

一种融入差分变异的变规模和声搜索算法

赵新超, 刘朝华

(北京邮电大学理学院, 北京 100876)

[摘要] 为了充分利用记忆库内保存的有益历史信息, 对解向量的微调进行差分变异操作的差分改进, 提出了一种融入差分变异操作的变规模和声搜索(linearly decreasing harmony search, LDHS)算法, 加强了算法的新路径探索性能。同时, 为了平衡记忆库的多样性和收敛性, 对记忆库的大小采取线性调整, 并研究了记忆库大小对 LDHS 算法性能的影响。最后, 将 LDHS 算法、基本的和声搜索(harmony search, HS)算法、3种改进的 HS 算法在 CEC 2014 的 8 个测试函数上分别进行不同维度独立运行 30 次实验, 对比结果表明, LDHS 算法能够更快地找到全局最优解, 并具有一定的稳定性。

[关键词] 和声搜索算法; 变规模; 差分变异; 历史信息

[中图分类号] TP 301.6

A Harmony Search Algorithm with Differential Mutation and Varying Population Size

ZHAO Xinchao, LIU Zhaohua

(School of Science, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: In order to make full use of the information of the harmony memory, a differential mutation operation was proposed to provide an improved direction for the fine tuning of the solution. A harmony search algorithm was also proposed with linearly decreasing population size-LDHS, which enhanced the exploration capability for the novel routes of the algorithm. At the same time, in order to balance the diversity and convergence of harmony memory, a linearly decreasing strategy was adopted to the population size of the harmony memory, and its effect was considered on the performance of the algorithm. Finally, the performance of the proposed algorithm was verified with the basic HS and other three improved HS variants on the 8 CEC 2014 benchmark functions in 30 independent runs on different dimensions. The experimental results showed that LDHS algorithm could find the even better solutions more efficiently, and also had better steadiness.

Keywords: harmony search algorithm; varying population size; differential mutation; historical information

0 引言

和声搜索(harmony search, HS)算法于2001年由韩国学者 Geem 等^[1]提出, 它模拟了和声音调微调的原理。该算法原理和结构简单, 参数少, 求解速度快, 并且具有鲁棒性和通用性, HS 算法已

[收稿日期] 2017-02-21

[修回日期] 2017-03-18

[基金项目] 国家自然科学基金项目(61375066, 61374204)

[作者简介] 赵新超(1976—), 男, 教授, 博士, 博导, 从事群体智能、运筹学及其应用研究, E-mail: zhaoxc@bupt.edu.cn。

被证明对于更复杂的优化问题具有很好的竞争力。而且,该算法已在各种优化问题中得到了广泛应用,如多维多极值函数优化^[2]、工程设计^[3]、管道优化设计^[4]、结构优化^[5]、车辆线路^[6]、联合供热^[7]、电力节约调度^[8]、多坝系统调度^[9]、供水管网优化设计^[10]、无线传感器网络应用^[11]、结构框架的优化^[12]等。文献[13]在研究HS算法的基础上,提出一种改进的和声搜索(improved harmony search, IHS)算法,它将参数PAR设计成随着迭代次数的增加而线性递增,参数bw随着迭代次数的增加呈指数递减。文献[14]提出了一种自适应全局最优和声搜索(self-adaptive global best harmony search, SGHS)算法,该算法使用了一种产生新解的方式和一种自适应的参数调整策略。文献[15]把HS算法与粒子群优化(particle swarm optimization, PSO)算法相结合,提出了一种新颖的全局优化和声搜索(intelligent global harmony search, IGHS)算法。

为了提高算法性能,本文提出了一种融入差分变异操作的变规模和声搜索(linearly decreasing harmony search, LDHS)算法,充分利用算法在搜索过程中的有益历史信息,在新算法中结合差分变异操作,同时引入和声记忆库多样性作为指导信息,对算法中记忆库的大小进行动态更新,不同程度提高了算法全局搜索和局部搜索的能力。同时,为了验证本文所提策略和算法的性能,将该算法与基本的HS算法、3种经典的HS算法变种(IHS, SGHS, IGHS)在CEC 2014的测试函数上进行对比仿真实验,验证了所提出策略和算法性能的有效性和竞争力。

1 基本 HS 算法

2001年,文献[1]提出了一种新颖的元启发式智能优化算法,该算法模拟了音乐创作中乐师们凭借自己的记忆,通过反复调整乐队中各乐器的音调,最终达到一个美妙的和声状态的过程。HS算法将乐器声调的和声类比为优化问题的解向量,对和声优劣的评价对应于优化问题的目标函数值。基本HS算法的操作步骤如下:

输入:

- HMS: 和声记忆库大小,
- HMCR: 和声记忆库选择概率,
- PAR: 和声记忆库微调概率,
- bw: 音调微调步长,
- MAX_NFE: 最大函数值计算次数,
- L_i, U_i : 第*i*维向量的上界和下界。

1: 随机初始化生成HMS个和声 x^1, x^2, \dots, x^{HMS} , $x_i^j = L_i + \text{rand}(0,1) \cdot (U_i - L_i)$, $i = 1, 2, \dots, N; j = 1, 2, \dots, HMS$ 。

2: 循环。

```
for each  $i \in [1, N]$  do
  if ( $\text{rand}(0,1) < \text{HMCR}$ ) then
     $x'_i = x_i^a$ , where  $a \in \{1, 2, \dots, \text{HMS}\}$ 
    if ( $\text{rand}(0,1) < \text{PAR}$ ) then
       $x'_i = x'_i \pm \text{rand}(0,1) \times \text{bw}$ 
    end if
  else
     $x'_i = L_i + \text{rand}(0,1) \cdot (U_i - L_i)$ ,
    ( $\text{rand}(0,1)$ 是(0,1)上均匀分布的随机数)
  end if
end for。
```

- 3: 对新解进行评估, 比较 $f(x')$ 和 $f(x^{\text{worst}})$, 并把其中较好的一个存入到和声记忆库, 丢弃较差解,
 $\text{if } f(x') < f(x^{\text{worst}}), \text{ then } x^{\text{worst}} = x', f(x^{\text{worst}}) = f(x')$ 。
- 4: 如果达到终止条件, 则停止, 并输出 HM 中最好的个体, 否则回到第 2 步。

2 LDHS 算法

2.1 引入差分变异操作的步长微调策略

HS 算法受启发于音乐家的音乐创作, 考虑到音乐家在对已有的旋律进行调整时, 会充分利用自己已有的知识和经验进行微调, 这样能明显加快音乐创作的进程, 提高音乐创作的质量。受此启发, 本文对产生新解过程中的步长微调步骤进行改进, 使其结合差分算法中的 DE/best/2/bin 的变异形式, 以便更好地利用有益的、多样化的历史经验, 给解的微调增加一个新的变异方向, 这相当于音乐家利用自身具备的知识与经验给出一些有益的调整方向。这种改进策略既利用了历史有益的信息, 又没有照搬历史经验, 能有效地促进寻优速度和优化结果。其具体形式如式 (1) 所示:

$$x_{\text{new}}(i) = x_{\text{best}}(i) + F[(x_{r_1}(i) - x_{r_2}(i)) + (x_{r_3}(i) - x_{r_4}(i))] \pm \text{rand}(0,1) \times \text{bw}, \quad (1)$$

其中: $i \in \{1, 2, 3, \dots, \text{DIM}\}$, DIM 是问题的维度; x_{new} 是新生成的一个中间解; $r_1, r_2, r_3, r_4 \in \{1, 2, 3, \dots, \text{HMS}\}$ 是互不相同的 4 个随机整数; $\text{rand}(0,1)$ 是 0 和 1 之间的随机数; x_{best} 是目前记忆库中适应度值最优的一个解; F 是缩放因子, 用来控制差分向量的缩放规模。

如果生成的新解超出了边界 $[L_i, U_i]$, 按照式 (2) 进行处理,

$$x_{\text{new}}(i) = \begin{cases} L_i, & \text{if } x_{\text{new}}(i) < L_i, \\ U_i, & \text{if } x_{\text{new}}(i) > U_i. \end{cases} \quad (2)$$

2.2 变规模策略

传统的 HS 算法通常设置 HMS 为固定值, 但是众所周知, HMS 的大小对于最后的优化结果有着重要的影响。当 HMS 较大时, 算法收敛速度较慢, 从而影响算法的寻优速度; 当 HMS 较小时, 又会导致算法容易陷入局部最优, 降低了算法找到全局最优解的可能性。基于这个思想, 本文对 HMS 进行调整, 即在算法初期需要大范围探测搜索时, 提供较大的群体规模; 随着算法寻优过程的进行, HMS 随着迭代次数的增加而线性减小, 到算法后期到达一个较小的群体规模, 有利于最后阶段搜索引擎对最优解的准确定位。对 HMS 的调整公式如下:

$$\text{HMS} = \text{round}(\text{HMS}_{\max} - (\text{HMS}_{\max} - \text{HMS}_{\min}) \times \text{NFE}/\text{MAX_NFE}), \quad (3)$$

其中: HMS_{\max} 是 HMS 的最大值, 最大值的设置与待解决的问题的维度 DIM 有关, 设置为 $\text{rate} \times \text{DIM}$, rate 是一个常数; HMS_{\min} 是 HMS 的最小值; MAX_NFE 是最大函数值计算次数; NFE 是当前函数值计算次数。

2.3 算法流程

2.3.1 随机初始化记忆库, 设置参数。

2.3.2 进行循环迭代, 直到达到终止条件 1) 根据迭代次数更新本次迭代的 HMS, 若记忆库减小, 则对记忆库中的解向量根据适应度的大小进行排序, 删除最差解; 2) 产生新解, 并对新解进行微调, 具体流程如下:

```

for each  $i \in [1, \text{DIM}]$ 
    if( $\text{rand}(0,1) < \text{HMCR}$ )
         $x_{\text{new}}(i) = x_{\text{best}}(i)$ 
        if( $\text{rand}(0,1) < \text{PAR}$ ) % ( $r_1, r_2, r_3, r_4 \in \{1, 2, 3, \dots, \text{HMS}\}$ , 且互不相同)
             $x_{\text{new}}(i) = x_{\text{new}}(i) + F[(x_{r_1}(i) - x_{r_2}(i)) + (x_{r_3}(i) - x_{r_4}(i))] \pm \text{rand}(0,1) \times \text{BW}$ 
        end if
    else

```

$$x_{\text{new}}(i) = L_i + \text{rand}(0,1) \cdot (U_i - L_i)$$

end if
end for。

2.3.3 更新记忆库 判断生成的新解是否比记忆库中的最差解要好，如果优于最差解，则代替库中的最差解。

3 仿真实验

本文对 LDHS 算法、基本 HS 算法以及 3 种改进的 HS 算法的性能进行实验对比。对于其他 4 种 HS 算法的参数设置均遵从原文献的建议，如表 1 所示。

表 1 参数设置
Tab. 1 Parameters set

算法 Algorithm	参数设置 Parameters set
HS	HMS = 5, HMCR = 0.9, PAR = 0.3, bw = 0.01
IHS	HMS = 5, HMCR = 0.9, PAR _{min} = 0.01, PAR _{max} = 0.99, bw _{min} = 0.0001, bw _{max} = (x ^U - x ^L)/20
SGHS	HMS = 5, HMCR _m = 0.98, PAR _m = 0.9, Lp = 100, bw _{min} = 0.0005, bw _{max} = (x ^U - x ^L)/10
IGHS	HMS = 5, HMCR = 0.995, PAR = 0.4
LDHS	HMS _{min} = 5, HMS _{max} = 5 × DIM, HMCR = 0.99, PAR = 0.7, F = 0.6, bw = 0.01

本文采用的测试函数是 CEC 2014 基准测试函数。对于所有的问题，搜索空间在[-100,100]，函数 f1~f3 是单峰函数；f4~f16 是简单的多峰函数；f17~f22 是混合函数；f23~f30 是复合函数，即结合的多个测试问题转化为一个复杂的问题。本文在 CEC 2014 4 类基准测试函数中各选取 2 个，分别是 f2、f3、f6、f7、f20、f21、f27、f28，为方便记录，在本文中重新被编号为 f1~f8。对于所有的问题的维数 DIM = 10, 30, 50，函数值计算次数分别为 DIM * 10 000，即：100 000，300 000，500 000，每个算法对每个函数独立运行 30 次，记录 30 次最终结果的平均值和标准差作为算法的对比依据。

4 仿真结果与分析

4.1 记忆库大小对算法的影响

为了研究HMS_{max} = rate * DIM 对算法性能的影响，本文在 DIM = 10 维上分别取 rate 为 1,3,5,7，对 12 个函数进行测试，结果表 2 所示。

从表 2 中可以看出，当 rate 为 1 和 3 时，算法表现较差，当 rate 增加到 5 时，8 个函数中有 6 个函数得到较好的结果。随着 rate 的增长，当 rate 为 7 时，此时参数取值对应的结果在 f3 和 f5 上结果较优。从表 2 中还可以看出，此时 HMS 对算法性能的影响不明显，所以在本文中取 rate = 5。这种变化的趋势与实际是相符合的，即 HMS 相当于音乐家经验积累的丰富程度，在经验积累较少时，随着经验积累的增加，会对音乐创作产生较大的影响，但是当经验积累达到一定程度时，经验再增加对于音乐创作效果优化的影响就不明显了。

4.2 LDHS 与 HS、IHS、SGHS、IGHS 对比分析

5 种算法分别在 10, 30, 50 维进行仿真实验，其中 30 维的数值仿真结果与 50 维的结果有相同的趋势。为了节省空间，本文给出 DIM = 10, 50 的测试结果，如表 3 和表 4 所示，表中“Best”、“Mean”、“Worst”和“SD”分别代表每一种算法对每一个函数独立运行 30 次所产生解的最优值、平均值、最差值和标准差。表中最后一行“Ranking”对应的“+ / ~ / -”分别表示 LDHS 算法与对应算法相比有优势、表现相当、有劣势的函数个数。

表 2 rate 变化对算法性能的影响 (DIM = 10)
Tab.2 Influence of rate change on performance (DIM = 10)

函数 Function	项目 Item	rate = 1	rate = 3	rate = 5	rate = 7
<i>f</i> 1	Mean	5.36E + 03	1.03E + 03	7.67E + 00	8.48E + 00
	SD	3.94E + 03	3.11E + 03	2.20E + 00	3.75E + 00
<i>f</i> 2	Mean	1.27E + 02	1.83E − 04	1.61E − 04	2.07E − 04
	SD	3.28E + 02	1.36E − 04	1.61E − 04	1.83E − 04
<i>f</i> 3	Mean	5.47E + 00	2.62E + 00	2.39E + 00	1.86E + 00
	SD	1.87E + 00	1.73E + 00	1.46E + 00	1.80E + 00
<i>f</i> 4	Mean	6.95E − 01	8.95E − 02	7.50E − 02	8.59E − 02
	SD	3.40E − 01	6.04E − 02	4.29E − 02	4.80E − 02
<i>f</i> 5	Mean	4.03E + 03	4.15E + 00	2.72E + 00	1.90E + 00
	SD	7.15E + 03	2.13E + 00	1.60E + 00	1.56E + 00
<i>f</i> 6	Mean	1.22E + 03	4.43E + 01	4.40E + 01	6.08E + 01
	SD	2.20E + 03	5.17E + 01	6.50E + 01	7.15E + 01
<i>f</i> 7	Mean	4.36E + 02	3.48E + 02	2.86E + 02	3.15E + 02
	SD	9.79E + 01	1.01E + 02	1.36E + 02	1.15E + 02
<i>f</i> 8	Mean	5.75E + 02	4.45E + 02	4.25E + 02	4.35E + 02
	SD	1.52E + 02	8.10E + 01	6.35E + 01	6.92E + 01

从表 3 和表 4 中可以看出，在 DIM = 10，50 时，LDHS 算法明显优于其他 4 种对比算法。首先，在这 2 个维度上，LDHS 算法与传统 HS 算法、3 种 HS 变种算法相比较，LDHS 算法单峰函数 *f*1、*f*2 上性能表现最好；其次，在多峰函数 *f*3、*f*4 上，LDHS 算法随着维度的增加，性能表现越好；再次，LDHS 算法相比于其他 4 种 HS 算法在混合函数 *f*5、*f*6 上性能表现最好，这说明，随着问题规模增加和问题难度的增加，LDHS 算法具有越来越明显的性能优势；最后，对于复合函数 *f*7、*f*8 问题，在 DIM = 10 和 50 的 8 种情形里，只有函数 *f*7 在 10 维的结果略差于 IGHS 算法，说明 LDHS 算法在求解较为复杂的优化函数时具有最好的性能。众所周知，现实生活中的问题都是较为复杂多变的，所以 LDHS 算法对解决现实生活中的问题具有一定的优势。综上所述，在 CEC 2014 基准函数上，LDHS 算法在 DIM ∈ {10,50} 8 个测试函数各 16 种的两两对比中，分别在 16，16，16，15 个情形中优于 HS、HIS、SGHS 和 IGHS 算法，表明该算法具有明显的最优整体性能和稳定性。

4.3 LDHS 算法性能图例分析

5 种 HS 算法运用在 CEC 2014 的 8 个函数上，当 DIM = 10，50 维时，运行 30 次产生迭代最优适应值的平均值，其统计结果见图 1 和图 2。

对于两个单峰函数 *f*1 和 *f*2，可以看出 LDHS 算法在 10，50 维的性能表现具有很好的竞争优势。在 DIM 为 50 维时，从图 2 中可以看出，LDHS 算法相比于其他 4 种算法在最后阶段找到最优解，具有跳出局部最优的能力。对于函数 *f*7，虽然在 DIM 为 10 维时，LDHS 算法的表现并不是最优，但是当维数达到 50 维时，LDHS 算法的性能表现最优。同时，从图 2 中还可以看出，对于 *f*5、*f*6 这 2 个混合函数，LDHS 算法在最后阶段跳出局部最优，找到最优解。所以 LDHS 算法具有更好的跳出局部最优的能力，以及在解决高维度问题时有更好的表现。从图 1 和图 2 中还可以看出，LDHS 算法在解决 *f*5、*f*6 这 2 个混合函数时，在 10 维和 50 维性能效果比其他 4 种算法要更优。

结合表 3、表 4、图 1、图 2 可以看出，LDHS 算法的性能在 CEC 2014 测试函数中的单峰函数、多峰函数、混合函数和复合函数上都表现出很好的竞争力，因此，LDHS 算法在解决复杂优化问题时具有一定的应用潜力。

表 3 LDHS 与 4 种 HS 算法在 CEC 2014 基准函数上的性能比较 (DIM = 10)

Tab.3 Performance comparison among LDHS and four HS algorithms on CEC 2014 (DIM = 10)

函数 Function	项目 Item	HS	IHS	SGHS	IGHS	LDHS
f1	Best	1.18E+02	4.51E-02	3.08E+00	1.48E+06	4.90E+00
	Mean	3.94E+03	3.49E+03	3.92E+03	2.37E+07	9.17E+00
	SD	3.61E+03	3.54E+03	4.04E+03	1.87E+07	2.51E+00
	Worst	1.18E+04	1.14E+04	1.08E+04	7.04E+07	1.57E+01
f2	Best	3.86E+01	6.76E+00	2.62E+01	8.53E+01	2.78E-05
	Mean	4.63E+03	3.85E+03	7.39E+03	4.52E+03	1.42E-04
	SD	3.58E+03	5.04E+03	7.48E+03	4.85E+03	1.09E-04
	Worst	1.19E+04	2.30E+04	2.73E+04	2.09E+04	4.56E-04
f3	Best	8.37E-01	8.97E-01	1.35E+00	2.06E+00	1.47E-02
	Mean	3.36E+00	3.08E+00	4.29E+00	4.07E+00	1.96E+00
	SD	1.21E+00	1.18E+00	1.50E+00	1.41E+00	1.87E+00
	Worst	5.75E+00	5.87E+00	7.64E+00	7.73E+00	6.32E+00
f4	Best	4.93E-02	2.22E-02	6.16E-02	8.68E-01	2.22E-02
	Mean	2.61E-01	1.63E-01	1.58E-01	1.40E+00	9.80E-02
	SD	1.73E-01	8.79E-02	1.11E-01	4.68E-01	5.29E-02
	Worst	6.69E-01	4.01E-01	5.76E-01	2.79E+00	2.17E-01
f5	Best	5.31E+01	3.85E+00	7.37E+00	7.20E+00	4.10E-01
	Mean	4.85E+03	7.26E+03	8.74E+03	5.67E+03	2.37E+00
	SD	6.31E+03	9.95E+03	1.08E+04	6.14E+03	1.83E+00
	Worst	2.18E+04	2.94E+04	3.06E+04	2.15E+04	8.93E+00
f6	Best	8.58E+00	1.98E+00	2.50E+02	7.81E+01	1.21E-01
	Mean	4.52E+03	3.86E+03	4.53E+05	1.69E+05	9.05E+01
	SD	6.43E+03	5.39E+03	5.74E+05	4.58E+05	7.35E+01
	Worst	2.18E+04	2.32E+04	2.06E+06	2.50E+06	1.80E+02
f7	Best	3.55E+02	3.35E+02	1.66E+00	3.50E+00	7.88E-01
	Mean	4.00E+02	3.99E+02	4.04E+02	1.81E+02	3.09E+02
	SD	2.22E+01	3.40E+01	8.66E+01	2.00E+02	1.13E+02
	Worst	4.48E+02	4.86E+02	5.29E+02	4.45E+02	4.12E+02
f8	Best	3.60E+02	3.76E+02	4.25E+02	3.80E+02	3.69E+02
	Mean	4.77E+02	5.00E+02	5.96E+02	5.08E+02	4.42E+02
	SD	8.05E+01	9.77E+01	1.19E+02	9.60E+01	7.31E+01
	Worst	6.80E+02	8.47E+02	8.74E+02	7.18E+02	6.26E+02
Ranking	+ / ~ / -	8/0/0	8/0/0	8/0/0	7/0/1	—

表 4 LDHS 与 4 种 HS 算法在 CEC 2014 基准函数上的性能比较 (DIM = 50)

Tab.4 Performance comparison among LDHS and four HS algorithms on CEC 2014 (DIM = 50)

函数 Function	项目 Item	HS	IHS	SGHS	IGHs	LDHS
<i>f</i> 2	Best	7.35E + 06	2.74E + 02	1.79E + 00	2.56E + 10	1.91E + 03
	Mean	1.54E + 07	4.67E + 04	4.09E + 04	3.55E + 10	9.84E + 03
	SD	6.14E + 06	1.55E + 05	8.96E + 04	5.08E + 09	8.33E + 03
	Worst	3.05E + 07	8.55E + 05	4.01E + 05	4.60E + 10	2.66E + 04
<i>f</i> 3	Best	3.14E + 03	3.26E + 03	2.96E + 03	6.01E + 04	1.01E + 02
	Mean	1.48E + 04	1.20E + 04	2.87E + 04	7.10E + 04	1.98E + 02
	SD	7.25E + 03	7.07E + 03	1.29E + 04	6.69E + 03	6.26E + 01
	Worst	2.97E + 04	2.87E + 04	5.53E + 04	8.71E + 04	3.35E + 02
<i>f</i> 6	Best	2.21E + 01	1.95E + 01	2.73E + 01	5.32E + 01	6.72E + 00
	Mean	2.90E + 01	2.69E + 01	3.33E + 01	5.82E + 01	1.44E + 01
	SD	3.65E + 00	4.05E + 00	3.57E + 00	2.72E + 00	3.59E + 00
	Worst	3.63E + 01	3.60E + 01	4.23E + 01	6.54E + 01	2.08E + 01
<i>f</i> 7	Best	1.06E + 00	6.31E − 02	5.69E − 08	2.17E + 02	2.20E − 03
	Mean	1.16E + 00	1.33E − 01	1.26E − 01	3.12E + 02	4.09E − 03
	SD	7.00E − 02	5.40E − 02	1.86E − 01	4.26E + 01	3.04E − 03
	Worst	1.40E + 00	3.27E − 01	6.39E − 01	3.78E + 02	1.55E − 02
<i>f</i> 20	Best	2.50E + 03	9.62E + 02	2.02E + 04	9.20E + 03	1.34E + 02
	Mean	8.54E + 03	7.66E + 03	6.59E + 04	1.79E + 04	2.31E + 02
	SD	4.55E + 03	4.01E + 03	3.09E + 04	5.07E + 03	7.13E + 01
	Worst	2.02E + 04	2.10E + 04	1.45E + 05	3.30E + 04	4.05E + 02
<i>f</i> 21	Best	8.18E + 05	2.13E + 04	5.30E + 05	5.55E + 05	9.17E + 03
	Mean	3.69E + 06	3.54E + 05	2.49E + 06	4.00E + 06	9.76E + 04
	SD	2.15E + 06	2.02E + 05	2.42E + 06	2.16E + 06	6.92E + 04
	Worst	7.78E + 06	8.42E + 05	1.22E + 07	7.68E + 06	2.83E + 05
<i>f</i> 27	Best	9.18E + 02	4.49E + 02	1.04E + 03	1.77E + 03	4.12E + 02
	Mean	1.13E + 03	1.05E + 03	1.29E + 03	1.95E + 03	6.19E + 02
	SD	1.06E + 02	1.40E + 02	1.22E + 02	6.68E + 01	9.26E + 01
	Worst	1.37E + 03	1.23E + 03	1.56E + 03	2.10E + 03	8.10E + 02
<i>f</i> 28	Best	1.31E + 03	1.30E + 03	1.81E + 03	2.73E + 03	9.92E + 02
	Mean	1.73E + 03	1.66E + 03	3.53E + 03	3.93E + 03	1.12E + 03
	SD	4.73E + 02	2.38E + 02	1.19E + 03	9.45E + 02	4.86E + 01
	Worst	3.46E + 03	2.34E + 03	5.98E + 03	6.32E + 03	1.27E + 03
Ranking	+ / ~ / −	8/0/0	8/0/0	8/0/0	8/0/0	—

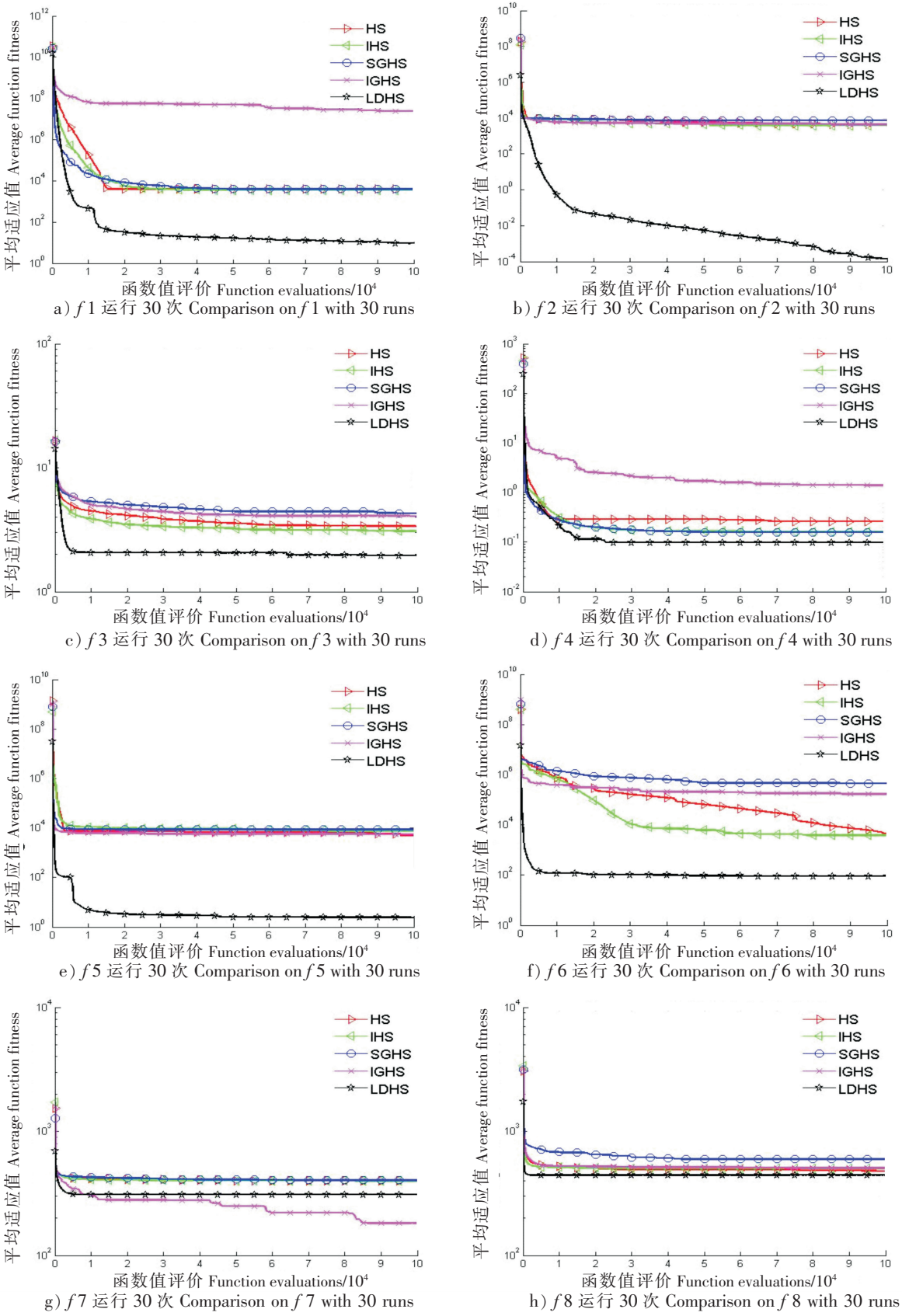


图 1 LDHS 和 4 种对比 HS 算法在 10 维上的性能比较

Fig.1 Performance comparison among LDHS and four competitors in 10 dimension

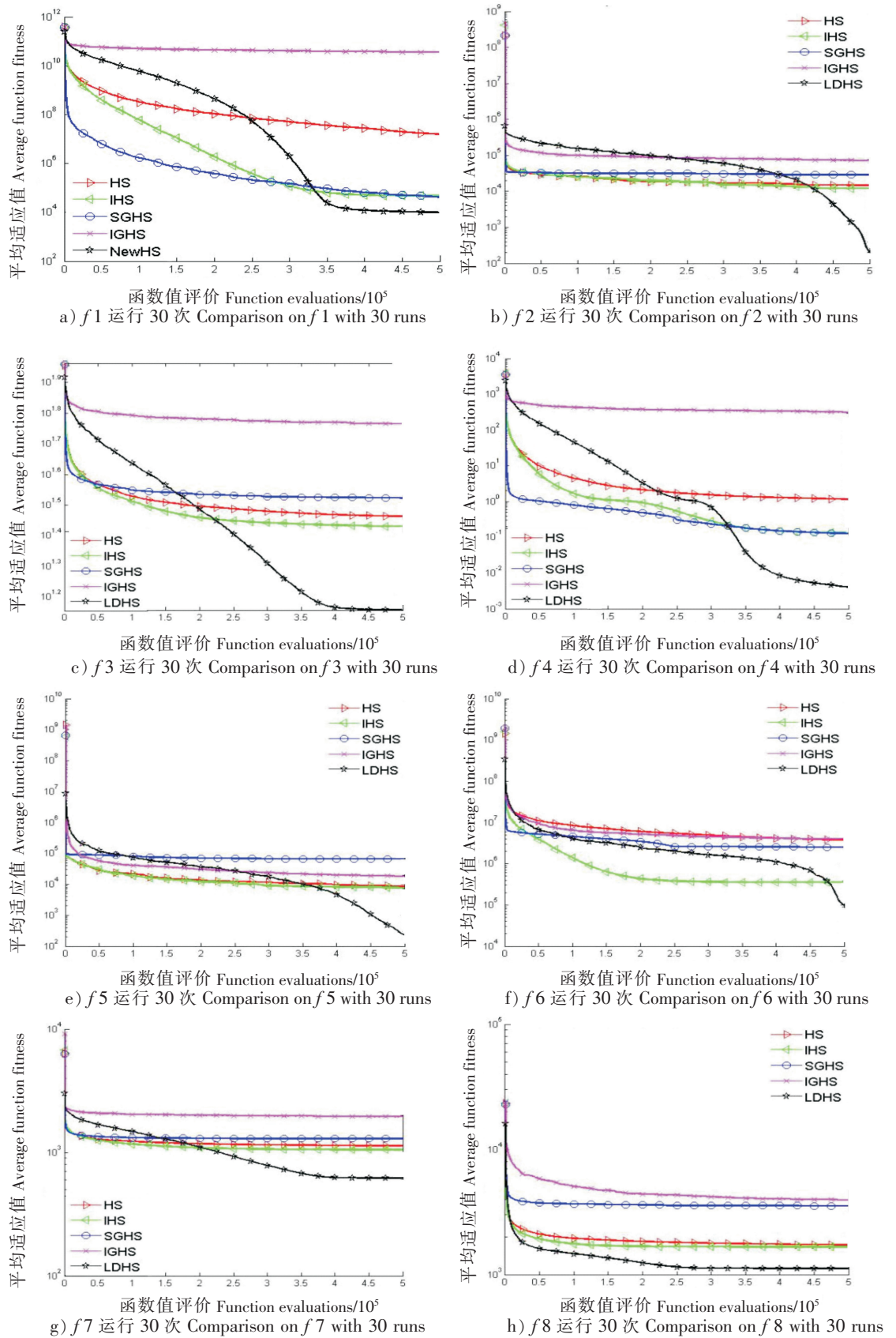


图 2 LDHS 和 4 种对比 HS 算法在 50 维上的性能比较

Fig.2 Performance comparison among LDHS and four competitors in 50 dimension

5 结论

本文结合音乐家利用已有的历史经验和知识对乐曲创作影响的情形,对基本 HS 算法搜索性能进行了研究,提出了一种 LDHS 算法,引入对和声记忆库的大小进行线性减小和利用差分变异操作机制的两种改进策略。结合记忆库中已有的历史信息,考虑到音乐创作的前期需要音乐家更多的知识与经验来提供优化方向,而后期的调整更加依赖于随机的微调。受此启发,本文引入了记忆库的大小线性递减的机制。同时,通过引入差分向量的多样性变异机制对算法的微调特性进行调整。算法对 CEC 2014 测试平台中的 8 个函数进行测试,结果表明,LDHS 算法在 8 个函数上表现比传统 HS 算法、3 种改进的 HS 算法性能更佳,尤其对混合函数更有竞争力,说明 LDHS 算法具有较好的解决实际优化问题的应用潜力。

[参考文献]

- [1] GEEM Z W, KIM J H, LOGANATHAN G V. A new heuristic optimization algorithm: harmony search [J]. Simulation, 2001, 76(2): 60-68.
- [2] 田永红,薄亚明,高美凤. 多维多极值函数优化的和声退火算法 [J]. 计算机仿真, 2004, 21(10): 79-82.
- [3] 雍龙泉. 和声搜索算法研究进展 [J]. 计算机系统应用, 2011, 20(7): 244-248.
- [4] GEEM Z W, KIM J H, LOGANATHAN G V. Harmony search optimization: application to pipe network design [J]. International Journal of Modelling and Simulation, 2002, 22(2): 125-133.
- [5] 邹德旋,高立群,吴建华,等. 混合差分进化和声搜索算法在结构工程中的应用 [J]. 东北大学学报(自然科学版), 2010, 31(6): 769-772.
- [6] 王英博,王琳,李扬,等. 改进的遗传和声算法及其在车辆路径中的应用 [J]. 计算机测量与控制, 2011, 19(12): 3068-3071.
- [7] VASEBI A, FESANGHARY M, BATHAEE S M T. Combined heat and power economic dispatch by harmony search algorithm [J]. International Journal of Electrical Power & Energy Systems, 2007, 29(10): 713-719. DOI:10.1016/j.ijepes.2007.06.006.
- [8] GEEM Z W. Optimal scheduling of multiple dam system using harmony search algorithm [C] //International Work-Conference on Artificial Neural Networks. Berlin: Springer Heidelberg, 2007: 316-323.
- [9] GEEM Z W. Harmony search optimisation to the pump-included water distribution network design [J]. Civil Engineering and Environmental Systems, 2009, 26(3): 211-221. DOI:10.1080/10286600801919813.
- [10] GEEM Z W. Particle-swarm harmony search for water network design [J]. Engineering Optimization, 2009, 41(4): 297-311. DOI:10.1080/03052150802449227.
- [11] NEZHAD S E, KAMALI H J, MOGHADDAM M E. Solving K-coverage problem in wireless sensor networks using improved harmony search [C] //IEEE. 2010 International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA). Fukuoka: IEEE, 2010: 49-55.
- [12] DEGERTEKIN S O. Improved harmony search algorithms for sizing optimization of truss structures [J]. Computers & Structures, 2012, 92: 229-241. DOI:10.1016/j.compstruc.2011.10.022.
- [13] MAHDAVI M, FESANGHARY M, DAMANGIR E. An improved harmony search algorithm for solving optimization problems [J]. Applied Mathematics and Computation, 2007, 188(2): 1567-1579. DOI:10.1016/j.amc.2006.11.033.
- [14] PAN Q K, SUGANTHAN P N, TASGETIREN M F, et al. A self-adaptive global best harmony search algorithm for continuous optimization problems [J]. Applied Mathematics and Computation, 2010, 216(3): 830-848. DOI:10.1016/j.amc.2010.01.088.
- [15] VALIAN E, TAVAKOLI S, MOHANNA S. An intelligent global harmony search approach to continuous optimization problems [J]. Applied Mathematics and Computation, 2014, 232: 670-684. DOI:10.1016/j.amc.2014.01.086.

(责任编辑 马建华 英文审校 黄振坤)