文章信息

文章历史：

2011年12月27日收到

以修订后的形式收到2012年6月23日

2012年10月10日接受

2012年10月25日在线可用

关键字：

粒子群优化（PSO）

多元化

邻里搜索

全局优化

抽象

粒子群优化（PSO）已显示出解决变体的有效性能

基准测试和实际优化问题。但是，它患有早产

由于多样性迅速丧失而趋同。为了提高其性能，这

论文提出了一种混合的PSO算法，称为DNSPSO，该算法采用多样性增强机制和邻域搜索策略来实现探索能力与开发能力之间的权衡。在一个场景上进行了全面的实验研究

基准功能，包括旋转的多峰和移动的高维问题。比较结果表明，DNSPSO在互联网上取得了可喜的业绩。

大多数测试问题。

2012 Elsevier Inc.保留所有权利。

1.简介

在过去的几十年中，已提出了几种基于群体智能的变体算法来解决复杂的基准测试和现实世界中的优化问题，例如，粒子群优化（PSO）[29]，蚁群优化（ACO）

[14]，人工蜂群（ABC）[27]，猫群优化[7]等。由于PSO的概念简单，易于实施

尽管效果不错，但它已在进化优化社区中流行。

在PSO中，总体（群）中的每个粒子都飞到其先前的最佳位置（最佳）和全局最佳位置

（最好）。尽管此运动的机制可以导致快速收敛，但是它会出现过早收敛的问题，这意味着在解决多峰问题时，它很容易陷入局部最小值。当粒子

朝着最佳和最佳方向移动时，粒子之间的差异（多样性）逐渐减小。当这个

发生这种情况时，搜索仅限于包含这些相似粒子的较小搜索空间。因此，找到新的可以解决问题的方案非常困难。如果减少多样性发生得太早，将导致过早的停滞。

众所周知，PSO的性能与颗粒的多样性高度相关，尤其是在尝试

避免过早收敛并摆脱局部最优。因此，保持PSO更高的多样性是一项至关重要的任务。

传统上，多样性度量用于分析进化算法（EA），而不是对其进行指导。最近，

Ursem [51]提出了一种以分集为指导的进化算法，该算法使用分集度量在

探索和利用行为。之后，在PSO中采用了一些类似的多样性指导策略来提高其性能[24,40,42,47,54]。

0020-0255 / $-请参见2012 Elsevier Inc.保留所有权利。

http://dx.doi.org/10.1016/j.ins.2012.10.012

author通讯作者。电话：+86 0791 88126661;传真：+86 0791 88126660

电子邮件地址：wanghui\_cug@yahoo.com.cn（H. Wang），sunhui2006 @ yahoo.com.cn（H. Sun），cl160 @ mcs.le.ac.uk（C. Li），shahryar.rahnamayan @ it

.ca（S. Rahnamayan），jspan @ cc.kuas.edu.tw（J.-s. Pan）。

大多数采用多样性指导的PSO算法通常基于监视群体的多样性。当分集下降到预定义常数以下时，将分集增强算符（例如排斥机制）应用于

避免过早收敛。当分集增加到另一个预定义的常数值时，分集减小

运算符（例如贪婪选择）用于支持快速收敛速度。但是，多样性的计算是

费时的任务。为了保持多样性并避免计算多样性，本文提出了一种新颖的多样性

PSO的增强机制。此外，采用邻域搜索策略来改善本地和全局

搜索能力。对45种著名基准函数的实验研究，包括旋转多峰和移位

高维问题，表明我们的方法获得了有希望的性能。

本文的其余部分安排如下。第2节介绍了一些与PSO相关的工作。第3节介绍了我们的建议

方法。第4节介绍了实验模拟，结果和讨论。最后，工作在第5节结束。

2.相关作品

PSO是一种基于种群的随机算法，始于随机产生的粒子的初始种群。对于

在D维空间中的搜索问题中，粒子表示由其速度和位置表示的潜在解。

在搜索过程中，每个粒子都被其先前的最佳粒子（pbest）和全局最佳粒子（gbest）所吸引。

遵循[43]。 vijðtþ1¼wvijðtÞc1 rand1ijðpbestijðtÞxijðtÞÞc2 rand2ijdgbestjðtÞxijðtÞÞ; ð1Þxijðtþ1Þ¼xijðtÞvijðtþ1Þ; ð2Þ

其中i = 1，2，...，N是粒子的索引，N是粒子的大小，Xi =（xi1，xi2，...，xiD）是第i个粒子的位置；

Vi =（vi1，vi2，...，viD）表示第i个粒子的速度； pbesti =（pbesti1，pbesti2，...，pbestiD）是先前的最佳位置，可为第i个粒子产生最佳适应度值；和gbest =（gbest1，gbest2，...，gbestD）是找到的全局最佳粒子

到目前为止的所有粒子。参数w，称为惯性因数，用于平衡全局和局部搜索能力

粒子[43]，rand1ij和rand2ij是在[0,1]，c1和

c2是两个控制社交和认知成分影响的学习因素，t = 1、2，...表示

迭代次数。

自从引入PSO以来，它已成为一种流行的优化器，并已广泛应用于实际问题中。在里面

在过去的几十年中，已经提出了PSO的许多变体。这些变量的简要概述如下。

Shi和Eberhart [43]为经典PSO引入了一个称为惯性权重w的参数。惯性重量用于

为了平衡勘探和开发能力，在搜索过程中线性减小的w是一个不错的选择[43]。

Clerc和Kennedy [8]提出了一个收缩因子，可以保证收敛并改善收敛

速度。 Bergh和Engelbrecht [2]对PSO的参数进行了全面的研究，并提供了正式的证明

每个粒子收敛到一个稳定点。

为了提高PSO的性能，已经提出了不同类型的拓扑。肯尼迪[28,30]分析

邻域拓扑对PSO的影响，并提出了四种不同的邻域拓扑。提出的结果

表明具有较小邻域的PSO在复杂问题上可能表现更好，而具有较大邻域的PSO

在简单的问题上可能会表现更好。 Suganthan [45]引入了一个邻域算子，它逐渐增加了

粒子的邻域大小，直到它覆盖了群中的所有粒子。 Hu和Eberhart [25]更新了

通过动态选择最接近当前粒子的m个粒子来每个粒子。 Mendes和Kennedy [37]提出了一种完全知情的PSO算法（FIPS），在该算法中，每个粒子的邻居（而不是pbest和gbest）用于

更新速度。基于FIPS，Mohais等人。 [38]通过重构提出了随机和动态邻域

就保留多样性而言。 Peram等。 [41]提出了一种改进的PSO，称为基于适应度比的PSO（FDR-PSO），它采用了一种新的速度更新方法。 Yano等。 [58]提出了一种混合动力

具有邻域搜索的PSO算法。当粒子的当前位置好于先前的最佳位置时，

该算法将在当前点附近搜索2D 1个点（D是尺寸大小）。尽管如此，

搜索点的数量随D的增加呈指数增长。为解决此问题，他们定义了最大数量

各种问题的搜索点数。但是，该算法收敛非常慢，因为邻居ood

搜索非常耗时。

Bergh和Engelbrecht [1]提出了一种用于PSO（CPSO-H）的协作方法来解决多模式问题。良

等。 [34]引入了一种全面的学习PSO（CPSO），以学习不同维度上其他粒子的经验。

Chen等。 [6]提出了一种通过动态链接发现和重组来提高PSO性能的新颖的PSO算法。前者是一种无成本且有效的链接识别技术，而后者则利用了已发现的

链接配置，以促进PSO的合作。 Ho等。 [22]介绍了一种用于PSO的正交方法来求解

任务分配问题。 Li和Yang [31,32]提出了一种用于功能优化的自适应学习PSO，其中

每个粒子的学习机制分为三个部分：其自身的历史最佳位置，最近的位置

邻居和全球最佳人选。通过使用这种单独的水平自适应技术，粒子可以很好地控制其勘探和开发的均衡高级行为。 Hsieh等。 [23]提出了一种有效的PSO人口利用策略

（EPUS-PSO），其中引入了人口管理器和解决方案共享策略。在EPUS-PSO中，人口规模为

变量。种群管理器可以根据搜索状态增加或减少粒子数。塞万提斯

等。 [4]提出了一种自适应密歇根州PSO（AMPSO）来减少搜索空间的尺寸。报告结果与

AMPSO能够改善最近邻分类器的性能。詹等。 [59]通过采用以下两种策略提出了一种自适应PSO（APSO）。第一个评估人口分布和粒子

适应性并标识当前的搜索状态。第二种利用精英学习策略帮助全球最佳

粒子跳出可能的局部最优值。

Sun等。 [48]提出了一种改进的向量PSO（IVPSO）来解决约束优化问题。 Wang等。 [56]

引入了自适应学习策略来提高CLPSO的性能。 Wang等。 [55]提出了另一个

通过采用广义的基于对立的学习（GOBL）改进了CLPSO。 Hsul等。 [26]提出了一种新颖的PSO

线性相控阵的多干扰消除设计算法。为了克服依赖于性能的PSO的缺点[3]，并入了四个不同的混沌序列以增强勘探能力[9]。 Chakr流产等。 [5]提出了对通用多目标PSO的简单分析。结果表明，惯性

因子和加速度系数控制算法在目标中向Pareto前沿的收敛行为

功能空间。在[10]中，PSO被认为是一个二输入一输出的反馈系统。 PSO的方法中集成了两个PID控制器，以提高群体的多样性。 Wang等。 [53]引入了广义的基于对立的学习和柯西突变来提高PSO的性能。仿真结果表明所提出的

方法（GOPSO）在解决低维问题方面表现出良好的前景（D 6 30）。但是，GOPSO并不是一个好选择

解决高维问题的选择（D = 100）。 Lu等。 [36]提出了一种新的混合PSO算法，其中将实值变异（RVM）运算符嵌入到PSO算法的三个变体中。在[33]中，李等人。建议修改

Broyden–Fletcher–Goldfarb–Shanno（BFGS）方法已整合到PSO中，以增强本地搜索

能力。仿真结果表明，BFGS可以有效地提高许多PSO变体的性能。

3.通过邻域搜索（DNSPSO）提高多样性的PSO

在本节中，提出了一种新的PSO变体，称为带有邻域搜索的分集增强PSO（DNSPSO）。的

DNSPSO采用两种策略，包括多样性增强机制和邻域搜索。

3.1。多样性增强机制

对于搜索过程中的PSO算法，分集丢失是一个严重的问题。为了保持群体的多样性，一些

已经提出了极好的增强多样性的策略。 Riget [42]提出了一种以多样性为导向的PSO，称为ARPSO，

它基于修改后的速度更新模型定义了排斥阶段。基本的PSO算法仅采用

吸引阶段，其中颗粒被其最佳和最佳吸引。群中的所有粒子快速移动到

粒子之间的相同方向和相似性增长非常快。结果，群体的多样性减小。

为了避免颗粒朝最佳颗粒吸引太快，引入了排斥相。当群的多样性

如果ARPSO降到预定义的恒定数dlow以下，则ARPSO切换到排斥阶段，在此阶段，粒子相互排斥

其他，然后多样性增加。当分集达到高级别（高）时，ARPSO切换回

吸引阶段。 ARPSO的整个搜索过程在开发和探索阶段之间交替-吸引

和排斥-多样性越来越低。但是，当多样性在dlow和dhigh之间保持不变时，ARPSO不会更改搜索行为。 Pant等人基于ARPSO。 [40]提出了一个中间阶段（称为积极冲突阶段）

在吸引和排斥之间。在中间阶段，既没有完全的吸引力也没有完全的排斥。每

先前的最佳粒子吸引了该粒子，而全局最佳粒子则对其进行了排斥。在[47]中，Sun等。介绍了

变异算子增强了种群多样性。当多样性低于dlow时，对gbest进行突变

以增加粒子之间的差异，即最佳和最佳。

尽管上述分集增强策略具有良好的性能，但它们的计算量很大

是时候监视和计算群多样性了。

分集ðtÞ¼1NXNi¼1

ff iff iff iff iff i i i i i i i i i i i i i i i i i i i i i i

XDj¼1xijðtÞxjðtÞ

2 r; ð3ÞxjðtÞ¼

PNi¼1xijðtÞN：ð4Þ

为了避免多样性计算并保持群体多样性，请使用r提出了一种新颖的多样性

机制。 对于每个粒子Pi（t），PSO的速度和位置更新方程式都会生成一个新的粒子Pi（t + 1）。

通过重新组合Pi（t）和Pi（t +1），按如下方式生成试验粒子TPi（t +1）=（TXi（t +1），TVi（t +1））：

TXijðtþ1Þ¼

Xijðtþ1Þ; 如果randjð0; 1Þ<prXijðtÞ; 否则（;ð5Þ

TVijðtþ1Þ¼Vijðtþ1Þ; ð6Þ

其中i = 1，2，...，N，j = 1，2，...，D，randj（0,1）是[0,1]内的统一随机数，而pr是预定义的概率。

TPi（t + 1）中位置矢量的每个元素都继承Xi（t + 1）和Xi（t）中的元素，且概率分别为pr和

1 pr。这类似于差分进化（DE）的交叉操作[44]。清晰的可视化方式

图1给出了生成TPi（t +1）的过程。 （5）较小的pr将增加TPi（t + 1）和Pi（t + 1）之间的不相似性。对于pr = 0的极端情况，TPi（t + 1）等效于Pi（t）。较大的pr将增加TPi（t +1）和Pi（t +1）之间的相似度。对于pr = 1，TPi（t + 1）等于Pi（t + 1）。

重组后，按如下所示使用贪婪选择机制：

Piðtþ1Þ¼TPiðtþ1Þ;如果fðTPiðtþ1ÞÞ6fðPiðtþ1ÞÞ

Piðtþ1Þ;否则; ð7Þ

其中f（）是适应性评估函数。不失一般性，本文仅考虑最小化问题。如果，

并且仅当试验粒子TPi（t + 1）优于Pi（t +1），然后用TPi（t +1）替换Pi（t +1）；否则，Pi（t +1）保持不变。

在PSO中，粒子在搜索过程中倾向于移动相同的位置。这意味着粒子变得与

增加迭代次数。当将试验粒子TPi（t +1）选入下一代时，

TPi（t +1）和Pi（t +1）将确定Pi（t +1）（由TPi（t +1）代替）与其余粒子之间的差异。

一群。群中较大的差异意味着较高的多样性。因此，pr的值控制着群体多样性。

较小的pr会使TPi（t +1）和Pi（t +1）之间的差异更大，从而导致较高的分集，而较大的pr则会降低分集。

通过将PSO和DE方案混合，提出了一些类似的算法，例如[15,16,60]。然而，

我们的方法不同于他们。在[15,16]中，DE算法被用来演化透水的最佳粒子（pbests）以增强

收敛。在[60]中，由DE算法提供的突变是最好的。

3.2。邻里搜索策略

像其他随机搜索算法一样，PSO在解决高度复杂问题时也存在过早收敛的问题

多峰问题。有时，次优接近于全局最优，被困个体的邻域可能包含全局最优。在这种情况下，搜索个人的邻居有助于找到更好的地方

解决方案。基于这种想法，一些优秀的邻域搜索策略已应用于一些自然界

算法[11,35,37,57]。

肯尼迪设计了四种不同的人口拓扑，包括圆形，圆形，星形和随机形[28]。报告的结果表明，连接少的总体拓扑在高度多模式的问题上可能表现更好，而

高度相互联系的人群更适合单峰问题。在[30]中，肯尼迪和门德斯调查了

各种总体拓扑对PSO的影响，以寻求一种在各种测试问题上表现良好的更好的结构。

Suganthan [45]提出了一个变量邻域算子。在优化的初始状态期间，邻域为

单个粒子本身。随着世代数量的增加，邻域逐渐扩展到包括所有

粒子。 Mendes等。 [37]提出了一种完全知情的PSO算法（FIPS），其中每个粒子的邻居代替

pbest和gbest中的，用于更新速度。 Peram等。 [41]开发了基于适应距离比的PSO（称为

FDR-PSO），其中每个粒子都被吸引到其邻居访问的最佳先前位置。胡和埃伯哈特

[25]使用动态邻域粒子群优化算法来解决多目标优化问题。在每一代中，经过计算

在距离其他每个粒子的距离上，每个粒子都找到了新的邻居。在新邻居中，每个粒子都找到本地

最好的粒子。 Ghosh等。 [20]提出了一个概率变量分析，该变量是在具有可变随机邻域拓扑的最佳PSO中粒子交互作用和信息交换的。

一直以来，我们都在寻求EA的勘探与开发之间的权衡。前者表示全局搜索

的能力，使算法探索可行搜索空间的每个区域，而后者意味着局部搜索

能力，并加速算法收敛到接近最优的解决方案。 EA的大多数改进都试图寻求一种

这两个阶段之间的理想平衡，以适应不同类型的问题。为了解决这个问题，本文提出

一种本地和全局全局邻域搜索策略，其灵感来自于差分进化（DE）[11]中的邻域突变算符。

假设Pi，i = 1，2，...，N是群中的第i个粒子，其中N是种群大小。 N个粒子有组织

根据它们的索引在圆拓扑上进行排序，使得PN和P2是P1的两个直接邻居[11]。图2呈现

圆形拓扑，群中有16个粒子。尽管附近有各种各样的拓扑结构（例如星形，

对于PSO [28]，则可以使用环形，金字塔形，四簇和圆形）[28]，环形拓扑结构简单易实现。对于每个粒子Pi

其k邻域半径由粒子Pi k，...，Pi，...，Pi + k组成，其中k为整数0 6 k 6 N 1 2。图2显示了

k邻域半径，其中k =2。根据我们的经验研究，参数k不会影响性能

DNSPSO。 k的不同值可获得相似的性能。在本文中，k设置为2。

3.2.1。本地邻里搜索（LNS）策略

对于每个粒子，其邻域可能会覆盖更好的解决方案。为了提高利用能力，提出了一种局部邻居搜索策略。在搜索粒子Pi的邻域期间，将生成一个试验粒子Li =（LXi，LVi），如下所示[52]：

Lxi¼r1 Xiþr2 pbestiþr3ðXcXdÞ; ð8Þ

LVi¼Vi; ð9Þ

其中Xi是第i个粒子的位置向量，pbesti是Pi的先前最佳粒子，Xc和Xd是以下粒子的位置向量

Pi，c，d 2 [i k，i + k] ^ c – d – i，r1，r2和r3的k邻域半径中的两个随机粒子是三个均匀随机

（0,1）内的数字，且r1 + r2 + r3 =1。对于所有j = 1、2，...，D，随机数r1，r2和r3相同，它们分别是

每一代都重新产生。 LNS机理的清晰解释在图3a中给出。 pbesti是

Xi的先前最佳粒子，因此它不在圆拓扑上。为了保持Pi的飞行方向，试验粒子Li保持

Pi的相同速度。

当局部最小值接近全局最优值时，局部邻域搜索策略是有效的。带有粒子

大的跳动很容易陷入局部最优状态。 LNS可以在当前搜索点附近生成试验粒子。这是

有助于逐步找到全局最优值

3.2.2。全球邻里搜索（GNS）策略

除了LNS之外，还提出了一种全球邻域搜索（GNS）策略，以增强探索能力。什么时候

搜索粒子Pi的附近区域，将生成另一个试验粒子Gi =（GXi，GVi），如下所示[52]

Gxi¼r4 Xiþr5 gbestþr6ðXeXfÞ; ð10Þ

GVi¼Vi； ð11Þ

其中gbest是全局最佳粒子，Xe和Xf是为整个群选择的两个随机粒子的位置矢量，

e，f 2 [1，N] ^ e – f – i，r4，r5和r6是（0，1）内的三个均匀随机数，并且r4 + r5 + r6 =1。随机数

对于所有j = 1，2，...，D，r4，r5和r6都相同，并且它们在每一代中都重新生成。 GNS策略有助于

解决多式联运问题。粒子位于不同的区域。因此，如果当前粒子落入局部最小值，

其他区域中的粒子可能会将被捕获的粒子向前拉。

对于LNS和GNS策略，新生成的试验粒子Li和Gi保持与其父Pi相同的速度。

这旨在遵循与父母相同的飞行方向和跳跃大小。

3.3。拟议的方法

DNSPSO算法采用两种策略，包括多样性增强机制和邻域搜索运算符。前者旨在间接增强种群多样性，而后者则着重于寻找种群的邻居。

粒子。

在每一代中，重组算子Eqs都会生成试验粒子TPi。 （eqdiv1）和（6）。如果TPi更好

而不是其父Pi，然后用TPi替换Pi；否则，我们保持Pi不变。完成此操作后，邻域搜索

策略以pns概率进行。如果满足概率pns，则生成两个试验粒子Li和Gi。然后，

Pi，Li和Gi中最合适的粒子被选为新的Pi。

算法1.建议的DNSPSO算法

1均匀随机地初始化群中的每个粒子；

2初始化pbest和gbest；

FE 3个，而MAX\_FE 6个

i = 1至N做4

5根据等式计算粒子Pi的速度。 （1）; 6根据公式更新粒子Pi的位置。 （2）; 7计算Pi的适应度值； 8个FEs ++;

/增强多样性的机制/ 9根据方程式生成一个新的试验粒子TPi。 （5）和（6）；

10计算TPi的适应度值；

11 FEs ++;

12在Pi和TPi之间选择一个钳工作为新的Pi（请参见等式（7））；

13更新pbesti和gbest；

14末

i = 1至N做15

/邻里搜索策略/

16如果rand（0,1）6 pns，则

17根据方程式生成试验粒子Li。 （8）和（9）；

18根据等式生成试验粒子Gi。 （10）和（11）；

19计算Li和Gi的适应度值；

20个FE = FE + 2;

21从Pi，Li和Gi中选择最合适的一个作为新的Pi；

22末

23更新pbesti和gbest；

24末

25结束

DNSPSO的主要步骤在算法1中进行了描述，其中Pi是群中的第i个粒子，N是种群大小，pns

是进行邻域搜索的概率，FE是适应度评估的次数，MAX\_FE是最大值

功能评估的次数。

4.实验验证

4.1。测试问题

在以下实验中使用了15种基准功能。这些问题已在以前的研究中得到利用

[34,53]。根据它们的性质，它们分为四类：单峰和简单多峰问题（f1 f2），

未旋转的多峰问题（f3 f7），旋转的多峰问题（f8 f11）和高维位移问题

（f12 f15）。本文使用的所有问题都是最小化问题。表1中列出了这些基准问题的简要说明。有关基准问题定义的更多详细信息，请参见[34,50]。

4.2。参数灵敏度分析

pr和pns的值可能会影响DNSPSO的性能。要为这两个控制参数选择更好的值，

我们研究了在变体pr和pns值下DNSPSO的性能。在本节中，pr和pns选自

分别设置{0.0，0.3，0.6，0.9}和{0.0，0.2，0.4，0.6，0.8,1.0}。因此，pr和pns的组合有4 6 = 24个可能的选择。

对于DNSPSO的其他参数，我们使用了以下设置。人口总数N设置为40。w = 0.7298，

c1 = c2 = 1.49618，k =2。对于D = 30，FE的最大数量（MAX\_FEs）设置为200,000 [34]。对于D = 100，

MAX\_FEs = 5000 D [50]。所有实验均进行了30次，平均误差值f（x）f（xo

）（f（xo

）是

记录了f（x）的全局最优值。

对于上述参数的不同选择，总共有24组结果。为了简化这些结果，

表2给出了相对较好的参数设置。从结果可以看出，较小的pr是合适的

对于低维问题（f1 f11），而较大的pr有助于解决高维问题（f12 f15）。如前所述，较小的pr会增加多样性。这减慢了DNSPSO的收敛速度。这是较小的pr在高维问题上表现不佳的可能原因之一。虽然DNSPSO的性能不佳

对pns值敏感，这并不意味着邻居搜索在搜索过程中无法正常工作

处理。根据我们的实验，当pns = 0.0时，DNSPSO的性能非常差。即使将pns设置为0.05，

DNSPSO效果也很好。

为了在统计水平上选择最佳的参数设置组，根据[13,17]的建议进行Friedman检验的平均排名。表3显示了使用不同参数设置的DNSPSO的平均排名。如图所示，

pr = 0.9，pns = 0.6达到最高排名。它表明pr = 0.9和pns = 0.6是相对最佳的选择。

测试套件。

4.3。不同策略的影响

该方法采用了两种策略：多样性增强机制和邻域搜索。为了研究这两种策略的效果，我们研究了标准PSO，具有多样性增强机制的PSO（DPSO），具有邻域搜索的PSO（NSPSO）以及包括这两种机制的DNSPSO的性能。这将有助于

分别验证这些策略的有效性。

对于DPSO，NSPSO和DNSPSO中使用的参数pr和pns，它们分别设置为0.9和0.6。对于其他参数，我们使用了与第4.2节中所述相同的设置。

表格1

实验中使用的基准问题，其中D是函数的维数，X 2 RD是定义域，而f（xo

）是

功能。

问题名称DX f（xo）f1球面函数30 [100，100] 0

f2 Rosenbrock的函数30 [2.048，2.048] 0

f3 Ackley的功能30 [32.768，32.768] 0

f4 Griewanks的函数30 [600，600] 0

f5 Weierstrass函数30 [0.5，0.5] 0

f6 Rastrigin的函数30 [5.12，5.12] 0

f7非连续Rastrigin函数30 [5.12，5.12] 0

f8旋转阿克莱功能30 [32.768，32.768] 0

f9旋转格里旺克斯的功能30 [600，600] 0

f10旋转的Weierstrass函数30 [0.5，0.5] 0

f11旋转Rastrigin的功能30 [5.12，5.12] 0

f12移球函数100 [100，100] 450

f13移位的Schwefel函数2.21 100 [100，100] 450

f14转移的Rosenbrock函数100 [100，100] 390

f15转移的格里旺克斯的功能100 [600，600] 180

Ë

表2

DNSPSO在不同的pr和pns下获得的平均误差值。九组参数设置中最好的结果以粗体显示。

函数D pr = 0.0 pr = 0.0 pr = 0.0 pr = 0.3 pr = 0.6 pr = 0.9 pr = 0.9 pr = 0.9 pr = 0.9

pns = 0.2 pns = 0.4 pns = 0.6 pns = 0.8 pns = 0.8 pns = 0.2 pns = 0.4 pns = 0.6 pns = 0.8

平均误差平均误差平均误差平均误差平均误差平均误差平均误差平均误差平均误差平均误差

f1 30 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00

f2 30 1.31E + 01 1.53E + 01 1.83E + 01 2.49E + 01 2.24E + 01 1.93E + 01 2.05E + 01 2.01E + 01 2.06E + 01

f3 30 5.89E 16 5.89E 16 5.89E 16 5.89E 16 5.89E 16 5.89E 16 5.89E 16 5.89E 16 5.89E 16

f4 30 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00

f5 30 6.48E 13 3.27E 13 5.34E 13 1.61E 13 1.59E 13 2.04E 13 1.68E 13 1.40 E 13 2.63E 13

f6 30 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00

f7 30 0.00E + 00 0.00E + 00 0.00E + 00 2.40E + 01 3.84E 01 5.98E 02 0.00E + 00 0.00E + 00 0.00E + 00

f8 30 5.89E 16 5.89E 16 5.89E 16 5.89E 16 5.89E 16 5.89E 16 5.89E 16 5.89E 16 5.89E 16

f9 30 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00

f10 30 2.35E 13 2.14 E 13 4.38E 13 3.98E + 01 2.65E + 01 3.71E + 01 3.76E + 01 2.69E + 01 3.86E + 01

f11 30 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00

f12 100 6.95E + 04 8.71E + 04 1.04E + 05 4.08E + 03 3.79E 01 3.57 E 13 1.95E 11 4.24E 11 7.39E 09

f13 100 5.23E + 01 5.49E + 01 5.39E + 01 3.44E + 01 3.76E + 01 2.43E + 01 3.34E + 01 2.38E + 01 2.70E + 01

f14 100 3.94E + 08 1.38E + 09 3.12E + 09 1.14E + 08 1.00E + 06 3.60E + 02 2.36E + 02 2.19E + 02 2.78E + 02

f15 100 1.08E + 02 8.35E + 01 4.99E + 01 8.29E 02 2.90E 01 1.65E 01 5.46E 04 2.93 E 06 1.88E 02

表3

Friedman测试使用不同参数设置的DNSPSO所获得的平均排名。最高排名

以粗体显示。

DNSPSO排名

pr = 0.0，pns = 0.2 4.63

pr = 0.0，pns = 0.4 4.57

pr = 0.0，pns = 0.6 4.30

pr = 0.3，pns = 0.8 4.27

pr = 0.6，pns = 0.8 4.73

pr = 0.9，pns = 0.2 5.40

pr = 0.9，pns = 0.4 5.64

pr = 0.9，pns = 0.6 6.30

pr = 0.9，pns = 0.8 5.17

表4

PSO，NSPSO，DPSO和DNSPSO所获得的平均误差值。四种算法中最好的结果以粗体显示。

函数D PSO平均误差NSPSO平均误差DPSO平均误差DNSPSO平均误差

f1 30 2.95E 92 0.00E + 00 5.33E 49 0.00E + 00

f2 30 2.36E + 01 1.84E + 01 1.74E + 01 2.01E + 01

f3 30 1.50E + 00 5.89E 16 1.34E + 00 5.89E 16

f4 30 4.65E 02 0.00E + 00 1.72E 02 0.00E + 00

f5 30 2.98E + 00 7.39E 13 1.78E 13 1.40E 13

f6 30 4.28E + 01 0.00E + 00 1.82E + 01 0.00E + 00

f7 30 1.60E + 01 0.00E + 00 9.27E + 00 0.00E + 00

f8 30 3.55E + 00 9.55E 13 2.89E + 00 5.89E 16

f9 30 1.97E 02 0.00E + 00 1.11E 16 0.00E + 00

f10 30 6.07E + 00 5.86E 01 3.22E + 01 2.69E + 01

f11 30 7.36E + 01 0.00E + 00 6.95E + 01 0.00E + 00

f12 100 3.70E + 04 9.17E + 04 5.06E + 04 4.24E 11

f13 100 3.92E + 01 3.46E + 01 3.81E + 01 2.38E + 01

f14 100 1.97E + 02 1.22E + 07 1.76E + 07 2.19E + 02

f15 100 7.90E + 02 9.04E + 01 8.39E + 00 2.93E 06

表4列出了以上四种算法的平均误差值。从NSPSO与PSO的比较来看，NSPSO

除f14以外，在所有测试案例中均取得了比PSO更好的结果。在此功能上，NSPSO和DPSO都几乎没有改进

解决方案的质量。由于f14是移动的Rosenbrock函数，并且许多PSO变体几乎找不到全局最优值

它位于一个细长的，抛物线形的平坦山谷中。这表明邻域搜索对于大多数测试都是有效的

功能。在其余的功能上，PSO在f1，f10和f14上的性能优于DPSO，而DPSO则优于PSO。 DPSO与PSO的比较证明了多样性增强机制的有效性。

通过结合这两种策略，DNSPSO可获得出色的性能。它在以下方面优于其他三种PSO算法

大多数测试功能。特别是对于高维问题（f12 f15），使用单一策略（多样性增强机制或邻域搜索）很难获得有希望的解决方案。对于f12和f15，NSPSO和DPSO

降至本地最小值，而DNSPSO取得令人满意的结果。多样性增强机制有助于增加

表2

DNSPSO在不同的pr和pns下获得的平均误差值。九组参数设置中最好的结果以粗体显示。

函数D pr = 0.0 pr = 0.0 pr = 0.0 pr = 0.3 pr = 0.6 pr = 0.9 pr = 0.9 pr = 0.9 pr = 0.9

pns = 0.2 pns = 0.4 pns = 0.6 pns = 0.8 pns = 0.8 pns = 0.2 pns = 0.4 pns = 0.6 pns = 0.8

平均误差平均误差平均误差平均误差平均误差平均误差平均误差平均误差平均误差平均误差

f1 30 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00

f2 30 1.31E + 01 1.53E + 01 1.83E + 01 2.49E + 01 2.24E + 01 1.93E + 01 2.05E + 01 2.01E + 01 2.06E + 01

f3 30 5.89E16 5.89E16 5.89E16 5.89E16 5.89E16 5.89E16 5.89E16 5.89E16 5.89E16

f4 30 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00

f5 30 6.48E13 3.27E13 5.34E13 1.61E13 1.59E13 2.04E13 1.68E13 1.40 E13 2.63E13

f6 30 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00

f7 30 0.00E + 00 0.00E + 00 0.00E + 00 2.40E + 01 3.84E01 5.98E02 0.00E + 00 0.00E + 00 0.00E + 00

f8 30 5.89E16 5.89E16 5.89E16 5.89E16 5.89E16 5.89E16 5.89E16 5.89E16 5.89E16

f9 30 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00

f10 30 2.35E13 2.14 E13 4.38E13 3.98E + 01 2.65E + 01 3.71E + 01 3.76E + 01 2.69E + 01 3.86E + 01

f11 30 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00 0.00E + 00

f12 100 6.95E + 04 8.71E + 04 1.04E + 05 4.08E + 03 3.79E01 3.57 E13 1.95E11 4.24E11 7.39E09

f13 100 5.23E + 01 5.49E + 01 5.39E + 01 3.44E + 01 3.76E + 01 2.43E + 01 3.34E + 01 2.38E + 01 2.70E + 01

f14 100 3.94E + 08 1.38E + 09 3.12E + 09 1.14E + 08 1.00E + 06 3.60E + 02 2.36E + 02 2.19E + 02 2.78E + 02

f15 100 1.08E + 02 8.35E + 01 4.99E + 01 8.29E02 2.90E01 1.65E01 5.46E04 2.93 E06 1.88E02

表3

Friedman测试使用不同参数设置的DNSPSO所获得的平均排名。最高排名

以粗体显示。

DNSPSO排名

pr = 0.0，pns = 0.2 4.63

pr = 0.0，pns = 0.4 4.57

pr = 0.0，pns = 0.6 4.30

pr = 0.3，pns = 0.8 4.27

pr = 0.6，pns = 0.8 4.73

pr = 0.9，pns = 0.2 5.40

pr = 0.9，pns = 0.4 5.64

pr = 0.9，pns = 0.6 6.30

pr = 0.9，pns = 0.8 5.17

表4

PSO，NSPSO，DPSO和DNSPSO所获得的平均误差值。四种算法中最好的结果以粗体显示。

函数D PSO平均误差NSPSO平均误差DPSO平均误差DNSPSO平均误差

f1 30 2.95E92 0.00E + 00 5.33E49 0.00E + 00

f2 30 2.36E + 01 1.84E + 01 1.74E + 01 2.01E + 01

f3 30 1.50E + 00 5.89E16 1.34E + 00 5.89E16

f4 30 4.65E02 0.00E + 00 1.72E02 0.00E + 00

f5 30 2.98E + 00 7.39E13 1.78E13 1.40E13

f6 30 4.28E + 01 0.00E + 00 1.82E + 01 0.00E + 00

f7 30 1.60E + 01 0.00E + 00 9.27E + 00 0.00E + 00

f8 30 3.55E + 00 9.55E13 2.89E + 00 5.89E16

f9 30 1.97E02 0.00E + 00 1.11E16 0.00E + 00

f10 30 6.07E + 00 5.86E01 3.22E + 01 2.69E + 01

f11 30 7.36E + 01 0.00E + 00 6.95E + 01 0.00E + 00

f12 100 3.70E + 04 9.17E + 04 5.06E + 04 4.24E11

f13 100 3.92E + 01 3.46E + 01 3.81E + 01 2.38E + 01

f14 100 1.97E + 02 1.22E + 07 1.76E + 07 2.19E + 02

f15 100 7.90E + 02 9.04E + 01 8.39E + 00 2.93E06

126 H. Wang等。 /信息科学223（2013）119–135

群多样性，同时减慢收敛速度。可以从非移位球体函数和移位球体函数看出

（f1和f12），与DPSO相比，PSO可获得更准确的解决方案。邻域搜索基于

最佳或最佳。这有助于加快PSO的收敛速度，但存在失去群体多样性的风险。通过将这两种策略进行混合，DNSPSO可以在勘探和开发能力之间取得平衡。

DNSPSO采用两种策略：多样性增强机制和邻域搜索，这可能会花费额外的计算时间。为了研究这些策略如何影响计算时间，我们在测试套件上计算了PSO，NSPSO，DPSO和DNSPSO的平均计算时间。计算配置如下。

系统：Windows XP（SP3）。

CPU：Intel Core 2 Duo CPU T6400（2.00 GHz）。

内存：2G。

语言：C ++。

编译器：Microsoft Visual C ++ 6.0。

MAX\_FE：2.00E + 05（D = 30）和5000 D（D = 100）。

在实验中，使用与第4.2节中所述相同的参数设置。的平均计算时间

表5中列出了测试套件上的四种算法。可以看出，DPSO比PSO消耗的时间更少。主要原因是

DPSO的迭代次数少于PSO。在实验中，所有算法都使用相同的MAX\_FE。对于PSO，最大值

世代数（MAX\_G）为MAX FE

N，而DPSO的MAX\_G为MAX FEs。

2N。因为DPSO评估两次适应度值

每一代。对于NSPSO，MAX\_G为MAX FE

NþN2分

¼个最大FE

2：2N。因此，NSPSO比DPSO和PSO消耗更少的迭代。的

计算时间的结果证实，NSPSO比PSO和DPSO花费更少的计算时间。对于DNSPSO，

MAX\_G为MAX FE

3：2N。虽然，DNSPSO比NSPSO花费更少的迭代，但它花费更多的计算时间。主要原因

是多样性机制的计算复杂度高于邻域搜索策略。以上

结果表明，多样性机制和邻域搜索的结合不会增加额外的

使用相同的MAX\_FE时DNSPSO的计算时间。

4.4。 DNSPSO与其他最新PSO变体的比较

4.4.1。 DNSPSO与CPSO-H，CLPSO和APSO的比较

本节对15种测试功能进行了DNSPSO与CPSO-H，CLPSO和APSO的比较研究。验证

我们提出的策略的一般性，我们将提出的多样性机制和邻域搜索策略整合到

CLPSO。这种新方法的组合称为DNSCLPSO。涉及的算法如下：

合作式PSO（CPSO-H）[1]。全面学习PSO（CLPSO）[34]。自适应PSO（APSO）[59]。具有广义的基于对立面学习的PSO（GOPSO）[53]。将CLPSO与建议的多样性机制和邻域搜索策略（DNSCLPSO）相结合。

我们的方法DNSPSO。

[34]中介绍了CLPSO-H和CLPSO的参数设置。根据[59]的建议，使用了ALPSO的相同参数设置。对于GOPSO和DNSPSO，w = 0.7298，c1 = c2 = 1.49618。在GOPSO中使用异议的可能性

设置为0.3。 DNSPSO中使用的参数k，pr，pns分别设置为2、0.9和0.6。对于DNSCLPSO，相同参数表5

PSO，NSPSO，DPSO和DNSPSO达到的平均计算时间（以秒为单位）。

功能D PSO时间NSPSO时间DPSO时间DNSPSO时间

f1 30 0.84 0.74 0.83 0.75

f2 30 0.86 0.70 0.86 0.75

f3 30 1.29 1.13 1.25 1.14

f4 30 1.38 1.16 1.28 1.20

f5 30 43.46 42.17 42.76 43.72

f6 30 1.26 1.05 1.26 1.14

f7 30 1.74 1.56 1.73 1.48

f8 30 2.57 2.32 2.48 2.44

f9 30 2.60 2.38 2.52 2.74

f10 30 49.60 46.81 48.54 47.06

f11 30 2.45 2.18 2.36 2.35

f12 100 6.88 4.91 6.95 4.95

f13 100 8.06 7.16 8.13 7.80

f14 100 28.92 27.24 31.61 27.72

f15 100 20.70 16.80 19.06 16.89

使用CLPSO和DNSPSO的设置。以上六个PSO算法使用相同的总体大小（N = 40）和适应性评估的最大数量（MAX\_FEs）。对于D = 30，MAX\_FEs设置为200,000 [34]。对于D = 100，MAX\_FEs设置为

5000 D [53]。所有实验均进行30次，平均误差值f（x）f（xo

）（f（xo

）是全局最优值

f（x））并记录方差。

表6列出了这6种PSO算法获得的平均误差值和标准差的结果。DNSPSO与其他算法之间的比较结果在表的最后一行中总结为“ w / t / l”，意思就是

与竞争对手相比，DNSPSO在W功能方面胜出，在T功能方面有联系，而在L功能方面则失败。

根据表6的结果，DNSPSO在12个功能上的性能优于CPSO-H，而CPSO-H在3个功能上仅获得更好的结果

功能。在15点上，CLPSO仅比DNSPSO获得更好的性能，而其余14点，DNSPSO的性能优于CLPSO。

功能。在两个功能上，APSO的效果均优于DNSPSO，而在其余13个功能上，DNSPSO的效果更好。 GOPSO和DNSPSO都可以在六个功能上找到全局最优值。对于其余9个功能，DNSPSO赢得7个，而

GOPSO仅赢2。

所有算法都针对已移动和未移动的Rosenbrock函数（f2和f14）处于局部最小值，其全局最优值为

在一个长的，抛物线形的平坦山谷中。大多数PSO算法都可以轻松找到谷底，但很难收敛到全局

最佳。对于移位的Schwefel函数（f13），尽管DNSPSO比其他算法具有更好的解决方案，但所有五个PSO

由于尺寸的影响，算法陷入局部极小值。对于低维Schwefel函数，许多PSO算法可以找到有前途的解决方案。随着尺寸的增加，这些算法的性能会受到严重影响。

通过将CLPSO与我们提出的策略相结合，DNSCLPSO在11个功能上实现了重大改进。特别是对于f3，f4，f9和f10，DNSCLPSO能够找到全局最优值，而CLPSO则属于局部最优值。但是，我们的

建议的方法并不总是适用于所有测试功能。对于f6和f7，CLPSO获得了合理的解决方案，而

DNSCLPSO卡住了。对于f14，DNSCLPSO几乎找不到好的解决方案。

图4显示了这六个PSO算法的收敛曲线。本文仅提出四个代表性收敛

高维函数图（f12 f15）。可以看出，对于转移的Sphere和转移的Rosenbrock函数，

在演进过程中，DNSPSO的收敛速度比其他四种算法快。对于移位的Schwefel函数，DNSPSO在演化的最后阶段显示出更快的收敛速度。对于转移的Griewank函数，DNSPSO在

DNSCLPSO实现了比其他算法的进化的开始和中间阶段更快的收敛

比演化的最后阶段的其他算法要好。

为了比较测试套件上多种算法的性能，我们根据

[13,18]的建议。表7显示了CPSO-H，CLPSO，APSO，GOPSO，DNSCLPSO和DNSPSO的平均排名。的

最高排名以粗体显示。如图所示，这六个算法的性能排名如下：DNSPSO，DNSCLPSO，GOP SO，CLPSO，CPSO-H和APSO。最高的平均排名是通过DNSPSO算法获得的。它表明DNS PSO是六种PSO算法中最好的一种。

为了比较DNSPSO和其他四种PSO算法之间的性能差异，我们进行了Wilcoxon

秩检验[12,19]。表8显示了在DNSPSO和其他五个算法之间进行比较时所得的p值。低于0.05的p值以粗体显示。从结果可以看出，DNSPSO明显优于

除GOPSO和DNSCLPSO之外的所有算法。尽管DNSPSO并不明显优于GOPSO和DNSCLPSO，

根据表7中所示的平均排名，DNSPSO的性能要优于它们。

4.4.2。基于实值变异的DNSPSO与PSO的比较

最近，Lu等。 [36]提出了一种新的混合PSO算法，其中实数突变（RVM）算子是

嵌入到PSO算法的三个变体中。对六个基准函数的实验结果表明，RVM

可以有效地提高PSO变体的性能。在本节中，我们将对DNSPSO与

基于RVM的PSO变体。涉及的算法如下。

表6

平均值函数误差值和标准偏差的比较结果，其中“ w / t / l”表示DNSPSO在w函数中获胜，在t函数中归结，而在

l功能，与其竞争对手相比。比较中最好的结果以粗体显示。

功能D CPSO-H [1] CLPSO [34] APSO [59] GOPSO [53] DNSCLPSO DNSPSO

f1 30 1.29E 36±7.61E 36 1.23E 13±3.09E 13 9.60E 66±1.57E 65 0.00E + 00±0.00E + 00 1.23E 50±2.86E 50 0.00E + 00±0.00E + 00

f2 30 1.37E + 01±9.86E + 00 2.08E + 01±1.24E + 01 1.83E + 01±1.46E + 01 1.48E + 01±9.57E 01 2.64E + 01±1.38E + 01 2.01E + 01±1.05E + 01

f3 30 2.25E 14±3.07E 14 1.85E 07±2.70E 07 1.09E 14±1.94E 14 3.43E–15±1.59E 15 5.89E 16±0.00E + 00 5.89E 16±0.00E + 00

f4 30 1.90E 02±8.81E 02 4.37E 09±5.06E 08 1.20E 02±9.14E 02 0.00E + 00±0.00E + 00 0.00E + 00±0.00E + 00 0.00E + 00±0.00E + 00

f5 30 4.74E 15±3.67E 14 5.62E 07±1.38E 06 4.77E 02±4.08E 01 1.04E 08±2.19E 08 0.00E + 00±0.00E + 00 1.40E 13±5.29E 14

f6 30 3.32E + 00±1.18E + 01 1.50E 04±6.96E 04 6.27E + 00±1.28E + 01 0.00E + 00±0.00E + 00 2.20E + 00±1.96E + 00 0.00E + 00±0.00E + 00

f7 30 6.67E 01±6.05E + 00 1.93E 03±6.45E 03 2.27E + 00±2.45E + 01 0.00E + 00±0.00E + 00 3.35E + 00±1.76E + 00 0.00E + 00±0.00E + 00

f8 30 1.82E 01±2.28E + 00 1.07E 05±3.77E 05 1.22E + 00±5.33E + 00 9.59E 13±0.00E + 00 5.89E 16±0.00E + 00 5.89E 16±0.00E + 00

f9 30 2.30E 02±1.01E 01 6.49E 05±2.74E 04 1.38E 02±8.30E 02 0.00E + 00±0.00E + 00 0.00E + 00±0.00E + 00 0.00E + 00±0.00E + 00

f10 30 8.33E + 00±2.00E + 01 2.99E + 00±4.11E + 00 8.40E + 00±1.37E + 01 2.64E 13±2.45E 13 0.00E + 00±0.00E + 00 2.69E + 01±2.87E-01

f11 30 7.33E + 01±1.15E + 02 5.48E + 01±4.69E + 01 7.09E + 01±1.20E + 02 0.00E + 00±0.00E + 00 1.57E + 01±1.03E + 01 0.00E + 00±0.00E + 00

f12 100 7.29E + 02±2.46E + 02 4.61E 10±6.99E 10 1.63E + 04±2.17E + 04 3.43E + 03±4.82E + 02 1.87E 10±5.73E 10 4.24E 11±3.26E 11

f13 100 3.39E + 01±3.38E + 01 4.47E + 01±3.71E + 01 1.01E + 02±1.79E + 02 3.37E + 01±3.63E + 00 2.99E + 01±1.85E + 01 2.38E + 01±1.84E + 00

f14 100 1.07E + 05±3.13E + 06 2.80E + 02±2.55E + 02 1.95E + 09±3.12E + 10 4.53E + 07±8.41E + 06 1.02E + 04±6.17E + 04 2.19E + 02±9.92E + 01

f15 100 3.88E + 00±1.42E + 01 1.50E 09±1.06E 08 1.19E + 02±1.53E + 03 2.15E + 00±1.73E 01 7.75 E 10±3.52E 10 2.93E 06±6.61E 06

w / t / l 12/0/3 14/0/1 13/0/2 7/6/2 8/4/3 –

图4. CPSO-H，CLPSO，APSO，GOPSO，DNSCLPSO和DNSPSO在高维函数（f12 f15）上的收敛曲线。

表7

Friedman测试得出的六个PSO算法的平均排名。的

最高排名以粗体显示。

算法排名

DNSPSO 4.87

DNSCLPSO 4.33

高普索4.27

CLPSO 3.07

CPSO-H 2.60

APSO 1.87

表8

DNSPSO与功能f1f15上的其他PSO算法之间的Wilcoxon测试。

低于0.05的p值以粗体显示。

DNSPSO与p值

CPSO-H 4.68E02

CLPSO 1.99E02

APSO 1.46E02

高普索2.14E01

DNSCLPSO 1.31E01

H. Wang等。 /信息科学223（2013）119–135129

带有RVM的BPSO（标准PSO）（BPSO-RVM）[36]。 CPSO（具有收缩因子的PSO）和RVM（CPSO-RVM）[36]。 CBPSO（具有收缩因子和惯性权重的PSO）与RVM（CBPSO-RVM）[36]。具有高斯突变的PSO（PSO-GM）[21]。我们的方法DNSPSO。

对于所有算法，公共参数使用相同的设置。根据[36]的建议，人口规模

（N）和适应性评估的最大数量（MAX\_FEs）分别设置为20和10,000 20（在[36]中，最大最大代数设置为10,000。）。

表9给出了六个基准函数的平均适应误差值和标准偏差的比较结果。

表格的最后一行将DNSPSO与其他四种算法的比较结果汇总为“ w / t / l”。如

可以看出，DNSPSO在所有测试功能上均优于BPSO-RVM和CPSO-RVM。 CBPSO-RVM和PSO-GM取得更好的结果

在Ronsenbrock功能上比DNSPSO更好，而在其余五个功能上DNSPSO表现更好。

4.5。 CEC 2005基准测试的比较结果

为了进一步验证DNSPSO的性能，在Windows 2000中使用了一组十个CEC 2005转换和旋转的基准功能。

这个部分。表10给出了这些功能的简单说明。他们的更详细定义可以在[46]中找到。

在实验中，将DNSPSO与其他五种PSO算法进行了比较。涉及的算法如下：

合作式PSO（CPSO-H）[1]。全面学习PSO（CLPSO）[34]。自适应PSO（APSO）[59]。具有广义的基于对立面学习的PSO（GOPSO）[53]。将CLPSO与建议的多样性机制和邻域搜索策略（DNSCLPSO）相结合。

我们的方法DNSPSO。

对于上述六种算法，除了MAX\_FE以外，使用与第4.4节中所述相同的参数设置。对于

在此测试套件中，根据[46]的建议，将MAX\_FE设置为3.00E + 05。对于每个测试功能，每种算法为

运行25次。在整个实验中，均会报告平均功能误差值。

CEC 2005基准上的平均功能误差值和标准偏差的比较结果列在

表11中最好的部分以粗体显示。总结了DNSPSO与其他算法的比较结果

在表的最后一行中用“ w / t / l”表示，这意味着DNSPSO与竞争对手相比，在w函数中胜出，在t函数中有联系，而在功能中却失去了。

表9

平均适应性误差值和标准偏差的比较结果，其中“ w / t / l”表示DNSPSO在w函数中获胜，在t函数中联系而在l函数中失败

功能，与其竞争对手相比。比较中最好的结果以粗体显示。

功能BPSO-RVM [36] CPSO-RVM [36] CBPSO-RVM [36] PSO-GM [21] DNSPSO

球面函数3.15E 38±1.77E 37 2.62E 116±1.70E 115 1.65E 146±2.48E 146 5.25E 53±1.52E 52 0.00E + 00±0.00E + 00

二次函数2.73E + 01±2.45E + 01 2.84E 04±5.24E 04 3.25E 21±1.16E 20 5.63E 04±9.27E 04 1.03E 125±5.19E 126

罗森布鲁克

功能

1.82E + 01±8.09​​E + 00 1.22E + 01±4.29E + 01 2.41E + 00±3.46E + 00 3.70E + 00±4.05E + 00 7.06E + 00±8.25E + 00

格里万克

功能

9.20E 03±1.10E 02 6.80E 03±1.16E 02 7.20E 03±1.03E 02 3.84E 04±3.20E 03 0.00E + 00±0.00E + 00

Rastrigin函数1.17E + 01±2.52E + 01 2.86E 01±1.88E + 01 1.68E 13±1.05E 12 4.56E + 00±1.10E + 01 0.00E + 00±0.00E + 00

Ackley功能1.22E 14±3.38E 15 1.52E 14±2.83E 15 1.17E 14±3.46E 15 8.74E 14±2.16E 15 5.89E 16±0.00E + 00

w / t / l 6/0/0 6/0/0 5/0/1 5/0/1 –

表10

实验中使用的十个CEC 2005基准函数，其中D为维度，f（xo

）是全局最优值。

函数名称D f（xo）fcec051位移球函数30450

fcec052转移了Schwefel的问题1.2 30 450

fcec053移位的旋转高条件椭圆函数30450

fcec054转移了Schwefel的问题1.2，有噪音30450

fcec055施韦费尔的问题2.6 30 310

fcec056转移的Rosenbrock的功能30390

fcec057移位的旋转格里旺克函数30180

fcec058移位的旋转阿克莱功能30 140

fcec059移位Rastrigin的功能30330

fcec0510移位旋转Rastrigin的功能30330

根据表11的结果，DNSPSO在7个功能上比CPSO-H取得更好的结果，而CPSO-H表现更好

比DNSPSO在2个功能上要好。与CLPSO和APSO相比，DNSPSO在7个功能上的性能要好于CLPSO和APSO。

APSO在三个功能上取得了更好的结果。除了fcec058以外，DNSPSO在所有功能上均超过GOPSO。在此功能上

它们都属于同一局部最优。

通过CLPSO与我们提出的策略的混合，DNSCLPSO在7个功能上的性能优于CLPSO。它表明我们的建议有助于提高PSO变体的性能。

表12显示了CPSO-H，CLPSO，APSO，GOPSO，DNSCLPSO和DNSPSO的平均排名。最高排名是

以粗体显示。如图所示，这五个算法的性能可以按平均排名分为以下顺序：

DNSPSO，DNSCLPSO，GOPSO，APSO，CLPSO和CPSO-H。最高的平均排名是通过DNSPSO算法获得的。

它表明DNSPSO是CEC 2005基准测试的六种PSO算法中最好的一种。

表11

CEC 2005基准上的平均功能误差值和标准偏差的比较结果，其中“ w / t / l”表示DNSPSO在w功能中胜出，

与竞争对手相比，它在t函数上具有约束力，而在l功能上则具有损耗。比较中最好的结果以粗体显示。

功能CPSO-H [1] CLPSO [34] APSO [59] GOPSO [53] DNSCLPSO DNSPSO

fcec051 4.26E 02±1.25E + 00 8.01E 13±3.40E 12 7.20E 14±1.38E 13 5.86E 14±6.37E 14 6.53E 14±5.41E 14 4.29E 14 ±3.96E 14

fcec052 1.34E + 03±1.14E + 04 3.38E + 03±3.19E + 03 1.82E 02±9.53E 02 1.26E 04±2.28E 04 2.85E 03±1.52E 03 2.95E 06 ±6.21E 06

fcec053 1.24E + 00±3.64E + 01 4.06E 09±1.27E 08 7.39E 14±1.43E 14 8.31E + 06±6.43E + 06 6.72E 01±2.30E + 00 5.21E + 05 ±4.17E + 05

fcec054 1.07E + 04±7.25E + 04 1.12E + 04±1.57E + 04 9.46E + 02±3.30E + 03 5.48E + 02±4.39E + 02 6.26E + 02±4.91E + 02 3.79E + 00 ±2.45E + 00

fcec055 2.74E + 04±3.12E + 04 1.86E + 04±1.26E + 04 2.21E + 02±3.27E + 02 6.58E + 03±5.16E + 03 8.41E + 03±2.56E + 03 2.98E + 03 ±6.24E + 03

fcec056 1.73E + 03±4.23E + 04 1.04E + 01±5.67E + 01 2.99E + 01±1.68E + 02 1.53E + 01±1.34E + 01 1.95E + 01±2.03E + 01 1.16E + 01 ±8.28E + 00

fcec057 1.52E + 03±1.68E + 03 5.42E + 03±1.43E + 03 5.63E + 03±2.85E + 03 1.20E + 00±1.17E + 00 4.28E + 01±2.55E + 01 2.96E 02 ±3.76E 02

fcec058 2.09E + 01±3.73E 01 2.10E + 01±1.91E 01 2.12E + 01±3.57E 01 2.09E + 01±2.39E 01 2.09E + 01±3.04E 01 2.09E + 01 ±1.83E 01

fcec059 3.55E + 01±5.88E + 01 1.99E 01±2.59E + 00 5.74E + 00±9.09E + 00 8.36E + 01±3.27E + 01 3.36E + 01±1.69E + 01 6.57E + 01 ±2.57E + 01

fcec0510 2.21E + 02±2.87E + 02 1.42E + 02±8.45E + 01 1.33E + 02±3.00E + 02 2.98E + 01±1.43E + 01 1.52E 06±2.13E 07 0.00E + 00 ±0.00E + 00

w / t / l 7/1/2 7/0/3 7/0/3 9/1/0 7/1/2 –

表12

Friedman测试在CEC上对六个PSO算法获得的平均排名

2005年基准。最高排名以粗体显示。

算法排名

DNSPSO 4.85

DNSCLPSO 3.95

高普索3.85

APSO 3.30

CLPSO 2.90

CPSO-H 2.15

表13

实验中使用的CEC 2010大型基准函数，其中D是维度，f（xo

）是全局最优值。

函数名称D f（xo）F1位移椭圆函数1000 0

F2位移Rastrigin的功能1000 0

F3 Shifted Ackley的功能1000 0

F4单组位移和m旋转椭圆函数1000 0

F5单组移动且旋转的Rastrigin函数1000 0

F6单组平移和旋转Ackley函数1000 0

F7单组位移m维Schwefel问题1.2 1000 0

F8单组位移的多维Rosenbrock函数1000 0

F9 D2m-组移位和m旋转椭圆函数1000 0

F10 D2m-Group移位和m旋转Rastrigin的功能1000 0

F11 D2m-Group移位和m旋转的Ackley函数1000 0

F12 D2m-Group位移m维Schwefel问题1.2 1000 0

F13 D2m-Group位移的m维Rosenbrock函数1000 0

F14 Dm-Group移位和m旋转的椭圆函数1000 0

F15 Dm-Group移位和m旋转Rastrigin函数1000 0

F16 Dm-Group移位和m旋转的Ackley函数1000 0

F17 Dm-Group位移的m维Schwefel问题1.2 1000 0

F18 Dm-Group位移的m维Rosenbrock函数1000 0

F19转移了Schwefel的问题1.2 1000 0

F20转移Rosenbrock的功能1000 0

4.6。 CEC 2010大型基准测试的比较结果

在第4.1节中，高维问题（f12-f14）是可分离的。要测试更复杂且不可分割的问题，

本节使用CEC 2010大型基准测试。表13给出了这些功能的简单描述。更多

他们的详细定义可以在[49]中找到。

在实验中，将DNSPSO与CLPSO，GOPSO，DNSCLPSO和CEC 2010 Special比赛的获胜者进行了比较

大规模全球优化会议和竞赛（MA-SW-链）[39]。 MA-SW-Chains是一种模因算法，

它通过链接不同的本地搜索应用程序，为每个个体分配取决于其功能的本地搜索强度。在此，我们不会在此测试套件上将DNSPSO与APSO和CPSO-H进行比较，因为这两种算法较差

比CLPSO和GOPSO

这五种算法的参数设置如下。对于CLPSO，GOPSO，DNSCLPSO和DNSPSO，人口规模设置为100。对于其余参数，使用与第4.4节中所述相同的设置。对于MA-SW-Chains，

与[39]相同的参数设置。根据[49]的建议，最大适应度评估数

所有算法的（MAX\_FEs）设置为3.00E + 06。每个算法针对一个函数运行25次。在整个实验中

报告平均功能值和标准偏差。

表14列出了CEC 2010大型基准测试的平均值函数值和标准偏差的比较结果，其中最佳和次佳结果分别以黑体和斜体突出显示。表格最后一行的DNSPSO与其他算法之间的比较结果汇总为“ w / t / l”，即

与竞争对手相比，DNSPSO在W功能方面胜出，在T功能方面有联系，而在L功能方面则失败。

根据表14的结果，DNSPSO在19个功能上的性能优于CLPSO，而CLPSO在f5上仅能获得更好的结果。

DNSPSO在所有测试功能上均超过GOPSO。通过将CLPSO与我们提出的策略相结合，DNSCLPSO在大多数功能上均取得了重大改进。 DNSCLPSO在七个功能上胜过DNSPSO，而DNSPSO获得了更好的功能

其余13个功能的结果。但是，DNSPSO和其他三个PSO变体不适合大规模优化

表14

CEC 2010大型基准测试的平均值函数值和标准偏差的比较结果，其中“ w / t / l”表示DNSPSO胜出

功能，与竞争对手相比，在功能上是紧密的，在功能上则是失败的。

功能CLPSO [34] GOPSO [53] MA-SW链[39] DNSCLPSO DNSPSO

F1 7.80E + 08±6.79E + 07 1.13E + 10±2.37E + 09 2.10E 14±1.99E 14 1.67E + 08±7.30E + 06 1.87E + 07±1.73E + 06

F2 6.06E + 03±1.40E + 02 9.25E + 03±5.62E + 03 8.10E + 02±5.88E + 01 5.66E + 03±8.29E + 01 5.85E + 03±4.23E + 02

F3 2.04E + 01±8.29E 02 1.97E + 01±7.21E 02 7.28 E 13±3.43E 13 1.91E + 01±7.89E 01 1.93E + 01±4.44E 02

F4 9.98E + 13±9.44E + 12 1.92E + 13±1.04E + 12 3.53E + 11±3.12E + 10 8.99E + 12±1.22E + 12 2.25E + 12±1.96E + 11

F5 1.44E + 08±2.63E + 07 1.62E + 08±1.46E + 08 1.68E + 08±1.04E + 08 8.52E + 07±1.43E + 07 1.57E + 08±2.32E + 07

F6 6.68E + 06±7.99E + 05 1.10E + 07±2.33E + 07 8.14E + 04±2.84E + 05 2.41E + 01±9.95E 01 1.75E + 06±2.75E + 05

F7 9.03E + 09±8.80E + 08 1.41E + 09±2.62E + 09 1.03E + 02±8.70E + 01 6.57E + 08±2.44E + 08 8.60E + 06±2.55E + 05

F8 1.11E + 08±1.40E + 07 9.68E + 07±1.24E + 10 1.41E + 07±3.68E + 07 9.19E + 07±3.50E + 07 1.31E + 07±4.65E + 06

F9 8.96E + 09±1.29E + 08 4.28E + 10±2.73E + 10 1.41E + 07±1.15E + 06 1.57E + 09±6.43E + 07 3.16E + 08±2.49E + 06

F10 1.24E + 04±2.80E + 02 9.69E + 03±1.28E + 04 2.07E + 03±1.44E + 02 7.63E + 03±2.57E + 01 6.90E + 03±1.88E + 02

F11 2.28E + 02±1.03E 01 2.01E + 02±1.35E + 02 3.80E + 01±7.35E + 00 1.68E + 02±9.13E + 00 1.76E + 02±5.74E + 00

F12 7.26E + 06±1.08E + 06 5.91E + 06±2.61E + 06 3.62E 06±5.92E 07 1.00E + 06±5.83E + 03 2.34E + 05±1.81E + 04

F13 6.73E + 08±1.55E + 08 5.23E + 09±3.28E + 09 1.25E + 03±5.72E + 02 2.20E + 08±3.41E + 07 1.15E + 06±1.81E + 05

F14 2.09E + 10±7.32E + 08 5.48E + 10±4.16E + 10 3.11E + 07±1.93E + 06 3.15E + 09±8.62E + 07 3.81E + 09±7.15E + 07

F15 1.47E + 04±2.49E + 02 1.24E + 04±9.40E + 03 2.74E + 03±1.22E + 02 8.26E + 03±8.78E + 01 7.52E + 03±4.34E + 01

F16 4.24E + 02±2.85E 01 3.97E + 02±2.58E + 02 9.98E + 01±1.40E + 01 4.04E + 02±9.27E 01 3.86E + 02±4.15E 01

F17 1.15E + 07±1.04E + 06 1.05E + 07±8.18E + 06 1.24E + 00±1.25E 01 1.56E + 06±2.41E + 04 6.13E + 05±6.03E + 03

F18 5.69E + 10±8.51E + 09 5.46E + 11±7.19E + 10 1.30E + 03±4.36E + 02 2.81E + 10±3.05E + 09 2.29E + 07±9.39E + 06

F19 3.82E + 07±8.43E + 06 8.78E + 06±5.42E + 06 2.85E + 05±1.78E + 04 5.05E + 06±7.09E + 04 5.33E + 06±1.66E + 06

F20 8.44E + 10±2.18E + 09 2.22E + 11±3.15E + 10 1.07E + 03±7.29E + 01 4.23E + 10±3.61E + 09 2.02E + 08±1.92E + 08

w / t / l 19/0/1 20/0/0 2/0/18 13/0/7 –

表15

Friedman测试获得的CEC 2010大型基准测试的平均排名。最高排名是

以粗体显示。

算法排名

MA-SW-链4.70

DNSPSO 3.70

DNSCLPSO 3.40

高普索1.65

CLPSO 1.55

问题。与CEC 2010大规模全球优化（MA-SW链）特别会议的竞赛获胜者相比[39]，DNSPSO在两个功能上的表现均优于MA-SW链。

表15显示了CLPSO，GOPSO，MA-SW-Chains，DNSCLPSO和DNSPSO的平均排名。最高排名是

以粗体显示。如图所示，这五个算法的性能可以按平均排名分为以下顺序：

MA-SW链，DNSPSO，DNSCLPSO，GOPSO和CLPSO。最佳算法和次佳算法是MA-SW-Chains和

DNSPSO。

5。结论

本文提出了一种称为DNSPSO的增强型PSO算法，用于解决复杂的优化问题。建议

该方法采用两种策略，包括多样性增强机制和两种邻域搜索策略。

前一种策略有助于通过调整粒子之间的差异来增加群体多样性。后者

由于先前的最佳粒子的吸引力和最佳粒子的吸引，该策略有利于加快收敛速度​​。

全球最佳粒子。通过结合这两种策略，DNSPSO可以在探索和开发能力之间进行权衡。为了验证DNSPSO的性能，在实验中测试了不同类型的基准功能。

参数pr和pns的值会影响DNSPSO的性能。仿真结果表明固定的pr和pns

并非适合所有功能。较小的pr适用于低维函数，而较大的pr适用于高维函数。 pns的值不会严重影响DNSPSO的性能，并且pns 2（0,1）可以达到

好的结果。

基于多样性增强机制（DPSO）或基于邻域搜索（NSPSO）的PSO无法获得有希望的结果。通过将它们组合在一起，DNSPSO获得了出色的性能。平均计算时间的结果表明

与标准PSO相比，在相同的最大适应度评估数下，该算法不会增加计算时间。

对于包括两种基准的低维问题，DNSPSO的效果要优于CPSO-H，

具有RVM的CLPSO，APSO，GOPSO和PSO。通过将我们提出的策略整合到CLPSO中，DNSCLPSO可以显着获得

改进。对于高维问题，尽管DNSPSO优于CLPSO，GOPSO和DNSCLPSO，但所有这些PSO

对于CEC 2010大型基准测试而言，变体并不是不错的选择。

参数k（邻域半径）可能会影响邻域搜索的有效性。如何调整k

在以后的工作中进行调查。

致谢

作者感谢编辑和匿名审稿人的详尽和建设性的意见，这些意见有助于我们提高这项工作的质量。这项工作得到了江西省教育厅科技计划计划项目（No. GJJ12641，GJJ12633，GJJ12307），南昌工程学院青年基金会（No. 1907-1886）的支持。

2012KJ021）和国家自然科学基金（No. 61070008，61165004，61261039）

1：1.5：1.5：3：8：2