

ГЛОБАЛЬНАЯ ОПТИМИЗАЦИЯ МЕТОДОМ РОЯ ЧАСТИЦ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ

А.П. Карпенко, Е.Ю. Селиверстов

Среди задач непрерывной конечномерной оптимизации самыми важными с практической точки зрения и, одновременно, самыми сложными являются задачи глобальной оптимизации. Относительно новым и быстро развивающимся классом методов решения задач глобальной безусловной оптимизации являются эвристические методы, среди которых выделяются поведенческие (имитационные) методы. Поведенческие методы основаны на моделировании коллективного поведения самоорганизующихся живых или неживых систем. Известность получили следующие поведенческие методы решения задачи глобальной безусловной оптимизации: метод поведения пчёл; метод колонии муравьев; рассматриваемый в данной работе метод роя частиц (метод PSO - particle swarm optimization) [1].

По аналогии с классификацией генетических методов можно выделить три следующих класса параллельных методов PSO:

1. методы, основанные на глобальной модели параллелизма;
2. миграционные методы, основанные на островной модели параллелизма;
3. методы, основанные на диффузной модели параллелизма.

В работе рассматривается метод, основанный на островной модели параллелизма.

Современные графические процессоры (ГП) являются мощными специализированными вычислителями. В настоящее время ГП все шире используются также для решения сложных вычислительных задач. Известны два основных производителя ГП для высокопроизводительных вычислений - AMD и NVidia. Архитектуры ГП этих производителей существенно отличаются [2]. В работе рассматривается использование ГП компании NVidia. Для программирования ГП в работе используется инструментальный CUDA (C Unified Driver Architecture) компании NVidia [3]. Архитектура CUDA стала основой для открытого стандарта гетерогенных вычислений OpenCL.

1. Постановка задачи и канонический метод роя частиц

Рассматривается задача глобальной безусловной минимизации вещественнозначной целевой функций $\Phi(X)$ в n -мерном арифметическом пространстве R^n :

$$\min_{x \in R^n} \Phi(x) = \Phi(x^*) \quad (1)$$

Множество частиц (рой частиц) обозначим $S = \{P_i, i \in [1:NN]\}$, где NN - количество частиц в рое S . В момент времени $t=0,1,2,\dots,T$ координаты частицы P_i определяются вектором $X_{i,t} = (x_{i,t,1}, x_{i,t,2}, \dots, x_{i,t,n})$, а ее скорость - вектором $V_{i,t} = (v_{i,t,1}, v_{i,t,2}, \dots, v_{i,t,n})$. Начальные координаты и скорости частицы P_i равны $X_{i,0} = X_i^0$, $V_{i,0} = V_i^0$ соответственно.

Итерации в каноническом методе PSO [4] выполняются по следующей схеме:

$$X_{i,t+1} = X_{i,t} + V_{i,t+1} \quad (2)$$

$$V_{i,t+1} = \alpha V_{i,t} + U[0, \beta] \otimes (X_{i,t}^b - X_{i,t}) + U[0, \gamma] \otimes (X_{g,t} - X_{i,t}). \quad (3)$$

Здесь $U[a, b]$ представляет собой n -мерный вектор псевдослучайных чисел, равномерно распределенных в интервале $[a, b]$; \otimes - символ покомпонентного умножения векторов; $X_{i,t}^b$ - вектор координат частицы P_i с наилучшим (в смысле (1)) значением целевой функции $\Phi(X)$ за все время поиска; $X_{g,t}$ - вектор координат соседней с данной частицы с наилучшим за время поиска значением целевой функции $\Phi(X)$; α, β, γ - свободные параметры алгоритма, рекомендации по выбору которых приведены, например, в работе [5].

Важнейшим понятием в методе PSO является понятие соседства частиц, которое определяется соответствующей топологией соседства. Топология соседства частиц задается неориентированным графом, вершины которого соответствуют частицам роя, а ребра связывают непосредственных соседей. В вычислительной практике часто используется топология соседства типа «кольцо» (lbest-топология) [1].

2. Параллельный метод роя частиц GIPSO

В работе рассматривается параллельный вариант метода роя частиц, основанный на островной модели параллелизма [6]. Идея методов данного класса состоит в том, что весь рой из NN частиц делится на NS островов (подроев), и частицы, принадлежащие каждому из островов, обрабатываются на своем процессоре.

После каждых $T_Z < T$ независимых итераций, острова обмениваются между собой лучшими частицами (в соответствие с используемой топологией соседства частиц).

Реализованный в работе параллельный вариант метода роя частиц, названный нами метод GIPSO (GPU Island Particle Swarm Optimization), построен на основе одного из вариантов канонического метода роя частиц, в котором координаты частиц вычисляются по формуле (2), а скорости – по модифицированной формуле (3)

$$V_{i,t+1} = \chi \left(V_{i,t} + \varphi \left(Y_{i,t} - X_{i,t} \right) \right), \quad (4)$$

где

$$\varphi = U[0, \beta] + U[0, \gamma], \quad Y_{i,t} = \frac{U[0, \beta] \otimes X_{i,t}^b + U[0, \gamma] \otimes X_{g,t}}{\phi}.$$

Кроме того, в отличие от канонического метода, введены ограничения на скорости частиц $-V^{\max} \leq V_i \leq V^{\max}$, где величины V^{\max} зависят от размера пространства поиска n .

Метод роя частиц, который задается формулами (2), (4), назовем CPSO (Classic Particle Swarm Optimization).

Острова, на которые разбивается рой S , обозначим S_i , $i \in [1 : NS]$. В программной реализации принято, что каждый из островов S_i содержит одинаковое количество частиц, N так, что общее количество частиц в рое S равно $NN = N \times NS$. Эволюция частиц каждого из островов S_k протекает в соответствии алгоритмом метода CPSO, который поэтому называется ядром метода GIPSO.

Итерации $[1, 2, \dots, T]$ разделяются на SC сезонов, каждый из которых включает в себя T_Z итераций: $T = T_Z \times SC$. В методе GIPSO итерации в рамках сезона для всех островов, а также для частиц в пределе острова выполняются параллельно.

С целью обмена информацией об исследованных различными островами S_i областях пространства поиска по завершении текущего сезона осуществляются миграции частиц между островами. В методе GIPSO используется две стратегии миграции частиц – перемещение и репликация.

Миграция перемещением частиц заключается во взаимном обмене наилучшей частицы $P_{i,best}$ острова S_i и наихудшей частицы $P_{j,worst}$ острова S_j , $i \neq j$. Выбор острова S_j может происходить случайным образом или в соответствии с некоторой топологии соседства островов, например, $lbest$ -топологии.

Миграция на основе репликации заключается в следующем. В каждом из подроев S_i находится лучшая частица $P_{i,best}$. Из этих частиц выбирается наилучшая частица $P_{k,best}$. Далее из подроя S_k осуществляется репликация (копирование) частицы $P_{k,best}$ во все остальные подрои S_i , $i \neq k$.

Метод роя частиц отличается от многих других эволюционных методов оптимизации наличием большого количества свободных настраиваемых параметров, некоторые из которых существенно влияют на основные характеристики алгоритма. Метод GIPSO, во-первых, наследует свободные параметры канонического метода PSO, во-вторых, имеет следующие собственные свободные параметры: NS - количество островов; NN - общее число частиц в рое; T - интервал миграции; стратегия миграции.

3. Отображение алгоритма GIPSO на архитектуру CUDA

Видеопроцессор архитектуры CUDA включает в себя несколько (4 - 32) потоковых мультипроцессоров (SM), каждый из которых состоит из нескольких потоковых процессоров (SP). В программной реализации метода GIPSO (программе GIPSO) каждому из островов поставлен в соответствие вычислительный блок. Параллельная обработка частиц в пределах каждого из вычислительных блоков производится на потоковых процессорах. Для реализации миграции частиц между островами в пределах каждого из блоков имеют место коммуникации между потоками. Для обеспечения высокой производительности и хорошей балансировки потоков ГП используется количество вычислительных блоков, значительно превосходящее количество мультипроцессоров.

В программе GIPSO учтены основные особенности ГП – указанная выше иерархическая система процессоров и неоднородная иерархическая память [2].

По спецификации Compute Capability 1.2 размер общей памяти ГП равен $G_{shm} = 16 \times 1024 = 16384$ байтов. Алгоритм GIPSO требует $K_{shm} = (3nN + N + n + 1)4$ байтов общей памяти. Из условия $K_{shm} < G_{shm}$ вытекает следующее ограничение на количество частиц в островах:

$$N < \frac{G_{shm}/4 - n - 1}{3n + 1} \quad (5)$$

Каждый из мультимикропроцессоров SM обладает набором регистров, общее количество которых в одном мультимикропроцессоре составляет $G_{reg} = 8192$ для спецификации Compute Capability 1.0 и $G_{reg} = 16384$ для спецификации Compute Capability 1.2.

Количество используемых в программе GIPSO регистров явно не может быть задано средствами библиотеки CUDA. В то же время, если локальные данные потока не помещаются в регистрах, компилятор NVCC использует для этих данных локальную память мультимикропроцессора (чрезвычайно медленную, по сравнению с регистровой памятью). Поэтому в программе GIPSO предприняты значительные усилия для того, чтобы использовать только регистровую память.

Пусть потоку требуется R_t регистров. Тогда для вычислительного блока, состоящего из J потоков, количество требуемых регистров определяется как

$$R_b = \left(R_t \times \text{round}(J, 32), \frac{G_{reg}}{32} \right). \quad (6)$$

Здесь функция округления $\text{round}(a, b) = \text{ceil}(a/b)b$. В зависимости от используемого варианта вычислительного ядра, выполнены реализации алгоритма GIPSO с количеством потоков $J = 32, 30, 7$. Количества используемых регистров R_t при этом равны N, Nn, NS , соответственно.

Требования (5), (6) и формируют ограничения на свободные параметры алгоритма. N, NS . Размерность пространства поиска n является определяющей при выборе количества частиц в островах N . В то же время, количество роев NS практически не зависит от величины n и должно только удовлетворять ограничению $NS \leq 320$. Именно это обстоятельство, в конечном счете, обуславливает хорошую масштабируемость метода роя частиц для архитектуры ГП CUDA.

4. Тестирование метода

При тестировании программы GIPSO использовались видеопроцессоры с 4 и 16 мультимикропроцессорами. В качестве тестовых функций использованы функции Экли, Растригина, Розенброка из стандартного набора тестовых функций, применяемых для анализа эффективности методов глобальной оптимизации [1]. Основные эксперименты выполнены в гиперкубе Π^8 , принадлежащем 8-мерному пространству R^8 , где $\Pi^8 = \{-20 \leq x_i \leq 20, i \in [1:8]\}$. Использовался также аналогичный гиперкуб Π^4 в пространстве R^4 .

Изучено влияние основных свободных параметров метода на его характеристики. Варьировалось количество островов NS , количества частиц в островах N , длительность сезона, стратегия миграции.

Сходимость метода проверялась при следующих значениях параметров алгоритма: $NS=4, N=32, n=8$. Эксперименты показали, что на всех указанных тестовых функциях глобальный минимум находится за относительно небольшое количество итераций (50 - 300).

Скорость сходимости метода, как функция числа островов NS , исследовалась при числе островов, равном 2, 4, 8, 16, 32, 64. Результаты исследования показали равномерное увеличение скорости сходимости по мере увеличения числа островов.

Скорость сходимости метода, как функция размерности пространства поиска n , была рассмотрена, как отмечалось выше, при $n=4$ и $n=8$. При малой размерности пространства поиска ($n=4$) выявлено слабое влияние размера роя на скорость сходимости. Приемлемые по скорости результаты показывают рои уже из 4, 8 частиц. При этом во всех случаях достигается глобальный экстремум. Для пространства поиска большей размерности ($n=8$) размер роя начинает оказывать влияние не только на скорость сходимости, но и на значение достигаемого экстремума. При использовании очень маленьких роев из 2, 4 частиц глобальный экстремум достигается далеко не всегда. Минимальный размер роя, при котором обеспечивается отыскание глобальных экстремумов всех рассматриваемых тестовых функций, составляет 8 частиц.

Сравнительная оценка производительности методов GIPSO, CPSO выполнена при следующих значениях свободных параметров: в методе GIPSO - $N=32, NS=4, T=50$; в методе CPSO - $N=32$. Выявлено, что во всех случаях метод GIPSO имеет лучшую скорость сходимости по сравнению с методом CPSO. Особенно очевидны эти преимущества на сложных тестовых функциях.

ЛИТЕРАТУРА:

1. Карпенко А.П., Селиверстов Е.Ю. Обзор методов роя частиц для задачи глобальной оптимизации (Particle Swarm Optimization) // "Наука и образование: электронное научно-техническое издание", www.technomag.edu.ru/, март, 2009.
2. Графические процессоры для высокопроизводительных вычислений. <http://parallel.ru/GPU/>

3. NVidia Corporation. NVIDIA CUDA Programming Guide Version 1.1. 11.29.2007.
4. Kennedy, J. Particle swarm optimization / J. Kennedy, R. Eberhart // In Proceedings of IEEE International conference on Neural Networks. — 1995. — Pp. 1942–1948.
5. Li-ping, Z. Y. Huan-jun, H. Shang-xu. Optimal choice of parameters for particle swarm optimization // Journal of Zhejiang University Science.— 2005.-06.-Vol. 6, No. 6.- Pp. 528–534 (<http://dx.doi.org/10.1007/BF02841760>)
6. Belal, M. Parallel models for particle swarm optimizers / M. Belal, T. El-Ghazawi // IJICIS. — 2004. — January. — Vol. 1. — Pp. 100–111.