doi:10.16104/j.issn.1673-1891.2019.03.014

基于Spark的DA算法并行化研究

唐 立,王利军
(安徽经济管理学院信息工程系,合肥 230031)

摘要:在对大规模数据进行蜻蜓算法优化时,由于要计算的维度过多,迭代次数过大,从而耗费大量运算时间,而基于Spark分布式计算可以减少大数据运算的耗时。将DA算法在Spark分布式计算平台下进行并行计算,把蜻蜓种群被分配到各个节点,每节点中蜻蜓个体信息通过多线程并行更新,然后共享全局最优解,从而提高大规模数据优化的运行速度。最后仿真实验的验证是由4个测试函数进行测试,验证结果显示:在保证正确率的前提下,基于Spark的DA算法在对大规模数据优化的计算用用时最少。

关键词:Spark;DA算法;并行化;大规模数据
中图分类号:TP301.6;TP18 **文献标志码**:A **文章编号**:1673-1891(2019)03-0066-04

Study on DA Algorithm Parallelization Based on Spark

TANG Li, WANG Lijun
(Department of Information Engineering, Anhui Institute of Economic Management, Hefei 230031, China)

Abstract: In the optimization of dragonfly algorithm for mass data, due to too many dimensions to be calculated and too many iterative computations, it takes a lot of operation time. However, the distributed algorithm based on Spark can reduce the operation time of big data. In this paper, the DA algorithm is used to conduct parallel computation on Spark distributed computing platform, and the dragonfly population is distributed to each node. The dragonfly individual information in each node is updated in parallel through multiple threads, and then the global optimal solution is shared, so as to improve the operation speed of mass data optimization. Finally, the verification of the simulation test is carried out through four test functions. The verification results show that on the premise of the accuracy ensured, the DA algorithm based on Spark takes the least time in the optimization calculation of mass data

Keywords: Spark; DA algorithm; parallelization; mass data

0 引言

蜻蜓算法(Dragonfly Algorithm)是2015年Mirjalili等通过模拟蜻蜓群体飞行、寻食和避敌的行为提出的一种新兴的元启发式优化算法,它具有较强的稳定性和优秀的全局寻优能力,从而在各个领域中得到应用,如神经网络优化[1]、深度学习[2]、单目标任务、多目标任务优化[3-4]等。随着数据量的增大,优化规模的扩大,DA算法所需要计算的维度增多,计算迭代次数过多,从而影响其优化性能[5]。分布式计算是利用集群计算对同一问题或不同问题进行求解计算,和使用单机计算形成对比,可以在一定程度上解决大量数据计算耗时问题。另一方面,Spark是基于内存计算的新一代大数据分布式计算平台,对大量数据分布式处理具有非常好的效果。为了提高大数据下DA算法的效率,研究了DA算法在Spark平台上的并行化方案,并通过仿真实验验证该方案的效率性和准确性。

1 相关概念

1.1 DA算法

DA算法是模拟蜻蜓群体飞行导航、捕猎、躲避天敌等行为对全局和局部同时进行搜寻,寻找最佳捕猎行为的算法寻优过程[6-7]。蜻蜓群体的行为可以分5种:

第1种:分离行为指蜻蜓个体与相邻其他蜻蜓之间的分离情况,其行为的数学表达式为:

$$S_i = -\sum_{j=1}^{N}(X - X_j) \quad , \tag{1}$$

其中,S_i是分离度,描述蜻蜓i与相邻蜻蜓的分离程

收稿日期:2019-05-14
基金项目:安徽省高校自然科学研究项目(KJ2019A0965);安徽省社会科学联合会课题(2018CX104)。
作者简介:唐立(1980—),男,安徽马鞍山人,副教授,硕士,研究方向:计算机应用、数据挖掘。

度值。蜻蜓所在的位置为X,相邻蜻蜓j的位置是X_j,相邻蜻蜓的总数为N。

第2种对齐行为:指蜻蜓个体与相邻其他蜻蜓的速度匹配,其行为数学表达式为:

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N}, \tag{2}$$

其中,蜻蜓i与相邻蜻蜓的对齐度为A_i,相邻蜻蜓的速度为V_j。

第3种内聚行为:是指蜻蜓个体与相邻其他蜻蜓的聚拢程度,其行为数学表达式为:

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X, \tag{3}$$

其中,蜻蜓i的内聚度为C_i。

第4种觅食行为:指蜻蜓个体寻找食物行为,其行为数学表达式为:

$$F_i = X^+ - X, \tag{4}$$

其中,蜻蜓i的觅食能力为F_i,食物的位置为 。

第5种避敌行为:指蜻蜓个体躲避外敌行为,其行为数学表达式为:

$$E_i = X^- + X, \tag{5}$$

其中,蜻蜓i的避敌能力是E_i,天敌的位置为X^-。

当蜻蜓个体附近有其他蜻蜓时,则蜻蜓可以通过上面5种行为方式来确定其飞行方向$\triangle X$和空中位置X,不断更新方向和位置,最终找到最优的结果。t是迭代次数,蜻蜓位置和方向迭代更新的表达式分别为式(6)和(7):

$$X_{t+1} = X_t + \triangle X_{t+1}, \tag{6}$$

$$\triangle X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + \triangle w \triangle X_t, \tag{7}$$

式(7)中,s、a、c、f、e、$\triangle w$分别为分离权重、对齐权重、内聚权重、觅食权重、天敌权重、惯性权重。

在当前蜻蜓个体周围无其他蜻蜓时,就需要利用Lévy()函数调整蜻蜓位置,促使它能够找到群体。Lévy()函数表达式如式(8):

$$\text{Lévy}(x) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{\frac{1}{\beta}}}, \tag{8}$$

$$\sigma = \left(\frac{\tau(1+\beta) \times \sin\frac{\pi\beta}{2}}{\tau(\frac{1+\beta}{2}) \times 2^{(\frac{\beta-1}{2})}} \right)^{\frac{1}{\beta}}, \tag{9}$$

其中,r_1,r_2为随机数,范围为0~1;β是常量,通常为0.5;$\tau(x)=(x-1)!$。

最终蜻蜓位置根据Lévy()函数更新,其数学表达式为式(10):

$$X_{t+1} = X_t + \text{Lévy}(d) \triangle X_t, \tag{10}$$

其中,d为维度。

1.2 基于Spark的分布式计算

近几年,分布式计算平台发展最为突出的有Hadoop和Spark,而Spark是新一代的代表。Spark相对于传统的Hadoop而言,它是基于内存计算的,当需要多次迭代运算时,数据传递的时间开销远远小于Hadoop。同时,Spark的多线程模式要比Hadoop的进程模式在任务启动上的开销要小很多。此外,Spark不仅可以利用自带的资源调度框架运行,还可以利用Hadoop的资源调度框架运行。总之Spark比Hadoop具有速度快、适用范围广、可扩展等特点[8]。

Spark运行架构如图1所示,它是由集群资源管理器(Cluster Manager)、节点控制器(Driver)、工作节点(Worker Node)和每个工作节点上的执行进程(Executor)组成。同时Spark还使用弹性分布式数据集(Resilient Distributed Dataset,RDD)进行存储数据,它可以让用户自定义分区,形成内部元素可并行化计算的分布式内存集合。Spark会根据RDD中的数据物理位置进行计算迁移,减少数据网络传递开销。同时,在计算过程中的运算中间值和最终值均保持在RDD中,也减少磁盘读取的时间开销。此外,RDD之间存在一种有向无环图(简称DAG)的依赖关系,如果在计算过程中产生数量丢失,可以根据这种依赖关系,恢复计算过程。这也是减少了Spark在计算上出错的概率。

Spark基本运行流程如图1所示。当一个Spark应用被提交时,节点控制器(Driver)创建一个Spark Context,并由Spark Context向资源管理器(Cluster Manager)申请资源,分配任务。资源管理器为每个工作节点(Worker Node)的多个Executor分配资源,并启动Executor进程。Spark Context根据RDD的依赖关系构建DAG图,DAG调度器(DAG

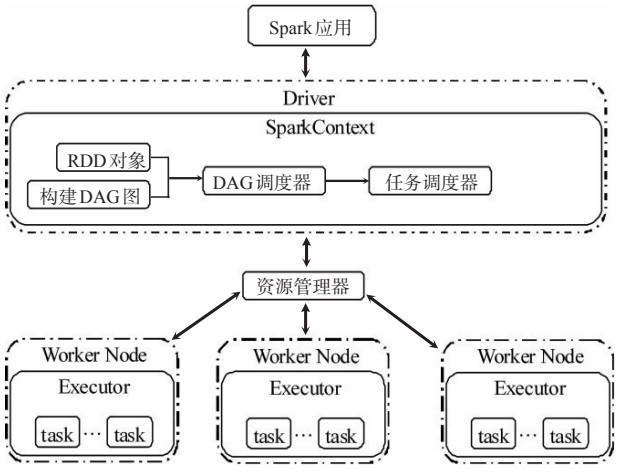

图1　Spark运行架构及运行流程图

Scheduler)把DAG图分解成多个任务集,并且也计算出任务集之间的依赖关系,然后把每个任务集传给任务调度器(Task Scheduler)进行处理,任务调度器将任务分发给Executor运行。最终Executor计算结果反馈工作节点,工作节点将结果返回Driver[9-10]。

2 基于Spark的并行DA算法

DA作为元启发式优化算法通常需要算法的代价比较大,它在演化过程中依靠5种行为和步长进行更新,产生大量的迭代运算。而Spark的RDD机制正好可以符合这个需求,因此,我们基于RDD设计了其种群中蜻蜓位置和步长更新以及目标函数评价的Executor多线程并行化方案,从工作节点对蜻蜓本身进行更新。而种群中蜻蜓个体之间交互通过全局最优位置的共享,是在主节点进行蜻蜓之间的交互,故此形成分布式主从式计算模式,在原有群寻优算法结果的基础上提高优化速度。其方案具体流程如下。

2.1 Spark的环境构建和资源的分配

当一个DA算法应用被提交时,首先为这个应用构建一个基本运行环境,即任务节点读取配置文件生成SparkConf对象,然后基于该对象创建一个SparkContext,由SparkContext负责和资源管理器(Cluster Manager)的通信和连接。另外,Spark通过读取外部数据资源将其转换为源RDD,然后利用RDD的算子操作在不同的RDD之间转换,最后得到结果RDD。在这期间要合理分配RDD分区数目使之与程序分配的计算资源相匹配,否则会影响Spark的并行化计算效率。

2.2 多节点并行化DA算法

如图2所示,作为Spark分布式框架,蜻蜓种群被分配到各个节点。并初始化蜻蜓方向$\triangle X$和空中位置X等变量,采用Spark的广播变量机制分发到各个计算节点。每个工作节点的Executor将依据RDD依赖关系构建DAG图,分配给各个线程具体任务。工作站里每个蜻蜓上的每一维的信息在Executor的线程任务里并行更新,计算每个蜻蜓适应度值后,蜻蜓个体的历史最优位置(被视为食物)和最差位置(被视为外敌)的更新在不同的任务线程里并行完成。对于种群的全局最优位置或最差位置在共享内存通过加锁机制在主节点上进行实现。具体步骤如下:

Step1:确定子种群规模N,蜻蜓种群RDD并行化,初始化DA参数;要计算的最大迭代次数M;空间维度d;惯性权重$\triangle w$;领域半径为r;蜻蜓5种行为权值(s, a, c, f, e)、位置X和飞行方向$\triangle X$。运用Spark的广播机制将初始参数分发到各个计算节点。

Step2:计算个体蜻蜓的适应度值。当迭代次数t>1时,建立上一代迭代的蜻蜓位置与当前蜻蜓位置的排序映射,通过蜻蜓个体适应度值,求得上代和本代的优秀蜻蜓个体,并记录当前最优个体位置X;利用欧氏距离公式算出当前最优解为食物位置X^+和最差解为敌人位置X^-,然后放入共享内存中。

Step3:蜻蜓种群学习通过种群历史最优位置的共享来实现,更新食物和外敌位置,并对5种行为权值(s,a,c,f,e)和惯性权重$\triangle w$进行更新。

Step4:运用公式(1)~(5)重新调整5种行为(S,A,C,F,E)的值,如果领域半径内有其他蜻蜓,则用公式(6)(7)更新位置向量X和方向向量$\triangle X$;如果没有,则用公式(10)更新位置向量X。

Step5:看是否达到最大迭代数,若达到,则运算结束;若没有,则当前的迭代次数加一并跳转Step2。

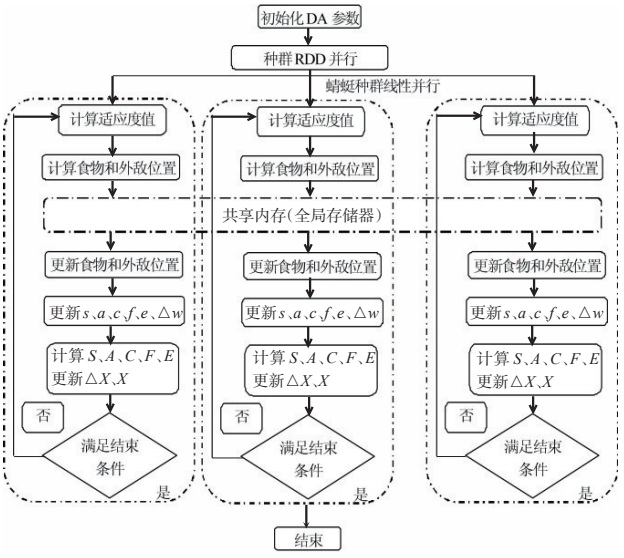

图2 基于Spark分布式框架下的并行DA算法

3 仿真实验

3.1 实验环境和参数

实验是由4台相同配置服务器组成的Spark集群计算平台,其中1台作为主控节点负责任务调度和收集计算结果,其余3台参与计算任务。每台机器的配置处理器为Intel Xeon E5-2609V3,主频为1.9 GHz,操作系统为CentOS6.9,所使用的Spark版本为2.2.1,Hadoop版本为3.0。

为了验证本文算法的有效性,选择2类不同特征的测试函数从不同角度对Spark下的并行DA算法进行测试。一类为单峰无约束函数,其中包含的2个标准测试函数是F_1、F_3;另一类为多峰无约束函

数,测试函数是F_5、F_7。在运算过程中的共同参数设置为:种群规模数N分别是500、1 000、1 500、2 000、2 500个,最大迭代次数为MIT=3 000次,最大评价次数为30 000次。

3.2 实验分析与比较

为了充分说明文本算法的性能,笔者选取单机下的DA算法、Hadoop下的并行DA算法和Spark下的并行DA算法分别对4个测试函数进行求解,并从最优解、最差解和均方差几个方面对各算法进行测评,如表1所示。

表1　测试函数计算统计结果

测试函数	算法	最优解	最差解	均方差
	DA	2.237	17.529	4.521
F_1	Hadoop-DA	1.892	18.537	3.871
	Spark-DA	1.887	17.682	4.259
	DA	4.084E-03	1.381	3.362E-01
F_3	Hadoop-DA	3.872E-03	1.156	3.064E-01
	Spark-DA	4.042E-03	1.249	3.261E-01
	DA	65.371	2.355E+02	38.418
F_5	Hadoop-DA	63.522	2.047E+02	32.439
	Spark-DA	55.395	1.882E+02	29.617
	DA	5.187E-02	8.791E-01	2.029E-01
F_7	Hadoop-DA	4.739E-02	7.842E-01	1.842E-01
	Spark-DA	4.281E-02	7.191E-01	1.635E-01

表1给出计算结果统计,从整体上来看,3种算法对目标求解的精度非常接近,这是因为本文对DA算法本身没有进行改变,其优化的最终结果也变化不大。同时也表明基于Spark下的并行DA算法在精确度上不低于其他算法。

更为重要的是,本文算法的目的是提高大数据规模优化计算用时效率,笔者将比较在不同群规模下对标准的DA算法、Hadoop下的并行DA算法和Spark下的并行DA算法在测试函数上所需的运行时间,如表2所示。

表2给出了DA算法、Hadoop下的并行DA算法和Spark下的并行DA算法在测试函数F_1、F_3、F_5、F_7上对不同的种群规模数上优化运行时间,从表2可以看出,随着种群规模的增大,3种算法的运行时间差距也不断变大。标准DA算法运行时间耗费最多,无法适应大规模运算。相对来说,Hadoop或Spark下的并行DA算法用时要少很多,但是Spark下的并行DA用时最少,是因为Spark基础内存计算,大大减少数据传递的开销,随着数据量增大,运行时间的优势越明显。

表2　3种算法在不同种群数下的运算时间　　s

测试函数	算法	种群规模数N/个				
		500	1 000	1 500	2 000	2 500
	DA	98.6	294.5	623.4	1167.6	1467.5
F_1	Hadoop-DA	52.3	123.6	223.7	271.2	325.9
	Spark-DA	48.6	80.3	104.5	126.7	152.7
	DA	112.3	327.4	668.2	1236.6	1512.3
F_3	Hadoop-DA	61.6	148.7	268.3	358.9	502.6
	Spark-DA	59.8	103.6	144.6	197.3	228.6
	DA	123.6	456.8	1024.6	1682.7	2256.8
F_5	Hadoop-DA	78.8	164.5	345.6	459.3	684.6
	Spark-DA	72.5	124.5	168.9	227.6	302.8
	DA	150.7	568.3	1278.4	2012.6	2658.3
F_7	Hadoop-DA	85.6	202.6	428.6	573.6	814.6
	Spark-DA	80.4	142.8	181.7	258.3	386.7

4 结语

为了解决DA算法对大规模数据优化性能降低的问题,本文提出基于Spark框架下并行DA算法方案,该算法主要通过节点线程并行更新蜻蜓个体信息,然后共享全局最优解,使DA算法在Spark框架并行计算,最后仿真实验验证该算法确实提高了DA算法对大规模数据优化的效率。下一步笔者研究的目标是:在提高速度的前提下,如何提高DA算法的精确度。

参考文献:

[1] 张霄,钱玉良,邱正,等.基于蜻蜓算法优化BP神经网络的燃气轮机故障诊断[J].热能动力工程,2019(3):26-32.

[2] 马彧廷,郭敏.基于极限学习与蜻蜓算法的小麦碰撞声信号检查与识别[J].电子设计工程,2016,24(5):8-11.

[3] 韩鹏,陈锋.一种改进的多目标蜻蜓优化算法[J].微型机与应用,2017,36(20):27-29.

[4] 崔东文,金波.DA-PNN湖库营养状态识别通用模型研究及应用[J].人民珠江,2015,36(6):101-105.

[5] MAHDAVI S, SHIRI M E, RAHNAMAYAN S. Metaheuristics in large-scale global continues optimization:A survey[J]. Information Sciences,2015,295:407-428.

[6] MIRJALILI S.Dragonfly algorithm:a new meta-heuristic opti-mization technique for solving single-objective,discrete,and multi-objective problems[J].Neural Computing & Appli-cations,2016,27(4):1053-1073.

[7] 吴伟民,吴汪洋,林志毅,等.基于增强个体信息交流的蜻蜓算法[J].计算机工程与应用,2017,53(4):10-14.

[8] 陈虹君,吴雪琴.基于Hadoop平台的Spark快数据推荐算法分析与应用[J].现代电子技术,2016,39(10):18-20.

[9] 桑渊博,曾建潮,谭瑛,等.基于分布式框架的并行PSO算法[J].中北大学学报(自然科学),2019,40(2):126-130.

[10] 陆俊尧,李玲娟.基于Spark的协同过滤算法并行化研究[J].计算机技术与发展,2019(1):85-89.

(责任编辑:蒋召雪)