетических алгоритмах), или над парами особей (оператор скрещивания). Перечисленные факты делают популяционные методы оптимизации практически идеальным объектом для реализации на параллельных вычислительных системах [3]. Ограниченный список паттернов взаимодействия делает возможным выполнение автоматического распараллеливания заданного популяционного алгоритма под заданную модель параллельного выполнения. Такая задача и является конечной целью настоящего исследования. Целями же данной работы являются:

- выделение, классификация и формализация паттернов взаимодействия между особями популяции;
- разработка средств высокоуровневого описания популяционных алгоритмов для решения задач непрерывной оптимизации;
- программная реализация разработанного подхода, позволяющая полностью автоматически генерировать для заданного алгоритма оптимизации программу на языке C++ с учетом указанной пользователем модели выполнения.

2. Модель популяционных алгоритмов оптимизации

Исследование структуры популяционных методов непрерывной оптимизации проводилось на следующем наборе алгоритмов: генетические алгоритмы, метод роя частиц, метод CSO, муравьиные алгоритмы, алгоритм бактериального поиска, алгоритм пчелиного поиска, гравитационный алгоритм. Предполагается, что данные, с которыми работает популяционный алгоритм, распределены между особями популяции и представлены атрибутами особей, скалярными или векторными, например, положение и скорость частицы в методе PSO. Общие для всех особей данные представлены атрибутами среды. В результате проведенного анализа были выделены следующие паттерны взаимодействия между особями популяции:

- **`FOREACH f`** — применение оператора **`f`** ко всем особям популяции. Примеры: оператор мутации в генетическом алгоритме, обновление состояния (положения и скорости) в методе роя частиц.
- **`PAIRWISE f`** – случайное разбиение популяции на пары и применение оператора **`f`** к каждой составленной паре. Примеры: операторы отбора и скрещивания в генетическом алгоритме, турнирный оператор в методе CSO.

- **REDUCE key global** – семейство паттернов, выполняющих редукцию (минимум, максимум, сложение и т.п.) атрибута особей **key** в глобальное значение (атрибут среды) **global**. Примеры: определение лучшего решения в популяции в методе PSO, поиск центроида популяции в алгоритме CSO.
- **ROULETTE key** – реализация отбора по атрибуту **key** с помощью метода рулетки. Примеры: отбор (в том числе ранговым методом) в генетических алгоритмах и алгоритме бактериального поиска.
- **DISTANCE pos f key** – вычисление для каждой особи популяции суммы значений функции **f**, примененной к расстояниям до всех других особей. Положение особи задается атрибутом **pos**, вычисленная сумма сохраняется в атрибуте **key**. Примеры: процедура роения в бактериальном поиске, вычисление сил в алгоритме гравитационного поиска.

Также в число паттернов были включены дополнительные паттерны взаимодействия со средой с учетом возможной параллельной реализации такого рода алгоритмов, например:

LOAD global — ввод данных (атрибутов среды).

SAVE global — вывод данных.

COPY id key global — копирование атрибута **key** особи с идентификатором **id** в атрибут **global** среды.

Помимо перечисленных выше паттернов в модель были добавлены вспомогательные команды, задающие списки атрибутов особей (**LOCALS**) и среды (**GLOBALS**), и управляющие команды (**BEGIN, END, BEGINLOOP, ENDLOOP**). Примеры описания генетического алгоритма и метода CSO в рамках предложенной модели приведены на рис. 1 (модификатор * перед именем атрибута означает, что данный атрибут является векторной величиной).

```
GLOBALS f_min
LOCALS *x, f
BEGIN
    FOREACH init
    BEGINLOOP
        FOREACH eval
        PAIRWISE select
        PAIRWISE cross
        FOREACH mutate
    ENDLOOP
    MIN f, f_min
    SAVE f_min
END
```

```
GLOBALS *g, f_min
LOCALS *x, *v, f
BEGIN
    FOREACH init
    BEGINLOOP
        FOREACH eval
        MEAN x, g
        PAIRWISE compete
        FOREACH update
    ENDLOOP
    MIN f, f_min
    SAVE f_min
END
```

Рисунок 1. Описание генетического алгоритма (слева) и алгоритма CSO (справа)

3. Программная реализация

На основе предложенной модели была разработана программная система для автоматической генерации кода определенного пользователем популяционного алгоритма для заданной им модели параллельного выполнения. На вход системе поступает описание алгоритма, состоящее из двух частей. В первой части описывается структура алгоритма в терминах перечисленных выше паттернов (как это показано на рис. 1). Во второй части пользователь определяет (на языке C++) содержание используемых им операторов, например, для генетического алгоритма – это операторы `init`, `eval`, `select`, `cross` и `mutate`. Такое высокоуровневое описание алгоритма преобразуется в готовую к компиляции программу на языке C++. Для организации такого преобразования возможны два принципиальных подхода. При первом подходе оно реализуется средствами целевого языка (C++) с использованием специальных библиотек шаблонных функций. При втором подходе такое преобразование выполняется отдельной программой (препроцессором), которая использует описание пользователя как шаблон для генерации целевой программы. В настоящей работе был реализован второй вариант в силу его большей гибкости, кроме того, такой подход позволяет строить более компактный и более оптимальный код, что должно положительно сказываться на его эффективности.

Программа-препроцессор была написана на языке Python. При ее вызове пользователь, помимо кода своего алгоритма, указывает и целевую модель выполнения (рис. 2). В настоящее время система поддерживает три модели выполнения:

`SEQ` – генерируется последовательная программа;

`ISL` – генерируется параллельная программа с использованием технологии MPI на основе островной модели популяционных алгоритмов;

`SUB` – субпопуляционная модель, в которой популяция разбивается на отдельные субпопуляции одинакового размера, каждая из которых обрабатывается одним процессорным узлом, также с использованием MPI.

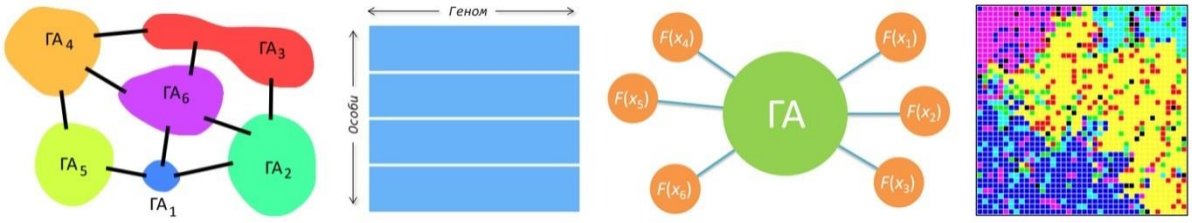

Рисунок 2. Модели параллельного выполнения: островная, субпопуляционная, master-slave и клеточная

В дальнейшем планируется расширить список моделей выполнения клеточной моделью (**`CEL`**) и моделью master-slave (**`MAS`**). Кроме того, предполагается предоставить пользователю дополнительную возможность использования технологии OpenMP.

4. Заключение

Описанная модель и ее программная реализация были протестированы на трех популяционных алгоритмах: генетическом алгоритме, методе роя частиц и методе CSO. Система выдает корректный (готовый к компиляции и выполнению) код на языке C++ в трех моделях выполнения (последовательная, островная и субпопуляционная) по одному и тому же описанию алгоритма. Важным свойством предложенного подхода является компактность описания алгоритмов оптимизации, которое оказывается в 3-4 раза короче окончательного кода этого же алгоритма на C++ (см. табл. 1).

Таблица 1. Сравнение размера (в строках) исходного и сгенерированного кода для трех алгоритмов и трех моделей выполнения

	Source	SEQ	ISL	SUB
Генетический алгоритм	61	175	204	244
Метод роя частиц	60	183	212	270
Метод CSO	65	188	217	257

Одним из возможных приложений предложенной модели является генерация так называемых параллельных *прокси-приложений*, эмулирующих параллельную работу того или иного алгоритма оптимизации. Сам алгоритм при этом описывается рядом параметров, определяющих, например, размер требуемой алгоритму памяти (в расчете на одну особь популяции), время выполнения отдельных операторов алгоритма и т.д. Такие прокси-приложения могут использоваться для исследования эффективности той или иной параллельной реализации, а также для оптимальной настройки аппаратных средств параллельной вычислительной системы под заданную реализацию алгоритма.

Настоящее исследование в дальнейшем предполагается продолжить в следующих направлениях:

- включение в систему других алгоритмов непрерывной оптимизации и реализация дополнительных паттернов взаимодействия внутри популяции;
- поддержка других моделей параллельного выполнения, в частности, модели master-slave и клеточной;
- внедрение в модель параметризации, позволяющей пользователю управлять тем или иным параллельным процессом при выполнении заданного алгоритма, например, частотой использования глобальных операторов алгоритма;
- разработка полноценного специализированного языка описания популяционных алгоритмов, позволяющего генерировать исполняемый код на разных языках программирования.

Список литературы

[1] Карпенко А.П. Современные алгоритмы поисковой оптимизации. М.: Изд-во МГТУ им. Н.Э. Баумана, 2014.

[2] Полуян С.В., Ершов Н.М. Применение параллельных эволюционных алгоритмов оптимизации в задачах структурной биоинформатики // Вестник УГАТУ. 2017. Т. 21, № 4.

[3] Ершов Н.М., Попова Н.Н. Естественные модели параллельных вычислений. М.: Изд-во МАКС Пресс, 2016.

THE ANALYSIS OF PARALLEL STRUCTURE OF POPULATIONAL OPTIMIZATION ALGORITHMS

Ershov N. [1, a], Poluyan S. [2, b]

[1] *Lomonosov Moscow State University, the Faculty of Computational Mathematics and Cybernetics MSU, Faculty of Computational Mathematics and Cybernetics, Russia, 119234, Moscow, GSP-1, Leninskiye Gory str., 1, bdg. 52*

[2] *Moscow Region State Educational Institution for higher professional education University "Dubna", 141982, Russia, Moscow region, Dubna, Universitetskaya str., 19*

E-mail: [a] ershov@cs.msu.ru, [b] svpoluyan@gmail.com

The paper is devoted to the problems of coarse-grain parallel implementation of the evolutionary and swarm optimization methods on the example of solving the problem of minimizing the functions of a real argument. A subpopulation and island paralleling schemes are considered. A classification of parallel interaction patterns between subpopulations, based on the structure analysis of a number of population optimization methods, is proposed. A software implementation is described in the form of a template functions library that offers the user a high-level tool for describing population optimization algorithms. The parametrization issues of optimization algorithms with the aim of studying the effectiveness of their parallelization are considered. This work was supported by the Russian Foundation for Basic Research (Grant No. 17-07-01562 A).

Keywords: optimization, swarm algorithms, genetic algorithms, parallel computing