计算机科学
COMPUTER SCIENCE

基于人工蜂群算法的多维函数优化加速方法

李辉, 韩林, 于哲, 王威

引用本文

李辉, 韩林, 于哲, 王威.基于人工蜂群算法的多维函数优化加速方法[J]. 计算机科学, 2022, 49(11A): 211200075-6.

LI Hui, HAN Lin, YU Zhe, WANG Wei. Acceleration Method for Multidimensional Function Optimization Based on Artificial Bee Colony Algorithm [J]. Computer Science, 2022, 49(11A): 211200075-6.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211200075

基于人工蜂群算法的多维函数优化加速方法

李 辉[1] 韩 林[2] 于 哲[3] 王 威[2]
1 青岛农业大学经济学院 山东青岛 266109
2 郑州大学国家超级计算郑州中心 郑州 450000
3 中国科学院微电子研究所 北京 100029

摘 要 人工蜂群算法在农业农村大数据应用开发中被广泛采用,但是串行人工蜂群算法时间的复杂度较高,不适用于多维函数的快速求解问题。针对串行人工蜂群算法对多维函数求解执行效率较低的问题进行分析,通过解析多维函数及人工依赖关系判定,提出了一种基于人工蜂群算法的多维函数优化加速方法,该方法包括任务划分、数据分布、同步操作和任务并行。为了证明方法的有效性,以海光处理器为硬件测试平台,对4个多维函数进行对比测试。实验结果表明,与串行人工蜂群算法对多维函数的求解速度相比,该方法对于4个多维函数的求解速度能得到大幅提升。

关键词:大数据;人工蜂群算法;多维函数;ROCm HIP模型;海光处理器
中图法分类号 TP391

Acceleration Method for Multidimensional Function Optimization Based on Artificial Bee Colony Algorithm

LI Hui[1], HAN Lin[2], YU Zhe[3] and WANG Wei[2]
1 School of Economics, Qingdao Agricultural University, Qingdao, Shandong 266109, China
2 National Supercomputing Center in Zhengzhou, Zhengzhou University, Zhengzhou 450000, China
3 Institute of Microelectronics of the Chinese Academy of Sciences, Beijing 100029, China

Abstract The artificial bee colony algorithm is widely used in the development of agricultural and rural big data applications, the serial artificial bee colony algorithm has a high time complexity and is not suitable for solving multi-dimensional problems quickly. According to the serial artificial bee colony algorithm, the problem of low execution efficiency of multi-dimensional function solving is analyzed, a multi-dimensional function optimization method based on the artificial bee colony algorithm is proposed after analyzing the multi-dimensional function and determining the artificial dependency relationship, which consists of task allocation, data distribution, synchronization operations and task parallelism. To demonstrate the efficacy of the proposed method, the Haiguang processor is used as a hardware test platform to compare and test four multi-dimensional functions. Experimental results show that the proposed method significantly outperforms the serial artificial bee colony algorithm in solving four multidimensional functions.

Keywords Big data, Artificial bee colony algorithm, Multidimensional function, ROCm HIP model, HYGON processor

1 引言

我国农业农村数据历史长、数量大、类型多,但长期存在核心数据缺失、数据处理质量不高、开发利用不够等问题,无法满足新时期农业农村发展需要[1]。农村网络基础设施建设数量的增长和网民数目的增加,需要更高效的农业农村数据的载体,并且应用市场的需求也逐步显现,特别是依托于超级计算、云计算等多种先进计算应用与分析方法的出现,为农业农村发展有效积累了海量数据处理的经验,为解决我国农业农村大数据发展面临的困难和问题提供了有效途径[2]。但大数据技术的超速发展使得需要处理的数据规模骤然增加,这就要求科研技术人员不断改进数据分析以及处理方法,不断提高数据处理效率。在对众多数据进行处理优化过程中,许多实际的数据可以采用数学建模的方式,将其抽象为函数问题,使用相应的优化算法对其进行改进。

针对函数优化问题的传统优化算法,如进化计算[3]、黄金分割法[4]和多项式近似法[5]等优化算法,已经无法满足现阶段农业农村大数据处理的需求,原因是其局部最优解的结果具有较大的局限性,并且数值之间的依赖性较强。因此需要并行性和随机性强的优化算法对数据进行优化处理,针对仿照生物类的群智能算法的研究尤为重要[6]。群智能算法具有原理简单、参数设置少、优化效果更好的特点,因此成为解决

基金项目:硅纳微结构与芯片器件超大规模计算应用与自主软件研发(201400211300)
This work was supported by the Ultra-large-scale Computing Applications and Independent Software Development of Silicon Nanostructures and Chip Devices(201400211300).
通信作者:李辉(leephil@163.com)

大规模复杂问题的有效方法[7]。

2 优化分析

人工蜂群((Artificial Bee Colony, ABC)算法是一种群智能算法[8]。该算法通过模拟自然界中蜜蜂寻找最好蜜源的过程来寻找问题的最优解,具备预设参数少、实现简便、并行性强等优点。该算法在早期能够解决一些问题,并广泛应用于神经网络、处理图像和路径规划等领域。但对于多个目标的大规模数据求最优解过程,比如对多维函数的求解问题,该算法求解耗时过长,仍不能满足行业大数据处理需求。目前,针对 ABC 算法的优化,国内外很多学者进行了相关的研究工作。文献[9]使用 CUDA 架构实现了并行 ABC 算法;文献[10]使用消息传递接口实现了分布式并行 ABC 算法;文献[11]基于 FPGA 实现了硬件并行体系结构的 ABC 算法。但是在当前恶劣的国际形势下,我们必须在国产软、硬件的基础上,研发属于本国的先进大数据安全技术,保证农业农村核心数据的安全,确保满足新时期农业农村发展的需要。

为了更加有效地提高 ABC 算法求解多维函数的效率,本文提出了一种基于人工蜂群算法的多维函数优化加速方法,该方法以国产海光处理器为硬件基础,重点研究了如何提高 ABC 算法对多维函数求最优解的效率。主要研究内容为:

(1)为了提高 ABC 算法对多维函数求最优解的效率,使用海光处理器作为硬件平台,对 ABC 算法中的函数进行了并行性分析。

(2)针对海光处理器的特点,提出了对 ABC 算法的并行加速方法,首先对计算任务进行划分,接着对计算任务中的数据进行分布处理,然后使用同步操作技术避免线程间的数据竞争;最后是对计算任务的并行实现。

(3)为了验证本文提出的基于人工蜂群算法多维函数优化加速方法的有效性,本文分别采用串行 ABC 算法和优化后的 ABC 算法对 4 个多维函数的求解速度进行测试。

2.1 关键函数

算法在求最优解的过程中,需计算大量适应度值。在处理多维度优化问题上,ABC 算法的时间消耗主要在计算适应度函数问题上[12]。ABC 算法在处理多维函数时,随着问题维度的增加,计算的数据也增加,需要消耗大量的时间来计算函数值。采用 4 个不同求极值的多维函数来测试 ABC 算法性能,函数[13]如表 1 所列。Sphere 函数的特点是单模态,其全局最小值在 $x=(0,0,\cdots,0)$ 处,值为 $f(x)=0$,其取值范围在 $[-100,100]$。Rosenbrock 函数的特点是单模态,不可分的,其全局最小值在 $x=(1,1,\cdots,1)$ 处,值为 $f(x)=0$,其取值范围在 $[-30,30]$。Sphere 函数和 Rosenbrock 函数属于单峰函数,表示在一个区间内只有一个局部极大的峰值。Griewank 函数的特点是多模态,并且不可分,其全局最小值在 $x=(0,0,\cdots,0)$ 处,值为 $f(x)=0$,其取值范围在 $[-600,600]$。Rastrigin 函数的特点是多模态,并且可以分离,其全局最小值在 $x=(0,0,\cdots,0)$ 处,值为 $f(x)=0$,其取值范围在 $[-5.12,5.12]$[14]。Griewank 函数和 Rastrigin 函数是多峰函数,表示一个区间内有多个局部的极大峰值。4 个标准测试函数的理论最值为 0,优化结果越接近 0,说明算法的优化效果更好[15]。

表 1 多维函数的基本信息

Table 1 Basic information of multidimensional functions

函数名	表达式	取值范围	最小值点
Sphere	$f(x)=\sum_{i=1}^{n}x_i^2$	$[-100,100]$	$f_1(\vec{0})=0$
Rosenbrock	$f(x)=\sum_{i=1}^{n}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	$[-50,50]$	$f_2(\vec{1})=0$
Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]$	$f_3(\vec{0})=0$
Rastrigin	$f(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	$[-5.12,5.12]$	$f_4(\vec{0})=0$

为了实现 ABC 算法对多维函数的加速效果,需要测试串行 ABC 算法处理以上多维函数的消耗时间,采用 Linux 系统下的性能分析工具对串行 ABC 算法的函数进行测试,定位到算法的性能瓶颈,表 2 为 ABC 算法热点函数分析。

表 2 ABC 算法热点函数分析

Table 2 Hot spot function analysis of ABC algorithm

函数名	Sphere	Rosenbrock	Griewank	Rastrigin
OnlookerBees	50.69	51.16	50.80	51.00
EmployedBees	46.14	45.31	46.00	45.10
Probabilities	1.40	1.50	1.40	1.90
ScoutBees	1.10	0.93	0.80	1.20
BestSource	0.67	1.10	1.00	0.80

表 2 的结果显示,OnlookerBees 与 EmployedBees 两个函数耗时百分比较高,时间开销较大,并且这两个函数中有大量的适应度值计算,可以对这两个函数进行深入分析。

EmployedBees 函数的功能是搜索产生新的蜜源,并且依据贪婪选择原理决定是否更新蜜源。若选择更新蜜源,EmployedBees 函数将选择的蜜源信息传递给 OnlookerBees 函数,OnlookerBees 函数根据轮盘赌选择的方法选择食物源,计算出每个蜜源位置上的概率。ScoutBees 函数的作用是在轮盘赌选择后的蜜源周围进一步搜索,寻找更好的蜜源。针对传统的 ABC 算法时间复杂度较高的问题,提出了一种基于 CPU+DCU 的 ABC 算法模型,如图 1 所示。

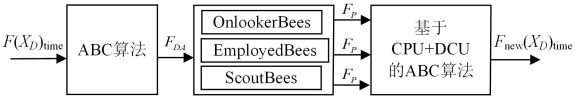


图 1 基于 CPU+DCU 的 ABC 算法模型

Fig. 1 ABC algorithm model based on CPU+DCU

图 1 中 $F(X_D)_{time}$ 是输入到 ABC 算法多维函数的运行时间,F_{DA} 代表对 ABC 算法的依赖分析操作,分析 OnlookerBees, EmployedBees 和 ScoutBees 这 3 个函数之间的关系,EmployedBees 的作用是更新蜜源;OnlookerBees 的作用是计算出每个蜜源位置的概率;ScoutBees 的作用是进一步搜索寻找更好的蜜源,三者之间独立运行,没有数据和算法之间的依赖关系,可以对三者进行并行运算。F_P 代表并行操作,即将 3 种蜜蜂的运算过程传输到 DCU 端进行优化处理。$F_{new}(X_D)_{time}$ 代表经过优化处理后的 ABC 算法对多维函数的运行时间。

2.2 优化方法

为了提升系统和程序的性能,可以采取很多优化方法

应用于系统和应用程序中。一般会针对源代码、多线程和多进程等方面进行优化处理,但是采用多进程方法在各个节点之间进行通信时,会产生大量的时间开销[16]。若只对代码进行循环展开等代码级别的优化方式,性能能够提升的空间十分有限。若对寄存器进行优化和指令流水重排,需要对存储器的结构体系和寄存器的配置非常熟悉,工作量巨大且效果不显著。采用多线程的优化方法能够并行处理多个计算任务,能够提升应用程序的性能。针对多维函数的 ABC 算法并行加速方案,可以利用 CPU+DCU 的异构架构模式,采用 HIP 异构语言的多线程技术并行优化,具体的优化方法如图 2 所示。

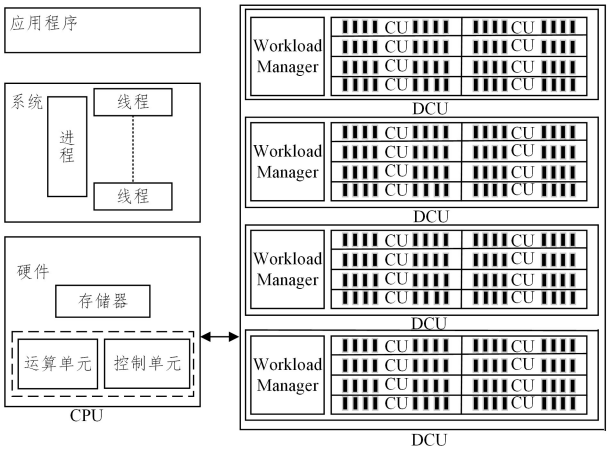


图 2　常用优化方法

Fig. 2　Common optimization methods

从图 2 中可以看出每个计算结点有 1 块 CPU 处理器和 4 块 DCU 加速卡,其中 DCU 加速卡上使用的编程模型为 AMD 公司开发的 ROCm(Radeon Open Computing platform) HIP(Heterogeous-compute Interface for Portability)异构编程模型[17]。HIP 异构编程模型将应用程序在 CPU 上运行的程序称作主机端程序,在 DCU 上运行的程序称为设备端程序。DCU 的主要计算结构是 CU(Compute Unit),每个 CU 包含 4 个 SIMD 计算组和 1 个标量计算单元(Scalar ALU),1 个标量计算单元被 CU 内 4 个 SIMD 共享,并且每个 CU 内拥有 64 kB 的共享内存 LDS(Local Data Shared),其中每个 CU 内的 LDS 各自独立[18]。可以利用 CPU 和 DCU 各自的计算优势,实现多级并行计算,提高并行程序整体的执行性能。

3　优化实现

本文通过对 ABC 算法的热点函数分析,提出了基于人工蜂群算法的多维函数优化方法,该方法基于 CPU+DCU 的结构,将 ABC 算法应用于曙光 Parastor300s 并行存储系统。分别采用了任务划分、数据分布、同步操作和任务并行的方法进行优化。任务划分主要是对蜜源和蜂群的数目进行线程任务的划分;数据分布是将蜜源和适应度值的信息放到共享内存中;同步操作负责解决共享内存中数据竞争的问题;任务并行方法实现了 ABC 算法在 CPU 和 DCU 中的具体功能。

3.1　任务划分

在 DCU 平台上,计算任务划分到不同的线程中,线程对蜜源和蜂群的数目进行计算任务的划分。将蜜源进行一维维度的划分,将线程划分成 block-Idx * blockDim+threadIdx 的形式,其中 blockDim 保存块中的线程数,blockIdx 是当前块的索引值,threadIdx 表示块中当前线程的数量。每个 block 又可以分成多个 thread,将寻找最优蜜源的过程进行一维 block 维度线程的划分,即 threadIdx 的形式。

HIP 模型的线程组织分为 block 和 thread 两个层级,其中 block 对内部的 thread 进行分配,grid 对内部的 block 进行分配。图 3 的 gridDim 是块数,blockIdx 是网格内当前块的索引值,划分成 N 块,threadIdx 是块内当前线程的索引,表示当前运行一个核心上的线程,其中线程块中的线程数 blockDim 的值是 N。

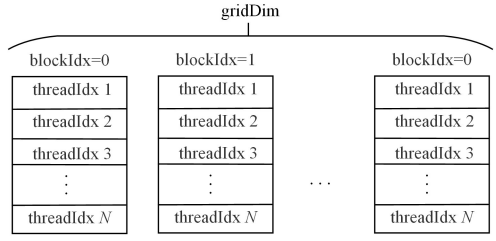


图 3　任务划分方式

Fig. 3　Task division method

任务划分算法的实现过程如算法 1 所示。

算法 1　task division algorithm

Input:blockIdx. x:block index within grid

blockDim. x:count the number of threads in a block in the x direction

threadIdx. x:thread index within block

bees:colony parameters

Output:newFitness:new fitness information

1. j=blockIdx. x * blockDim. x+threadIdx. x
2. if(j < bees)
3. fitness[j]=fit. fitness
4. i=threadIdx. x
5. if(i < employed bees)
6. $x_i'=x_i+\Phi_i(x_i-x_{i'})$
7. else
8. limit[i]=limit[i]+1
9. if(i <onlooker bees/2)
10. $P_i=\dfrac{fit(x_i)}{\sum_{N=1}^{SN} fit(x_n)}$
11. else
12. limit[i]=limit[i]+1

3.2　数据分布

为了提升对访问内存的处理效率,减少内存中全局内存的访问次数,要尽可能多地使用共享内存[19]。全局内存属于板载显存,而共享内存属于片上内存,因此共享内存的读写速度比全局内存快得多[20]。在寻找最优蜜源的过程中需要多次读写全局内存,访问全局内存速度较慢,而且如果不是连续访问,访问速度会变得更慢,因此尝试使用共享内存来代替全局内存实现数据的存储与计算。在数据进行计算之前,先将即将计算的数据从全局内存读取到共享内存中,需要计算时从共享内存中直接读取,并将结果临时存在共享内存里,待计算完全结束后将最终的计算结果写回全局内存。若蜜源和适应度值的信息全部都传输到 DCU 芯片的全局内存中,那么

读和写入相关的数据都会增加大量的运行时间。为了提升DCU的运算性能，使用了DCU架构中的共享内存部件。

图4中全局内存中的蜜源和适应度信息被存储到共享内存中，每一个block都有自己的共享内存空间，block里面的每个线程都可以访问其内部共享内存中的数据。线程之间进行相互协作，大大减少了所需的全局内存的带宽，使用共享内存可以提升数据访问操作的速度。

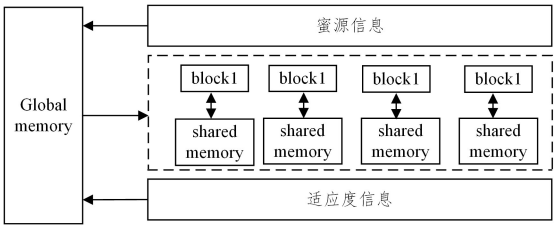


图4 数据分布划分

Fig. 4 Data distribution division

3.3 同步操作

通过采用数据分布的方法，Block里面的线程可以采用协同计算的方式，这种方式能够增加计算效率。但是当线程对共享存储空间的数据进行访问时，会出现多个线程同时访问到共享内存中存储的数据，对存储的数据进行写操作，最终从内存中取得错误数据的情况。线程之间数据竞争的问题是并行计算常见问题，解决数据竞争最简单的方法就是建立线程之间的屏障，将线程都同步进行。在block内可以使用syncthreads语句创建屏障，用于同步线程之间的运算。

图5中的syncthreads语句用于协调同一个block中线程之间的通信，将block内的各个线程分开，使得每一部分对共享存储器独自进行操作，避免出现各自划分的部分访问到共享内存中的数据。

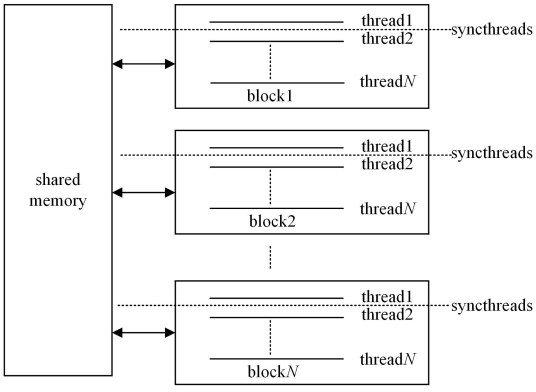


图5 线程同步操作

Fig. 5 Thread synchronization operation

图5的这种划分结构可以避免多个线程同时对共享存储器中的数据进行计算，导致出现结果不一致的情况。算法2是对线程同步操作的具体实现。

算法2 thread synchronization algorithm

Input: bees: colony parameters; employed bees: update the bee

Output: limit: solutions can not be improved

1. if(j<bees):
2. fitness[j]=fit. fitness
3. syncthreads()
4. for i∈[0…generations)do
5. if i<employed bees
6. $x_i'=x_i+\Phi_i(x_i-x_{i'})$
7. if i <onlooker bees/2
8. $P_i=\frac{fit(x_i)}{\sum_{N=1}^{SN} fit(x_n)}$
9. syncthreads()
10. limit[i]=limit[i]+1
11. end for

3.4 任务并行

经过任务划分、数据分布和同步操作对蜜源信息和适应度信息处理后，在DCU端实现数据的计算任务，在CPU端启动任务并行的信号后，在DCU加速部件中执行任务。

从图6可以看出CPU端主要负责计算任务相关参数的初始化和数据的传输过程，ABC算法的计算过程主要传输到DCU加速部件中进行计算。

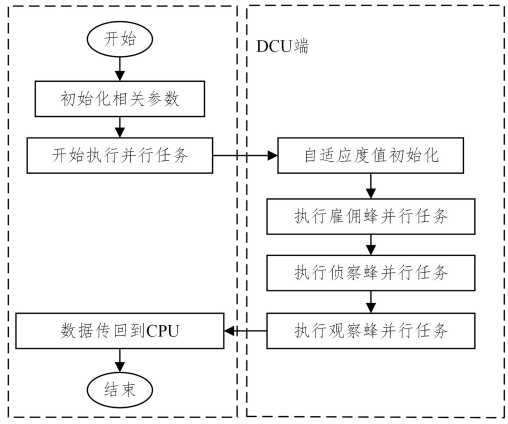


图6 并行任务划分

Fig. 6 Parallel task division

并行任务的划分是先对适应度值进行初始化，判断蜂群数量是否大于分配的线程数量，然后执行各个蜂群种类的并行计算任务，蜜蜂在对应的蜜源及其附近进行搜索，并计算出最好的蜜源，对于满足条件的蜜源，更新其位置信息，然后继续寻找更好的蜜源。算法3是具体并行任务划分的实现过程。

算法3 the task parallel algorithm

Input: blockIdx. x: block index within grid

blockDim. x: count the number of threads in a block in the x direction

threadIdx. x: thread index within block

bees: colony parameters

Output: newFitness: new fitness information

limit: solutionscan not be improved

1. Fitness Initialization
2. hipMemcpy(dev_bees, hipMemcpyHostToDevice)
3. if(i<bees): fitness[i]=fitness(hive[i])
4. parallel artificial bee colony
5. for i∈[0…generations)do
6. if i<employed bees
7. if(newFitness < fitness[i]):
8. fitness[i]=newFitness
9. else
10. limit[i]=limit[i]+1
11. ifi <onlooker bees/2

```
12. if(newFitness < fitness[selected_tournament]):
13. fitness[selected_tournament]=newFitness
14. else
15. limit[i]=limit[i]+1
16. if(i < bees)
17. if(limit[i] >=limit_solutions)
18. newFitness=fitness(hive[i])
```

4　实验结果与分析

4.1　实验环境

实验验证所需要提供的实验平台采用的是国产混合异构的超级计算系统，该系统单个计算节点采用了一颗 32 核心的 Hygon CPU 和 4 块 Hygon DCU 加速部件，使用调度系统对计算任务进行调度和管理，实验环境具体的参数如表 3 所列。

表 3　实验环境

Table 3　Experimental environment

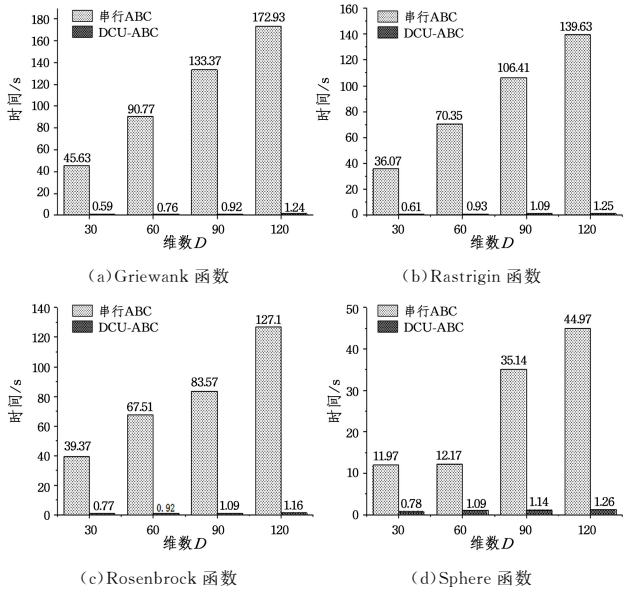
名称	描述
服务器	TC8600H
操作系统	CentOS7.6
CPU	Hygon C86 7165 CPU @2.0 GHz 24 核
DCU	Hygon DCU 1
编译器	GCC7.2
CPU 内存/GB	128

4.2　实验结果与分析

ABC 算法的串行和优化后算法的最大迭代次数都设定为 150，最大循环次数设定为 3000，局部寻找最优解的重复次数按照公式 limit=0.25×NP×D 来计算[21]。Limit 是限度，超过这个限度 EmployedBees 就会变成 ScoutBees；NP 表示蜂群规模的数量；D 表示维数，代表函数变量的个数。

进行两组对比实验，第一组实验在国产海光处理器上进行，在实验测试的过程中，控制蜂群的规模数量 NP 不变，将维数 D 增加。为了对比国产海光处理器的串行人工蜂群算法和优化后人工蜂群算法在不同维数 D 的算法性能，将海光处理器优化后的人工蜂群算法命名为 DCU-ABC 算法，并将 DCU-ABC 算法放到单个计算节点上运行。将蜂群规模的数量 NP 设定为 1000，分别对表 1 的多维函数选取 30，60，90 和 120 这 4 种不同维数进行实验，将表 1 的多维函数作为串行 ABC 算法和 DCU-ABC 算法的测试对象，使用性能分析工具统计出两种算法对不同多维函数的运行时间。每种算法对不同多维函数运行 30 次，取平均值作为结果。图 7 分别是 DCU-ABC 算法和串行 ABC 算法对 Griewank 函数、Rastrigin 函数、Rosenbrock 函数和 Sphere 函数在不同维数下的运行时间，单位为 s。

如图 7 所示，随着不同维数 D 的增加，两种算法的运行时间也相应增加，并行 ABC 算法取得了很好的加速效果。对于 Griewank 函数而言提升的效果最好，串行 ABC 算法在维数为 90 的情况下，运行的时间 133.37s，而 DCU-ABC 算法只有 0.92s，由此数据可以看出 DCU-ABC 算法能够大幅度提升对多维函数的运算效率。对于 Sphere 函数而言提升的幅度不太大，串行 ABC 算法在维数为 60 的情况下，运行的时间 12.17s，而 DCU-ABC 算法为 1.09s。不同多维函数在不同维数的情况下有着不同的性能提升效果，原因在于函数的计算复杂度有所差异。



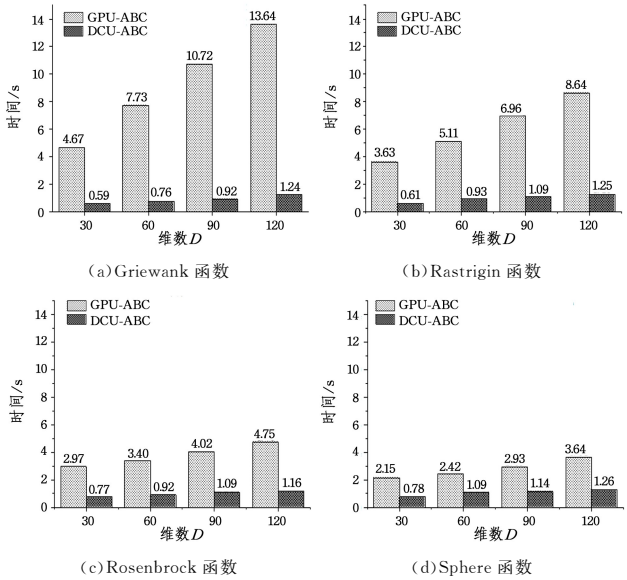
(a)Griewank 函数　(b)Rastrigin 函数

(c)Rosenbrock 函数　(d)Sphere 函数

图 7　串行 ABC 算法和 DCU-ABC 算法不同维数的运行时间

Fig. 7　Running time of serial ABC algorithm and DCU-ABC algorithm in different dimensions

第二组实验是为了进一步评估优化后的 ABC 算法在国产海光处理器对多维函数的求解效率，以国产海光处理器为基础的服务器和科学计算 GPU 服务器(NVIDIA Tesla P100-PCIE-12GB)分别对优化后的 ABC 算法求解多维函数的效率进行了测试对比，将在科学计算 GPU 服务器上优化后的 ABC 算法命名为 GPU-ABC 算法。在实验测试的过程中，将表 1 的多维函数作为 DCU-ABC 算法和 GPU-ABC 算法的测试对象，使用性能分析工具统计出两种算法对不同多维函数的运行时间。蜂群规模的数量 NP 设定为 1000，分别对表 1 的多维函数选取 30，60，90 和 120 这 4 种不同维数进行实验。DCU-ABC 算法和 GPU-ABC 算法对不同多维函数运行 30 次，取平均值作为结果。图 8 分别是 DCU-ABC 算法和 GPU-ABC 算法对 Griewank 函数、Rastrigin 函数、Rosenbrock 函数和 Sphere 函数在不同维数的运行时间，单位为 s。



(a)Griewank 函数　(b)Rastrigin 函数

(c)Rosenbrock 函数　(d)Sphere 函数

图 8　DCU-ABC 算法和 GPU-ABC 算法不同维数的运行时间

Fig. 8　Running time of DCU-ABC algorithm and GPU-ABC algorithm in different dimensions

如图 8 所示，并行 ABC 算法在国产海光处理器对多维函数的求解效率要优于科学计算 GPU 服务器。如 Griewank 函数，设置不同维数，DCU-ABC 算法比 GPU-ABC 算法的求解效率更好，Griewank 函数的维数设置为 30 时，DCU-ABC 算法的求解效率是 GPU-ABC 算法的 7.92 倍；Griewank 函数的维数为 90 时，算法的求解性能最好，DCU-ABC 算法的求解效率是 GPU-ABC 算法的 13.5 倍。通过利用 DCU-ABC 算法和 GPU-ABC 算法对 4 个多维函数求解效率的测试，实验结果表明，并行 ABC 算法在国产海光处理器对多维函数的求解效率性能较好。

结束语 以国产自主研发的海光处理器为基础，针对海光处理器上 ABC 算法中函数性能较低的情况，分别采用了任务划分、数据分布、同步操作和任务并行的方法，缓解了串行 ABC 算法对函数计算效率低、耗时长等问题。为了验证以上优化方法的正确性和有效性，对优化方法进行了实验测试，实验结果表明，相比串行 ABC 算法和 GPU-ABC 算法，该方法能够得到较好的性能提升，提高了对多维函数的运算效率，能够进一步高效处理农业农村发展带来的海量数据。接下来将扩充到多个运算节点，同时还可以采用向量化的优化方法，更加充分地发现算法中的并行性，处理其中的数据依赖关系，大幅度挖掘程序加速比的潜力，最大化发挥以海光处理器为基础平台的各种并行机制的作用，最大程度地提升算法的执行效率。

参 考 文 献

[1] The State Council . Implementation Opinions of the Ministry of Agriculture on Promoting the Development of Agricultural and Rural Big Data. [R/OL]. (2015-09-30)[2016-04-10]. http://www.gov.cn/gongbao/content/2016/content_5061698.htm.

[2] XU S, WANG W, ZHANG J, et al. High Performance Computing Algorithm and Software for Heterogeneous Computing[J]. Journal of Software, 2021, 32(8): 2365-2376.

[3] VIKHAR P A. Evolutionary algorithms: A critical review andits future prospects[C]// 2016 International conference onglobal trends in signal processing. IEEE Computer Society, 2016: 261-265.

[4] AKHTARUZZAMAN M, SHAFIE A A, RAIHAN S M, et al. Golden ratio, the Phi, and its geometrical substantiation[C]// 2011 IEEE Student Conference on Research and Development. ACM, 2011: 425-430.

[5] CHEN Y, BEAULIEU N C. A simple polynomial approximation to the Gaussian Q-function and its application[J]. IEEE Communications Letters, 2009, 13(2): 124-126.

[6] CHAKRABORTY A, KAR A K. Swarm intelligence: A review of algorithms[J]. Nature-Inspired Computing and Optimization, 2017, 10(2): 475-494.

[7] ZHAO R, LIU Q, LI C, et al. Performance Comparison and Application of Swarm Intelligence Algorithms in Crowd Evacuation [C]// Proceedings of the 2020 4th International Conference on Management Engineering. ACM, 2020: 47-51.

[8] KARABOGA D, BASTURK B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm[J]. Journal of Global Optimization, 2007, 39(3): 459-471.

[9] LUO G H, HUANG S K, CHANG Y S, et al. A parallel Bees Algorithm implementation on GPU[J]. Journal of Systems Architecture, 2014, 60(3): 271-279.

[10] BANHARNSAKUN A, ACHALAKUL T, SIRINAOVAKUL B. Artificial bee colony algorithm on distributed environments [C]// 2010 second world congress on nature and biologically inspired computing (NaBIC). IEEE Computer Society, 2010: 13-18.

[11] MUÑOZ D M, LLANOS C H, COELHO L S, et al. Accelerating the artificial bee colony algorithm by hardware parallel implementations[C]// 2012 IEEE 3rd Latin American Symposium on Circuits and Systems(LASCAS). IEEE Computer Society, 2012: 1-4.

[12] KARABOGA D, AKAY B. A comparative study of artificial bee colony algorithm[J]. Appl Math Comput, 2009, 214: 108-132.

[13] KARABOGA D, AKAY B. A comparative study of artificial bee colony algorithm[J]. Applied Mathematics and Computation, 2009, 214(1): 108-132.

[14] GUO B X. Research on Photovoltaic Power Forecasting Based on Intelligent Water Drop Algorithm and Neural Network[D]. Beijing: North China Electric Power University, 2016.

[15] KANG S, QIAN X Z, GAN L. ParallelSaNSDE for Many-Core Sunway Processor[J]. Journal of Frontiers of Computer Science and Technology, 2021, 15(10): 2015-2024.

[16] YUAN L, ZHANG Y Q, BAI X R, et al. Research on Locality-aware Design Mechanism of State-of-the-art Parallel Programming Languages[J]. Computer Science, 2020, 47(1): 7-16.

[17] WANG Y C, HU H, WILLIA M, et al. Performance evaluation of Sugon exascale prototype with GTC-P[J]. Computer Engineering & Science, 2020, 42(1): 1-7.

[18] Introduction toamd gpu programming with hip [EB/OL]. (2019-09-06) [2019-12-23]. https://www.olcf.ornl.gov/wp-content/uploads/2019/09/AMD_GPU_HIP_training_20190906.pdf.

[19] LI J C. Research on Image Dehazing Heterogeneous Acceleration Method Based on FPGA+CPU[D]. Xi'an: Xidian University, 2017.

[20] CHEN P, ZHAO H L, TAO C, et al. Block-run-based connected component labelling algorithm for GPGPU using shared memory [J]. Electronics Letters, 2011, 47(24): 1309-1311.

[21] KARABOGA D, AKAY B, OZTURK C. Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks[C]// International Conference on Modeling Decisions for Artificial Intelligence. IEEE Computer Society, 2007: 318-329.

LI Hui, born in 1978, Ph.D, lecturer. His main research inte- rests include risk management and actuarial and so on.