대규모 협동진화 차등진화

신성윤[1,*] · 탄쉬지에[2] · 신광성[3]· 이현창[3]

[1]군산대학교 · [2]중국 구강대학교 · [3]원광대학교

Large Scale Cooperative Coevolution Differential Evolution

Seong-Yoon Shin[1,*] · Xujie Tan[2] · Kwang-Seong Shin[3] · Hyun-Chang Lee[3]

[1]Kunsan Nationl University · [2]Jiujiang University. · [3]Wonkwang University

E-mail : s3397220@kunsan.ac.kr / txj_2003@163.com / {waver0984, hclglory}@wku.ac.kr

요 약

미분 진화는 연속 최적화 문제에 대한 효율적인 알고리즘이다. 그러나 대규모 최적화 문제를 해결하기 위해 미분 진화를 적용하면 성능이 빠르게 저하되고 런타임이 기하급수적으로 증가한다. 이 문제를 극복하기 위해 Spark(SparkDECC라고 함)를 기반으로 하는 새로운 협력 공진화 미분 진화를 제안한다. 분할 정복 전략은 SparkDECC에서 사용된다.

ABSTRACT

Differential evolution is an efficient algorithm for continuous optimization problems. However, applying differential evolution to solve large-scale optimization problems quickly degrades performance and exponentially increases runtime. To overcome this problem, a new cooperative coevolution differential evolution based on Spark (referred to as SparkDECC) is proposed. The divide-and-conquer strategy is used in SparkDECC.

키워드

Differential evolution, optimization problem, SparkDECC, divide-and-conquer strategy

Ⅰ. 소개

스파크 클라우드 플랫폼은 버클리 대학교[1]에서 제안한 분산 데이터 처리 프레임워크로, 많은 분야에서 성공적으로 적용되고 있다. Spark는 반복 계산 관계형 쿼리를 효과적으로 지원할 수 있는 새로운 데이터 추상화 모델인 RDD(Resilient Distributed Dataset)를 제안합니다. RDD 모델은 인메모리 컴퓨팅을 기반으로 하기 때문에 디스크 데이터를 자주 읽고 쓰는 맵리듀스 모델의 단점을 피하고 효율성을 높인다.

Ⅱ. Spark

반복 컴퓨팅을 보다 효율적으로 지원하기 위해 Spark 플랫폼은 MapReduce 클라우드 모델을 확장한다. Spark 플랫폼은 RDD와 누산기라는 두 가지 중요한 추상화를 제공한다. RDD는 기본적으로 메모리에 공존하는 읽기 전용 분할 레코드 세트를 제공하는 내결함성 병렬 데이터 구조이다. 브로드캐스트는 데이터를 각 노드에 캐시하는 공유 변수로, 데이터를 전송할 필요가 없고, 통신 오버헤드를 줄이며, 통신 성능을 향상시킨다. Spark API는 RDD를 위한 두 가지 작업, 즉 변환과 작업을 제공한다. 변환은 지연되고 각 데이터 파티션에 대해 동일한 작업을 수행하고 새 RDD를 반환한다. 그리고 작업 연산자는 RDD에서 작업을 트리거하고 값을 마스터 제어 노드로 반환합니다. RDD의 내부 구현 메커니즘은 데이터 액세스를 보다 효율적으로 만들고 중간 결과의 메모리 소비를 방지하며 반복 계산을 보다 효율적이고 빠르게 만드는 반복자를 기반으로 한다.

Ⅲ. SparkDECC

대규모 최적화 문제. 그러나 인구 규모가 증가함에 따라 CC 프레임워크에 필요한 시간이 빠르게 증가한다. CC 프레임워크의 수렴 속도를 향상시키

* corresponding author

Table 1. Comparison of SparkDE, OXDE, CoDE, jDE, and PSO algorithms for solving the results

	OXDE		CoDE		jDE		PSO		SparkDECC	
F	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	4.76E+02	1.67E+02	1.50-05	1.74E-05	2.46E-06	1.23E-05	2.30E+06	2.79E+04	**5.85E-13**	**1.62E-13**
f_2	5.27E+01	6.89E+00	8.89E-01	9.74E-01	**1.74E-10**	**8.62E-10**	7.16E+04	3.30E+04	6.60E-07	1.00E-07
f_3	1.54E+06	2.33E+05	**4.50E+04**	**6.12E+03**	7.54E+04	1.48E+04	8.68E+08	6.51E+07	5.31E+07	7.20E+06
f_4	2.59E+01	1.84E+00	**2.67E+01**	**1.95E+00**	5.04E+01	4.58E+00	4.01E+02	7.98E+00	9.76E+01	2.22E-01
f_5	2.28E+04	7.08E+03	2.39E+03	2.12E+02	2.18E+03	2.21E+02	9.23E+12	1.29E+12	**1.62E+03**	**1.62E+02**
f_6	7.86E+03	8.86E+02	5.26E+01	6.26E+01	1.06E+04	2.98E+03	2.30E+06	1.29E+05	**1.60E-01**	**4.73E-01**
f_7	5.68E+00	7.25E-01	**9.06E-01**	**1.03E-01**	2.91E+01	1.60E+01	3.48E+13	3.87E+12	3.62E+00	1.67E-01
f_8	-4.17E+05	7.27E+02	-1.89E+05	5.14E+03	**-4.19E+05**	**3.18E-01**	-1.34E+05	4.51E+03	-6.11E+04	1.18E+03
f_9	4.94E+02	5.32E+01	4.59E+03	2.10E+02	**2.98E+00**	**4.03E+00**	2.33E+06	1.35E+05	1.10E+04	3.93E+01
f_{10}	6.49E+00	2.79E-01	2.25E+00	1.82E-01	5.12E+00	7.06E-01	2.15E+01	3.77E-02	**4.55E-08**	**8.16E-09**
F_{11}	5.02E+00	1.19E+00	7.70E-03	2.29E-02	8.91E-01	7.39E-001	5.75E+02	4.15E+01	**3.54E-14**	**1.03E-14**
f_{12}	3.01E+00	7.11E-11	5.75E-02	4.85E-02	1.32E+06	2.20E+06	6.99E+12	7.75E+11	**7.46E-04**	**2.58E-03**
f_{13}	1.94E+03	3.40E+02	1.15E+02	7.53E+01	1.14E+07	8.17E+06	7.87E+12	8.78E+11	**8.79E-04**	**3.04E-03**
−/+/≈	8/3/2		7/4/2		7/4/2		12/1/0			

기 위해 클라우드 컴퓨팅의 장점을 CC 프레임워크와 결합하고 Spark 알고리즘(SparkDECC) 기반의 협력 공진화 알고리즘을 제안한다.

SparkDECC 알고리즘은 먼저 무작위 그룹화 방법에 따라 고차원 문제를 여러 개의 저차원 하위 문제로 분해한다. 하나의 하위 문제는 하위 개체군에 해당하고 전체 문제에서 각 하위 문제의 위치 정보를 보존한다. keyi 값에 따라 저차원 부분군은 RDD의 해당 파티션에 분산되고 각 파티션의 부분군은 DE 알고리즘의 돌연변이 및 교차 선택을 병렬로 실행한다.

부분 모집단의 개별 적합도 값을 계산할 때 마지막 라운드의 최적 개체를 선택하여 완전한 모집단을 구성하고 로컬 최적화를 수행하도록 협력한다. 해당 파티션에서 여러 세대의 진화를 거친 후 저차원 하위 개체군은 위치 정보에 따라 새로운 완전한 개체군으로 병합되며 전역 검색을 통해 최적의 개체를 반환하는 SparkDECC 알고리즘의 흐름도는 그림 1과 같다.

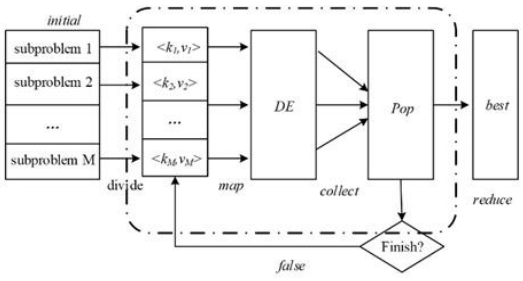

Fig. 1. Flowchart of SparkDECC algorithm

Ⅳ. 실 험

Table 1은 SparkDECC와 OXDE, CoDE, JDE, PSO의 평균최적값과 표준분산을 비교한 것이다. 5가지 알고리즘의 매개변수 설정은 일관되며 각 알고리즘은 독립적으로 25번 실행됩니다. Wilcoxon 순위 합 테스트는 네 가지 알고리즘의 실험 결과를 분석하는 데 사용됩니다. 유의 수준은 0.05로 -, +, ≈은 각각 열등, 우수, 같음을 나타냅니다.

Ⅴ. 결 론

대규모 최적화 문제를 해결하기 위해 미분 진화를 적용하면 성능이 빠르게 저하되고 런타임이 기하급수적으로 증가하는 문제를 극복하기 위해 Spark(SparkDECC라고 함)를 기반으로 하는 새로운 협력 공진화 미분 진화를 제안하였는데, 분할정복 전략은 SparkDECC에서 사용된다.

References

[1] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, I. Stoica, “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing,” in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, San Jose, CA, pp. 15-28, 2012. DOI: 10.5555/2228298.2228301.