一种求解高维复杂优化问题的动态自适应和声搜索算法

拓守恒 邓方安
(陕西理工学院数学与计算机科学学院 汉中 723000)

摘 要 为了更好地提高求解高维复杂优化问题的能力,提出一种动态自适应和声搜索(DSHS)算法。该算法采用正交试验来设计算法的初始化和声记忆库;利用多维动态自适应调整算子和单维和声微调算子相结合的策略进行和声创作;改进和声音调调解步长,从而增强算法的扰动能力,避免其陷入局部搜索。通过6个标准Benchmark函数测试表明,该算法在全局搜索能力、收敛速度和稳定性方面都有明显提高。

关键词 高维优化问题,动态自适应,和声搜索算法
中图法分类号 TP18 **文献标识码** A

Dynamic Self-adaptive Harmony Search Algorithm for Solving High-dimensional Complex Optimization Problems

TUO Shou-heng DENG Fang-an
(School of Mathematics and Computer Science, Shaanxi University of Technology, Hanzhong 723000, China)

Abstract This study presented a dynamic self-adaptive harmony search (DSHS) algorithm to solve high-dimensional optimization problems. In the proposed DSHS algorithm, the orthogonal experimental design algorithm was used to initialize population; two new harmony adjustment operators, multi-dimensional dynamic adaptive adjustment operator and one-dimensional tones fine-tuning operator, were integrated to the improvisation scheme. For avoiding the search being trapped in local optimum, an improved band width adjustment algorithm was employed to enhance the disturbance performance. 6 benchmark function experiments show that the proposed algorithm has strong convergence velocity, stabilization and capacity of space exploration on solving high-dimensional complex optimization problems, compared with most other approaches.

Keywords High-dimensional, Dynamic self-adaptive, Harmony search algorithm

1 引言

仿生群体智能优化算法以对其优化问题的数学模型复杂度要求较低、优化过程与初始值无关、搜索速度快等优势,成为求解大规模复杂问题的研究热点。目前典型仿生群体智能优化算法主要有遗传算法(Genetic Algorithm:GA)、微粒子群算法PSO、人工蜂群优化算法ABC、蚁群算法ACO、模拟退火算法SA和差分进化算法DE等。仿生智能优化算法通过模仿生物的进化规则(GA,DE)或生物觅食规则(PSO、ACO、ABC)等来进行优化问题求解,已经得到广泛应用。

和声搜索(Harmony Search:HS)算法是Geem和Kim等[1]提出的一种新型的仿生智能优化算法。HS算法模拟音乐家在音乐创作中不断调整各个乐器或音符,形成美妙动听的和声。同样地,在优化问题求解中,通过逐步调整各个解中的决策变量的值使其向全局最优解靠近。目前,该方法已在组合优化[2]、无约束函数优化[3]等问题中得到了广泛应用,Esmaile Khorram[4]将HS算法应用到热力和电力经济调度问题中,Sinem Kulluk[5]应用HS算法训练神经网络,雍龙泉[6]采用HS算法求解绝对值方程。

HS算法在求解低维优化问题时,求解速度快,且稳定性好。但该算法是逐步淘汰和声库中最差和声向量,当问题的维数和复杂度较高时,往往需要多次迭代求解,因此非常容易早熟而陷入局部搜索,并且难以跳出局部搜索区域。针对该缺点,学者们提出了一些改进算法:改进的和声算法(Improve Harmony Search, HIS)[7]、全局最优和声搜索算法(Global-best Harmony Search, GHS)[8]、自适应全局最优和声搜索算法(Self-adaptive Global-best Harmony Search, SGHS)[9,10]等。IHS、GHS和SGHS算法对HS算法从收敛性和搜索空间探索能力都有一定的提升,但对一些高维多模态复杂问题还是有一些不足,为此,本文在上述算法的基础上提出一种求解高维复杂优化问题的动态自适应和声搜索算法(Dynamic Self-adaptive Harmony Search, DSHS)。

到稿日期:2011-12-07 返修日期:2012-03-05 本文受国家高技术研究发展计划(863计划)(2008AA01A303),国家自然科学基金(81160183),宁夏自然科学基金项目(NZ11105),宁夏卫生厅科研项目(2011033),陕西理工学院"汉水文化"省级重点学科课题(SLGH1226)资助。

拓守恒(1978—),男,讲师,硕士,CCF会员,主要研究方向为进化计算、人工智能、神经网络等;邓方安(1963—),男,博士,教授,主要研究方向为人工智能、粗糙集。

2 和声算法在优化问题中的应用

2.1 优化问题数学模型

高维优化问题可用如下数学模型进行描述：

min $f(X)$，$X=(x_1,x_2,\cdots,x_D)$，Where $X\in S$，$S\subseteq R^D$，x_i ($LB_i\leqslant x_i\leqslant UB_i$，$i=1,2,\cdots,D$)是 X 的决策变量，$i=1,2,\cdots,D$。

2.2 基本和声算法(Harmony Search，HS)

基本和声搜索算法的基本步骤如算法 1。

算法 1 HS 算法

Step 1 设置初始参数。(1)决策变量个数 D。(2) 各决策变量的取值范围[LB,UB]。(3)和声记忆库的大小 HMS。(4) 和声记忆的保留概率 HMCR。(5)音调调节概率 PAR 与调整步长 BW。(6)最大搜索代数 MaxT。

Step 2 在搜索空间内随机初始化和声记忆库 HM，并计算每个和声向量的适应值。

Step 3 即兴创作一个新和声 X_{new}，创作规则如下：

```
For (j=1 to D)
  IF (rand(0,1) < HMCR) then
    ① X_new(j)=X_a(j) where a∈(1,2,…,HMS)
      IF (rand(0,1) < PAR) then
    ② X_new(j)=X_new(j)+r1 * BW, where r1 ∈ (-1,1)
      EndIF
  Else ③X_new(j)=LB_j+rand(0,1) * (UB_j-LB_j)
  Endfor
```

Step 4 IF $f(X_{new})<f(X_{worst})$ then $X_{worst}=X_{new}$

Step 5 如果即兴创作次数达到 MaxT，则结束并输出和声记忆库中最好和声向量，否则转至 Step 3 继续。

下面是和声创作的 3 个基本规则：

①根据和声选择概率 HMCR，从和声记忆库中选择。

②利用声调微调概率 PAR，决定是否对和声进行微调。BW 是声调微调步长。

③在搜索空间中，随机即兴创作新和声音符。

在算法 1 中，*HMCR*，*PAR* 和 *BW* 都是初始时给的一个固定值，这要求必须给出非常合适的一组值，才可能引导算法搜索到最优解，该算法的求解精度不高，并且很容易早熟而停止搜索。对此，IHS 算法对 *PAR* 和 *BW* 进行了改进，使其能够随着迭代次数 t 的增加自动调整。

2.3 IHS(Improve Harmony Search)算法

IHS 算法是对 HS 算法在 Step 3 中的 *PAR* 和 *BW* 进行动态调整，见式(1)和式(2)。

$$PAR(t)=PAR_{\min}+\frac{PAR_{\max}-PAR_{\min}}{\mathrm{MaxT}}\times t \tag{1}$$

$$BW(t)=BW_{\max}\cdot\exp\left(\frac{\ln\left(\frac{BW_{\min}}{BW_{\max}}\right)}{\mathrm{MaxT}}\times t\right) \tag{2}$$

IHS 算法通过式(1)和式(2)在进化过程中自动更新和声调整概率 *PAR* 和调整步长，有效地提高了算法的适应能力。在优化初期 *PAR* 较小，调整步长 *BW* 较大，主要进行全局探索，到后期逐步缩小步长 *BW*，主要通过局部的微调来提高解的精度。

2.4 SGHS 算法

SGHS 算法对 HS 的改进主要是在利用规则①时，从和声记忆库中选择当前全局最优和声 X_{best} 的第 j 维作为 X_{new} (j)，该算法能够明显加快搜索速度，为避免算法早熟，将规则②和规则①位置互换。*SGHS* 算法对微调步长 *BW* 也进行了调整，如式(3)所示。

$$BW(t)=\begin{cases}BW_{\max}-\dfrac{BW_{\max}-BW_{\min}}{\mathrm{MaxT}}\times 2t, & t\leqslant \mathrm{MaxT}/2\\ BW_{\min}, & t>\mathrm{MaxT}/2\end{cases} \tag{3}$$

3 本文动态自适应和声搜索算法

算法 IHS，GHS 和 SGHS 对 HS 算法的改进基本都是通过调整 *PAR* 与 *BW*，使其能够随着迭代次数的增加逐步调整 *PAR* 与步长 *BW*，以避免算法过早成熟。但通过实验发现，随着问题维数的增加，这些算法收敛速率和求解精度都明显降低，且还是很容易陷入局部搜索。这是由于它们在创作新和声向量 X_{new} 时，其每一维都需要依据和声创作的 3 个基本规则重新生成，并且只有在 X_{new} 好于目前和声记忆库 HM 中最差和声向量 X_{worst} 时，才会被装入 HM 中。这样对于一个高维优化问题，在优化后期存在如下 2 个问题：

(1)如果 HM 中全局最优和声向量 X_{best} 的部分维陷入局部搜索后，和声记忆库 HM 中的其它和声向量很容易在相应的维上被“吸引”到该维的局部搜索区域。

(2)在利用新和声产生规则时，高维决策向量相互冲突，在部分维上得到了改进，另外部分维可能反而退化，此消彼长，难以在短时内消除冲突，导致收敛速度降低。

针对上述问题，本文提出多维动态自适应新和声调整算法和单维和声微调算法产生新和声，算法思想见 3.1 节和3.2 节。

3.1 多维动态自适应和声调整算法

算法 2 多维动态自适应和声调整算法

```
For (j=1 to D)
  rnd(j)=SP
  IF rnd(j)==1
    IF (rand(0,1) < HMCR) then
      X_new(j)=X_r1(j)+rand*[X_r2(j)-X_r3(j)],
      (where r1,r2,r3∈(1,2,…,HMS))
      IF (rand(0,1) < PAR) then %和声微调
        X_new(j)=X_new(j)+rand(0,1)*BW_j(t),
      EndIF
      Else
    X_new(j)=LB_j+rand * (UB_j-LB_j)
    EndIF
  Else X_new(j)=X_ra(j), where ra∈(1,2,…,HMS)
EndFor
```

算法 2 中，

$$SP=Round(rand(0,1)\times ep) \tag{4}$$

式中，$ep=\exp\left(-\dfrac{(x-u)^2}{2\delta^2}\right)$，$x=1-\dfrac{t}{\mathrm{MaxT}}$，$u=1.2$，$\delta=1.3$。

图 1 中描述了 SP 的变化过程，高斯函数 ep 根据 t 来逐步调整新和声产生策略，在优化初期，为了保证较强的全局探索能力，需要以较大的概率进行差分变异重组、微调和随机扰动，而从和声记忆库选择的概率较小。到后期，为了提高精度，主要通过和声记忆库选择进行调整。

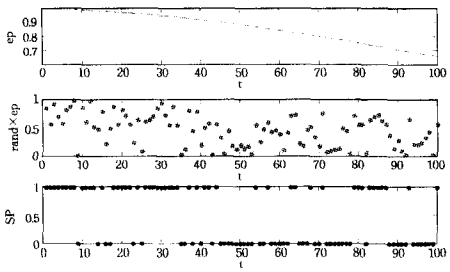

图 1　SP 变化过程

3.2　单维和声微调算法

由于高维优化问题很容易在部分维度上陷入局部搜索，为了避免该问题，本文提出单维和声微调算法，该算法产生的新和声仅仅是对和声记忆库中的某一和声向量的一个决策向量进行调整，使其具有更高的改进概率。算法描述如下：

算法 3　单维和声微调算法

$X_{new}=X_{rb}$, where rb∈(1,2,…,HMS)

d=fix(D * rand(0,1))+1；从 1～D 中随机选取第 d 维

IF rand(0,1)<PAR, $X_{new}(d)=X_{new}(d)+N(0,BW_d(t))$

Else xa= LB_d+rand(0,1) * (UB_d-LB_d)

$X_{new}(d)=X_{new}(d)+$rand * (xa−$X_b(d)$), where b∈(1,2,…,HMS)

EndIF

在算法 2 和算法 3 中，PAR 采用式(1)进行计算，BW 采用下列公式。

$$BW=\exp\left[-\frac{(x-1)^2}{2\times 0.3^2}\right]\times(BW_{max}-BW_{min})+BW_{min} \quad (5)$$

式中，$x=1-\frac{t}{MaxT}$。

由图 2 可以看出，式(2)中，BW 的值在起始阶段下降很快，当下降到接近 BW_{min} 时逐步平稳，也就是说，式(2)很快会进入较小的微调阶段，对于一个大空间、大规模高维复杂问题，算法的种群扰动能力下降太快，很容易导致搜索速度下降。另外，当搜索陷入局部搜索后，由于扰动能力较差，导致搜索难以跳出局部区域。而改进后的式(5)中，BW 的值下降平缓，能够保持较强的局部扰动能力，并且可以将 BW_{min} 值设置得很小来提高最优解的精度，且不会使得搜索较早陷入局部微调阶段，如图 3 所示。

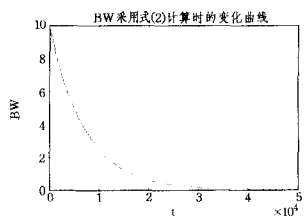

图 2　IHS 算法中，BW 的变化曲线

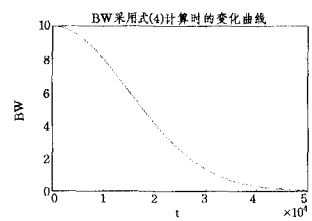

图 3　本文算法 2 和算法 3 中 BW 的变化曲线

3.3　基于正交试验设计的动态自适应和声搜索算法

本文结合算法 2 和算法 3 提出一种求解高维多峰值问题的动态自适应和声搜索算法（Dynamic Self-Adaptive Harmony Search，DSHS）。具体描述如下：

算法 4　动态自适应和声搜索算法（Dynamic Self-Adaptive Harmony Search，DSHS）

Step 1　设置初始参数 HMS，HMCR，PAR，BW，LIMIT and MaxGen.

Step 2　为了保证初始解分布均匀性，采用正交试验设计算法[11]初始化和声记忆库 HM，并计算每个和声向量的适应值。

Step 3　即兴创作一个新和声 X_{new}，创作过程如下：

IF r1<MDP 采用算法 2 创作新和声。

Else 利用算法 3 产生新和声。

Step 4　IF $f(X_{new})<f(X_{worst})$ then

$X_{worst}=X_{new}$, limit=0

Else limit=limit+1

If limit>LIMIT then

X_{worst}=LB+ Rand(1,D) * (UB−LB), limit=0

EndIF

EndIF

Step 5　如果即兴创作次数达到 MaxT，则结束并输出和声记忆库中最好和声向量，否则转至 Step 3 继续。

在算法 4 中，MDP 是多维动态自适应和声调整算法选择概率，LIMIT 是和声记忆库更新阈值，如果长时间内，和声记忆库没有得到更新，而需要丰富和声记忆库中和声音调的多样性来增强算法的空间探索能力，因此，随机产生新和声来替换和声记忆库中的最差和声向量。

4　仿真分析

为了评估本文算法（DSHS）的性能，选取了 6 个有标准的 benchmark 函数（f1-Sphere，f2-Schwefel，f3-Rosenbrock，f4-Rastrigin，f5-Ackley，f6-Griewangk）[9]进行测试，实验测试结果分别和标准的和声算法 HS、改进的和声算法 IHS 和自适应全局和声搜索算法 SGHS 进行了比较和分析。

4.1　运行环境与参数设置

在测试中，微机硬件环境为华硕笔记本电脑，13450CPU，2GB 内存，在 MATLAB 2009(a)软件平台上进行编程实现。各种算法参数设置如表 1 所列。

表 1　各种算法测试参数设置比较

算法	HMS	HMCR	PAR	BW	MaxT	其它
HS	5	0.9	0.3	0.01	5E+4	
IHS	5	0.9	PAR_{min}=0.01 PAR_{max}=0.99	BW_{max}=(UB−LB)/20 BW_{min}=0.0001	5E+4	
SGHS	5	0.98	PAR_{min}=0.01 PAR_{max}=0.99	BW_{max}=(UB−LB)/10 BW_{min}=0.0005	5E+4	
DSHS	5	0.85	PAR_{min}=0.03 PAR_{max}=0.9	BW_{max}=(UB−LB)/10 BW_{min}=0.00001	5E+4	MDP=0.35 LIMIT=100

4.2　实验运行结果

在本文算法的所有测试函数中，分别设置函数的维数 D=30，D=100，使程序各自独立运行 30 次，记录平均最优解（Mean），并计算出标准差（Std），统计结果如表 2 和表 3 所列。在表 2 和表 3 中，将本文算法 DSHS 和 HS、IHS 和 SGHS 进行了比较。表 4 中，当算法 DSHS 在 D=200 维时，给出算法结束条件（目标最优解 Gbest 或最高函数评价次数 FETs），独立运行 30 次，统计函数平均评价次数（FETs）、平均最优解和优化用时的平均值。

表 2　测试函数的优化结果比较(D=30)

函数	函数最小值	平均值 标准差	HS	IHS	SGHS	DSHS
f1	0	Mean	0.000187	0.000712	0.0	0.0
		Std	0.000032	0.000644	0.0	0.0
f2	0	Mean	0.17152	1.097325	0.000102	8.3877e−6
		Std	0.072851	0.181253	0.000017	3.7556e−6
f3	0	Mean	160.29	94.3232	30.9297	14.6592
		Std	66.69	59.8473	41.0549	18.3709
f4	0	Mean	0.497584467	2.7176085	2.198725e−05	1.1645e−08
		Std	0.79600679	1.0434473	4.939711e−06	3.7609e−11
f5	0	Mean	1.130004	1.893394	0.484445	1.84385e−6
		Std	0.407044	0.314610	0.356729	6.6149e−7
f6	0	Mean	1.119266	1.120992	0.050467	0.00617568
		Std	0.041207	0.040887	0.035419	0.01171294

表 3　测试函数的优化结果比较(D=100)

函数	函数最小值	平均值 标准差	HS	IHS	SGHS	DSHS
f1	0	Mean	8.683062	8.840449	0.000002	2.64155e−7
		Std	0.775134	0.762496	0.000003	7.39945e−8
f2	0	Mean	82.926284	82.548978	0.017581	0.00304201
		Std	6.717904	6.341707	0.021205	4.56293e−4
f3	0	Mean	18687.218474	23820.9159765	621.749360	126.15183815
		Std	3051.8394862	2981.26808037	583.889593	51.57029206
f4	0	Mean	319.02846923	210.641788076	12.353767	0.273618
		Std	26.016889071	13.62771747312	2.635607	0.2206172
f5	0	Mean	13.857189	13.801383	0.31599	4.63419e−6
		Std	0.284945	0.530388	0.0410263	6.68538e−8
f6	0	Mean	128.91464	110.405944	0.027932	5.74537e−5
		Std	15.012159	8.48646620	0.009209	9.95127e−5

从表 2 与表 3 可以发现，DSHS 算法对于函数 $f1-f6$，在 D=30 和 100 时，除 Rosenbrock($f3$)之外，基本都获得了最优解，其实 $f3$ 也已经接近最优解，只是精度不够高。在函数评价次数相同时，DSHS 算法对所测函数 30 次独立运行的平均值最优解(mean)和标准差(std)都明显优于 HS、IHS 和 SGHS 算法。

表 4 中对维度在 200 维时，通过设置算法优化结束条件(最高函数评价次数，最优目标值)来统计本文算法 DSHS 的运行代价，结果显示，除了 Rosenbrock($f3$)函数之外，基本都能够在 1E+005 次函数评价以内获得满足条件的最优解。Rosenbrock($f3$)函数也能够在 150000 次新和声的创作与调节(函数评价)中，获得近似最优解。

表 4　D=200 时 DSHS 算法对 6 个函数的测试结果

函数	结束条件	函数评价次数(次)	最优解	平均用时(s)
f1	Gbest<0.01	37303	0.0096	61.4630
f2	Gbest<0.1	101708	0.0994	127.0982
f3	Gbest<100 FETs=1.5E+5	1.5E+5	159.2224	178.8644
f4	Gbest <1	42161	0.9830	68.6567
f5	Gbest <0.1	53584	0.0926	79.2665
f6	Gbest <0.1	62342	0.0932	103.5359

4.3　实验结果分析

在测试函数 $f1-f6$ 中，$f3$ 与 $f4$ 是放生智能优化算法最难以获得全局最优解的 2 个函数，下面通过 $f3$ 与 $f4$ 的收敛曲线图与最优解分布图比较本文算法 DSHS 的有效性。

图 4 与图 5 分别描述了 Rastrigin 与 Rosenbrock 函数在 D=100 时的优化过程收敛曲线。从图中的收敛曲线来看，本文算法从开始到结束都优于其它 3 种算法(HS、IHS、SGHS)，说明 DSHS 算法在求解高维优化问题时具有优势。另外从收敛曲线的走势来看，本文 DSHS 算法曲线一直处于下降状态，没有出现平缓的水平曲线，这说明算法 DSHS 能够保持群体的积极活跃性，并不断地向全局最优解靠近，种群不会过早成熟而陷入停滞状态，能够获得高精度的全局最优解。

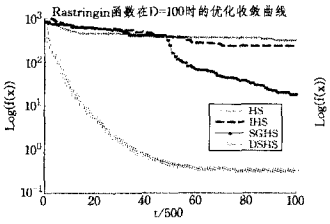

图 4　Rastrigin 函数在 D=100 时的优化收敛曲线

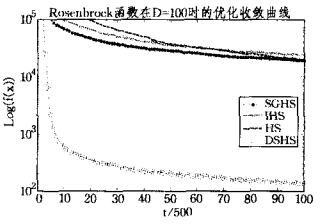

图 5　Rosebrock 函数在 D=100 时的优化收敛曲线

4.3.1　稳定性分析

为了测试本文算法的稳定性，将 30 次测试结果利用盒图的形式进行描述(见图 6、图 7)，观察其运行结果分布。

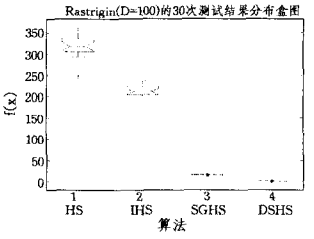

图 6　Rastrigin(D=100)测试结果分布图

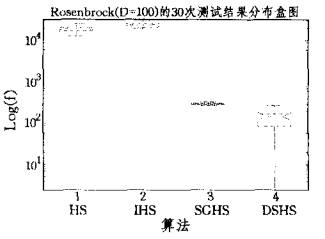

图 7　Rosenbrock(D=100)测试结果分布图

由图 6、图 7 可以看出，算法 DSHS 的 30 次运行结果分布基本都比较集中，没有很广大跳跃。DSHS 算法的最差结果都优于算法 HS、IHS 和 SGHS，体现出算法 DSHS 具有良好的稳定性。

结束语　文中对提出的动态自适应和声算法 DSHS 采用 6 个高维 Benchmark 函数进行验证，通过实验结果分析了算法的有效性；利用优化过程收敛曲线分析了算法的收敛性与全局性；采用盒图描述了 30 次运行结果的分布，进而分析算法的稳定性和有效性。结果表明，改进后的算法不论从全局搜索能力还是收敛性、稳定性方面都较算法 HS、IHS 和 SGHS 有所提升。

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] 高立群，依玉峰，郑平，等. 和声搜索算法在求解最短路径问题中的应用[J]. 东北大学学报：自然科学版，2011(6)：769-772

[3] 陈莹珍，高岳林. 混沌的自适应和声搜索算法[J]. 太原理工大学学报，2011(2)：141-144

[4] Khorram E, Jaberipour N. Harmony search algorithm for solving combined heat and power economic dispatch problems[J]. Energy Conversion and Management, 2011, 152: 1550-1554

(下转第 246 页)

(27)(25)∧(26)⇒$A_{42}=d_5$

(28)(1)∧(23)⇒$A_{23}=a_1$

(29)(5)∧(23)⇒$A_{54}=e_5$

(30)$A(2)\wedge(8)\wedge(28)\Rightarrow(A_{22}=a_3)\vee(A_{24}=a_3)\vee(A_{25}=a_3)$

(31)(29)∧(3)⇒$A_{24}\neq a_3$

(32)(30)∧(31)⇒$(A_{22}=a_3)\vee(A_{25}=a_3)$

(33)(3)∧(32)⇒$((A_{22}=a_3)\wedge(A_{52}=e_1))\vee((A_{25}=a_3)\wedge(A_{55}=e_1))$

(34)$A(5)\wedge(9)\wedge(29)\Rightarrow(A_{51}=e_3)\vee(A_{52}=e_3)\vee(A_{55}=e_3)$

(35)(25)∧(12)⇒$A_{51}\neq e_3$

(36)(34)∧(35)⇒$(A_{52}=e_3)\vee(A_{55}=e_3)$

(37)(12)∧(36)⇒$((A_{32}=c_3)\wedge(A_{52}=e_3))\vee((A_{35}=c_3)\wedge(A_{55}=e_3))$

(38)(33)∧(37)⇒$A_{52}\in\{e_1,e_3\}\wedge A_{55}\in\{e_1,e_3\}\Rightarrow\{A_{52},A_{55}\}=\{e_1,e_3\}$

(39)$A(5)\wedge(9)\wedge(29)\wedge(38)\Rightarrow\{A_{52},A_{53},A_{54},A_{55}\}=\{e_1,e_2,e_3,e_5\}\Rightarrow A_{51}=e_4$

(40)$A(7)\wedge(15)\Rightarrow\forall j.(A_{5j}=e_4\rightarrow A_{3(j-1)}=c_2\vee A_{3(j+1)}=c_2)$

(41)(39)∧(40)⇒$A_{32}=c_2$

(42)(37)∧(41)⇒$(A_{35}=c_3)\wedge(A_{55}=e_3)$

(43)$A(5)\wedge(9)\wedge(29)\wedge(39)\wedge(42)\Rightarrow\{A_{51},A_{53},A_{54},A_{55}\}=\{e_2,e_3,e_4,e_5\}\Rightarrow A_{52}=e_1$

(44)(3)∧(43)⇒$A_{22}=a_3$

(45)$A(2)\wedge(8)\wedge(28)\wedge(44)\Rightarrow a_2\in A_2-\{A_{21},A_{22},A_{23}\}\Rightarrow(A_{24}=a_2)\vee(A_{24}=a_2)$

(46)(42)∧(13)⇒$A_{25}\neq a_2$

(47)$A(2)\wedge(45)\wedge(46)\Rightarrow A_{24}=a_2$

(48)$A(2)\wedge(8)\wedge(28)\wedge(44)\wedge(47)\Rightarrow A_{25}=a_5$

(49)(2)∧(48)⇒$A_{45}=d_1$

(50)(13)∧(47)⇒$A_{34}=c_4$

(51)$A(3)\wedge(25)\wedge(41)\wedge(42)\wedge(50)\Rightarrow c_5\in A_3-\{A_{31},A_{32},A_{34},A_{35}\}\Rightarrow A_{33}=c_5$

(52)(6)∧(51)⇒$A_{43}=d_3$

(53)(10)∧(41)⇒$(A_{41}=d_2)\vee(A_{43}=d_2)$

(54)(52)∧(53)⇒$A_{41}=d_2$

(55)$A(4)\wedge(27)\wedge(49)\wedge(52)\wedge(54)\Rightarrow d_4\in A_4-\{A_{41},A_{42},A_{43},A_{45}\}\Rightarrow A_{44}=d_4$

由(47)、(55)知,命题成立。

由定理1的证明过程可知推论1成立。

推论1 形式系统的Γ定理组成结果矩阵

$$A_{5\times5}\triangleq\begin{pmatrix}b_2&b_3&b_1&b_5&b_4\\a_4&a_3&a_1&a_2&a_5\\c_1&c_2&c_5&c_4&c_3\\d_2&d_5&d_3&d_4&d_1\\e_4&e_1&e_2&e_5&e_3\end{pmatrix}$$

比较表1和结果矩阵$A_{5\times5}$,得出谜题的解:德国人养鱼。如表2所列。

表2 爱因斯坦谜题的唯一解

房间位置	1	2	3	4	5
颜色	黄	蓝	红	绿	白
国籍	挪威	丹麦	英国	**德国**	瑞典
香烟	Dunhill	Blends	Pall mall	Prince	Blue master
宠物	猫	马	鸟	**鱼**	狗
饮料	水	茶	牛奶	咖啡	啤酒

结束语 给出了爱因斯坦谜题的形式化描述,构造了一个形式系统Γ,在此基础上给出求解谜题的推理方法,从而得到了唯一解。如果把Γ嵌入著名的定理证明器PVS[5,6],只需把Γ的逻辑语言描述改为PVS程序语言描述,即可得到PVS程序——执行程序,就可以半人工、半自动地求解谜题。而使用PVS的好处是工具可以自动化简一部分证明过程,从而实现半自动推理。基于SAT的方法[2]定义了125个原子命题,因而求解该谜题在最坏情况下需时$O(2^{125})$,在事先不知道解的情况下,难以在有效时间内找到谜题的解。新的基于一阶逻辑形式推理的方法避免了蛮力搜索巨大状态空间所需时间,同时给出了符合人脑智能的推理过程,这是我们方法的优势所在。

参考文献

[1] Yeomans J. Solving Einstein's riddle using spreadsheet optimization [J]. Informs transaction on education, 2003, 3(2): 55-63

[2] 田聪,段振华,王小兵. Einstein谜的SAT求解[J]. 计算机科学, 2010, 37(5): 184-186

[3] David M. Model theory: an introduction [M]. Springer, 2002

[4] 陆钟万. 面向计算机科学的数理逻辑[M]. 北京:科学出版社, 2002

[5] PVS Specification and Verification System [EB/OL]. http://pvs.csl.sri.com/

[6] Newborn M. Automated Theorem Proving: Theory and Practice [M]. Springer-Verlag, 2001

(上接第243页)

[5] Kulluk S, Ozbakir L, Baykasoglu A. Self-adaptive global best harmony search algorithm for training neural networks[J]. Procedia Computer Science, 2011(3): 282-286

[6] 雍龙泉. 基于凝聚函数的和声搜索算法求解绝对值方程[J]. 计算机应用研究, 2011(8): 2922-2926

[7] Mahdavi M, Fesanghary M, Damangir E. An improved harmony search algorithm for solving optimization problems[J]. Applied Mathematics and Computation, 2007, 188(2): 1567-157

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

[9] Pan Quan-ke, Suganthan N P N, et al. A self-adaptive global best harmony search algorithm for continuous optimization problems[J]. Applied Mathematics and Computation, 2010, 216: 830-848

[10] Mahamed G H O, Mehrdad M. Global-best harmony search[J]. Applied Mathematics and Computation, 2008, 198(2): 634-656

[11] 拓守恒,汪文勇. 求解高维多模优化问题的正交小生境自适应差分演化算法[J]. 计算机应用, 2011, 31(4): 1094-1098