Artificial Intelligence and Robotics Research 人工智能与机器人研究, 2015, 4(4), 32-39
Published Online November 2015 in Hans. http://www.hanspub.org/journal/airr
http://dx.doi.org/10.12677/airr.2015.44005

Hans汉斯

The Improvement of Harmony Search Algorithm

Kangli Zhang, Shouyuan Chen, Zengzhen Shao

School of Information Science and Engineering, Shandong Normal University, Jinan Shandong
Email: shouyuanchen@163.com

Received: Oct. 24th, 2015; accepted: Nov. 7th, 2015; published: Nov. 11th, 2015

Abstract

Harmony search algorithm is a new heuristic global search algorithm, which has been successfully applied in many combinatorial optimization problems. The better performance of genetic algorithm, simulated annealing algorithm and tabu search is demonstrated on the problem. However, the harmony search algorithm is easy to fall into local search. This paper makes an improvement on the generation of the initial solution vector and the harmony bank so as to improve the efficiency of the algorithm.

Keywords

Harmony Search Algorithm, Initial Solution Vector, Harmony Bank

和声搜索算法的改进

张康丽，陈寿元，邵增珍

山东师范大学信息科学与工程学院，山东 济南
Email: shouyuanchen@163.com

收稿日期：2015年10月24日；录用日期：2015年11月7日；发布日期：2015年11月11日

摘 要

和声搜索算法是新近问世的一种启发式全局搜索算法，在许多组合优化问题中得到了成功应用。在有关

文章引用: 张康丽, 陈寿元, 邵增珍. 和声搜索算法的改进[J]. 人工智能与机器人研究, 2015, 4(4): 32-39.
http://dx.doi.org/10.12677/airr.2015.44005

问题上展示了较遗传算法、模拟退火算法和禁忌搜索更好的性能。但是，和声搜索算法在迭代时容易陷入局部搜索的状态，本文针对这一问题，对生成初始解向量进行了改进，对更新和声记忆库进行了改进。从而提高了算法的执行效率，避免算法陷入局部最优。

关键词

和声搜索算法，初始解向量，和声记忆库

1. 引言

2001 年，韩国学者 Geem Z W 提出了和声搜索算法 HS (Harmony Search)，该算法是一种新的启发式智能搜索算法，HS 灵感来源于音乐创作中和声的调节，音乐家们在原有和声的基础上对各个乐器的音调进行调节，逐步实现一个自己满意的新和声。

HS 算法通用好，依赖于问题程度低，一直受到广大研究人员的关注。它被应用到各种优化问题中，例如水网设计优化、土木工程结构优化、公交线路车辆调度优化等众多领域。文献[1]提出了一种求解数值优化问题的轮盘赌自适应和声搜索算法；文献[2]阐述了和声搜索算法的起源、思想、基本步骤，并分析了参数 HMS、HMCR、PAR 对该算法的影响；文献[3]首先描述了原始和声搜索算法的特点和流程，然后对和声搜索算法繁多的变种及其应用领域进行梳理和分类，最后指出和声搜索算法的未来研究方向；文献[4]在和声算法中引入自适应的参数调节方式及遗传算子，使和声算法能够解决离散型数学问题；文献[5]提出了一种提取种群中优良个体的算法。总的来说，和声搜索算法容易陷入局部最优的问题并没有引起足够的重视。本文中提出一种 ALHS 算法对此进行了部分改进，该算法首先采用区间分割法生成初始解向量，避免了收敛于局部最优解的可能性；然后利用了遗传算法的基于排序的概率分配方法和轮盘赌选择方法，提高和声记忆库的多样性，避免算法陷入局部最优，提高了算法的效率。实验结果表明改进的算法效率高于 HS 算法[2]。

2. 和声搜索算法

和声搜索算法原理

和声搜索算法的基本思想：首先对和声记忆库大小 HMS (Harmony Memory Size)和各种参数进行初始化，然后按照一定的概率生成新的和声并且对新和声进行微调，如果新和声优于原和声记忆库中最差的和声则进行替换，按照上述步骤进行迭代直到找到最优解或满足停止条件为止[6]。HS 算法基本步骤为：

Step 1：确定需要优化问题的目标函数和 HS 的基本参数。

$$\max|\min g(X)$$

$$s.t.X=(x_1,\ x_2,\cdots,x_n)^{\mathrm{T}}$$

$$x_i\in[x_{i\min},x_{i\max}]\ i=1,2,\cdots,n$$

其中 $g(X)$ 是需要进行优化的目标函数，X 是由 n 个决策向量组成的解向量，x 是决策向量，x 的下界和上界分别是 $x_{i\min}$ 和 $x_{i\max}$。初始化的参数主要包含：

n，决策变量的个数(I ≥ 1)

x_i，决策变量的范围 $x_{i\min}\ \le\ x_i\ \le\ x_{i\max}$

HMS，和声记忆库的解向量个数

HMCR，和声记忆库的保留概率

PAR，新生成的解向量的微调概率

bw，(band width)微调时使用的干扰带宽

iter，迭代次数计数器(初始值为 0)

Itermax，算法的最大迭代次数(停止条件)

Step 2：根据 Step 1 的约束对和声记忆库进行随机初始化。

根据下式可得到解向量的各个决策变量 $x_i \in [x_{i\max}, x_{i\min}]$，从而得到和声记忆库中各个初始化解向量。

$$x_{ij} = x_{i\min} + (x_{i\max} - x_{i\min}) \times r$$
$$s.t. j = 1, 2, \cdots, HMS$$
$$i = 1, 2, \cdots, n$$
$$r \in (0,1) \text{之间的随机数}$$

Step 3：构建新的解向量

新的解向量 $X_{new} = (x_{1new}, x_{2new}, \cdots, x_{nnew})^{\mathrm{T}}$，其中每个决策变量 x_{inew} 可通过三中方式产生。

1) 选择原解向量中的对应决策变量 $x_{inew} \in U\{x_{i1}, x_{i2}, \cdots, x_{iHMS}\}$；

2) 在决策变量的取值范围内随机选择 $x_{inew} \in [x_{i\max}, x_{i\min}]$；

3) 对于前两种方式中某些决策变量新型概率微调。具体构建流程如下：

```
For i=1 to n do
    If r<HMCR then //从原决策向量中选择
        xinew                    ∈U{xi1,xi2,⋯,xiHMS}
        If r<PAR //对原决策变量进行微调
            xinew=xinew±r×bw
        End if
    Else //随机选择产生新的决策向量
        xinew=ximin+(ximax-ximin) ×r
    End if
End for
```

Step 4： 更新和声记忆库。Xworst 是和声记忆库中最差的解向量。

Xworst =Xnew,g(Xnew)优于 g(Xworst)

=Xworst,g(Xnew)差于 g(Xworst)

Step 5：判断算法是否满足停止条件。$i_{ter} = i_{ter} + 1$，若 $i_{ter} < I_{termax}$ 则跳转到 Step 3 继续迭代，否则输出最优解，算法结束。

3. 改进的和声搜索算法

3.1. 生成初始解向量的改进

由于和声记忆库的的初始解向量是随机生成的，解向量在解空间的的分布状况可能不合理，在迭代时很容易陷入局部搜索的状态，在一定程度上影响算法的搜索性能[2]。为了增强遍历性，本文首先采用区间分割法，把各决策变量的取值范围划分成HMS个子区间(*n*个子区间构成子解空间)，再在各个子解空间中随机生成一个初始解向量，从而产生初始和声记忆库[7]。

初始解向量采用区间分割法生成，可以保证随机产生的各个解向量具有差别性，在解空间上的分布

状况均匀，避免了收敛于局部最优解的可能性，加快了和声搜索算法的收敛速度，增强了全局搜索的能力[8]。

$$V_j = \left\{ x = x_i \middle| x_i \in \left[x_{i\min} + (j-1)\Delta d, x_{i\min} + j\Delta d \right] (i = 1,2,\cdots,n) \right\}$$
$$s.t.\ \Delta d = \left(x_{i\max} - x_{i\min} \right) / HMS$$
$$j = 1,2,\cdots,HMS$$

其中V_j为分割得到的子解空间，Δd是各个决策向量子区间大小。n个决策向量子区间构成一个子解空间。

3.2. 更新和声记忆库的改进

传统HS的选择更新策略是：选择和声记忆库中的最差解向量，若新的解向量优于最差解向量则进行替换，否则不作处理。这种选择更新策略严重影响了和声记忆库的种群多样性[9]。这样经过了若干次迭代后，和声记忆库中的较差的解向量没有得到更新，而且只更新最差的解向量也不利于种群的进化，种群多样性降低。为了避免算法陷入局部最优，提高和声记忆库的多样性，本文引入了基于遗传算法的基于排序的概率分配方法和轮盘赌选择方法[1]。

1) 基于排序的概率分配方法(Rank-Based Probability Assignment)

将和声记忆库中的解向量按照适应度值进行排序，概率与具体的适应度值无关，按照解向量在顺序表中的排名确定解向量的被选择概率。计算选择概率有两种方法本文采用第一种方法计算选择概率[4]。

方法一(由 Baker 提出，其中 i 是解向量在顺序表中的排名)：

$$P(i) = 1/HMS\left[\lambda_+ - \left(\lambda_+ - \left(\lambda_+ - \lambda_- \right)(i-1)/(HMS-1) \right) \right]$$
$$s.t. \lambda_+ \in [1.2]$$
$$\lambda_+ + \lambda_- = 2$$

方法二(由 Michalewicz 提出，θ 是排名第一的解向量的概率)：

$$P(i) = \theta(1-\theta)i - 1$$

本文采用第一种方法计算选择概率，按照适应度值从小到大进行排序，则适应度值最小的解向量被选择的概率 p(0) = 0，也就是说最优的解向量永远不会被替换[10]。

2) 轮盘赌选择方法(Roulette Wheel Selection)

通过概率分配方法得到选择概率后，计算各个解向量的累计概率$q(i) = \Sigma ij = 1p(j)$，每次迭代产生一个$r \in (0,1)$之间的随机数，通过与累计概率比较，判断是否选择进行更新。

若$r \le q(1)\ i = 1$，则选择第 1 个解向量

若$q(i-1) \le q(i)\ i > 1$，则选择第 i 个解向量[11]

根据上述算法基本思想，算法部分代码(使用改进后的算法更新和声库)表示如下：

```
1) def refreshALHS():
2) global ALHS
3) newHM = createNew(ALHS)
4) for i in range(4):
    for i in range(4 - i):
        if calFxy(ALHS[i][0], ALHM[i][1]) > calFxy(ALHS[i + 1][0], ALHS[i + 1][1]):
            temp = ALHS[i]
```

```
                ALHS[i] = ALHS[i + 1]
                ALHS[i + 1] = temp
        Pi = []
        lanmuda = round(random.uniform(1,2), 2)
        r = random.random()
    5) for i in range(5):
            if i == 0:
                Pi.append(0)
            else:
                temp = (1.0 / HMS) * (lanmuda - (lanmuda    - (2 - lanmuda)) * (float(i - 1) / (HMS - 1)))
                Pi.append(round(temp, 2))
        if r < sum(Pi[:2]):
            ALHS[1] = newHS
    6) if sum(Pi[:2]) <= r and r < sum(Pi[:3]):
            ALHS[2] = newHS
    7) if sum(Pi[:3]) <= r and r < sum(Pi[:4]):
            ALHS[3] = newHS
    8) if sum(Pi[:4]) <= r and r < sum(Pi):
            ALHS[4] = newHS
```

4. 实验结果分析

4.1. Rosenbrock 测试函数

试验环境：Windows 64 位操作系统；Python2.7.10 语言；PyCharm 4.5 平台。目标函数为 Rosenbrock 函数[12]，该函数是一个用来测试最优化算法性能的非凸函数。Rosenbrock 函数每个等高线大致呈抛物线形，其全域最小值也位于抛物线形山谷中。但山谷内的值变化不大，要找到全域最小值相当困难。其全域最小值位于(x, y) = (1,1)点，数值为 f(x,y) = 0。实验参数：HMS = 5，HMCR = 0.9，PAR = 0.3，Bw = 0.01。

1) 稳定性测试

从图 1，图 2 中可以看到：改进后的 ALHS 算法搜索到的目标函数的取值大部分在 0 附近，波动较小；未改进的 HS 算法搜索的目标函数的取值偏离最优值较多，波动较大。这就表明，ALHS 算法在寻求最优值方面优于 HS 算法。通过实验可得(表 1)，整体上，ALHS 算法优于 HS 算法，但是随着迭代次数越来越多，效果明显度降低。

2) 性能测试

通过图 3 可以得出：随着迭代次数的增加，HS 算法陷入局部最优，ALHS 算法并没有陷入局部最优，这也充分说明了改进的算法有效地解决了陷入局部最优的问题，从而改善了算法的性能并且解决了长久以来和声算法的最大问题。

4.2. 其它测试函数(图 4~7)

综上所述：改进的算法 ALHS 在稳定性测试以及性能测试(是否陷入局部最优)方面都要优于 HS (为改进的和声算法)。

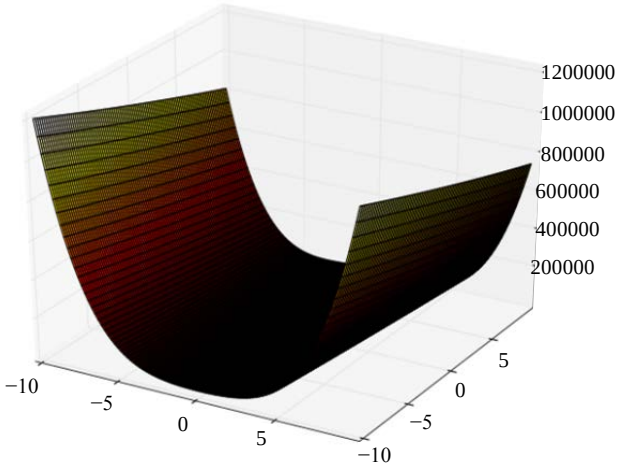

Figure 1. Rosenbrock function
图 1. Rosenbrock 函数图

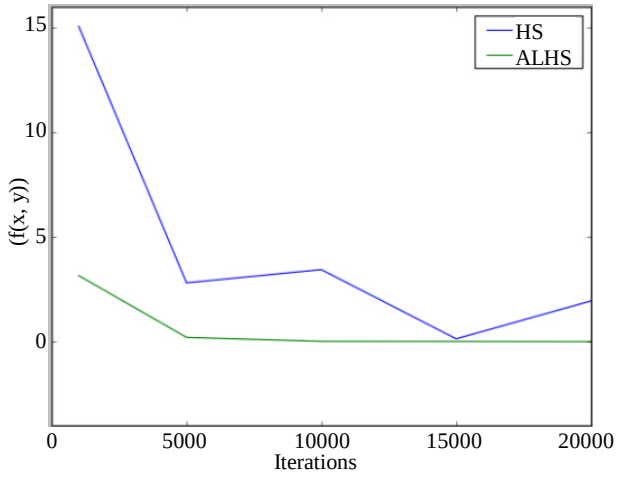

Figure 2. Comparison results of stability test algorithm
图 2. 稳定性测试算法对比结果图

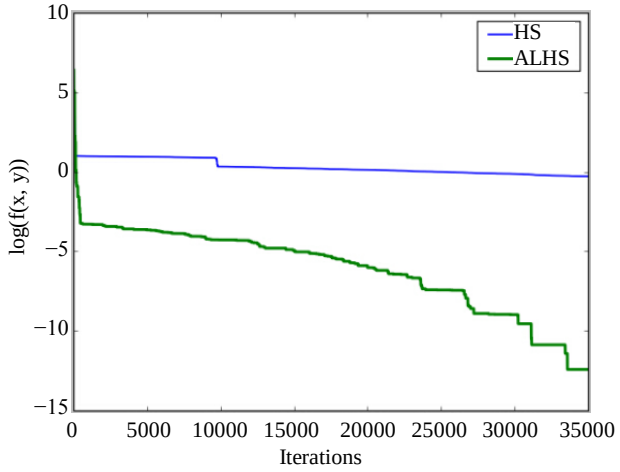

Figure 3. Performance results comparison chart
图 3. 性能结果对比图

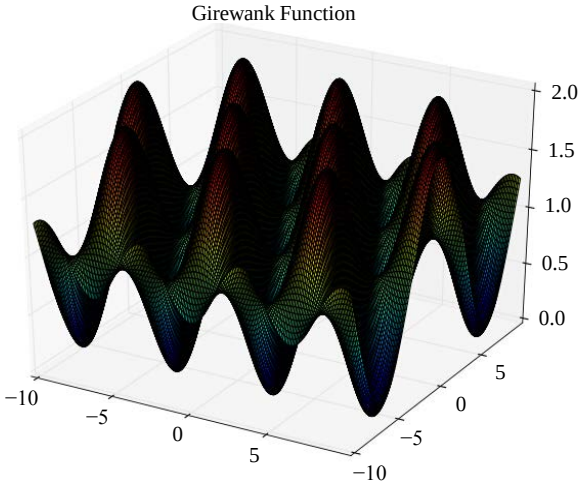

Figure 4. Girewank function
图 4. Girewank 函数

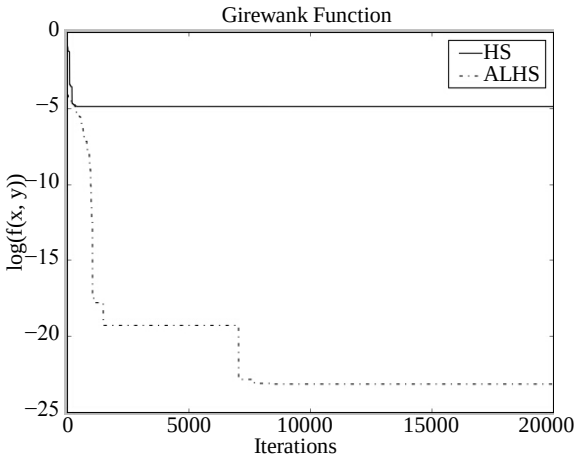

Figure 5. Performance results comparison chart
图 5. 性能结果对比图

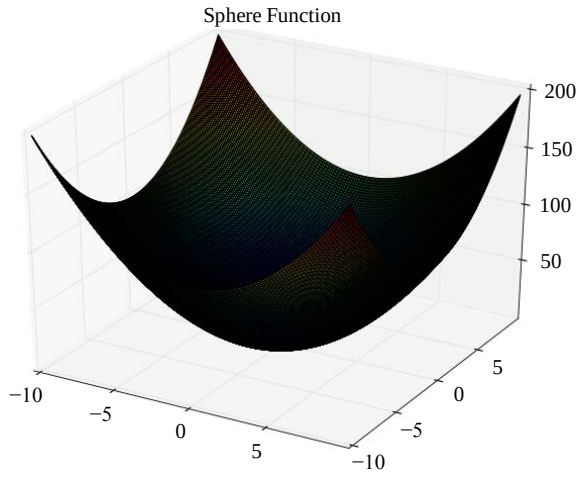

Figure 6. Sphere function
图 6. Sphere 函数

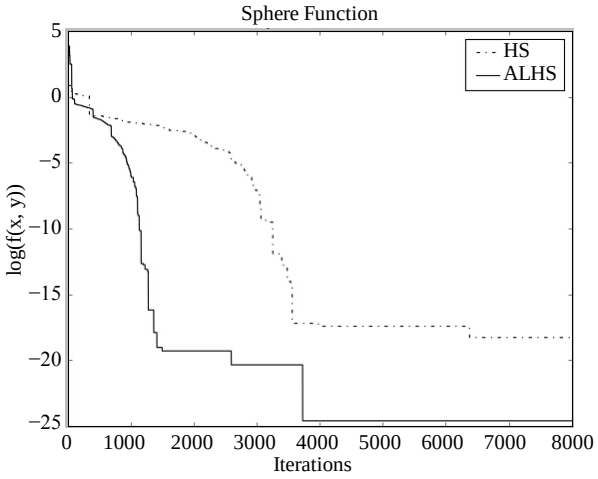

Figure 7. Sphere function performance results
图 7. Sphere 函数下性能结果图

Table 1. HS algorithm and ALHS algorithm to find the optimal value of the data
表 1. HS 算法与 ALHS 算法寻找最优值数据结果图

算法最优值 f(x,y)\迭代次数	1000	5000	10,000	15,000	20,000
HS-f(x,y)	15.0733812	2.80783099	3.43908	0.13252056	1.9447976
ALHS-f(x,y)	3.1549648	0.2028839	0.0139705	0.0096196	0.0008727

5. 结论

本文针对和声算法的搜索性能不足，提出了一种改进的和声算法 ALHS。通过对生成初始解向量的改进，对和声记忆库的改进，改善了算法的性能。另外，本文详细说明了该算法的基本思想，通过实验对比分析具体证明了算法的有效性。在未来工作中，将重点在和声算法的新的应用领域进行研究[5]。

参考文献 (References)

[1] 李永林, 叶春明, 刘长平. 轮盘赌选择自适应和声搜索算法[J]. 计算机应用研究, 2014, 31(6): 1665-1668.

[2] 薛亚娣. 和声搜索算法综述[J]. 技术天地, 2014, 3.

[3] 周雅兰, 黄韬. 和声搜索算法改进与应用[J]. 计算机科学, 2014, 6(41): 52-75.

[4] 赵鸿飞, 张琦, 朱春生. 基于改进自适应和声遗传算法的装配序列优化研究[J]. 计算机应用研究, 2013, 8(30): 2357-2364.

[5] 杨树欣, 李盼池. 和声搜索算法的改进研究[J]. 计算机技术与发展, 2015, 25(4): 93-97.

[6] Ingram, G. and Zhang, T.H. (2009) Overview of Applications and Developments in the Harmony Search Algorithm. *Music-Inspired Harmony Search Algorithm*, **191**, 15-37.

[7] 李峰刚, 魏炎炎, 杨龙. 基于和声算法异构 Hadoop 集群资源分配优化[J]. 计算机工程与应用, 2014, 50(9): 98-102.

[8] 黄鉴, 彭其渊. 多样性保持的和声搜索算法及其 TSP 求解[J]. 计算机应用研究, 2013, 12(20): 3583-3585.

[9] Das, S., Mukhopadhyay, A. and Roy, A. (2011) Exploratory Power of the Harmony Search Algorithm: Analysis and Improvements for global Numerical Optimization. *IEEE Transactions on Systems*, **41**, 89-106.

[10] 李树荣, 陈国霞, 雷阳. 一种加快局部收敛速度的改进和声搜索算法[C].//*Proceedings of the* 31*st Chinese Control Conference*, 2012: 2368-2373.

[11] 张琛, 詹志辉. 遗传算法选择策略比较[J]. 计算机工程与设计, 2009, 30(23): 5471-5478.

[12] 王慧敏, 贺兴时, 威孟龙. 一种改进的和声搜索算法[J]. 纺织高校基础科学学报, 2013, 26(3): 383-387.