计算机科学
COMPUTER SCIENCE

基于网格化粒子群搜索算法的最大浮点误差并行检测方法

冀立光, 周蓓, 杨鸿儒, 周玉畅, 崔梦琦, 许瑾晨

引用本文

冀立光, 周蓓, 杨鸿儒, 周玉畅, 崔梦琦, 许瑾晨. 基于网格化粒子群搜索算法的最大浮点误差并行检测方法[J]. 计算机科学, 2026, 53(2): 124-132.

JI Liguang, ZHOU Bei, YANG Hongru, ZHOU Yuchang, CUI Mengqi, XU Jinchen. Parallel Detection Method of Maximum Floating-point Error Based on Gridding Particle SwarmOptimization Algorithm [J]. Computer Science, 2026, 53(2): 124-132.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.250200014

基于网格化粒子群搜索算法的最大浮点误差并行检测方法

冀立光　周　蓓　杨鸿儒　周玉畅　崔梦琦　许瑾晨
信息工程大学网络空间安全学院　郑州 450001
(1194868658@qq.com)

摘　要　浮点计算程序广泛应用于航空航天、人工智能、国防军事、金融结算等领域,浮点程序的计算精度和性能直接关系到相关应用的安全和效果。最大浮点误差值是衡量浮点计算程序精度的核心关键指标,浮点误差的累积效应也会导致难以承受的灾难,因此需要研发一款精准高效的浮点数最大误差检测工具,为研究人员及时采取优化和干预措施提供支撑作用。对此,将浮点数最大误差检测问题转换为目标函数最大值搜索问题,充分发挥国产申威平台的主从架构两级并行计算模式的算力优势,深度挖掘粒子群启发式搜索算法的性能和精度潜能,采用"网格搜索、独立培养、分层汇聚、动态适应"的思想优化粒子群算法,根据搜索过程所处的不同阶段针对性地设置相关搜索参数,使得改进后的算法在搜索精度和搜索性能两个方面均有所提高。该算法为精确检测浮点数最大误差提供了一种新的实用工具和思路参考,同时进一步丰富了国产申威平台的工具库。

关键词:浮点数;误差检测;粒子群优化算法;并行计算;申威平台
中图分类号　TP314

Parallel Detection Method of Maximum Floating-point Error Based on Gridding Particle Swarm Optimization Algorithm

JI Liguang,ZHOU Bei,YANG Hongru,ZHOU Yuchang,CUI Mengqi and XU Jinchen
School of Cyberspace Security,University of Information Engineering,Zhengzhou 450001,China

Abstract　Floating-point computing programs are widely used in aerospace,artificial intelligence,national defense and military,financial settlement and other fields. The computing accuracy and performance of floating-point programs are directly related to the safety and effectiveness of related applications. The maximum floating-point error is the key indicator to measure the accuracy of floating-point computing programs,and the cumulative effect of floating-point errors will also lead to unbearable disasters,so it is necessary to develop an accurate and efficient float-point maximum error detection tool to provide support for researchers to take timely optimization and intervention measures. The proposed algorithm transforms the problem of maximum error detection into the problem of searching for the maximum value of the objective function,gives full play to the computing power advantages of the master-slave architecture two-level parallel computing mode of the domestic Sunway platform,deeply excavates the performance and accuracy potential of the particle swarm heuristic search algorithm,and optimizes the particle swarm algorithm with the idea of grid search,independent cultivation,hierarchical convergence and dynamic adaptation. According to the different stages of the search process,the relevant search parameters are set,so that the improved algorithm achieves improvement in both search accuracy and search performance. This provides a new practical tool and thinking reference for accurately detecting the maximum error of floating-point numbers,and further enriches the tool library of domestic Sunway platform.

Keywords　Floating-point,Error detection,Particle swarm optimization algorithm,Parallel computing,Sunway platform

1　引言

人工智能的进步正以前所未有的速度助推人类的发展,数值计算、逻辑推理、智能聊天、方案撰写、音视频编辑等功能越来越成熟[1]。这些看似神奇的功能背后,实则都需要十分强大的算力作支撑。云计算、大数据、机器学习等学科的发展,无一例外都离不开高性能计算[2]。计算机等运算设备在进行科学计算时,主要以处理浮点数为主,因此,实数在计算机中以浮点数的形式存储并参与运算[3]。但是用浮点数近似存储实数时,本身就会产生误差,并且在运算的过程中还存在计算误差的累积效应,可能导致计算结果误差较大,甚至是完全错误[4]。同时,浮点数的计算是非自包含的,计算结果还会产生溢出、除零错等浮点异常情况,这也会造成计算过程中断或者无法产生正确运算结果的情况。如果不能及时准确地检测出浮点数计算的最大误差,不把误差控制在合理的范围,就会导致灾难性的后果。卫星导航系统高度依赖浮点数计算精度来确定接收机的位置信息,如果浮点误差太大,就会导致偏航等危险。海湾战争期间,沙特阿拉伯的达摩地区部署的美

到稿日期:2025-02-05　返修日期:2025-04-26
通信作者:许瑾晨(atao728208@126.com)

国爱国者导弹,其导航系统浮点计算误差累积效应导致浮点误差超过规定范围,未能成功拦截伊拉克的飞毛腿导弹。飞毛腿导弹最终击中了美国的一个兵营,造成士兵死亡的重大事故[5]。金融结算对计算的精度和性能要求更高,一旦出错,后果十分严重。例如,在 JavaScript 中使用 double 类型进行货币累加运算时,可能会出现精度损失,导致最终结果与预期不符。1982 年,温哥华证券交易所推出了一项初始值为 1000 的股票指数,舍入误差导致每天丢失一点指数,这个问题直到 22 个月后才被发现,当时的指数是 524.811,而实际指数是 1098.892。无独有偶,由于浮点数转整数出现的整数溢出异常,在 1996 年欧洲 Ariane-5 火箭测试发射时出现了严重的升空自爆现象[6],造成了 3.7 亿美元的巨额经济损失。此外,在 2010 年,由控制软件异常导致的丰田汽车刹车问题,同样给个人带来了极大的安全隐患[7]。由此可见,浮点误差的危害不容小觑,必须正确处理。

浮点数计算是科学计算不可或缺的重要组成,浮点数计算的精度和性能直接影响高性能计算应用的效率和准确性[8]。人类要想利用机器进行快速复杂的运算来解决科学研究和工程实验中的问题,就必须合理利用浮点数进行计算,将误差控制在一个可接受的程度。最大浮点数误差是度量浮点数计算误差大小的关键指标,如果最大误差在允许的区间内,就不会产生重大危害,浮点数计算仍然可以支撑人类的科学研究和工程实验[9]。因此,设计一款准确有效的浮点数最大误差检测工具意义重大,影响深远。

通过梳理和分析浮点数误差检测研究的发展脉络和主要方法,并借鉴相关学科的理论和工程经验,将浮点数最大误差检测问题转换为搜索问题,同时引入粒子群搜索算法并加以优化改造,利用并行计算的模式缩小每个进程对应的搜索区间长度,增加粒子群规模和迭代次数,从而提高最大浮点数误差的检测速度和精度,生成一款可靠高效的浮点误差检测工具,进而减少浮点误差导致的严重后果。

本文的主要创新点如下:

1)将浮点数最大误差检测问题转换为目标函数的最优解搜索问题,可以充分借助现有搜索算法的快速迭代发展来解决检测最大浮点误差问题;

2)针对浮点误差检测研究的特点,按照“网格搜索、独立培养、分层汇聚、动态适应”的思路对粒子群搜索算法逻辑进行优化;

3)充分发挥国产申威平台的主从架构两级并行计算能力,对优化后的粒子群算法进行并行化改编,在检测精度和性能两个方面都实现了提升。

本文第 2 章介绍了浮点误差检测和粒子群算法的研究现状,第 3 章介绍了基础知识,第 4 章详细论述了基于网格化粒子群搜索算法的最大浮点误差并行检测方法的实现,第 5 章进行了对比实验和结果分析,最后总结全文并展望未来。

2 研究现状

研究浮点数最大误差检测是一个涉及多个领域的重要课题,随着计算机硬件性能的提升和各类算法的不断提出,人们对浮点计算的精确性和可靠性提出了更高的要求。浮点数是实数的一个子集,其分布是不均匀的[10]。在计算机中,每一个实数都需要用相邻的浮点数近似表示并进行各类计算,这本身就会产生舍入误差。舍入误差的积累在多次运算后尤为明显,可能导致最终结果与预期值相差甚远。同时,在计算过程中,计算机会对浮点数进行规格化处理,当一个特别大的浮点数和一个很小的浮点数相加时,在浮点数相加之前,通常需要将两个数的指数对齐。如果两个数的指数相差很大,那么在对齐过程中,小数可能需要向右移动很多位(即乘以 2 的某个幂次),这可能导致其有效位数减少,甚至变为 0(即发生下溢),从而无法在最终结果中体现出来,产生大数“吃”小数的情况,这种情况也会增加浮点数误差。面对浮点误差产生的不同原因和机理,科研工作者也设计出不同的浮点误差检测和精度优化方法。目前主流的技术路线是将浮点误差检测问题转换为目标函数最值搜索问题,利用现有各类成熟高效的启发式搜索算法助力最大误差检测,在搜索空间内搜索触发最大浮点误差值的输入值[11]。2014 年以来,一些研究人员使用搜索的方法对浮点误差进行初步探索[12-13]。这些研究人员直接使用现有的搜索算法与框架,未针对浮点数误差问题的特性进行深入研究,导致误差的检测效率普遍不高。之后,更多的科研人员通过结合浮点数分布规律和浮点数计算特性,对搜索算法进行针对性改进和优化,使得浮点误差检测的精度不断提高,比较常见的搜索算法有 LGSA[14],BEA[15],DEMC[16]等。还有研究人员通过检测触发浮点异常的条件来检测浮点误差。2013 年,Barr 等[17]提出工具 Ariadne。该工具利用浮点数的特定性质,将浮点数理解为实数运算,给定一系列会触发异常的条件;然后使用约束求解器求解符号执行的路径约束和触发异常的条件约束,找到可能会触发异常的实数值,再检测对应的浮点数值及其临域的浮点数值是否会触发异常。Ariadne 使用 KLEE 作为符号执行工具,使用 Z3 作为约束求解工具。

对于浮点数最大误差检测,选用高效精准的搜索算法非常重要,粒子群搜索算法参数丰富,搜索精度和搜索性能较好。通过研读和梳理相关研究文献,发现针对粒子群算法[18]进行各类优化和改进的方法较为常见,主要是以粒子群算法为主体,深度融合遗传算法[19]、模拟退火算法[20]、文化算法等各类算法的优势,增强优化后搜索算法的全局搜索能力和局部搜索精度,提升优化后算法整体的搜索精度和性能。

综合来看,目前的浮点数最大误差检测研究方法发展较快,各类检测算法精度不断提高,性能越来越好,但是主要以串行计算和小规模并行计算为主,采用众核处理器的大规模并行计算模式较少,特别是专为国产申威平台量身设计的算法数量更为有限。因此,本文将最大误差检测问题转换为目标函数最大值搜索问题,这样就可以充分利用搜索算法研究成果来支撑最大浮点误差检测研究。鉴于粒子群算法参数设置灵活,搜索逻辑简洁,搜索精度较高,因此本文基于标准粒子群算法,按照“网格搜索、独立培养、分层汇聚、动态适应”的原则对其进行算法逻辑优化和并行化改编,设计出一款适用于国产申威并行计算平台的浮点数最大误差并行检测工具,来支撑浮点数最大误差检测研究,同时也进一步丰富申威平台的工具库。

3 基础知识

3.1 粒子群算法

粒子群搜索算法(Particle Swarm Optimization, PSO)是1995年由Eberhart教授和Kennedy博士提出的[21-22],是一种利用群体智能(Swarm Intelligence, SI)实现优化的方法。该算法的主要思想来源于对鸟群捕食行为时信息传递的仿生算法。Kennedy和Eberhart发现,鸟群在飞行过程中经常会出现突然改变方向、散开、聚集等行为,这些行为看似不可预测,但是鸟群整体的行为却保持着一定的规律;同时,个体与个体之间也保持着最适宜的距离。通过对这种类似生物群体行为的研究,研究人员发现生物群体中存在着一种社会信息共享机制,这种机制为群体的进化提供了一种参考。在粒子群算法中,每一个粒子代表鸟群中的一只鸟,表示待优化问题一个可能的解。粒子在解空间中的搜索方式就是模拟鸟群的觅食过程,通过模拟群体(如鸟群、鱼群等)的集体行为来寻找问题的最优解。粒子群算法的主要执行过程如下。

1)粒子表示:每个粒子代表待优化问题的一个解,具有位置和速度两个属性。位置参数表示解在搜索空间中的当前状态,速度参数决定粒子如何移动以搜索新的解。

2)初始化:在算法开始时,用特定算法生成一组粒子,并初始化它们的位置和速度等粒子群参数。

3)速度更新:粒子群中每个粒子的速度根据粒子本身的历史最优位置和整个粒子群的历史最优位置进行更新。速度更新公式通常包括当前速度、认知部分和社会部分3个部分。

4)位置更新:粒子的新位置由其当前位置和更新后的速度共同决定。

5)迭代过程:粒子群算法通过不断迭代更新粒子的速度和位置,使粒子群逐渐趋近于最优解。在每次迭代中,都会评估每个粒子的适应度,即目标函数的值,并更新局部最优值和全局最优值。

6)终止条件:算法通常设置一定的终止条件,如达到最大迭代次数、满足预设的精度要求或解的质量不再提高等,以结束迭代过程。

3.2 浮点数概述

浮点数F是实数R的一个子集,在计算机中以二进制形式存储,具体表达方式是以2为底的科学计数法形式。本文研究的浮点数采用的是IEEE-754(2008)标准,主要依据Muller等[23]的著作。其规范形式为$f=(-1)^s\times m\times 2^e$,分为3部分,分别是符号位$s$、指数位$e$、尾数位$m$。因为浮点数的特殊结构,浮点数的分布是不均匀的,呈现出越靠近0越密集、越远离0越稀疏的规律,分布趋势如图1所示。

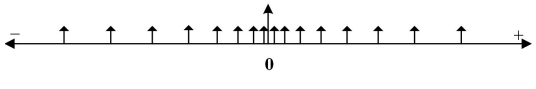

图1 浮点数分布示意图

Fig. 1 Illustration of floating-point number distribution

因为浮点数是以科学计数法表示的,所以浮点数的表示方式并不唯一。为了方便计算和提高浮点数精度,通常会将浮点数按照固定的格式变换,使用规格化的科学记数法。

规格化的规范要求为:尾数的最高有效位必须为1(在IEEE 754标准中,这个1是隐含的,不直接存储在尾数字段中),并且尾数的其他位尽可能用来表示有效数字。例如,数值大小相同的两个二进制浮点数,$a_0=(-1)^0\times 1.0101\times 2^{-1}$和$a_1=(-1)^0\times 0.010101\times 2^1$,其中$a_0$就是规格化的形式,$a_1$就不是规格化后的形式。对于二进制的浮点数系统,规格化数的尾数位的首位默认为1,非规格化数默认为0,都为隐藏位,无需明确存储。

规格化操作有以下两种基本类型。

1)右规:当尾数的有效位数不足时,需要将尾数右移,同时阶码增加,直到尾数变成规格化形式。右规操作可能会导致阶码溢出,需要特殊处理。

2)左规:当尾数的有效位数超出尾数字段的范围时,需要将尾数左移,同时阶码减少,直到尾数变成规格化形式。左规操作通常不会导致阶码溢出,但可能会导致尾数丢失有效位。

3.3 浮点误差类型

3.3.1 绝对误差

浮点程序的计算值$f(x)$和理论精确值$F(x)$之差的绝对值就是绝对误差($Error_{\text{abs}}$)[24]。

$$Error_{\text{abs}}=|f(x)-F(x)| \tag{1}$$

绝对误差直接表示测量值与真实值之间的差异,使得人们能够直观地了解测量的准确程度。但是绝对误差只考虑了测量值与真实值之间的绝对差异,没有考虑到测量值本身的大小。因此,在比较不同量级的测量值时,绝对误差可能无法准确反映测量的相对准确性。

3.3.2 相对误差

绝对误差与真实值的比值就是相对误差($Error_{\text{rlt}}$)[25],浮点数的相对误差通常用百分比表示,它反映了计算机进行浮点数运算的精度限制。相对误差是一个比值,没有单位,这使得它能够在不同单位或量级的测量值之间进行比较,提供了更通用的误差评估标准,具体计算式如下:

$$Error_{\text{rlt}}=\left|\frac{f(x)-F(x)}{f(x)}\right| \tag{2}$$

然而,相对误差对测量值大小敏感且是一个比值,因此当测量值较小时,即使绝对误差很小,相对误差也可能很大。这可能会导致其对某些精确测量的准确度评估产生偏差。

3.3.3 Bits误差

本文的对比实验中采用了Bits误差($Error_{\text{Bits}}$)。Bits误差的定义如式(3)、式(4)所示:

$$DB_{\text{number}}\{F(x),f(x)\}=|\{a_i\in\mathbb{F}|\min(F(x),f(x))\leqslant a_i\leqslant\max(F(x),f(x))\}| \tag{3}$$

$$Error_{\text{Bits}}\{F(x),f(x)\}=\log_2(DB_{\text{number}}\{F(x),f(x)\}) \tag{4}$$

其中,DB_{number}表示精确值$F(x)$与计算值$f(x)$之间相差的浮点数的位数,$Error_{\text{Bits}}$度量精确值$F(x)$与计算值$f(x)$两者的数值在数位上有多少比特位是不同的。例如,双精度浮点数1.0和2.0之间相差4 503 599 627 370 496个浮点数,对其取log2得52,说明1.0和2.0之间相差了52位,Bits误差为52。

4 算法实现流程

基于网格化粒子群搜索算法的最大浮点误差并行检测方法(Parallel Detection Method of Maximum Floating-point Error Based on Gridding Particle Swarm Optimization Algo-

rithm，PSOPDM)，在标准粒子群搜索算法的基础上进行优化和改进，并在国产申威平台上进行并行化实现和执行，最后应用于浮点数最大误差检测研究。算法的总体设计思路可以概括为网格搜索、独立培养、分层汇聚、动态适应的并行处理流程。优化后的算法有效改善了标准粒子群算法“局部搜索能力强，全局搜索能力不足”的先天缺陷。具体地，首先将搜索区域合理分割为相互完全独立的子区域，在各自的子区域中定制粒子种群的规模，并针对性地设置相应区域的搜索参数，目的是让不同的粒子种群在所属区域多样拓展，独立搜索，从根本上避免粒子群之间的干扰和误导，从源头上避免粒子群搜索算法过早地收敛于局部极值处并无法跳出的“早熟”现象。算法在各搜索区域上独立培养粒子种群之后，各自并行迭代执行优化后的粒子群算法，计算出每一个搜索子区域中的局部最大浮点误差值，最终通过数据汇聚筛选模块处理输出整个搜索区域的全局最大浮点误差值。具体流程如图 2 所示。

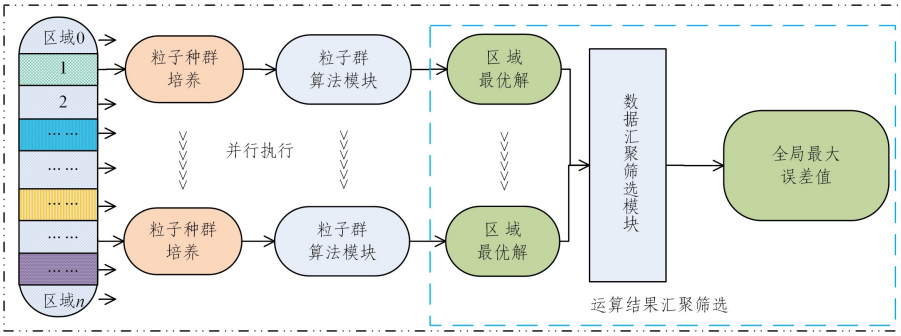

图 2 PSOPDM 算法流程

Fig. 2 Flow of PSOPDM algorithm

4.1 精准分割搜索区域

标准粒子群搜索算法(PSO)在迭代初期容易陷入局部最优值，导致其全局搜索能力较差[26]。为了从源头上彻底解决这一问题，本文算法深度借鉴国产申威平台的并行架构特点，结合浮点数分布规律，将目标搜索区域即待检测表达式的定义域进行网格化分割，缩短每一个搜索子区域的长度。本文算法充分利用参与并行计算的申威平台的核组号(Rank-ID)和计算从核号(My-ID)将搜索区域进行网格化分割：靠近“零”端的搜索子区域长度较小，因为在该处附近的浮点数分布密集，同时该区域也是浮点误差较大的热点区域；远离“零”端的搜索区域长度梯度递增，因为远离“零”端的浮点数分布密度逐步降低，触发最大浮点数误差的概率下降。各搜索区域保持完全独立，粒子种群间相互隔离，互不干扰，在所属的区域内多向拓展，深度搜索，从根本上避免了经典粒子群搜索算法容易陷入局部最优值而难以跳出极值点的缺陷，增强了粒子群对全局搜索的能力。首先利用核组号(Rank-ID)将目标搜索区域划分为不同的汇聚搜索区域，即一个核组处理一块汇聚搜索区域。核组内的从核号(My-ID)将所在的汇聚搜索区域进行二次划分，将汇聚搜索区域分割为 64 块末端搜索区域，每个计算从核负责一块末端搜索区域的搜索任务，分割的基本原理如图 3 所示。

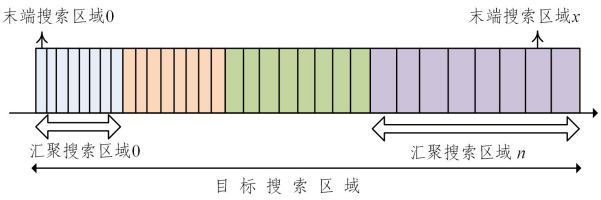

图 3 搜索区域分割示意图

Fig. 3 Illustration of search area segmentation

4.2 粒子种群培养

不同的搜索区域在长度和触发最大浮点数误差的概率方面不同，因此必须针对各搜索区域进行粒子种群的定制化培养，确保粒子种群的搜索能力与所属搜索区域的特点高度契合，达到提升搜索效率和精度的目的。粒子种群培养主要在粒子群规模、搜索参数设置两方面进行定制化设计，遵循的基本原则是在浮点数密集的搜索区域设置的粒子数量较多，在浮点数稀疏的搜索区域设置的粒子数量相对较少。具体的搜索参数设置将结合 4.3 节粒子群算法模块进行详细介绍。

4.3 粒子群算法模块

标准粒子群算法具有参数设置较少、搜索速度快、逻辑简单的特点，应用范围广泛。标准粒子群算法的数据处理过程主要有输入、粒子群初始化、粒子评价、最优解更新、粒子群速度和位置更新、输出等主要环节。标准粒子群算法执行流程如图 4 所示。

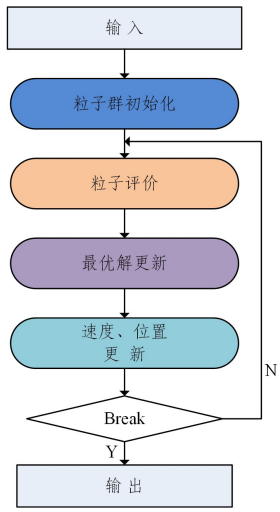

图 4 标准粒子群算法执行流程图

Fig. 4 Flowchart of standard particle swarm optimization algorithm execution

4.3.1 输入

粒子群算法的输入主要包含搜索区域 Rang 和目标优化函数 $F(x)$。对于单参数的目标函数,搜索区域是一个区间,为搜索算法提供了优化的评价依据和搜索区域。

目标优化函数 $F(x)$:浮点数最大误差检测就是检测出浮点数计算误差值中的最大值。设 $G(x)$是在浮点数 x 处计算的精确值,$g(x)$是在浮点数 x 处的计算值。在本文算法中,精确值 $G(x)$的计算是调用 128 位的高精度数学函数库双 Double 库来执行计算的,双 Double 库不仅精度高,计算性能也很优秀。求解相对误差($Error_{rlt}$)时,目标优化函数 $F(x)=|G(x)-g(x)|/g(x)$。求解 Bits 误差($Error_{Bits}$)时,$F(x)=\log_2(DB_{number}\{G(x),g(x)\})$,$DB_{number}\{G(x),g(x)\}$是精确值 $G(x)$与计算值 $g(x)$之间相差的浮点数的个数。Bits 误差表示精确值和计算值两者的数值在数位上有多少比特位是不同的。其他搜索参数会在粒子群初始化时完成相关配置和设定。

4.3.2 粒子群初始化

设置算法参数:优化后的粒子群算法在初始化过程中需要设置算法的一些重要参数,主要包括粒子种群的规模(*NUM_PARTICLES*)、个体学习因子和社会学习因子(c_1,c_2)、惯性系数(*INERTIA_WEIGHT*)以及最大迭代次数(*MAX_ITER*)。部分重要参数初始化数据如表 1 所列。

表 1 粒子群算法主要参数信息

Table 1 Main parameters information of particle swarm optimization algorithm

参数名称	参数符号	初始值	是否动态调整
粒子群规模	*NUM_PARTICLES*	1000	是
个体学习因子	c_1	0.3	是
社会学习因子	c_2	2	是
惯性系数	*INERTIA_WEIGHT*	0.9	是
最大迭代次数	*MAX_ITER*	1000	否

初始化粒子:定义一个结构体数组,包含位置(Position)、速度(Velocity)、个体最优位置(Personal-best-position)和个体最优值(Personal-best-value)4 个元素,在搜索空间中随机生成每个粒子的初始位置、初始速度,将每个粒子的初始位置设为其个体最优位置,根据目标优化函数 $F(x)$和初始个体位置(Position)计算得出初始个体最优值(Personal-best-value)。当算法参数设置和各粒子个体完成初始赋值后,粒子群初始化工作就全部完成了。

4.3.3 粒子评价

粒子群中的粒子个体在初始化工作完成后需要根据目标优化函数($F(x)$)计算每个粒子的适应度值(*objective-function*( )),适应度值在后续的搜索和计算过程中用于评估粒子个体最优解的优劣,是粒子群算法的一项重要指标。

4.3.4 最优解更新

最优解更新主要分为粒子个体最优解更新和粒子群最优解更新。比较每个粒子个体当前的适应度值与历史适应度值,在本文算法中取最大值作为该粒子个体的最优适应度值。粒子群中的全部粒子个体最优适应度值的最大值设置为全局最优值(G-best),各粒子同步更新对应数据。

4.3.5 速度、位置更新

粒子群初始化完成后,不断迭代更新粒子的各项元素,包括速度、位置等。

$$V_{(i+1)}=w\cdot V_i+c_1\cdot(P\_best_i-x_i)+c_2\cdot(G\_best_i-x_i) \tag{5}$$

$$X_{(i+1)}=X_i+V_{(i+1)} \tag{6}$$

式(5)是粒子速度更新公式,粒子当前的速度$V_{(i+1)}$决定了该粒子的运动速度和运动方向,粒子之前的速度V_i反映了粒子之前的状态,代表粒子对自身状态的记忆,直接决定了粒子自身全域开拓,全局搜索的能力。w 是惯性权重参数,是表示过去速度对当前速度影响程度的重要参数,通过调整惯性权重参数的大小来平衡全局搜索和局部搜索之间的关系。较大的惯性权重,对提高粒子的全局搜索能力有利;较小的惯性权重参数,对提高粒子的局部搜索精度有利[27]。惯性权重太大,粒子群容易错过最优解,导致算法不收敛。为了提高标准粒子群的整体搜索效果,本文算法动态调整惯性参数,实现粒子惯性权重参数的定制化设置,算法在迭代搜索的过程中,将惯性权重 w 从 0.9动态减小为 0.2,保证了粒子群兼顾全局搜索的“大范围”和局部搜索的“高精度”。

式(5)中第二部分$c_1\cdot(P_best_i-x_i)$(其中c_1是个体学习因子)是粒子的“认知部分”,代表粒子独立思考和独立搜索的能力。第三部分$c_2\cdot(G_best_i-x_i)$(其中c_2是社会学习因子)是粒子的“社会部分”,反映了粒子个体受到粒子群体的影响程度的大小,体现了粒子群内部的信息共享和协同能力。学习因子c_1和c_2共同作用,以平衡“自身能力”和“群体协作”对于粒子运行到个体最优值和全局最优值的加速权重。在搜索前期,为避免陷入局部极值难以跳出,需要粒子群进行广泛的全局搜索,因此$c_2>c_1$。在搜索后期,需要进一步提升局部搜索的能力,因为粒子群初步定位到了最优值的基本范围,因此需要提高每个粒子个体独立搜索的能力,也就是需要增强认知能力,因此$c_1>c_2$。

粒子群中各粒子在速度和位置更新后,依据目标优化函数计算新的个体极值,同步更新群体最优值,循环迭代,直到搜索到满足算法预定设置的条件或者达到迭代次数的最大值,结束迭代,输出结果。

4.3.6 输出

粒子群算法模块的输出值是该模块所属搜索区域的局部最大误差值Err_{Local}。

4.4 运算结果汇聚筛选

本文算法充分利用国产申威平台主从异构两级并行架构设置算法数据汇聚流程。目标搜索区域通过核组号和从核号分割为一个个独立的较小的末端搜索区域,每个核组所属的 64 个末端搜索区域共同形成汇聚搜索区域,整个搜索区域划分为不同的汇聚搜索区域。每个末端搜索区域都会并行分配一个计算从核搜索所属末端搜索区域的末端最

大误差值Err_{End}，从单个核组负责的汇聚搜索区域筛选出64个从核末端最大误差值中的汇聚层最大误差值Err_{Lyer}，最后利用*MPI_Reduce*()函数计算出整个搜索区域中浮点数误差的最大值——全局最大误差Err_{All}。数据汇聚流程如图5所示。

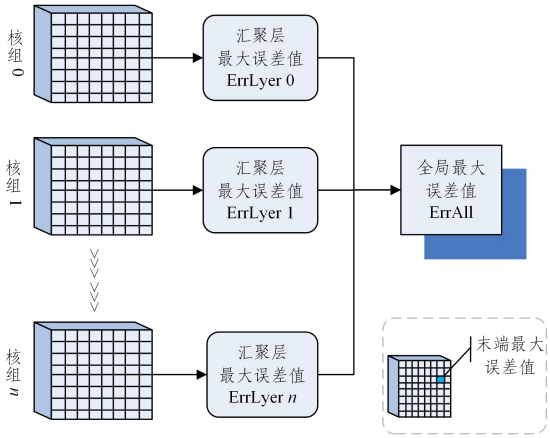

图5 数据汇聚流程图

Fig. 5 Flowchart of data aggregation

5 实验分析

PSOPDM算法和Herbie工具[28]、S3FP工具[12]以及HSED工具[29]的对比实验是在国产某申威平台上执行的。该申威平台上部署的是国产众核处理器，每一个CPU包含4个核组，每个核组包含64个计算从核(核心阵列)与1个管理主核(运算控制核心)。每个核组配置共享内存，每个从核配置有局部存储，计算从核从局部存储读取数据的速度快于从共享存储中读取数据的速度。该国产申威平台支持主从异构两级并行架构模型，提供与国际接轨的并行编程标准支持，包括MPI3.0，OpenMP3.1，Pthreads，OpenACC2.0，支持消息并行模型、共享编程模型、加速编程模型，能满足科学计算课题移植和开发的多样性需要。

该国产申威平台上编写程序采用并行C语言，分为主核代码和从核代码两个独立子程序，二者分别编译后需要再次混合编译，方可执行程序，得出最大误差和运行时间。在对比实验中，与对应的串行程序对比性能。此外，为了验证PSOPDM检测浮点算术表达式误差的有效性，本文使用FPBench基准测试集[30]中32个单参算术表达式与目前主流的误差检测工具Herbie，S3FP以及HSED进行精度评估和性能评估。其中Herbie采用的是Bits误差，S3FP和HSED采用的是相对误差。

5.1 测试用例

本文的基准测试信息如表2所列，其中FPBench基准测试集中共有46个单参算术表达式，本文选取了32个作为测试对象，排除了14个包含循环或判断或表达式重复的基准。检测区间依据FPBench给定的默认区间，如果FPBench没有给出默认区间，则依据[0.01，100]区间，该区间为常用区间。

表2 测试用例信息

Table 2 Test cases information

编号	FPBench	测试区间
1	sqroot	[0，1]
2	sqrt_add	[1，1000]
3	exp1x	[0.01，0.5]
4	exp1x_log	[0.01，0.5]
5	NMSEexample37	[0.01，100]
6	NMSEproblem336	[0.01，100]
7	NMSEexample39	[0.01，100]
8	NMSEproblem341	[0.01，100]
9	NMSEsection311	[0.01，100]
10	NMSEproblem345	[0.01，100]
11	NMSEproblem337	[0.01，100]
12	verhulst	[0.1，0.3]
13	predatorPrey	[0.1，0.3]
14	logexp	[0.01，8]
15	sine	[−1.570796326 79，1.570796326 79]
16	carbonGas	[0.1，0.5]
17	NMSEproblem341	[0.01，100]
18	NMSEexample38	[0.01，100]
19	NMSEproblem334	[0.01，100]
20	NMSEproblem333	[0.01，100]
21	NMSEproblem331	[0.01，100]
22	NMSEexample36	[0.01，100]
23	NMSEexample35	[0.01，100]
24	NMSEexample34	[0.01，100]
25	NMSEexample31	[0，100]
26	test05_nonlin1_r4	[1.000 01，2]
27	test05_nonlin1_test2	[1.000 01，2]
28	intro-example-mixed	[1，999]
29	sineOrder3	[−2，2]
30	bsplines3	[0，1]
31	NMSEexample310	[0.001，1]
32	NMSEproblem343	[0.001，1]

5.2 精度结果及分析

将PSOPDM与目前几款主流的误差检测工具Herbie，S3FP，HSED进行对比。针对相同的测试用例，在相同的输入区间内，将不同检测工具检测出的最大误差值Err_{max}进行对比。检测出的最大误差值更大，说明其检测能力更优。

5.2.1 PSOPDM与Herbie的精度对比实验

Herbie输出的是Bits误差，因此PSOPDM与Herbie对比的是Bits误差，实验对比结果如图6所示，图中横坐标为选取的测试用例(因为区域有限，按照5.1节中测试用例的顺序，选用编号代替测试用例的名称)，纵坐标为Bits误差的值。

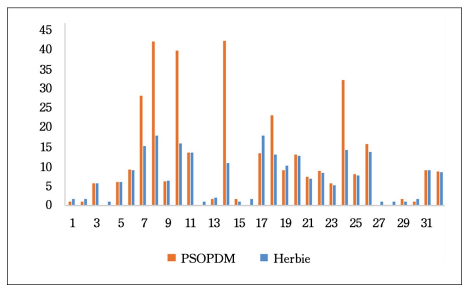

图6 PSOPDM和Herbie精度对比

Fig. 6 Precision comparison between PSOPDM and Herbie

在PSOPDM与Herbie对比实验的32个测试用例中，PSOPDM有18项优于Herbie，其中有7项优势明显；有8项

差于 Herbie;6 项差别不大。实验表明,PSOPDM 的检测精度整体效果优于 Herbie 的检测精度。

5.2.2 PSOPDM 与 S3FP 精度对比实验

S3FP 输出的是最大相对误差,因此 PSOPDM 与 S3FP 对比的是最大相对误差。PSOPDM 与 S3FP 的对比实验结果如图 7 所示,图中横坐标是测试用例的编号值,纵坐标为最大相对误差(x)的数学变换 $f(x)=\log_{10}(x)+20$。

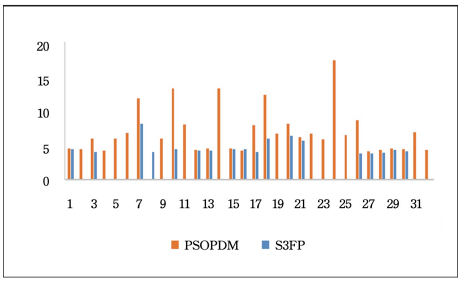

图 7 PSOPDM 和 S3FP 精度对比

Fig. 7 Precision comparison between PSOPDM and S3FP

在 PSOPDM 与 S3FP 的对比实验中,PSOPDM 有 30 项优于 S3FP,其中有 20 项优势明显。S3PF 工具需要指定 TIMEOUT 参数,以控制搜索时间预算,有 14 项测试用例无法得出最终结果。实验表明,PSOPDM 的检测精度优于 S3FP 工具的检测精度。

5.2.3 PSOPDM 与 HSED 精度对比实验

HSED 输出的是最大相对误差,因此 PSOPDM 与 HSED 对比的是最大相对误差。PSOPDM 与 HSED 对比实验结果如图 8 所示,图中横坐标是测试用例的编号值,纵坐标为最大相对误差(x)的数学变换 $f(x)=\log_{10}(x)+20$。

在 PSOPDM 与 HSED 的对比实验中,在 32 个测试用例中,PSOPDM 有 17 项优于 HSED,其中有 3 项优势明显;同时,有 10 项实验结果基本相等,差别很小;有 5 项对比结果小于 HSED。实验表明,PSOPDM 工具的检测精度优于 HSED 工具的检测精度。

精度对比实验结果表明,PSOPDM 工具在检测浮点数计算最大误差方面具有较高的精度,实验精度高于 Herbie,S3FP,HSED 检测工具。PSOPDM 工具在和 S3FP 以及 HSED 检测工具的精度对比实验中,实验效果要优于和 Herbie 精度的对比实验效果,这说明 PSOPDM 工具检测相对误差方面的精度要优于 Bits 误差检测。

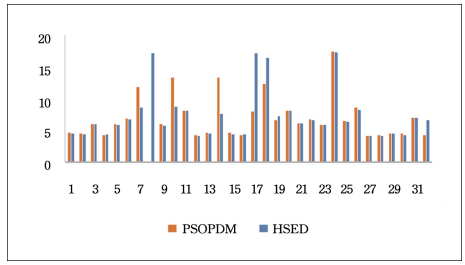

图 8 PSOPDM 和 HSED 精度对比

Fig. 8 Precision comparison between PSOPDM and HSED

5.3 粒子群算法优化前后精度对比

PSOPDM 算法是基于经典粒子群算法(PSO),经过针对性优化设计,并且在国产申威平台上进行并行化改编执行的综合优化算法。本节随机选取了 8 个测试用例,具体名称和测试区间信息如表 3 所列。对比实验结果以相对误差为例,对比未经优化的经典粒子群算法和经过综合优化后的 PSOPDM 算法在最大浮点误差检测领域的搜索精度,实验结果如表 4 所列。

表 3 测试用例信息

Table 3 Test cases information

编号	测试用例名称	测试区间
1	sqroot	[0,1]
2	NMSEproblem336	[0.01,100]
3	NMSEproblem337	[0.01,100]
4	carbonGas	[0.1,0.5]
5	NMSEproblem331	[0.01,100]
6	test05_nonlin1_r4	[1.00001,2]
7	NMSEexample310	[0.001,1]
8	NMSEproblem343	[0.001,1]

表 4 PSO 和 PSOPDM 的搜索精度值

Table 4 Search accuracy values of PSO and PSOPDM

对比算法	sqroot	NMSE-problem336	NMSE-problem337	carbonGas	NMSE-problem331	test05_nonlin1_r4	NMSE-example310	NMSE-problem343
PSO	4.41885×10^{-16}	8.86320×10^{-14}	5.53210×10^{-13}	2.08490×10^{-16}	1.64924×10^{-14}	4.13671×10^{-13}	1.11355×10^{-13}	2.21856×10^{-16}
PSOPDM	4.44038×10^{-16}	8.92610×10^{-14}	1.65071×10^{-12}	2.08504×10^{-16}	1.77950×10^{-14}	5.54837×10^{-12}	1.12355×10^{-13}	2.24045×10^{-16}

分析 PSOPDM 算法和 PSO 算法的对比实验数据发现,对于同一测试用例和相同的测试区间,PSOPDM 算法检测出的最大浮点误差值全部高于 PSO 算法检测出的最大浮点误差值,说明 PSOPDM 算法的搜索精度全面高于 PSO 算法的搜索精度,综合优化结果有效。

5.4 算法性能对比

5.4.1 并行版本程序和串行版本程序性能对比

为了对比实现相同功能的串行版本和并行版本的基于网格化粒子群搜索算法的最大误差检测算法的运行性能,选取了 NMSEexample39,logexp,NMSEproblem331,test05_nonlin1_r4 这 4 个测试用例。在并行版本中的程序调用了 60 个核组,每个核组包含 64 个计算从核,每个计算从核的粒子群规模为 1000,粒子群的规模总计达到了 3840000 个。在串行版本中的算法也采用同样规模的粒子群,其余搜索参数也保持一致。二者执行相同的程序,运行时间如图 9 所示。

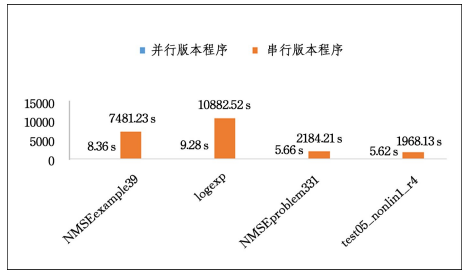

图 9 串行程序和并行程序性能对比

Fig. 9 Performance comparison between serial and parallel programs

从图 9 中可以看出，并行版本的基于网格化粒子群搜索算法的最大误差并行检测算法在执行时间上远远小于串行版本的基于网格化粒子群搜索算法的最大误差检测算法，二者相差了 3 个数量级。由此可见，并行计算模式的应用极大地提高了最大浮点误差检测算法的性能。

5.4.2 PSOPDM 和 S3FP 性能对比

本小节选取了 NMSEexample39，logexp，NMSEproblem331，test05_nonlin1_r4 这 4 个测试用例，分别利用 PSOPDM 工具和 S3FP 工具执行误差检测，对比两者的检测性能，实验结果如图 10 所示。

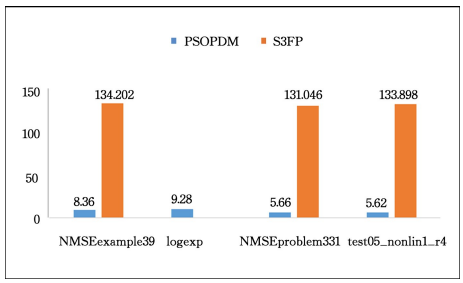

图 10 PSOPDM 和 S3FP 的性能对比

Fig. 10 Performance comparison between PSOPDM and S3FP

从图 10 中可以看出，4 个测试用例中，PSOPDM 工具在 NMSEexample39，NMSEproblem331 和 test05_nonlin1_r4 这 3 个测试用例的执行性能优于 S3FP 工具。其中在 logexp 测试用例中，S3FP 工具无法在规定的时间内检测出结果。因此，PSOPDM 工具执行的时间全部小于 S3FP 工具的执行时间，表明 PSOPDM 工具拥有更好的检测性能。

结束语 基于网格化粒子群算法的最大浮点误差并行检测方法（PSOPDM）充分发挥国产申威平台的主从异构两级并行计算模式的算力优势，引入启发式搜索算法粒子群算法，科学划分搜索区域，从源头上规避了经典粒子群算法容易陷入局部极值的劣势，在各搜索区域培养粒子种群，动态适时调整搜索参数，独立并行迭代搜索，通过并行计算数据汇聚算法筛选出搜索最优解。这种方法大幅提高了检测浮点数最大误差的精度和性能，为高效检测浮点数最大误差方法提供了一种新的方法和工具，同时也为国产申威平台提供了一种高效的检测工具和算法。

然而，PSOPDM 算法还存在一些不足。一是 PSOPDM 算法暂时只能支持单参数表达式的浮点误差检测，后续还可以增加多参数表达式浮点误差检测能力。二是 PSOPDM 算法暂时只能在国产申威平台上运行，因为程序代码架构和语法的原因，暂时还无法在通用平台上运行。在后续的研究中可以按照相应平台的编程语言和架构进行改编，使之适应更多的平台。三是 PSOPDM 算法的并行能力仍有提升的空间，可以进一步提高并行规模和并行效率，从理论上推断，如果进一步优化并行算法，浮点误差检测精度还会进一步提升。

浮点数最大误差检测是一项重要的工作，在工程项目和学术研究领域扮演着“拉警报、响预警”的作用，可以一定程度上避免重大损失，规避“黑天鹅”和“灰犀牛”事故的发生，为重大工程推进和重要科学研究保驾护航。

参 考 文 献

[1] MACHIANI H N，TALEIZADEH A A，TOLOO M，et al. Designing a new sustainable healthcare network considering the COVID-19 pandemic：Artificial intelligence-based solutions[J]. Expert Systems with Applications，2025，260：125357-125357.

[2] NING D. Computer Software Design Based on Cloud Platform High-Performance Computing[J]. Journal of Physics：Conference Series，2021，1915(3)：032005.

[3] LUCKIJ G，DOLHOLENKO O. Development of floating point operating devices[J]. Technology Audit and Production Reserves，2023，5(2)：11-17.

[4] OHTA Y，OZAKI K. Extension of floating-point filters to absolute and relative errors for numerical computation[J]. Journal of Physics：Conference Series，2019，1218(1)：012011.

[5] UBLAIR M，OBENSKI S，BRIDICKAS P. Patriot missile defense：software problem led to system failure at Dhahran，Saudi Arabia[R]. Washington：United States Government Accountability Office，1992.

[6] Wikipedia. Ariane-5 flight 50 1[EB/OL]. http：//en. wikipedia. Org/wiki/Ariane-5-Fligh t_501.

[7] CNN. Toyota：Software to blame for Prius brake problems[EB/OL]. http：//edition. cnn. com /2010/WORLD/asiapcf/02/04/japan. prius. complaints/.

[8] BAGNARA R，BAGNARA A，BISELLI F，et al. Correct approximation of IEEE 754 floating-point arithmetic for program verification[J]. Constraints，2022，27(1/2)：29-69.

[9] YI X，CHEN L，MAO X，et al. Efficient automated repair of high floating-point errors in numerical libraries[J]. Proceedings of the ACM on Programming Languages，2019，3：1-29.

[10] SARMA R，BHARGAVA C，KOTECHA K，et al. An Evolutionary Normalization Algorithm for Signed Floating-Point Multiply-Accumulate Operation[J]. Computers，Materials & Continua，2022，72(1)：481-495.

[11] ZOU D M. Search-oriented error testing and analysis of floating-point programs [D]. Beijing：Peking University，2020.

[12] CHIANG W F，GOPALAKRISHNAN G，RAKAMARIC Z，et al. Efficient search for inputs causing high floating-point errors[C]// ACM SIGPLAN Notices. ACM，2014：43-52.

[13] CHILENSKIJ J，MILLER S P. Applicability of modified condition/decision coverage to software testing[J]. Software Engineering Journal，1994，9(5)：193-200.

[14] ZOU D，WANG R，XIONG Y，et al. A genetic algorithm for detecting significant floating- point inaccuracies [C]// Proceedings of the 37th International Conference on Software Engineering. IEEE，2015：529-539.

[15] FU Z，BAI Z，SU Z. Automated backward error analysis for numerical code [C]// Proceedings of the 2015 ACM SIGPLAN International Conference on Object－Oriented Programming，Systems，Languages，and Applications. ACM，2015：639-654.

[16] YIN X，CHEN L，MAO X，et al. Efficient Automated Repair of High Floating-Point Errors in Numerical Libraries[C]// Proceedings of the ACM on Programming Languages. 2019.

[17] BARR E T, VO T, LE V, et al. Automatic detection of floating-point exceptions[C]// ACM SIGPLAN Notices. 2013: 549-560.
[18] HUAYU F, DIAN L, HAIBING H, et al. A fast PSO algorithm based on Alpha-stable mutation and its application in aerodynamic optimization[J]. Xibei Gongye Daxue Xuebao/Journal of Northwestern Polytechnical University, 2022, 40(6): 1385-1393.
[19] MARCELA B S S, ARRIGO C, ALBERTOC T. Genetic algorithm with a Bayesian approach for multiple change-point detection in time series of counting exceedances for specific thresholds[J]. Journal of the Korean Statistical Society, 2023, 52(4): 982-1024.
[20] SEIFOLLAHI S, BAGIROV A, BORZESHI Z E, et al. A simulated annealing-based maximum-margin clustering algorithm [J]. Computational Intelligence, 2019, 35(1): 23-41.
[21] EBERHART R C, KENNEDY J. A New Optimizer Using Particle Swarm Theory[C]// Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Nagoya, 1995: 39-43.
[22] KENNEDY J, EBERHART R C. Particle Swarm Optimization [C]// Proceeding of IEEE International Conference on Neural Networks. 1995: 1942-1948.
[23] MULLER J M, BRUNIE N, DE DINECHIN F, et al. Handbook of Floating-Point Arithmetic [M]. Springer International Publishing, 2018.
[24] SOLOVYEV A, JACOBSEN C, RAKAMARIĆ Z, et al. Rigorous Estimation of Floating-Point Round-off Errors with Symbolic Taylor Expansions[C]// In 20th International Symposium on Formal Methods. New York: ACM, 2015: 532-550.
[25] CHIANG W F, BARANOWSKI M, BRIGGS I, et al. Rigorous floating-point mixed-precision tuning[C]// Symposium on Principles of Programming Languages. New York: ACM, 2017: 300-315.
[26] LIU L X. High performance data processing of distributed database and multi-core processor based on particle swarm optimization[J]. Journal of Electronics and Information Science, 2023, 8(4): 45-51.
[27] SUGANTHAN P N. Particle Swarm Optimizer with Neighborhood Operator[C]// Proceedings of Congress on Evolutionary Computation. 1999: 1958-1962.
[28] PANCHEKHA P, SANCHEZ-STERN A, WILCOX J R, et al. Automatically improving accuracy for floating point expressions [J]. ACM SIGPLAN Notices, 2015, 50(6): 1-11.
[29] ZHANG Z Y, XU J C, HAO J W, et al. Hierarchical search algorithm for error detection in floating-point arithmetic expressions [J]. The Journal of Supercomputing, 2023, 80: 1183-1205.
[30] CATTANEO D, BELLO A D, CHERUBIN S, et al. Embedded Operating System Optimization through Floating to Fixed Point Compiler Transformation[C]// 21st Euromicro Conference on Digital System Design. Institute of Electrical and Electronics Engineers, 2018: 172-176.

JI Liguang, born in 1992, postgraduate. His main research interest is high-performance computing.

XU Jinchen, born in 1987, Ph. D, associate professor. His main research interest is high-performance computing.

（责任编辑：何杨）