文章编号: 1001-0920 (2014) 07-1174-07 DOI: 10.13195/j.kzyjc.2013.0721

一种求解0-1背包问题的二进制修正和声搜索算法

欧阳海滨, 高立群, 孔祥勇, 刘宏志

(东北大学 信息科学与工程学院，沈阳 110004)

摘 要: 针对0-1背包问题, 提出一种二进制修正和声搜索算法. 该算法修正了即兴创作过程, 对参数PAR进行动态调整, 同时提出一种随机修复机制, 有效修复不可行的和声, 增强算法的局部搜索. 采用一种可行和声初始化方式, 保证初始和声都是可行的, 整个搜索过程完全采用0-1二进制模式, 对14个0-1背包问题进行测试. 将所提出算法与其他算法进行比较, 结果验证了所提出算法的有效性.

关键词: 0-1背包问题；二进制修正和声搜索算法；局部搜索；随机修复机制

中图分类号: TP273 **文献标志码:** A

A binary modified harmony search algorithm for 0-1 knapsack problem

OUYANG Hai-bin, *GAO Li-qun*, *KONG Xiang-yong*, *LIU Hong-zhi*

(College of Information Science and Engineering，Northeastern University，Shenyang 110004，China．Correspondent: OUYANG Hai-bin，E-mail：oyhb1987@163.com)

Abstract: A binary modified harmony search algorithm is proposed to solve the 0-1 knapsack problem(KP). In the algorithm, the improvisation process is modified and the parameter PAR is adjusted dynamically, as well as a stochastic repair operator is developed to effectively repair infeasible harmony and enhance local search. Besides, a feasible harmony initialization method is used to guarantee initial harmony feasible. The 0-1 binary model is completely used in the whole search process. Fourteen 0-1 knapsack problems are tested. The proposed algorithm is compared with other algorithms, and the statistical results demonstrate the effectiveness of the proposed algorithm.

Key words: 0-1 knapsack problem；binary modified harmony search algorithm；local search；stochastic repair operator

0 引 言

0-1背包问题由Fayard[1]首次提出, 是运筹学和计算机科学领域中一类典型的组合优化NP完全问题. 0-1背包问题一经提出便得到许多学者的关注和研究, 至今已涌现出各种求解0-1背包问题的优化算法, 大致可以分为两类: 一类是基于数学理论的传统近似算法, 如动态规划算法、Lagrange算法、分支定界算法等; 一类是受自然现象启发的智能随机算法, 如模拟生物进化的遗传算法和差分进化算法、模拟鸟类迁移的粒子群算法、受蜜蜂采蜜启发的人工蜂群算法等. 但是, 传统近似算法所需要的计算量和存储空间随着背包维数的增加而快速增大, 难以有效地解决大规模的0-1背包问题. 近几年, 智能随机算法逐渐应用于0-1背包问题的求解, 并取得了良好的结果. 文献[2]提出了一种解决0-1背包问题的新颖全局和声搜索算法(NGHS), 通过位置更新和变异操作提高算法的优化性能, 有效求解了一系列0-1背包问题. 文献[3]提出了一种基于贪婪策略的化学反应优化算法(CROG), 该算法设计了一种融合贪婪策略和随机选择的修复函数, 对不可行解进行了有效修复, 并对0-1背包问题进行了求解, 所得结果优于遗传算法、蚁群算法和量子进化算法. 文献[4]将学习型和声搜索算法(LHS)应用于0-1背包问题, 通过学习策略增强了算法的全局搜索. 文献[5]利用改进的自适应二进制和声搜索算法(ABHS)优化0-1背包问题, 通过一种可伸缩的自适应策略提高算法的搜索能力和鲁棒性.

和声搜索算法(HS)是由Geem等[6]于2001年提出的一种启发式智能算法. 基于HS的优势和特点, 近几年HS已成功应用于许多实际优化问题中, 如电力系统环境经济调度问题、地下水管理、混合稳态自适

收稿日期: 2013-06-02；**修回日期:** 2013-09-02.

基金项目: 国家自然科学基金项目(60674021).

作者简介: 欧阳海滨(1987−), 男, 博士生, 从事智能优化与系统建模的研究；高立群(1949−), 男, 教授, 博士生导师, 从事智能优化与图像处理等研究.

应模糊控制器的设计和中等规模的时尚零售供应链销售预测问题等. 然而, 启发式智能优化算法存在收敛精度不高、易陷入局部最优的缺点, HS 算法也不例外, 这便需要研究者们进一步改进和修正, 以弥补启发式智能优化算法的不足. 对于 HS 算法, 研究者们从算法的参数方面进行改进, 提出了改进和声搜索算法 (IHS)[7]、自适应和声搜索算法 (SAHS)[8]、带有动态控制参数的和声搜索算法 (NDHS)[9]; 从算法操作策略上进行修正, 提出了全局最好和声搜索算法 (GHS)[10]、新颖的全局和声搜索算法 (NGHS)[11]、智能调整和声搜索算法 (ITHS)[12]; 从算法融合方面进行调整, 提出了动态多种群粒子群 HS 算法 (DMS-PSO-HS)[13]. 这些修正的 HS 算法促进了 HS 算法的发展, 在一定程度上提高了算法的优化精度.

本文针对 0-1 背包问题的特点, 提出一种二进制修正和声搜索算法 (BMHS). 该算法采用二进制创作模式, 利用全局最好和声的信息创作新的和声, 并设计了一种简单的随机修复机制, 快速修复即兴创作后产生的不可行的和声. 通过大量 0-1 背包实例的测试表明, 本文算法所得结果优于其他 7 种算法, 验证了 BMHS 算法的快速搜索能力和有效性.

1　0-1 背包问题

0-1 背包问题可以描述如下: 已知 D 个物体, 第 j $(j = 1, 2, \cdots, D)$ 个物体的重量和价值分别用 w_j 和 p_j 表示, 背包所能容纳的物体总重量上限为 C, 其目标是在背包能够承受的重量范围内尽可能使背包中的物体价值总和最大化. 具体数学模型为

$$\begin{aligned} &\max f(X) = \sum_{j=1}^{D} p_j x_j; \\ &\text{s.t.} \sum_{j=1}^{D} w_j x_j \leqslant C, \\ &\quad x_j = 0 \text{ or } 1,\ j = 1, 2, \cdots, D. \end{aligned} \tag{1}$$

2　基本和声搜索算法

HS 算法与遗传算法、粒子群算法和差分进化算法相比, 最大的不同在于, HS 算法通过随机的概率指导和声向量的每一维和声分量进行和声记忆库考虑、基音调整和随机变异, 最终得到一个新的和声向量, 并不断更新和声记忆库.

基本 HS 算法的流程如下.

Step 1: 算法参数初始化. 给定和声记忆库大小 (HMS)、和声记忆库考虑概率 (HMCR)、基音调整概率 (PAR)、和声微调步长 (bw)、最大迭代次数 K.

Step 2: 给定范围 $[x_i^L, x_i^U]$, 其中 x_i^L 和 x_i^U 分别为第 i 维变量的下限和上限. 根据下式随机初始化, 产生 HMS 个和声向量存入和声记忆库 HM:

$$x_i = x_i^L + \text{rand} \times (x_i^U - x_i^L). \tag{2}$$

Step 3: 基于 HMCR、PAR 和 bw 进行即兴创作, 产生新的和声向量. 具体伪代码如下所示, 其中 N 为问题的维数, 在进行基音调整时, 若随机数大于 0.5, 则取“+”号, 若小于 0.5, 则取“−”号.

```
for i = 1 to N %即兴创作过程开始
  if rand<HMCR %和声记忆库考虑操作
r ∈ {1, 2, ··· , HMS} x_i^new = x_i^r
    if rand<PAR %基音调整操作
    x_i^new = x_i^r ± rand × bw %进行基音调整
    end if
  else %随机变异操作
    x_i^new = x_i^L + rand × (x_i^U − x_i^L)
  end if
end for %即兴创作结束.
```

Step 4: 更新和声记忆库. 判断新和声 x^{new} 是否优于当前 HM 内最差和声 x^{worst}, 若是, 则用 x^{new} 代替 x^{worst}.

Step 5: 判断终止准则. 如果当前迭代次数 k 大于最大迭代次数 K, 则终止运行 HS 算法, 否则重复执行 Step 3 和 Step 4.

由此可见, 即兴创作是 HS 算法的核心步骤, HS 通过即兴创作产生新的和声, 利用更新策略更新和声记忆库, 最终得到一个优美和声, 即最优解. 根据具体的问题, HS 需设计合理有效的即兴创作和更新机制. 另外, HS 注重全局的随机搜索, 缺乏先验知识的指导信息, 迭代搜索盲目性较大. 为了有效地求解 0-1 背包问题, 下一节将介绍一种二进制修正和声搜索算法.

3　二进制修正和声搜索算法

基于 0-1 背包问题的特点, 本文改变以往连续就近取整的搜索方式, 采用完全的二进制搜索模式进行算法设计. 下面具体介绍可行解初始化方式、指导型即兴创作、随机修复机制和参数动态调整, 并给出算法的详细流程.

3.1　可行解初始化方式

0-1 背包问题的一个解对应一个和声, 且保证和声每一维上的值只能为 0 和 1, 0 表示物体没有被选中, 1 表示物体被选中. 在满足算法中的和声是一个二进制串的同时, 为了保证初始的和声都是可行的, 并且具有良好的广泛随机性, 本文提出一种可行解初始化方式, 利用随机数相比较方式初始化和声. 如果初始化的和声不满足约束, 则重新初始化, 直到得到一个可行的和声, 具体伪代码如下:

for $i = 1:1$:HMS

约束 G(i) $= 0$

for $j = 1:1:N$

if rand$_1$ < rand$_2$ $x(i,j) = 0$;

else $x(i,j) = 1$;

end if

end for

计算约束 $G(i)$

while $G(i) > C$ 判断约束是否满足条件

如果不满足约束, 则重新初始化

end while

end for.

3.2 指导型即兴创作

在HS中, 即兴创作过程采用3种操作规则: 1) 根据和声记忆库的考虑概率, 在和声记忆库中选择, 称为和声记忆库的考虑; 2) 根据基音调整概率和和声微调步长逐步调节和声变量, 称为基音调整; 3) 采用随机扰动策略探索新的和声变量, 称为随机变异.本文结合最优和声对整个和声记忆库具有指导作用的群智能思想, 利用最优和声进行基音调整, 同时针对0-1背包的特点, 采用0和1随机初始化模式修正随机变异操作, 并将此即兴创作过程命名为指导型即兴创作. 具体详细的指导型即兴创作的伪代码如下:

for $j = 1:1:N$

if rand$_3$ < HMCR %和声记忆库的考虑

$r = $ ceil(rand·HMS), $\gamma \in \{1, 2, \cdots, \text{HMS}\}$

$x^{\text{new}}(j) = x(r,j)$;

if rand$_4$ < PAR %基音调整

bw(j) = round(rand) %对随机值进行四舍五入取整

if rand$_5$ < 0.5

$x^{\text{new}}(j) = \text{bw}(j)$;

else

$x^{\text{new}}(j) = x^{\text{best}}(j)$;

end if

end if

else $x^{\text{new}}(j) = $ round(rand) %随机变异操作

end if

end for.

其中 rand 与 rand$_i(i = 1, 2, \cdots)$ 为0和1之间的随机数; ceil(x) 为取大于 x 的最小整数, 例如0.1取整为1; round(x) 为对 x 按照四舍五入规则取整. 由指导型即兴创作过程可知, 在基音调整中, 新产生的和声吸收了全局最好和声 x^{best} 的一部分信息, 通过最好和声的信息与随机信息的融合, 有利于向最好和声靠拢, 加强了最好和声附近存在的潜在更好的和声的搜索, 在一定程度上增强了算法的邻域搜索.

3.3 随机修复机制

如果新产生的和声是不可行的, 则采用一种随机修复机制对此和声进行修复. 随机选择需要修复的维数数目, 根据该数目独立选择随机的不相同的维数, 如果此维为1则置0, 并修正约束, 最后判断是否满足约束, 如果满足则停止修复, 否则重新进行随机修复. 为了更好地理解, 下面给出具体的伪代码:

$j = 1$; while GNEW> C

$d = \text{ceil}(\text{rand} \cdot N/4)$;

$\text{DD} = \text{randperm}(N)$;

if $x^{\text{new}}(\text{DD}(j)) == 1 \quad x^{\text{new}}(\text{DD}(j)) = 0$;

$\text{GNEW} = \text{GNEW} - w(j)$; end if

$j = j + 1$; if $j > d$ $j = 1$; end if

end while,

其中 randperm(N) 为对 1 ~ N序号进行随机排序.

3.4 参数动态调整

在本文算法中, PAR控制着算法进行基音调整的概率. 为了较好地调整参数PAR值, 总结了以往文献中提出的PAR动态调整方式, 主要有以下几种:

$$\text{PAR}_k = \text{PAR}_{\min} + \frac{\text{PAR}_{\max} - \text{PAR}_{\min}}{K} \times k^{[7]}; \quad (3)$$

$$\text{PAR}_k = \text{PAR}_{\min} + \frac{\text{PAR}_{\max} - \text{PAR}_{\min}}{K-1} \times (k-1)^{[12]}; \quad (4)$$

$$\text{PAR}_k = \text{PAR}_{\max} - \frac{\text{PAR}_{\max} - \text{PAR}_{\min}}{K} \times k^{[5]}; \quad (5)$$

$$\text{PAR}_k = \text{PAR}_{\min} + (\text{PAR}_{\max} - \text{PAR}_{\min}) \exp(-k/K)^{[9]}; \quad (6)$$

$$\text{PAR}_k = m \exp(nk^2) n = \frac{1}{K^2 - 1} \ln \frac{\text{PAR}_{\max}}{\text{PAR}_{\min}},$$

$$m = \text{PAR}_{\min} \exp(-n)^{[4]}; \quad (7)$$

$$\text{PAR}_k = \text{PAR}_{\min} \exp\left(\frac{k}{K} \ln \frac{\text{PAR}_{\max}}{\text{PAR}_{\min}}\right)^{[6]}. \quad (8)$$

其中: k 和 K 分别为当前迭代次数和最大迭代次数, $\text{PAR}_{\min}$ 和 $\text{PAR}_{\max}$ 分别为最小基音调整概率和最大基音调整概率. 可以看出: 1) PAR随着迭代次数呈线性变化(如式(3)~(5)), PAR在整个搜索过程中的变化速率是恒定的; 2) PAR随着迭代次数呈指数变化(如式(6)~(8)), PAR在整个搜索过程中的变化速率是变化的. PAR在迭代早期变化应该比较缓慢, 这样更能挖掘局部信息, 随着迭代次数的增加, PAR的值也逐渐加快, 在迭代后期变化快一些, 这样能加快搜索速度. 因此, 本文设计参数PAR如下:

$$\mathrm{PAR}_k=\begin{cases}0.3,\ k<K/2;\\ m\exp(n\times k)n=\dfrac{1}{K-1}\ln\dfrac{\mathrm{PAR}_{\max}}{\mathrm{PAR}_{\min}};\\ m=\mathrm{PAR}_{\min}\exp(-n),\ \text{otherwise.}\end{cases}\quad(9)$$

本文基于文献[7,10]对参数进行分析, 通过具体仿真测试, 设置 $\mathrm{PAR}_{\min}=0.01$, $\mathrm{PAR}_{\max}=0.3$, HMCR $=0.99$, HMS $=5$.

3.5 BMHS 算法流程

在基本 HS 算法流程的基础上, 给出本文算法的求解步骤.

Step 1: 算法和问题参数初始化.

Step 2: 根据可行解初始化方式初始化和声记忆库.

Step 3: 基于指导型即兴创作, 产生新的和声.

Step 4: 对不可行的新和声进行随机修复操作.

Step 5: 更新和声记忆库, 判断新和声 x^{new} 是否优于当前 HM 内最差和声 x^{worst}, 若是, 则用 x^{new} 代替 x^{worst}.

Step 6: 核查终止准则. 如果当前迭代次数 k 等于最大迭代次数 K, 则终止运行 BMHS 算法, 否则重复执行 Step 3 ~ Step 5.

由本文算法流程可知, 在整个迭代过程中应用全局最好和声指导新和声的调整, 并且完全采用二进制模式创作, 通过随机数的变化扰动扩大搜索空间, 同时通过随机修复机制增强局部搜索, 最终获得较优的搜索区间, 快速地创作出优美悦耳的和声即最优的解.

3.6 BMHS 算法分析

在 BMHS 算法中, 一个新和声的产生包含 3 部分信息, 分别是和声记忆库中随机的某一个和声的信息、最好和声的信息、随机数调整和变异的信息. 在和声记忆库中, 每一个和声都是迭代后留存较好和声, 相当于精英的保存, 因为 BMHS 算法只更新和声记忆库中的最差和声, 属于一种贪婪更新策略, 这样保证和声记忆库的和声总在不断优化, 质量也在提高. 每次迭代中, 新和声都会一定概率地受到当前最好和声的影响, 并吸收最好和声的部分信息, 使新和声靠近最好和声, 挖掘当前最好和声附近更好的和声. 此外, 如果当前全局最好和声是局部最优解, 则为了防止算法陷入当前最好和声的局部最优邻域, 随机调整和变异随机数, 有效地减小陷入局部最优的可能. 因此, BMHS 算法在一定迭代次数后能够收敛, 且迭代次数足够多时可以收敛到全局最优.

4 实验分析

为了测试本文算法的有效性, 对 0-1 背包问题进行两组实验, 包括 10 个小维数 0-1 背包问题和 4 个大规模 0-1 背包问题. 实验 1 选取文献[2,5]给出的 10 个小维数的背包问题进行求解, 并与最近文献所提出的各种算法进行比较. 实验 2 按照文献[2]提出的大规模 0-1 背包实例产生流程, 得到 4 个大规模的 0-1 背包实例, 然后考察本文算法优化大规模 0-1 背包实例的有效性和可行性.

实验 1　以 10 个 0-1 背包问题为研究对象, 将所提出算法与 HS 算法[6]、ABHS 算法[5]、DBPSO 算法[14]、HQEA 算法[15]、CROG 算法[3]、LHS 算法[4]和 NGHS 算法[2]进行比较, 各算法参数设置如表 1 所示. 所有算法均利用 Matlab 7.1 编程实现, 在 Intel(R) CORE(TM)2 Quad CPU Q9400 @2.66GHz, 2.66 GHz, 3.50 GB 的内存物理地址扩展、Microsoft Windows XP 系统的电脑上独立运行 50 次. 10 个背包问题的参数见文献[2], 在同等迭代次数的条件下, 表 2 记录了 8 种算法所获得的优化结果. 其中: best 为最优值, worst 为最差值, mean 为平均最优值, ART 为平均运行时间, SR 为算法的成功率, Std 为标准差.

由表 2 可见, 针对以上 10 个 0-1 背包问题, 每种算法均能找到每个问题的全局最优值. 从平均最优值看, HS 算法求解所有背包问题未能稳定地搜索到全局最优解, 说明 HS 算法容易陷入局部最优. DBPSO 算法和 HQEA 算法能够稳定寻找到 KP3、KP4、KP7 和 KP9 问题的最优值, 但对于余下的问题也不能稳定有效地求解出全局最优值. ABHS 算法、CROG 算法、LHS 算法、NGHS 算法和本文算法能够稳定地找到 10 个背包问题的全局最优解. 从方差、最差值、成功率来看, ABHS 算法、CROG 算法、LHS 算法、NGHS 算法和本文算法均优于 HS 算法、DBPSO 算法

表 1　各算法的参数设置

算法	参　数
HS	HMCR $=0.9$, HMS $=5$, PAR $=0.33$, bw $=x^U-x^L$
DBPSO	$c_1=2$, $c_2=2$, $x=0.8$, $v_{\max}=6$, $v_{\min}=-6$, population $=30$
CROG	KElossRate $=0.8$, InitialKE $=100$, PopSize $=20$, MoleColl $=0.2$, buffer $=0$
NGHS	Pm $=2/N$, HMS $=5$(low $-$ dimensional), HMS $=30$(high dimensional)
ABHS	$C=15$, PAR $=0.2$, HMS $=30$, NGC $=20$
HQEA	$\alpha=1$, $\beta=3$, $\rho=0.9$, $u=60$, $Q=10$
LHS	$\mathrm{PAR}_{\max}=0.99$, $\mathrm{PAR}_{\min}=0.01$, HMS $=5$, bw $=0.001$, $L=1.5$
BMHS	HMCR $=0.99$, $\mathrm{PAR}_{\min}=0.01$, $\mathrm{PAR}_{\max}=0.33$, HMS $=5$

表 2 8 种算法所获得的 KP1～KP5 的优化结果

函数	指标	HS	ABHS	DBPSO	HQEA	CROG	LHS	NGHS	BMHS
KP1	best	295	295	295	295	295	295	295	295
	worst	288	295	288	293	295	295	295	295
	mean	294.57	295	293.3	293.97	295	295	295	295
	ART	0.533 3	0.336 5	0.365 2	1.025 4	0.857 4	0.403 2	0.305 8	0.334 7
	SR/%	80	100	75	80	100	100	100	100
	Std	0.102 6	0	2.520 9	2.132 4	0	0	0	0
KP2	best	1 024	1 024	1 024	1 024	1 024	1 024	1 024	1 024
	worst	1 018	1 024	999	979	1 024	1 024	1 024	1 024
	mean	1 023.64	1 024	1 022.8	1 020.3	1 024	1 024	1 024	1 024
	ART	0.678 3	0.402 5	0.465 3	1.003 5	0.976 5	0.402 5	0.389 6	0.312 9
	SR/%	94	100	95	95	100	100	100	100
	Std	1.643 7	0	2.441 2	4.065 4	0	0	0	0
KP3	best	35	35	35	35	35	35	35	35
	worst	28	35	35	35	35	35	35	35
	mean	34.86	35	35	35	35	35	35	35
	ART	0.302 9	0.465 3	0.308 6	0.735 6	0.650 1	0.402 3	0.220 1	0.273 1
	SR/%	98	100	100	100	100	100	100	100
	Std	0.98	0	0	0	0	0	0	0
KP4	best	23	23	23	23	23	23	23	23
	worst	17	23	23	23	23	23	23	23
	mean	22.98	23	23	23	23	23	23	23
	ART	0.402 3	0.403 8	0.302 1	0.654 7	0.546 3	0.376 4	0.328 6	0.281 9
	SR/%	98	100	100	100	100	100	100	100
	Std	0.14	0	0	0	0	0	0	0
KP5	best	481.07	481.07	481.07	481.07	481.07	481.07	481.07	481.07
	worst	401.02	481.07	430.92	423.46	481.07	481.07	481.07	481.07
	mean	476.5	481.07	464.09	463.31	481.07	481.07	481.07	481.07
	ART	0.458 6	0.403 2	0.456 2	0.879 5	0.879 6	0.403 6	0.387 5	0.379 8
	SR/%	88	100	75	80	100	100	100	100
	Std	13.28	0	18.05	20.846	0	0	0	0
KP6	best	52	52	52	52	52	52	52	52
	worst	43	52	47	46	52	52	52	52
	mean	51.03	52	51.06	50.36	52	52	52	52
	ART	0.456 3	0.367 5	0.421 1	0.765 8	0.786 5	0.402 3	0.330 5	0.337 4
	SR/%	82	100	90	85	100	100	100	100
	Std	0.94	0	1.5	1.03	0	0	0	0
KP7	best	107	107	107	107	107	107	107	107
	worst	105	107	107	107	107	107	107	107
	mean	105.64	107	107	107	107	107	107	107
	ART	0.402 5	0.312 5	0.453 1	0.756 4	0.740 1	0.400 2	0.387 5	0.337 2
	SR/%	62	100	100	100	100	100	100	100
	Std	2.86	0	0	0	0	0	0	0
KP8	best	9767	9767	9767	9767	9767	9767	9767	9767
	worst	9732	9767	9736	9765	9767	9767	9767	9767
	mean	9753.1	9767	9752.8	9766.2	9767	9767	9767	9767
	ART	0.412 6	0.398 6	0.467 3	0.867 5	1.000 2	0.402 3	0.376 5	0.42
	SR/%	94	100	85	97	100	100	100	100
	Std	10.85	0	8.391	1.524 1	0	0	0	0
KP9	best	130	130	130	130	130	130	130	130
	worst	118	130	130	130	130	130	130	130
	mean	128.8	130	130	130	130	130	130	130
	ART	0.431 2	0.403 2	0.475 3	0.623 1	0.832 1	0.402 1	0.343 1	0.383 3
	SR/%	98	100	100	100	100	100	100	100
	Std	1.68	0	0	0	0	0	0	0
KP10	best	1 025	1 025	1 025	1 025	1 025	1 025	1 025	1 025
	worst	1 018	1 025	1 018	1 018	1 025	1 025	1 025	1 025
	mean	1 023.2	1 025	1 023.6	1 022.8	1 025	1 025	1 025	1 025
	ART	0.473 2	0.512 5	0.496 5	0.867 5	0.967 5	0.401 4	0.396 5	0.393 5
	SR/%	94	100	96	97	100	100	100	100
	Std	1.42	0	1.522 2	1.443 4	0	0	0	0

和 HQEA 算法. 这表明 ABHS 算法、CROG 算法、LHS 算法、NGHS 算法和本文算法求解 0-1 背包问题具有良好的优化潜力. 从算法运行的平均时间看, 各个算法运行需要的时间基本差别不大, 本文算法、ABHS 算法和 NGHS 算法略微占优势, 运行时间较短, 整体上表明本文算法求解 0-1 背包问题具有较强的竞争力.

实验 2　为了进一步考察本文算法求解大规模 0-1 背包问题的优化性能, 设计 4 个大规模的 0-1 背包实例, 维数 N 分别设置为 100、500、1 000、1 200. 对于 0-1 背包问题的每个维数, 重量和价值的值均随机产生, 重量在 5～20 之间取值, 价值在 50～100 之间取值. 0-1 背包问题的 4 个重量容限的值分别设置为 985、3 676、10 410 和 11 992, 求解 4 个背包实例的最大迭代次数分别设置为 30 000、50 000、100 000 和 300 000, 每个背包实例随机产生后便保持不变. 实验中, 考虑求解小维数背包问题时, ABHS 算法、CROG 算法、LHS 算法、NGHS 算法和本文算法具有较好的优化性能, 但文献 [5] 显示出 ABHS 算法优于 NGHS 算法, 因此, 本文算法、ABHS 算法、CROG 算法和 LHS 算法对大规模背包实例进行求解的参数设置保持不变, 独立运行 40 次, 得到的优化结果如表 3 所示. 根据迭代过程中最好值的变化, 给出算法的迭代收敛曲线如图 1～图 4 所示.

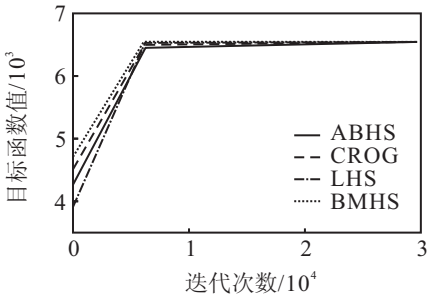

图 1　KP11 的最优值收敛曲线

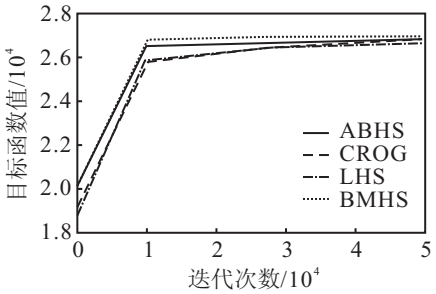

图 2　KP12 的最优值收敛曲线

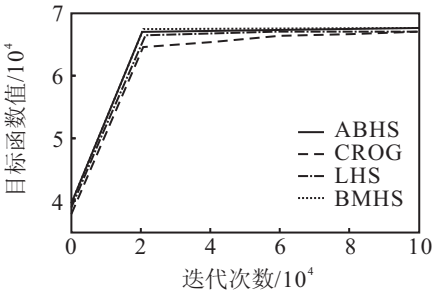

图 3　KP13 的最优值收敛曲线

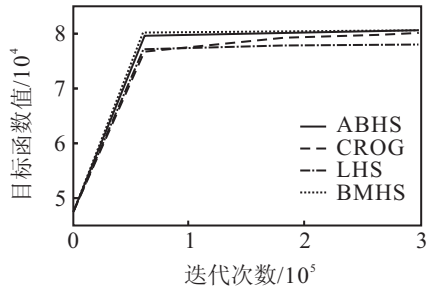

图 4　KP14 的最优值收敛曲线

由表 3 可见, 在标准差方面, 本文算法求解 KP11 所得的结果劣于 ABHS 算法和 CROG 算法, 求解 KP12 所得标准差结果没有 ABHS 算法好, 在其他方面, 本文算法所求得的结果在最好值、平均值和最差值方面都是 4 种算法中最好的, 显示出良好的优化潜力.

表 3　4 种算法获得的优化结果

函数	算法	最好值	最差值	平均值	标准差
KP11	ABHS	6.54e+03	6.53e+03	6.53e+03	3.47e+00
	CROG	6.54e+03	6.53e+03	6.54e+03	2.22e+00
	LHS	6.54e+03	6.52e+03	6.53e+03	2.63e+01
	BMHS	6.56e+03	6.54e+03	6.55e+03	1.03e+01
KP12	ABHS	2.68e+04	2.66e+04	2.67e+04	3.53e+01
	CROG	2.67e+04	2.66e+04	2.66e+04	7.32e+01
	LHS	2.67e+04	2.66e+04	2.67e+04	6.57e+01
	BMHS	2.68e+04	2.67e+04	2.68e+04	5.10e+01
KP13	ABHS	6.78e+04	6.74e+04	6.76e+04	1.75e+02
	CROG	6.69e+04	6.66e+04	6.67e+04	1.06e+02
	LHS	6.78e+04	6.75e+04	6.76e+04	8.19e+01
	BMHS	6.79e+04	6.77e+04	6.78e+04	3.56e+01
KP14	ABHS	8.01e+04	7.97e+04	7.99e+04	1.84e+02
	CROG	8.01e+04	7.99e+04	8.00e+04	8.86e+01
	LHS	8.01e+04	8.01e+04	8.01e+04	2.30e+01
	BMHS	8.02e+04	8.02e+04	8.02e+04	1.09e+01

由图1～图4可知: 在迭代早期, BMHS算法、ABHS算法、CROG算法和LHS算法均具有快速的收敛速度, 各个算法的收敛斜率相差不大, 本文算法略大. 其原因是本文算法利用最好和声指导即兴创作, 快速提高和声的质量, 产生更优美的和声. 在迭代后期, LHS算法和CROG算法过早陷入局部最优, 导致无法搜索到更好的解, ABHS算法和BMHS算法逐渐显示出良好的搜索性能, 但就精度而言, BMHS算法比ABHS算法好.

5 结 论

本文提出了一种求解0-1背包问题的二进制修正和声搜索算法. 基于群智能思想的启发, 采用最优和声指导新和声的即兴创作方式, 设计了可行的和声初始化和不可行和声随机修复机制, 在一定程度上增强了算法的局部搜索, 同时对关键参数PAR进行动态调整, 有效平衡了算法的局部搜索和全局搜索. 通过10个经典的小维数0-1背包问题和4个大规模0-1背包问题的仿真研究, 表明BMHS求解0-1背包问题具有良好的优化潜力和稳定性, 尤其在求解大规模0-1背包问题中脱颖而出, 优于其他文献提出的ABHS算法、CROG算法和LHS算法, 是一种可靠有效的算法.

下一步的工作是采用多和声记忆库协同创作方式, 进一步提高算法的优化潜力. 此外, 考虑反向学习的竞争优势, 扩展出双向创作竞争模式, 增强局部搜索的有效性, 并进一步扩大应用范围, 例如多维背包问题、可靠性优化问题等.

参考文献(References)

[1] Fayard D, Plateau G. Resolution of the 0-1 knapsack problem comparison of methods[J]. Mathematical Programming, 1975, 8(1): 272-307.

[2] Zou D X, Gao L Q, Wu J H, et al. Solving 0-1 knapsack problem by a novel global harmony search algorithm[J]. Applied Soft Computing, 2011, 11(2): 1556-1564.

[3] Tung K T, Li K L, Xua Y M. Chemical reaction optimization with greedy strategy for the 0-1 knapsack problem[J]. Applied Soft Computing, 2013, 13(4): 1774-1780.

[4] 李若平, 欧阳海滨, 高立群, 等. 学习型和声搜索算法及其在0-1背包问题中的应用[J]. 控制与决策, 2013, 28(2): 205-210.
(Li R P, Ouyang H B, Gao L Q, et al. Learned harmony search algorithm and its application to 0-1 knapsack problems[J]. Control and Decision, 2013, 28(2): 205-210.)

[5] Wang L, Yang R X, Xua Y, et al. An improved adaptive binary harmony search algorithm[J]. Information Sciences, 2013, 232: 58-87.

[6] Geem Z W, Kim J H, Loganathan G V. A new heuristic optimization algorithm: Harmony search[J]. Simulation, 2001, 76(2): 60-68.

[7] 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.

[8] Wang C M, Huang Y F. Self-adaptive harmony search algorithm for optimization[J]. Applied Mathematics and Computation, 2010, 37(4): 2826-2837.

[9] Chen J, Pan Q K, Li J Q. Harmony search algorithm with dynamic control parameters[J]. Applied Mathematics and Computation, 2012, 219(2): 592-604.

[10] Omran M G H, Mahdavi M. Global-best harmony search[J]. Applied Mathematics and Computation, 2008, 198(2): 643-656.

[11] Zou D X, Gao L Q, Wu J H, et al. Novel global harmony search algorithm for unconstrained problems[J]. Neurocomputing, 2010, 73(16/17/18): 3308-3318.

[12] Parikshit Yadav, Rajesh Kumar, Panda S K, et al. An intelligent tuned harmony search algorithm for optimization[J]. Information Sciences, 2012, 196: 47-72.

[13] Zhao S Z, Suganthan P N, Pan Q K, et al. Dynamic multi-swarm particle swarm optimizer with harmony search[J]. Expert Systems with Applications, 2011, 38(4): 3735-3742.

[14] Kennedy J, Eberhurt R. A discrete binary version of the particle swarm algorithm[C]. Proc of the IEEE Int Conf on 1997 Computational Cybernetics and Simulation Systems Man and Cybernetics. Orlando, 1997, 5: 4104-4108.

[15] 覃朝勇, 黄景文, 郑建国, 等. 求解背包问题的混合量子进化算法[J]. 小型微型计算机系统, 2011, 32(2): 305-309.
(Qin C Y, Huang J W, Zheng J G, et al. Hybrid quantum-inspired evolutionary algorithm for knapsack problem[J]. J of Chinese Computer Systems, 2011, 32(2): 305-309.)

(责任编辑: 郑晓蕾)