ФИЗИКА ЭЛЕМЕНТАРНЫХ ЧАСТИЦ И АТОМНОГО ЯДРА
2024. Т. 55, вып. 3. С. 620–623

РАЗРАБОТКА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ПОДДЕРЖКИ ЭВОЛЮЦИОННЫХ И РОЕВЫХ ВЫЧИСЛЕНИЙ ДЛЯ РЕШЕНИЯ ЗАДАЧ МНОГОМЕРНОЙ ОПТИМИЗАЦИИ

*А. Г. Николашкин *, Н. М. Ершов ***

Московский государственный университет им. М. В. Ломоносова, Москва

Рассматривается создание программной системы для поддержки популяционных методов при решении сложных задач оптимизации высокой размерности. Анализируются существующие программные решения в данной области с учетом возможности параллельного выполнения рассматриваемого класса алгоритмов. Описываются модель и структура предлагаемой программной системы, позволяющей проводить параметрическую и структурную настройку реализованных в ней алгоритмов, а также создавать новые алгоритмы и автоматически выполнять их параллелизацию.

The paper is devoted to the development of a software system to support population based methods for solving complex high-dimensional optimization problems. The existing software solutions in the field are analyzed taking into account the possibility of parallel execution of the considered class of algorithms. The model and structure of the proposed software system are described, which allows one to perform parametric and structural tuning of algorithms implemented in it, as well as to create new algorithms and to perform parallelization automatically.

PACS: 89.20.Ff; 07.05.Tp

ВВЕДЕНИЕ

Многомерные задачи глобальной оптимизации достаточно часто возникают в различных областях фундаментальной и прикладной наук. При решении таких задач алгоритмы, оперирующие популяцией пробных решений, как правило, оказываются более эффективными, чем классические траекторные или переборные алгоритмы [1]. К популяционным алгоритмам, прежде всего, относятся эволюционные, такие как генетические алгоритмы и метод дифференциальной эволюции, а также роевые алгоритмы, например метод роя частиц и алгоритм бактериального поиска.

* E-mail: nagvv97@mail.ru
** E-mail: ershovnm@gmail.com

В настоящее время существует достаточно большое количество популяционных алгоритмов и их различных модификаций. Однако, несмотря на это разнообразие, их объединяют общие свойства, такие как стандартная постановка задачи оптимизации, одинаковая структура выполнения и схожие используемые операторы над популяцией. Соответственно, идея разработки программной системы для работы с популяционными алгоритмами оказывается естественной и востребованной. И такие библиотеки создаются и разрабатываются на протяжении последних десятков лет.

СУЩЕСТВУЮЩИЕ РЕШЕНИЯ

В настоящее время, даже если исключить заброшенные проекты, существует немало решений в рассматриваемой области. И каждое из них предлагает разный пользовательский опыт и функционал. Для нас одним из приоритетных функционалов является возможность ускорения вычислений через параллелизацию. При этом не менее важными являются поддержка библиотекой готовых алгоритмов и тестовых задач, возможность реализации пользовательских алгоритмов и других вспомогательных функций.

Далеко не все библиотеки имеют возможность добавления новых алгоритмов. Например, библиотеки GeneAl и PySwarms[2] реализуют только генетические алгоритмы и метод роя частиц соответственно. А библиотека pymoo [3] хоть и реализует широкий набор различных алгоритмов, но не поддерживает добавление пользовательских алгоритмов.

Если оценить поддержку параллелизации вычислений, то оказывается, что большинство решений ограничиваются только параллелизацией вычисления целевой функции. Такой подход рационален, ведь, как правило, именно вычисление целевой функции является самой тяжелой операцией в работе алгоритмов оптимизации. Параллелизация только этой части уже может дать хорошее ускорение с минимальными затратами. Из рассмотренных нами библиотек лучшую поддержку параллелизации вычислений имеет pagmo2/pygmo2 [4], предоставляющая не только параллелизацию вычисления целевой функции, но и встроенную поддержку островной модели. Однако из коробки предлагаются возможности параллелизации только по потокам и процессам в рамках одного компьютера. Если нужно использовать другие инстурменты параллелизации, например MPI, то необходимо реализовать это самостоятельно через предусмотренный для этого интерфейс.

БИБЛИОТЕКА INSECTAE

При разработке библиотеки Insectae ставилось целью создание такого инструмента поддержки эволюционных и роевых вычислений, который предлагал бы возможность структурной и параметрической настройки реализованных в системе алгоритмов, стандартизированные средства для

разработки новых алгоритмов, эффективную параллелизацию вычислений и тестирующую подсистему для выбора наиболее эффективного алгоритма. В качестве языка программирования был выбран язык Python, так как он позволяет добиться большей гибкости в структуре и организации библиотеки, а также упрощает пользовательский опыт.

В качестве основы была использована модель организации алгоритмов, в которой выполнение алгоритма разбито на три этапа, а данные разделены на локальные и глобальные. Этапы выполнения следующие: *инициализация* — создание популяции, настройка параметров особей и среды; *основной цикл* — итеративное выполнение заданных операторов и проверка условия останова; *завершение* — финальная обработка полученных данных. Локальные данные — это данные, принадлежащие конкретным особям, а глобальные, соответственно, общей среде выполнения. При этом особи могут получать доступ к локальным данным других особей только в рамках специальных процедур. Также для работы с популяциями определен набор специальных шаблонов операторов, называемых «паттернами». Использование паттернов для разработки алгоритмов позволяет выполнять автоматическое распараллеливание любых алгоритмов, описанных в терминах данных паттернов.

Библиотека следует объектно-ориентированному подходу и предлагает несколько базовых классов. Из них за выполнение алгоритмов отвечают классы Goal, Target, Stop и Algorithm. Первые три определяют решаемую задачу и условие останова. А класс Algorithm отвечает за выполнение итераций алгоритма, за интерфейс доступа к паттернам, а также за встраивание дополнительных процедур для расширения или изменения логики уже имеющихся реализаций алгоритмов без необходимости их переписывания, например добавление элитизации. Другим базовым классом является Metrics, отвечающий за сбор и обработку статистики.

ПАРАЛЛЕЛИЗАЦИЯ ВЫЧИСЛЕНИЙ

Эволюционные и роевые алгоритмы в силу своих особенностей относительно легко поддаются параллелизации вычислений. Есть несколько способов выполнить параллелизацию [5], из них рассматриваются следующие: модель master-slave для распределения вычисления целевой функции, островная модель, независимые параллельные выполнения алгоритмов, как правило, в рамках тестовой системы и оптимизации метапараметров.

В настоящее время библиотека Insectae предлагает первый из упомянутых способов параллелизации вычислений через интерфейс паттернов. Этот механизм обеспечивается специальным абстрактным классом Executor. Реализации данного класса служат обертками библиотек и фреймворков, предоставляющих интерфейс map/starmap. Данный интерфейс в том или ином виде доступен в большинстве библиотек,

включая mpi4py, dask, scoop, apache spark, ray. Класс Executor обеспечивает единообразность интерфейса, нивелируя различия в конкретных реализациях интерфейса map/starmap. Также он позволяет пробрасывать и выставлять различные параметры нижележащим библиотекам и предоставлять специфичные реализации паттернов, если использование специализированных методов библиотек может обеспечить более эффективную реализацию, нежели интерфейс map/starmap.

НАПРАВЛЕНИЯ ДАЛЬНЕЙШЕЙ РАБОТЫ

Библиотека Insectae в настоящее время находится на относительно начальном этапе разработки, и еще предстоит стабилизировать ее интерфейс. Разработку библиотеки предполагается продолжать, в первую очередь, в направлении улучшения поддержки параллелизации вычислений. В дальнейшем имеются планы по расширению набора готовых алгоритмов и вспомогательного функционала. Это включает тестирующую подсистему для автоматизации и визуализации тестирования и сравнения реализованных в библиотеке алгоритмов.

СПИСОК ЛИТЕРАТУРЫ

1. *Karpenko A. P.* Modern Search Optimization Algorithms. M.: MSTU Bauman, 2014.
2. *Miranda L. J. V.* PySwarms: A Research-Toolkit for Particle Swarm Optimization in Python // J. Open Source Softw. 2018. V. 3, No. 21. P. 433; https://doi.org/10.21105/joss.00433.
3. *Blank J., Deb K.* pymoo: Multi-Objective Optimization in Python // IEEE Access. 2020. V. 8. P. 89497–89509.
4. *Biscani F., Izzo D.* A Parallel Global Multiobjective Framework for Optimization: pagmo // J. Open Source Soft. 2020. V. 5, No. 53. P. 2338; https://doi.org/10.21105/joss.02338.
5. *Sudholt D.* Parallel Evolutionary Algorithms // Springer Handbook of Computational Intelligence / Ed. by J. Kacprzyk, W. Pedrycz. Berlin; Heidelberg: Springer, 2015. P. 929–959; https://doi.org/10.1007/978-3-662-435052.46.