SparkDE:一种基于RDD云计算模型的并行差分进化算法

谭旭杰[1] 邓长寿[1] 董小刚[1] 袁斯昊[1] 吴志健[2] 彭 虎[2]
(九江学院信息科学与技术学院 九江 332005)[1] (武汉大学软件工程国家重点实验室 武汉 430072)[2]

摘 要 云计算MapReduce并行编程模型广泛应用于数据密集型应用领域,基于该模型的开源平台Hadoop在大数据领域获得了成功应用。然而,对于计算密集型任务,特别是迭代运算,频繁启动Map和Reduce过程将导致负载过大,影响计算效率。弹性分布式数据集(RDD)是一种基于内存的集群计算模型,有效地支持迭代运算,能够克服负载过大的问题。因此提出基于RDD模型的并行差分进化算法SparkDE。SparkDE首先将整个种群划分为若干个独立岛,然后将一个岛对应RDD中的一个分区,每个岛在RDD的一个分区中独立进化指定代数后,利用迁移算子在岛之间交换信息。利用标准测试问题对SparkDE、基于MapReduce模型的MRDE和基本DE进行对比实验研究。实验结果表明SparkDE求解精度高,计算速度快,加速效果明显,可以作为云计算平台的下一代优化器。

关键词 并行差分进化算法,岛模型,弹性分布式数据集,转换操作,控制操作
中图法分类号 TP18 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.9.022

SparkDE:A Parallel Version of Differential Evolution Based on Resilient Distributed Datasets Model in Cloud Computing

TAN Xu-jie[1] DENG Chang-shou[1] DONG Xiao-gang[1] YUAN Si-hao[1] WU Zhi-jian[2] PENG Hu[2]
(School of Information Science and Technology, Jiujiang University, Jiujiang 332005, China)[1]
(State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China)[2]

Abstract MapReduce is a popular cloud computing model which has been applied in data-intensive fields, and Hadoop which is based on MapReduce has been successfully used in dealing with big data. However, when dealing with computation-intensive tasks, particularly iterative computation, frequent loading of Map and Reduce processes will lead to overload. Resilient distributed dataset has been implemented in Spark, and it is an in-memory clustering computing model which can overcome this shortcoming efficiently. In this paper, a parallel version of differential evolution based on RDD (resilient distributed datasets) model named SparkDE was proposed. In SparkDE, the whole population is divided into several islands which evolve on their own, and then each island is deployed into a partition of RDD. After evolution for predefined generation in each island, migration operator is used calculation between islands. A wide range of benchmark problems are adopted to conduct numerical experiments. Compared with MapReduce (MRDE) based DE and classical DE, the results show that SparkDE can achieve higher accuracy of solution and faster speed of computation. The speedup of SparkDE is obvious. Thus SparkDE can serve as the next generation of optimizer in cloud computing.

Keywords Parallel differential evolution, Island model, Resilient distributed datasets, Transformation operation, Action operation

1 引言

传统的进化算法,例如遗传算法(GA)、进化策略(ES)、粒子群优化算法(PSO)等已经在许多领域获得了成功应用[1,2]。进化算法中,差分进化(DE)是一种基于群体的高效全局优化算法,广泛用于求解连续优化问题。与GA,ES,PSO等传统进化算法相比,DE算法在许多问题的求解上表现出了良好性能,并成功应用于许多实际问题[3,4]。近几年,DE算法的成功应用引起了学术界和工业界的广泛关注,研究人员提出了提高DE算法性能的不同方法[5-7]。然而,对于现实中的许多工程应用问题,每次评价目标函数都需要使用大量计算资源和计算时间。因此利用DE算法求解时,需要比较长的时间才能获得结果,无法满足工程实践的需要。而云计算是一种分布式并行计算平台,拥有大量的廉价计算资源[8]。因此,充分利用云计算平台的丰富计算资源可以有效减少目标函数的计算时间。此外,目前大数据处理领域存在

到稿日期:2015-07-25 返修日期:2015-09-07 本文受国家自然科学基金资助项目(61364025),武汉大学软件工程重点实验室开放基金资助项目(SKLSE2012-09-39),江西省教育厅科学技术资助项目(GJJ13729,GJJ14742)资助。

谭旭杰(1978-),男,硕士,讲师,主要研究方向为智能计算,E-mail:txj_2003@163.com;邓长寿(1972-),男,博士,教授,主要研究方向为智能计算和数据挖掘;董小刚(1979-),男,硕士,讲师,主要研究方向为智能计算;袁斯昊(1979-),男,硕士,讲师,主要研究方向为智能计算;吴志健(1963-),男,教授,博士生导师,主要研究方向为智能计算、并行计算、智能信息处理;彭 虎(1981-),男,博士,主要研究方向为智能计算及其在软件工程中的应用。

许多复杂优化问题，非常有必要将差分进化算法扩展到云计算平台，为大数据处理中的复杂优化问题求解提供新方法。Google 实验室提出的 MapReduce 模型[8]是集群环境下常用的并行编程模型与框架，基于 MapReduce 模型的开源软件 Hadoop 在大数据领域获得广泛应用。已有研究人员尝试利用 MapReduce 模型增强蚁群算法[9]和 DE 算法的性能[10,11]。然而，MapReduce 模型是一个简单通用的批处理计算模型，缺乏并行计算各阶段进行数据共享的有效机制。因此对于其它类型的计算，比如迭代运算、交互式和流式计算，MapReduce 无法提供有效的支撑[12]。针对上述问题，美国伯克利大学提出了全新的统一大数据处理框架 Spark，Spark 平台提出了一种全新的抽象的弹性分布式数据模型(RDD)[13]。RDD 模型的本质是数据共享，并提供相关的操作接口，能够适应大部分的数据并行算法和模型，在某种程度上 RDD 模型是对 MapReduce 模型的一种扩展。本文基于 Spark 平台，利用 RDD 数据模型和岛模型，提出一种全新的并行差分进化算法 SparkDE。

2 基本 DE 算法

DE 算法是一种基于种群的实数编码进化算法，其主要进化操作是初始种群、变异、交叉以及选择等 4 个操作。

(1)初始种群：DE 算法首先在一个 D 维空间依据式(1)随机地产生 NP 个受上下界约束的个体 $x_{i,0}$，$i\in[1,NP]$。

$$x_{ij,0}=x_j^{\min}+rand*(x_j^{\max}-x_j^{\min}) \tag{1}$$

其中，$j\in[1,D]$，$x_j^{\min}$ 和 $x_j^{\max}$ 分别为第 j 维空间上的上下边界值。

(2)变异算子

变异算子 DE 算法的主要进化算子。DE 算法利用当前种群中随机选择的个体之间的差异向量进行变异操作，得到的结果记为 $v_{i,g+1}$。DE 算法有多种变异算子，其效率各有不同，本文采用常用的 DE/rand/1/bin 变异算子。其具体描述如式(2)所示。

$$v_{i,g+1}=x_{r1,g}+F*(x_{r2,g}-x_{r3,g}) \tag{2}$$

其中，$r1,r2,r3\in[1,NP]$，且 $i,r1,r2,r3$ 两两互不相同。

(3)交叉算子

交叉算子依据交叉概率 CR，对中间个体 $v_{i,g+1}$ 与 $x_{i,g}$ 进行重组，获得候选个体 $u_{i,g+1}$。二项式交叉是 DE 算法常采用的交叉算子，如式(3)所示：

$$u_{ij,g+1}=\begin{cases}v_{ij,g+1}, & \text{if } rand<CR \text{ or } j=R(i)\\ x_{ij,g}, & \text{otherwise}\end{cases} \tag{3}$$

其中，$i=1,2,\cdots,NP$，$j=1,2,\cdots,D$，$R(i)\in[1,D]$，$rand$ 是一个均匀分布的随机数，$CR\in[0,1]$为交叉概率。

(4)选择算子

DE 算法的选择算子采用“贪婪”策略，即让候选个体 $u_{i,g+1}$ 和当前种群的 $x_{i,g}$ 依据目标函数的适应度值进行优劣竞争，获胜者进入下一代种群。其具体定义如式(4)所示：

$$x_{i,g+1}=\begin{cases}u_{i,g+1}, & \text{if } f(u_{i,g+1})\leqslant f(x_{i,g})\\ x_{i,g}, & \text{otherwise}\end{cases} \tag{4}$$

3 基于 RDD 云计算模型的并行差分进化算法——SparkDE

3.1 云计算模型

3.1.1 MapReduce 模型

Google 公司在函数式编程语言中的 Map 和 Reduce 函数的基础上，提出了 MapReduce 云计算模型。MapReduce 把对大数据集的常见操作任务划分为两步：Map 过程和 Reduce 过程。

$$\text{Map}:[k1,v]\rightarrow list(k2,v2) \tag{5}$$

$$\text{Reduce}:[k2,list(v2)]\rightarrow list(k3,v3) \tag{6}$$

用户自定义的 Map 过程以[$k1$，$v1$]键值对作为输入，经过处理产生一组中间输出数据的[$k2$，$v2$]键值对。MapReduce 框架聚合所有相同的中间键 $k2$ 的相应值，交由用户自定义的 Reduce 过程处理。Reduce 过程以 $k2$ 及对应的 $v2$ 列表作为输入，通过合并与 $k2$ 相同的 $v2$ 后，返回得到另外一系列[$k3$，$v3$]对，作为最终的输出结果写入文件系统 HDFS。MapReduce 一次处理流程如图 1 所示。

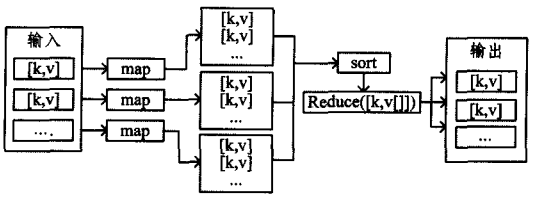

图 1 MapReduce 编程模型

由图 1 中的 MapReduce 编程模型可知，MapReduce 擅长进行“批处理”运算，可用于数据密集型领域。然而，进化计算在迭代求解过程中，每一次迭代的输出将作为下一次迭代的输入，而且每一次迭代计算的数据都不相同。因此，如果利用 MapReduce 实现进化算法，则必须将每一次迭代转化为一次任务提交，这样就会导致频繁读取文件，因此需要占用很长时间。

3.1.2 弹性分布式数据模型 RDD

针对 MapReduce 处理迭代运算等计算密集型运算时耗时较长的不足，伯克利大学推出的全新的大数据框架 Spark 中的 RDD 模型可以有效地降低迭代运算的时间消耗。RDD 是一种基于内存运算的集群计算模型，参加运算的数据在内存中共享，从而有效地支持迭代运算。

(1)RDD 抽象

RDD 提供了一个抽象来支持工作集合(working set)的应用。RDD 不仅支持数据流模型，同时也能够在不同工作集合有效地进行通信。对工作集合进行操作时，RDD 仅支持粗粒度(coarse-grained)的转换，即一种操作将作用于所有记录。此外，RDD 是一个分布于集群中各个结点数据集上的集合，在此数据集上可以进行并行运算。RDD 的创建有两个途径：来源于文件系统或者来自程序中的 Scala 集(collection)。

(2)Spark 中的 RDD 编程模型

Spark 是首个高效的、采用通用语言进行交互式分析数据的集群系统[13]。在 Spark 中，用对象表示 RDD，“转换”定义为在对象上进行的操作。创建了 RDD 之后，转换操作(Transformation Operation)用来对这些 RDD 进行转换。行动操作(Action Operation)将结果返回程序或者将结果写入文件系统。RDD 模型如图 2 所示。

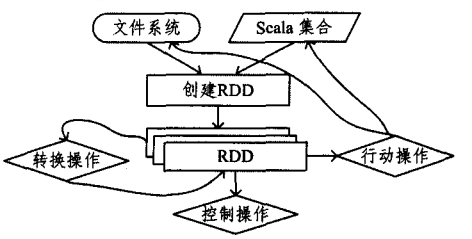

图 2 Spark 中的 RDD 模型

(3)RDD常见操作

RDD中的主要操作包括3类:转换操作、控制操作和行动操作。其中转换操作从一个RDD生成一个新的RDD;控制操将RDD持久化到内存,方便后续的操作;行动操作将结果返回到程序。

3.2 SparkDE

为了有效地在RDD模型上对DE算法进行并行化,本文采用基于岛模型(Island)[14]的差分进化算法。本文利用Spark提供的广播变量(Broadcast)协助实现不同岛之间数据的迁移。SparkDE首先输入种群数目 NP、变异概率 F、交叉概率 CR、岛的数目、岛之间迁移方式等参数,然后初始化种群并创建RDD,并将岛与RDD中的分区一一对应;接下来在每个岛内按照指定的代数进化之后,利用控制操作生成一个新RDD,并广播到每个结点,按照环形拓扑依次利用一个岛的最优值替换另外一个岛的最差值。SparkDE的具体程序流程如图3所示。

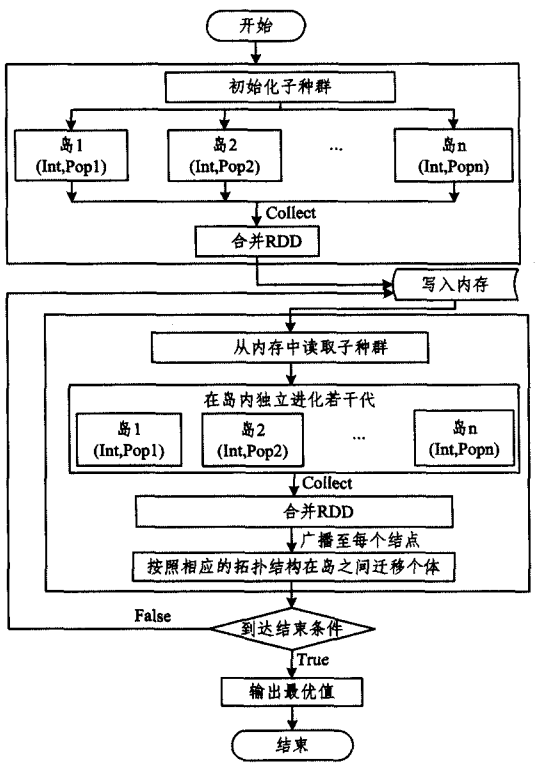

图3 SparkDE流程图

4 数值实验

4.1 测试问题

为了测试本文提出的SparkDE算法的性能,选择文献[15,16]所使用的测试集进行实验,该测试集包含13个不同的问题,其中F1－F7是单模问题,F8－F13是多模问题,关于这些测试问题的详细定义,请参阅文献[15]。本文选取30维问题进行实验。

4.2 实验设置

本文实验使用3台x86结构的计算机组建集群。每台机器配置一致:64bit Corei7 CPU,主频3.4GHz,内存8GB,Ubuntu13.10操作系统,安装使用Hadoop2.2.0和Spark1.2.0,利用Scala语言实现了SparkDE。MRDE采用文献[11]提供的程序。

实验时SparkDE中岛的数目为5,NP为60,$F=0.5$,$CR=0.9$,每个岛指定迭代次数为100,目标函数评价次数最大为3E5,迁移方式选择环形拓扑方式,即利用相邻岛的最优值替换本岛中的最差值。MRDE的参数、DE的参数与SparkDE的相关参数一致,3个算法的目标函数最大评价次数相同。

4.3 实验结果

实验时,每个算法针对每个问题独立运行20次,表1记录了所求出的最优值、平均值、最差值和方差,3种方法中最好的结果用粗体表示。

表1 SparkDE,MRDE和DE算法的求解结果比较

函数	算法	Best	Worst	Mean	Std
F1	SparkDE	**1.73E−016**	**9.63E−016**	**4.88E−016**	**2.41E−016**
	MRDE	6.30E−015	6.25E−014	2.75E−014	1.38E−014
	DE	3.21E−004	1.92E−003	1.02E−003	3.79E−004
F2	SparkDE	**4.56E−012**	**4.00E−011**	**1.50E−011**	**8.28E−012**
	MRDE	9.16E−011	4.74E−010	2.42E−010	1.16E−010
	DE	5.21E−002	1.10E−001	7.62E−002	1.49E−002
F3	SparkDE	**1.37E−001**	7.64E−001	**3.35E−001**	1.79E−001
	MRDE	1.84E−001	**5.95E−001**	3.73E−001	**1.28E−001**
	DE	4.61E+002	1.07E+003	8.13E+002	1.83E+002
F4	SparkDE	**0.00E+000**	**0.00E+000**	**0.00E+000**	**0.00E+000**
	MRDE	**0.00E+000**	**0.00E+000**	**0.00E+000**	**0.00E+000**
	DE	1.26E+000	2.57E+000	2.11E+000	3.33E−001
F5	SparkDE	1.80E+001	**2.03E+001**	**1.93E+001**	7.04E+001
	MRDE	**1.32E+001**	2.11E+001	1.99E+001	**1.66E+000**
	DE	2.46E+001	2.70E+001	2.60E+001	6.19E−001
F6	SparkDE	**0.00E+000**	**0.00E+000**	**0.00E+000**	**0.00E+000**
	MRDE	**0.00E+000**	**0.00E+000**	**0.00E+000**	**0.00E+000**
	DE	**0.00E+000**	**0.00E+000**	**0.00E+000**	**0.00E+000**
F7	SparkDE	**1.36E−003**	**3.63E−003**	**2.46E−003**	6.65E−004
	MRDE	2.65E−003	9.60E−003	6.22E−003	**1.87E−003**
	DE	1.40E+002	3.65E+002	2.51E−002	5.91E−003
F8	SparkDE	−1.10E+004	−9.11E+004	−9.61E+003	4.40E+002
	MRDE	**−8.23E+003**	**−6.73E+003**	**−7.51E+003**	**3.30E+002**
	DE	−8.15E+003	−6.41E+003	−7.12E+003	4.62E+002
F9	SparkDE	**4.19E+001**	**7.74E+001**	**5.82E+001**	**8.29E+000**
	MRDE	7.75E+001	1.29E+002	1.06E+002	1.38E+001
	DE	1.84E+002	2.13E+002	1.97E+002	7.46E+000
F10	SparkDE	**3.15E−009**	**1.77E−008**	**6.87E−009**	**3.06E−009**
	MRDE	3.33E−008	6.78E−008	4.99E−008	9.60E−009
	DE	5.96E−003	1.61E−002	1.05E−002	2.13E−003
F11	SparkDE	**0.00E+000**	**2.67E−015**	**8.49E−016**	**8.39E−016**
	MRDE	1.97E−014	3.03E−013	9.48E−014	7.17E−014
	DE	1.87E−003	2.54E−002	4.47E−003	5.06E−003
F12	SparkDE	6.33E−003	6.33E−003	6.33E−003	2.49E−018
	MRDE	**5.53E−016**	**1.17E−014**	**2.98E−015**	**2.39E−015**
	DE	1.31E−004	1.36E−3	4.49E−004	3.62E−004
F13	SparkDE	**1.00E−001**	**1.00E−001**	**1.00E−001**	**1.42E−017**
	MRDE	−1.15E+000	−1.15E+000	−1.15E+000	5.35E−014
	DE	5.27E−004	4.42E−003	1.66E−003	9.69E−004

表1的结果表明,对于13个问题的求解,SparkDE与MRDE的求解结果明显优于DE算法。对于F4和F6,SparkDE的结果与MRDE的结果相当;对于F12的求解结果,SparkDE稍逊于MRDE;对于其他9个问题,SparkDE的结果均优于MRDE的结果。总体而言,SparkDE的求解精度优于MRDE和DE。

为了描述SparkDE在计算时间上的优势,图4和图5给出了SparkDE和MRDE的20次求解所需时间的对比,时间单位为毫秒。

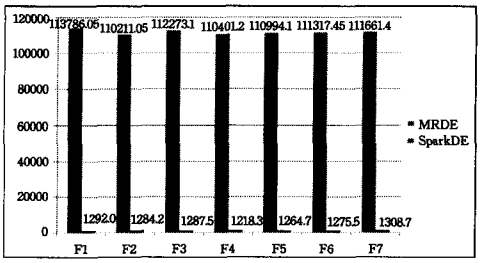

图 4　SparkDE 与 MRDE 运行时间比较(F1－F7)

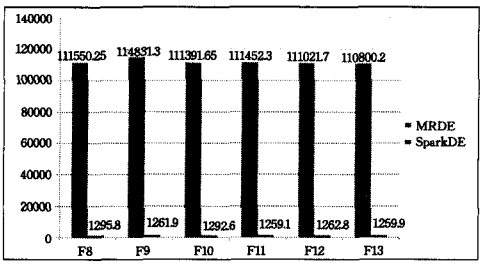

图 5　SparkDE 与 MRDE 运行时间比较(F8－F13)

从图 4 和图 5 的耗费时间可以计算出 MRDE 所需时间是 SparkDE 的 85～91 倍,因此基于 RDD 模型的 SparkDE 比基于 MapReduce 模型的 MRDE 要快得多,加速效果明显。

4.4　加速比实验

RDD 模型分区涉及到对此 RDD 进行并行计算的粒度,每一个 RDD 分区的计算操作都在一个单独的任务中执行[12]。因此本文以不同的分区探讨 SparkDE 的加速比。采用式(7)[17]定义的加速比公式研究 SparkDE 的加速比。

$$S_m(N_p)=\frac{T_m(1)}{T_m(N_p)} \tag{7}$$

其中,$T_m(1)$代表利用 1 个分区执行 m 次所需平均时间;$T_m(N_p)$表示 p 个分区执行 m 次所需平均时间。本文实验中 $m=20$,实验时单模函数选择 F1,F3 和 F5,多模函数选择 F9,F10 和 F11。针对每个问题,设定 3 种不同的最大目标函数即 3E5,1.5E5 和 3E4。

图 6－图 11 的加速比表明,SparkDE 具有较好的加速比。此外,加速比与目标函数的最大评价次数关系不明显,表明了加速比的稳定性。SparkDE 采用 5 个岛、RDD 使用 5 个分区时,每个岛在每个分区中独立运行,因此加速比接近 5 倍,达到最佳加速效果。

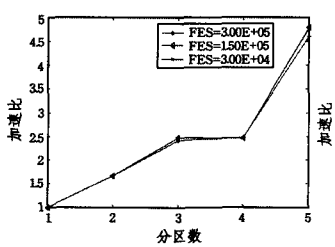

图 6　函数 F1 的加速比

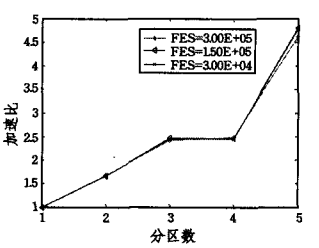

图 7　函数 F3 的加速比

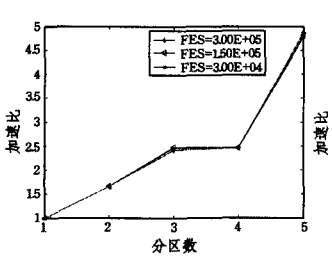

图 8　函数 F5 的加速比

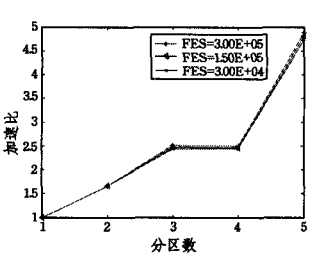

图 9　函数 F9 的加速比

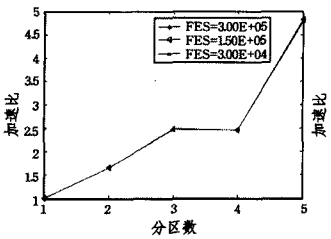

图 10　函数 F10 的加速比

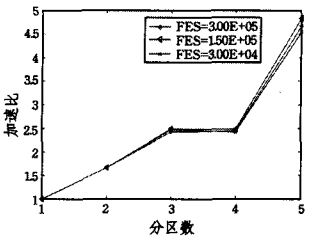

图 11　函数 F11 的加速比

结束语　本文利用多岛模型的 DE 算法,基于云计算 RDD 模型,提出一种全新的并行差分进化算法 SparkDE。SparkDE 将每个岛分别与 RDD 模型中的一个分区对应,每个岛上的 DE 算法独立并行进化。利用 Scala 语言在 RDD 模型的开源平台 Spark 上实现了 SparkDE,利用 13 个标准测试问题,与基于 MapReduce 模型的 MRDE、标准 DE 算法进行了实验对比研究,结果表明 SparkDE 求解精度高,速度快。与 MRDE 相比,SparkDE 的速度是其 85～91 倍。此外,选择 6 个问题进行加速比实验,实验结果表明当岛的数量与分区数量相同时,加速比几乎呈线性,加速效果良好。

在未来的工作中,将每个岛内的 DE 算法并行化,增加 SparkDE 的并行粒度。此外,利用更多的结点和 SparkDE 求解大规模优化问题也将是下一步的重要工作。

参 考 文 献

[1]　王宇平. 进化计算的理论和方法[M]. 北京:科学出版社,2011

[2]　Eiben A E,Smith J. From evolutionary computation to the evolution of things[J]. Nature,2015,521(7553):476-482

[3]　Vesterstrom J,Thomsen R. A comparative study of differential evolution,particle swarm optimization,and evolutionary algorithms on numerical benchmark problems[C]// IEEE on CEC 2004. IEEE,2004:1980-1987

[4]　Rocca P,Oliveri G,Massa A. Differential evolution as applied to electromagnetics [J]. Antennas and Propagation Magazine,IEEE,2011,53(1):38-49

[5]　Shen Ji-nan,Liang Fang,Zheng Ming-hui. Deep predation and secondary gradient acceleration differential evolution algorithm [J]. J. Huazhong Univ. of Sci. & Tech. (Natural Science Edition),2014,42(11):23-27,33(in Chinese)
沈济南,梁芳,郑明辉. 深度捕食二次梯度加速差分进化算法[J]. 华中科技大学学报(自然科学版),2014,42(11):23-27,33

[6]　Chen Ying,Lin Ying,Hu Xiao-min. Parallel Differential Evolution with Multi-Population and Multi-Strategy[J]. Journal of Frontiers of Computer Science and Technology,2014,8(12):1502-1510(in Chinese)
陈颖,林盈,胡晓敏. 多种群多策略的并行差分进化算法[J]. 计算机科学与探索,2014,8(12):1502-1510

[7]　Bi Xiao-jun,Liu Guo-an,Xiao Jing. Dynamic Adaptive Differential Evolution Based on Novel Mutation Strategy[J]. Journal of Computer Research and Development,2012,49(6):1288-1297 (in Chinese)
毕晓君,刘国安,肖婧. 基于新变异策略的动态自适应差分进化算法[J]. 计算机研究与发展,2012,49(6):1288-1297

[8]　Dean J,Chemawat S. MapReduce:simplified data processing on large clusters[J]. Communications of the ACM,2008,51(1):107-113

(下转第 139 页)

AS类型，虽然可以根据AS度数来进行识别(P型AS的度数相对较高)，但效果不佳；而某些区别AS类型的模型，如GHITLE[13]，度分布特性较差。H. Tangmunarunkit发现AS规模与度数之间存在正相关关系并将其关系函数化[14]，但并未考虑到AS类型对相关关系的影响。构造更加完备的AS级模型是未来研究者们需要解决的问题。

其次，现有的HOT类模型适合AS内部建模，但是同样没有考虑到AS类型对模型的影响。考虑到AS不同的功能，P型AS需要转运过路流量而C型AS不需要，那么两类AS的内部结构应当是有所区别的，很有可能呈现出这样的现象：P型AS内部更接近网状，而C型AS内部更接近树状。是否确实存在这样的现象以及这样的现象对建模的影响，都需要研究者们进行更多的研究。

第三，ITDK数据库尽管提供了几乎是现有最完善的路由器级拓扑信息，但是仍有所遗漏，这对域间参数映射函数化造成了不利影响，我们期待ITDK和其他网络研究组织能够提供更加完善的网络拓扑数据。

结束语 域间路由器级连接是AS级拓扑与域内拓扑的连接桥梁，研究域间路由器级连接特点对Internet建模具有重要意义。本文提出了域间参数映射概念，通过对现网数据的分析实现了域间参数映射的函数表达，并指出了在Internet建模方面应当进一步解决的问题。

参 考 文 献

[1] Schuchard M, Vasserman E, Mohaisen A. Losing control of the internet: using the data plane to attack the control plane [C]// NDSS. The Internet Society, 2011: 101-116

[2] Studer A, Perrig A. The Coremelt attack [M]// Computer Security-ESORICS 2009. 2009: 37-52

[3] Li L, Alderson D, Willinger W, et al. A first-principles approach to understanding the internet's router-level topology[J]. Acm Sigcomm Computer Communication Review, 2004, 34(4): 547-560

[4] Faloutsos M, Faloutsos P, Faloutsos C. On power-law relationships of tile internet topology[J]. Sigcomm, 1999, 29: 251-262

[5] Magoni D, Pansiot J J. Internet topology modeler based on map sampling [C] // International Symposium on Computers and Communications. 2002: 1021-1027

[6] Fabrikant A, Koutsoupias E, Papadimitriou C H. Heuristically Optimized Trade-Offs: A New Paradigm for Power Laws in the Internet[C]//29th International Colloquium Conference on Automata, Languages and Programming. 2002: 110-122

[7] Chang H, Jamin S, Willinger W. Internet connectivity at the AS-level: An optimization-driven modeling approach [C]// Proc. of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research. 2003: 210-225

[8] Scholtes I, Botev J, Esch M, et al. TopGen-Internet Router-Level Topology Generation Based on Technology Constraints [C]// Proc of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops. 2008: 102-112

[9] Quoitin B, Van V, Francoiset P, et al. IGen: Generation of Router-level Internet Topologies through Network Design Heuristics [C]//Proceedings of 21st International Teletraffic Congress. 2009: 109-117

[10] Liu Y, Pan L, B Li. Complex Internet route-level topology model [J]. Computer Engineering and Applications, 2012, 48(28): 1-5

[11] AS-Relationship [DB/OL]. (2013-11-01). http://www.caida.org/data/as-relationships

[12] The Internet Topology Data Kit [DB/OL]. (2013-07). http://topo-data.caida.org/

[13] Launois C. Generator of Hierarchical Internet Topologies using LEvels (GHITLE) [OL]. (2001). http://openresources.info.ucl.ac.be/ghitle

[14] Tangmunarunkit H, Doyle J, Govindan R, et al. Does AS size determine degree in as topology? [J]. ACM SIGCOMM Computer Communication Review, 2001, 31(5): 7-8

(上接第119页)

[9] Wang Zhao-yuan, Li Tian-rui, Yi Xiu-wen. Approach for Development of Ant Colony Optimization Based on MapReduce[J]. Computer Science, 2014, 41(7): 261-265(in Chinese)
王诏远，李天瑞，易修文. 基于MapReduce的蚁群优化算法实现方法[J]. 计算机科学，2014，41(7)：261-265

[10] Zhou C. Fast parallelization of differential evolution algorithm using MapReduce[C]// Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. ACM, 2010: 1113-1114

[11] Pavlech M. Framework for Development of Distributed Evolutionary Algorithms Based on MapReduce [C] // Annals of DAAAM for 2011 & Proceedings of the 22nd International DAAAM Symposium Intelligent Manufacturing & Automation: Power of Knowledge and Creativity. Vienna: DAAAM International Vienna. 2011: 1475-1476

[12] 夏俊鸾，刘旭辉，邵赛赛，等. Spark大数据处理技术[M]. 北京：电子工业出版社，2015

[13] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing[C] // Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. USENIX Association, 2012: 1-16

[14] Jumonj T, Chakraborty G, Mabuchi H, et al. A novel distributed genetic algorithm implementation with variable number of islands[C]//IEEE Congress on Evolutionary Computation, 2007 (CEC 2007). IEEE, 2007: 4698-4705

[15] Yao X, Liu Y, Lin G. Evolutionary programming made faster [J]. IEEE Transactions on Evolutionary Computation, 1999, 3 (2): 82-102

[16] Yang Z, Tang K, Yao X. Large scale evolutionary optimization using cooperative coevolution [J]. Information Sciences, 2008, 178(15): 2985-2999

[17] Kiyouharu T, Takashi I. Concurrent Differential Evolution Based on MapReduce[J]. International Journal of Computers, 2010, 4(4): 161-168