ПАРАЛЛЕЛЬНЫЕ ПОПУЛЯЦИОННЫЕ АЛГОРИТМЫ ОДНО- И МНОГОЦЕЛЕВОЙ ОПТИМИЗАЦИИ

А.П. Карпенко

МГТУ им. Н.Э. Баумана, Москва, Россия

Введение

Для эффективного решения сложных задач глобальной оптимизации в 80-х годах прошлого века начали интенсивно разрабатывать стохастические поисковые алгоритмы оптимизации. В разных публикациях такие алгоритмы называют поведенческими, интеллектуальными, метаэвристическими, вдохновленными (инспирированными) природой, роевыми, многоагентными, популяционными (*population*) и т.д. Популяционные алгоритмы предполагают одновременную обработку нескольких вариантов решения задачи оптимизации и представляют собой альтернативу классическим «траекторным» поисковым алгоритмам, в которых в области поиска эволюционирует только один кандидат на решение этой задачи.

Классические алгоритмы решения задачи многоцелевой оптимизации (МЦО-задачи) основаны на сведении ее к совокупности задач глобальной одноцелевой оптимизации. Относительно новый и быстро развивающийся класс МЦО-алгоритмов образуют алгоритмы Парето-аппроксимации, предполагающие предварительное построении некоторой конечномерной аппроксимации множества, а тем самым, и фронта Парето. Наиболее известные алгоритмы Парето-аппроксимации построены на основе эволюционных и, чаще всего, генетических алгоритмов. В настоящее время с этой целью все шире используют другие популяционные алгоритмы.

Практически значимые задачи одно- и многоцелевой оптимизации характеризуют нелинейность, недифференцируемость, многоэкстремальность, овражность, отсутствие аналитических выражений (плохая формализованность) и высокая вычислительная сложность оптимизируемых функций, высокая размерность пространства поиска, сложная топология области допустимых значений и т.д. Данные обстоятельства делают актуальной разработку параллельных алгоритмов решения этих задач.

Подавляющее большинство популяционных алгоритмов предложено в англоязычной литературе, в которой вместо традиционного для русскоязычного читателя термина *метод* часто используют термин *алгоритм*. Для того, чтобы избежать возможной неоднозначности чтения, мы также используем последний термин.

1.Постановка задач одно- и многоцелевой оптимизации

Множеством допустимых значений вектора варьируемых параметров X является ограниченное и замкнутое множество

$$D_X=\{X:G(X)\geq 0\}\subseteq\{X\}=R^{|X|}$$

где $G(X)$ - некоторая ограничивающая вектор-функция; $R^{|X|}$ – $|X|$ мерное арифметическое пространство.

Рассматриваем одноцелевую задачу глобальной условной минимизации

$$\min_{X\in D_X} f(X)=f(X^O)=f^O \quad , \tag{1}$$

где $f(X)$ - скалярная целевая функция (критерий оптимальности), $f(X^O)=f^O$ – ее искомое минимальное значение.

Целевая вектор-функция $F(X)=(f_1(X),f_2(X),...,f_{|F|})$ со значениями в пространстве целей $\{F\}$ определена в области D_X . Лицо, принимающее решения (ЛПР), стремится минимизировать в этой области каждую из частных целевых функций $f_1(X),f_2(X),...,f_{|F|}$, что условно записываем в виде

$$\min_{X\in D_X} F(X)=F(X^O)=F^O \quad , \tag{2}$$

где векторы X^O, F^O - искомое решение МЦО-задачи.

Вектор-функция $F(X)$ выполняет отображение множества D_X в некоторое множество D_F целевого пространства, которое называется множеством достижимости. Введем на множествах D_X , D_F отношение доминирования. Говорим, что вектор $F_1=F(X_1)\in D_X$ *доминирует* вектор $F_2=F(X_2)\in D_X$, если среди равенств и неравенств $f_k(X_1)\leq f_k(X_2)$, имеется, хотя бы одно строгое; $k\in[1:|F|]$. Вектор X_1 доминирует вектор X_2 , если $F(X_1)$ доминирует $F(X_2)$.

Выделим из множества D_F подмножество точек D_F^O - *фронт Парето* МЦО-задачи (1), которые не доминируются другими точками этого множества и среди которых нет доминирующих друг друга.

Множество $D_X^O \in D_X$, соответствующее множеству, D_F^O называется *множеством Парето* указанной МЦО-задачи. Таким образом, если $X \in D_X^O$, то $F(X) \in D_F^O$.

2. Популяционные алгоритмы одно- и многоцелевой оптимизации

Общая схема популяционных алгоритмов одноцелевой оптимизации включает в себя следующие шаги.

1) *Инициализация популяции*. В области поиска тем или иным образом создаем некоторое число начальных приближений к искомому решению задачи – инициализируем популяцию агентов.

2) *Миграция агентов популяции*. С помощью некоторого набора *миграционных* операторов, специфических для каждого из популяционных алгоритмов, перемещаем агентов в области поиска таким образом, чтобы, в конечном счете, приблизиться к искомому минимуму целевой функции.

3) *Завершение поиска*. Проверяем выполнение условия окончания итераций и, если оно выполнено, завершаем вычисления, принимая лучшее из найденных положений агентов популяции в качестве приближенного решения задачи. Если указанные условия не выполнены, возвращаемся к выполнению шага 2.

Можно предложить несколько классификаций популяционных алгоритмов оптимизации. Выделяем следующие классы таких алгоритмов [1]:

- эволюционные алгоритмы, включая генетические;
- популяционные алгоритмы, вдохновленные живой природой (алгоритмы роя частиц, колонии муравьев, пчелиного роя, искусственной иммунной системы, бактериальной оптимизации и т.д.);
- алгоритмы, вдохновленные неживой природой (алгоритмы гармонического, гравитационного, электромагнитного поиска и т.д.);
- алгоритмы, инспирированные человеческим обществом (алгоритмы эволюции разума, стохастического диффузионного поиска, «культурные» алгоритмы, меметические алгоритмы и т.д.);
- прочие алгоритмы (самоорганизующийся миграционный алгоритм, алгоритмы рассеянного поиска, прокладки путей и т.д.).

Алгоритмы решения МЦО-задачи чрезвычайно разнообразны. Существует несколько способов классификации этих алгоритмов. Используем классификацию, в соответствие с которой выделяют алгоритмы зондирования, априорные, апостериорные, адаптивные алгоритмы, а также алгоритмы *Парето-аппроксимации*. Ограничимся рассмотрением последних алгоритмов.

Обозначим Θ^F , Θ^X *архивные множества*, содержащие не доминируемые точки F_j^Θ и соответствующие им точки X_j^Θ ; $j \in [1:|\Theta|]$ Большинство популяционных алгоритмов Парето-аппроксимации использует итерационное уточнение множеств точек в архивах Θ^F , Θ^X . Если при этом на данной итерации появляется новая точка F_i , доминирующая некоторые точки из архива Θ^F , то все доминируемые точки, а также соответствующие им точки из архива Θ^F , удаляем. При удовлетворении некоторого критерия останова, текущее содержимое архивов Θ^F , Θ^X полагаем искомой аппроксимацией фронта D_F^O и множества Парето D_X^O соответственно.

В популяционных алгоритмах Парето-аппроксимации новые точки для архивов Θ^F , Θ^X «поставляет» популяция S агентов s_i ; $i \in [1:|S|]$. Текущие координаты агента s_i в пространстве поиска обозначаем X_i , а в целевом пространстве - $F_i = F(X_i)$.

Основной проблемой построения популяционных алгоритмов Парето-аппроксимации является конструирование фитнесс-функции, обеспечивающей перемещение агентов популяции s_i в направлении множества Парето D_X^O , а соответствующих точек F_i - в направлении фронта Парето D_F^O . В силу, как правило, меньшей размерности пространства $\{F\}$ по сравнению с размерностью пространства $\{X\}$, ответ на вопрос о направлении и шаге перемещения агентов обычно отыскивают в терминах целевого пространства, а не пространства параметров. Важно также, что относительно фронта Парето, в отличие от множества Парето, имеется некоторая априорная информация [2].

По способу определения приспособленности агентов выделяем популяционные алгоритмы Парето-аппроксимации на основе лексикографической селекции, чередующихся целевых функций, а также на основе ранжирования агентов. Последние алгоритмы непосредственно используют концепцию доминирования по Парето и нашли наибольшее применение в вычислительной практике. Основным понятием алгоритмов данного класса является понятие *ранга* агента популяции [2].

3. **Распараллеливание популяционных алгоритмов оптимизации**

В литературе представлено несколько способов классификации параллельных популяционных алгоритмов оптимизации. Наиболее известна классификация, в соответствии с которой выделяют

- алгоритмы, использующие глобальную модель параллелизма;
- миграционные алгоритмы, основанные на островной модели параллелизма;
- алгоритмы, в основе которых лежит диффузная модель параллелизма;
- алгоритмы, использующие другие модели параллелизма [3].

3.1. Глобальная модель параллелизма. Параллельные алгоритмы, построенные на основе данной модели (*global model*), представляют собой параллельные аналоги соответствующих последовательных

алгоритмов. Алгоритмы используют параллелизм по данным и ориентированы на организацию параллельных вычислений по типу *master-slave*.

Мастер-процесс в этом случае выполняется на *host*-процессоре параллельной ЭВМ и реализует собственно популяционный алгоритм. При решении МЦО-задачи на *host*-процессоре также хранятся и обрабатываются архивные множества Θ^F, Θ^X. Каждый из подчиненных (рабочих) процессов назначают на выполнение одному из процессоров ЭВМ. Рабочий процесс производит вычисления значений фитнесс-функции и после каждой итерации отсылает мастер-процессу ее значение. Мастер-процесс на основе этих данных вычисляет новые фазовые координаты агентов и посылает их рабочим процессам. И так далее.

Параллельные вычисления в алгоритмах рассматриваемого класса могут быть как синхронными, так и асинхронными.

Глобальная синхронная модель параллелизма (*global synchronous model*) предполагает, что текущая итерация завершается только после того, как мастер-процессом получены значения фитнесс-функции от всех рабочих процессов. Глобальной синхронной модели параллелизма соответствует статическая балансировка загрузки рабочих процессоров параллельной ЭВМ. Условиями высокой производительности алгоритма, использующего данную модель параллелизма, являются гомогенность вычислительной системы и одинаковое время вычисления значений фитнесс-функции в любой допустимой точке пространства параметров. Обеспечить выполнение этих условий обычно не удается. Глобальная синхронная модель параллелизма может быть эффективно реализована на *MIMD*-системах с общей и распределенной памятью, а также на *GPU*.

Глобальная асинхронная модель параллелизма (*global asynchronous model*) основан на том, что мастер-процесс получает данные от рабочих процессов не после глобальной синхронизации, а в любой момент времени по мере готовности этих данных. На основе полученной информации мастер-процесс обновляет фазовые координаты соответствующих агентов популяции и немедленно возвращает их свободным рабочим процессам для продолжения итераций. В алгоритме, реализующем данную модель параллелизма, легко обеспечить динамическую балансировку загрузки процессоров. Глобальная асинхронная модель параллелизма ориентирована на *MIMD*-вычислительные системы с общей или распределенной памятью. Данную модель невозможно эффективно реализовать в рамках парадигмы *GPU*.

Достоинство параллельных популяционных алгоритмов, основанных на глобальной модели параллелизма, состоит в простоте получения и использования глобальной информации о популяции. Недостатком алгоритмов данного класса являются возможные высокие накладные расходы на коммуникации.

3.2. Островная модель параллелизма. Параллельные популяционные алгоритмы, построенные на основе островной модели (*island model*) параллелизма, в некоторых публикациях называют *миграционными* алгоритмами или алгоритмами *коммутирующей мультипопуляции*. Суть алгоритмов этого класса состоит в следующем. Создаем мультипопуляцию $\boldsymbol{S}= S_1 \cup S_2 ... \cup S_{|P|}$, состоящую из числа субпопуляций (островов) S_i, равного числу используемых рабочих процессоров $|P|$ параллельной ЭВМ. Каждый остров обрабатывает свой процессор системы. Обмен данными между островами выполняем после каждых t^{mig} независимых итераций (сезонов) в соответствии с используемой топологией соседства островов. В МЦО-задачах каждый из островов S_i имеет свои локальные архивные множества Θ_i^F, Θ_i^X, а глобальные архивы Θ^F, Θ^X поддерживает *host*-процессор.

Островная модель использует статическую балансировку загрузки параллельной ЭВМ и обеспечивает, как правило, высокую производительность при выполнении следующих условий: вычислительная система является гомогенной; размеры субпопуляций одинаковы и достаточно велики; условием завершения итераций каждой из субпопуляций S_i является достижение одного и того числа итераций.

Известны следующие рекомендации относительно предпочтительности использования глобальной и островной моделей. Если пропускная способность коммуникационной сети ЭВМ является невысокой и мала вычислительная сложность фитнесс-функции, то лучше выбирать островную модель. Также островную модель лучше использовать в случае популяции очень большого размера. В остальных случаях более предпочтительной является глобальная модель.

Варианты островной модели различаются топологиями связей островов, длительностями сезона t^{mig}, стратегиями миграции, режимами обмена агентами, а для МЦО-задачи еще и правилами обновления глобальных архивов Θ^F, Θ^X.

3.3. Диффузная модель параллелизма. Алгоритмы этого класса можно считать частным случаем алгоритмов, основанных на островной модели параллелизма. Каждая из субпопуляций в этом случае включает в себя только одного агента ($|S_i|=1$), так что число островов равно числу агентов и, одновременно, числу используемых процессоров вычислительной системы, то есть $|S|=|P|$.

Основная идея диффузных алгоритмов состоит в том, что лучших агентов определяют параллельно и только среди соседних агентов (в смысле используемой топологии их соседства). Для этого в каждом сезоне миграции процессор, обрабатывающий агента s_i, получает от всех тех процессоров, на которых обрабатываются агенты, соседние с данным, их фазовые координаты и соответствующие значения фитнесс-функции. Такие коммуникации выполняются параллельно всеми $|P|$ процессорами.

Коммуникационные расходы в диффузной модели напрямую определяет используемая топология соседства агентов, и эти расходы являются высокими в случае, если соответствующий граф имеет высокую связность. Алгоритмы данного класса обеспечивают высокую производительность только при невысокой связности указанного графа и/или высокой вычислительной сложности фитнесс-функции.

В целом, алгоритмы, основанные на диффузной модели параллелизма, ориентированы на *MIMD*-параллельные вычислительные системы с общей памятью или сильно связные системы с распределенной памятью (например, гиперкуб или двумерный тор), имеющие быстрые коммуникационные сети.

Вариантом диффузной модели параллелизма можно считать модель, используемую *клеточным* генетическим алгоритмом (*cellular genetic algorithm*) [5]. Топология связей агентов между собой в этом случае фиксирована и представляет собой двумерный тор. Каждый агент обменивается данными только с четырьмя своими соседями (сверху, снизу, слева, справа).

3.4. Другие модели параллелизма. Кроме рассмотренных, выделяют

- модель не коммутирующей мультипопуляции,
- пространственно вложенную модель,
- модель расщепления,
- модель коэволюции,
- комбинированные модели.

Модель *не коммутирующей мультипопуляции* является частным случаем островной модели, когда непосредственный обмен данными между субпопуляциями отсутствует. Достоинством данной модели является минимум коммуникационных расходов, а недостатком – слабое использование потенциала мультипопуляции. Можно считать, что алгоритмы некоммутирующей мультипопуляции представляют собой параллельный вариант метода мультистарта.

Пространственно вложенную модель параллелизма (*spatially embedded*) целесообразно использовать для распараллеливания, например, популяционного алгоритма типа хищник-жертва, когда агенты популяции связаны с узлами некоторой виртуальной сетки [6]. Данная модель предполагает декомпозицию этой сетки на одинаковые фрагменты (в случае гомогенной параллельной ЭВМ) и их обработку различными процессорами.

Модель параллелизма *расщеплением* используют при высокой размерности вектора варьируемых параметров. Идея алгоритмов этого класса состоит в том, что каждому из процессоров параллельной ЭВМ назначают для обработки равное число различных компонентов этого вектора (если его размерность кратна числу процессоров и вычислительная система гомогенна) [7].

Ко-эволюционную модель параллелизма можно считать частным случаем или развитием островной модели, ориентированным на распараллеливание ко-алгоритмов. Модель ко-эволюции предполагает, что каждая из ко-эволюционирующих субпопуляций выполняется на своем процессоре. Основной проблемой данной модели является проблема балансировки загрузки используемой параллельной ЭВМ, поскольку размеры субпопуляций в процессе ко-эволюции меняются и, возможно, весьма существенно [8].

Примером широко используемой *комбинированной* модели параллелизма может служить комбинация островной и глобальной моделей, когда каждая из субпопуляций имеет свое множество «рабов», оценивающих приспособленность агентов этой субпопуляции. При этом «раб» может обрабатывать как одного агента, так и их некоторую совокупность. Примером комбинированной модели может служить также объединение диффузной модели и модели расщепления. В этом случае каждому агенту назначают равное число процессоров (группу). В пределах каждой группы выделяют один процессор, отвечающий за моделирование эволюции данного агента, а остальным процессорам группы поручают обработку соответствующих частей вектора варьируемых параметров.

Заключение

Опыт решения сложных задач одно- и многоцелевой оптимизации показывает, что применение одного алгоритма оптимизации далеко не всегда приводит к успеху. Поэтому в последние годы большое внимание уделяется гибридизации классических и неклассических оптимизационных алгоритмов. В гибридных алгоритмах, объединяющих либо различные алгоритмы, либо одинаковые алгоритмы, но с различными значениями свободных параметров, эффективность одного алгоритма может компенсировать слабость другого [9]. Гибридные алгоритмы оптимизации могут требовать специальных подходов к их распараллеливанию, например, если для базового алгоритма достаточна статическая балансировка загрузки, для соответствующего гибридного алгоритма может быть необходима динамическая балансировка.

Для повышения эффективности современных популяционных алгоритмах широко используют также методы адаптивного и само-адаптивного управления их свободными параметрами (*parameter control*), то есть методы решения задачи мета-оптимизации этих алгоритмов в процессе решения исходной задачи оптимизации [10]. Распараллеливание таких алгоритмов также имеет свою специфику и требует самостоятельного рассмотрения.

ЛИТЕРАТУРА:

1. А.П. Карпенко "Популяционные алгоритмы глобальной поисковой оптимизации. Обзор новых и малоизвестных алгоритмов" // Приложение к журналу «Информационные технологии», 2012, № 7, с. 1-32.
2. А.П. Карпенко, Е.В. Митина, А.С. Семенихин "Популяционные методы аппроксимации множества Парето в задаче многокритериальной оптимизации. Обзор" // Наука и образование: электронное научно- техническое издание, 4, 2012, (http://www.technomag.edu.ru/doc/363023.html).
3. T. El-Ghazali et al. "Parallel Approaches for Multiobjective Optimization" / Multiobjective Optimization, Springer-Verlag Berlin Heidelberg, 2008, pp. 349–372.
4. Л.А. Гладков, В.В. Курейчик, В.М. Курейчик "Генетические алгоритмы: Учебное пособие". — М: Физматлит, 2006, С. 320.
5. E. Alba, B. Dorronsoro "Cellular Genetic Algorithms", Springer, 2008, P. 247.
6. Li Xiaodong "A real-coded predator-prey genetic algorithm for multiobjective optimization" // EMO-2003, pp. 207-221.
7. K.M. El-Abd "A taxonomy of cooperative search algorithms" // Hybrid Metaheuristics. Second International Workshop, 2005, Vol. 3636, pp. 32–41.
8. Е.Ю. Воробьева, А.П. Карпенко "Ко-эволюционный алгоритм глобальной оптимизации на основе алгоритма роя частиц" // Наука и образование: электронное научно- техническое издание, 2013, №11, DOI:10.7463/1113.0619595 (http://technomag.bmstu.ru/doc/619595.html).
9. А.П. Карпенко "Гибридные популяционные алгоритмы параметрической оптимизации проектных решений" // Информационные технологии, Приложение, 2013, №12, с. 6 -15.
10. P.K. Krus, J. Ölvander (Andersson) "Performance index and meta-optimization of a direct search optimization method" // Engineering Optimization, 2013, Vol. 45 (10), pp. 1167–1185.