第 37 卷 第 21 期
Vol.37 No.21

计 算 机 工 程
Computer Engineering

2011 年 11 月
November 2011

• 人工智能及识别技术 • 文章编号：1000—3428(2011)21—0170—03 文献标识码：A 中图分类号：TP18，O157

用于数独求解的几何粒子群优化算法设计

肖华勇，马 雷，温金环

(西北工业大学理学院，西安 710129)

摘 要：针对只有唯一解的数独问题(即标准数独)，利用改进的几何粒子群优化算法进行求解，将几何粒子群优化算法应用到数独中，解决数独求解过程中存在的局部最优解问题。通过实例讨论求解过程中最佳参数的选择，并得出较理想的结果。实验结果表明，该方法能够有效解决数独问题。

关键词：数独；唯一解；几何粒子群优化；适应度函数

Design of Geometric Particle Swarm Optimization Algorithm for Sudoku Solving

XIAO Hua-yong, MA Lei, WEN Jin-huan

(School of Science, Northwestern Polytechnical University, Xi'an 710129, China)

【Abstract】This paper proposes a new method to solve standard sudoku with unique solution, applying the improved Geometric Particle Swarm Optimization(GPSO) algorithm. It includes specific details of applying the GPSO algorithm to sudoku and improved techniques to some problems in the process of solving sudoku. It discusses the choice of the best parameter in the process of solution through the example, and gives the ideal result. Experimental results show that this method can effectively solve the sudoku problem.

【Key words】sudoku; unique solution; Geometric Particle Swarm Optimization(GPSO); fitness function

DOI: 10.3969/j.issn.1000-3428.2011.21.058

1 概述

数独(sudoku)源于 18 世纪瑞士数学家欧拉发明的拉丁方阵，后经日本人的改良而得以流行。棋盘由一个 3×3 的九宫格组成，每个格子又分成一个小九宫格，共九九八十一个小格子。游戏规则简单，就是将 1~9 这 9 个数字填入该棋盘，要求每个数字在每一行每一列或者每一小九宫格都只出现一次。经过近十多年的发展，现在已经有了很多解决数独问题的方法以及用计算机解决的人工智能算法[1]，国内外还有许多比较成熟的数独网站，里面提供了很多供在线解题者使用的技巧[2]。本文针对唯一解的数独问题，给出一种利用改进几何粒子群优化算法求解的方法。

2 几何粒子群优化算法

几何粒子群优化算法(Geometric Particle Swarm Optimization, GPSO)是在粒子群优化算法的基础上提出来的。粒子群优化(PSO)算法[3]是一种与遗传算法(Genetic Algorithm, GA)类似的进化算法，也是一种基于叠代的优化工具，系统初始化为一组随机解，通过某种方式叠代寻找最优解。近年来发展较快，在许多典型理论问题上应用广泛，如组合优化、约束优化、多目标优化、动态系统优化等。

规范的 PSO 算法程序的伪代码如下：

算法 1

1:for all 粒子 i do
2: 初始化粒子位置 $x_i \in U(a,b)$和速度$v_i = 0$
3:end for
4:while 目标最优值没有找到 do
5: for all 粒子 i do
6: 将目前为止的个体历史最优位置赋予 X_i
7: 将全部种群的历史最优位置赋予 G
8: end for
9: for all 粒子 i do
10: 根据式(1)更新速度：

$$v_i(t+1) = wv_i(t) + \phi_1 R_1\left(G(t) - x_i(t)\right) + \phi_2 R_2\left(X_i(t) - x_i(t)\right) \quad (1)$$

11: 根据式(2)更新位置

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

12: end for
13: end while

GPSO 算法其实是粒子群优化算法的推广，它将其推广到组合空间。这 2 个算法的主要区别在于粒子群优化算法是依靠速度变化不断向最优解靠近的，而在 GPSO 算法中没有速度，它是依靠单个粒子与其历史最优值以及全局最优值的交叉(凸组合)达到不断向最优解靠近的目的。在 GPSO 算法中也有 ω, ϕ_1, ϕ_2，都为正数且和为 1。

算法程序的伪代码如下：

算法 2

1:for all 粒子 i do
2: 在搜索空间中随机初始化粒子位置 x_i
3:end for
4:while 停止条件没有达到 do
5: for all 粒子 i do
6: 将目前为止的个体历史最优位置赋予 $\tilde{x}_i$
7: 将全部种群的历史最优位置赋予 $\tilde{g}$
8: end for
9: for all 粒子 i do
10: 通过式(3)随机凸组合更新位置：

作者简介：肖华勇(1970－)，男，副教授、博士，主研方向：几何粒子群优化算法；马 雷，硕士；温金环，讲师、博士

收稿日期：2011-04-12 **E-mail：**malei19850315@163.com

$$x_i = GX\left((x_i,\omega),(\quad,\phi_1),(\tilde{x}_i,\phi_2)\right) \quad (3)$$

11:　　变异 x_i
12:　end for
13: end while

3　几何粒子群优化算法的研究及改进

数独游戏是满足以下 4 个条件的约束问题[4]:

(1)每一单元(行、列、小九宫格)都是由 1~9 这几个固定的数字组成。

(2)每一行都是 1~9 的一个序列。

(3)每一列都是 1~9 的一个序列。

(4)每一小九宫格都是 1~9 的一个序列。

上面的 4 个约束，可以分为 2 类：(1)硬约束，不管该数独棋盘是单个粒子解、局部最优解、还是全局最优解，所有的数独棋盘都是必须满足的；(2)软约束，只满足部分约束并且约束满足的程度可以对该数独解的优劣进行度量。

依据解决数独游戏的算法思路，上面的 4 个条件可以分为下面的 2 类：

(1)硬约束：固定的位置和每一行都是 1~9 的一个序列。

(2)软约束:每一列和每一小九宫格都是 1~9 的一个序列。

GPSO 算法的思想是首先产生若干粒子以形成种群，由于所有粒子棋盘都必须满足硬约束，因此产生的初始种群也满足。接下来通过软约束满足的程度来度量数独解的好坏，而软约束满足的程度是用适应度函数来计算的。

本文采用的适应度函数为：数独棋盘上的每一行中不重复数字的个数(重复数字按一个数字计算)，每一列中不重复数字的个数(重复数字按一个数字计算)以及每一小九宫格中不重复数字的个数(重复数字按一个数字计算)之和。通过比较该函数的值来判断此时的解是否更优于前面的解。当每一行、每一列以及每一小九宫格都含有 1~9 这 9 个数字时，适应度函数为 9×9+9×9+9×9=243，此时的解为最优解且适应度函数达到最大值。当每一行、每一列以及每一小九宫格至少包含一个数字时，此时的适应度函数达到最小值，即 9×1+9×1+ 9×1=27。

产生初始种群以后，在适应度函数的引导下，通过交叉和变异一步一步向最优解逼近。交叉操作是在单个粒子解、历史最优解以及全局最优解之间进行的，其方法是随机产生与 3 个父类同样长度的交叉掩码，然后根据交叉掩码逐行进行交叉，具体的操作过程如下(假如交叉掩码和 3 个父类的某一行)：

交叉掩码：1 2 2 3 1 3 2

父类 P1：1 2 3 4 5 6 7

父类 P2：3 5 1 4 2 7 6

父类 P3：3 2 1 4 5 7 6

新子类 O：1 5 3 4 2 7 6

交叉掩码是随机产生的，长度与父代的长度相同。掩码中的 1、2、3 出现的频率代表了重组中各个父代的权重 w_a、w_b、w_c。掩码的数字表示在具体位置上选择那个父代，使其余 2 个父代在该位置上与之相同。在一个父代中，一个位置上的数字是通过与该父代中另一个位置上的数字交换而改变的。上例重组的过程如下：掩码按照从左到右的顺序，在第 1 个位置上的掩码是 1，意味着在第 1 个位置上，父代 1 不变，父代 2 和父代 3 要通过变换与父代 1 相同，在父代 2 中交换数字 3 和数字 1，在父代 3 中交换数字 3 和数字 1。重组在更新的父代中继续进行，父代 1 没有改变，当前的父代 2 和父代 3 是原来的父代 2 和父代 3 各自变换后的结果。在第 2 个位置上的掩码是 2，意味着在第 2 个位置上，父代 2 不变，父代 1 和父代 3 要通过变换与父代 2 相同，同第 1 个位置，在父代 1 中交换数字 2 和数字 5，在父代 3 中交换数字 2 和数字 5，父代 1 和父代 3 的第 2 个位置都变为 5。继续这样重组第 3 个位置，一直到最后一个位置。在 3 个父代都相同的位置上，不管掩码是多少，都不会引起变换，所以它们自然而然不会涉及到交换中。3 个父代交叉融合形成的排列就是新子代。在此过程中，可以通过交叉掩码交换得到新交叉掩码排序，再用新交叉掩码进行交叉操作。这种交叉重组可以很容易地应用于任意数量的父代。

当某个解的适应度为 243 时，该解为数独的最优解。若解的适应度没有达到 243，但是比较接近 243(例如大于 238)时，为了得到最优解，还必须再进行交叉变异，这就很大程度上降低了算法的效率。其实此时的解在交换距离下已经很逼近最优解，只需要根据数独游戏的规则选取某些位置的数字进行交换即可,在经过较少次数的交换后直接得到最优解。这样就避免了交叉变异，从而提高了算法的效率。但是，若经过多次交换后仍没有得到最优解，则此解可能就是数独的局部最优解，应该设法跳出局部最优解，以便程序重新搜索解空间。

为了更好地理解该问题，本文引入如下的定义及定理[5]。

定义 1　一行(或列或九宫格)中不同元素的个数叫做该行(或列或九宫格)的秩。

定义 2　若一行(或列或九宫格)的秩为 9，那么称该行(或列或九宫格)满秩。

算法中的粒子群都满足硬约束，故该问题中所有解都是行满秩的，而列和九宫格不一定是满秩的，只有当解为最优解时，所有的行、列和九宫格才是满秩的。

定理 1　若存在一列不满秩，则必存在另一列也不满秩。

证明：如果有一列不满秩(不妨设秩为 8)，那么在该列中必定缺少某个数字 i。所以，在其余的 8 列中有 9 个数字 i，由鸽巢原理可知，必定有一列中存在至少 2 个 i。

定理 2　若一个解中有一个九宫格不满秩，那么这个解的适应度值最大为 239。

证明：设已知有一个九宫格 A 不满秩(不妨设秩为 8)，那么该九宫格中必定缺少某个数字 i。因为硬约束条件，同行的 3 个九宫格中必然有 3 个数字 i,那么在与该九宫格中同行的另外 2 个九宫格中有 3 个 i，从而由鸽巢原理可知，必然有另一个九宫格 B 也不满秩，其至少有 2 个 i，此时的适应度最大为 243−2=241，分 2 种情况讨论：

(1)若与九宫格 A 同列的 2 个九宫格满秩，由于九宫格 A 必有两相同数字 j，因此在九宫格 A 所在的 3 列中有 4 个 j，从而由鸽巢原理可知必定存在一列有 2 个 j，那么这列不满秩。由定理 1 可知，还存在另一列不满秩。所以，此时适应度最大为 241−2=239。

(2)若与九宫格 A 同列的 2 个九宫格有一个不满秩，那么存在另一同行的九宫格也不满秩，所以，此时适应度最大为 241−2=239。

定理 3　如果一个解的适应度为 241，那么必定有且仅有两列不满秩。

证明：由定理 2 可知，该解的九宫格都满秩。由定理 1 可知，该解有两列不满秩。

由定理 2 和定理 3 可知，当适应度为 241 时，可通过尝

试交换不满秩的两列中的元素来得到最优解。如图 1 所示，这是在几何粒子群优化算法求解数独时得到的几组解。在求解过程中得到解 a，虽然它的适应度为 241，但是不能通过简单的交换不满秩的两列——第 7 列和第 9 列中的元素来得到最优解，所以，解 a 属于局部最优解。反观解 b，它的适应度也为 241，但通过交换第 4 列的第 4 个数字 1 和第 6 列的第 4 个数字 7，就直接得到最优解。所以，在适应度为 241 时，可以尝试交换不满秩的两列中的元素来得到最优解，这样就有效减少搜索。事实上就图 1 中的数独而言，在 20 次成功得到最优解的运算中，有 11 次是在适应度为 241 时，通过交换不满秩的两列中的元素来得到的。

5	3			7				
6			1	9	5			
	9	8						
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

(a)原始数独

5	3	2	6	7	8	9	1	4
6	7	4	1	9	5	3	2	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	2	4	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

(b)适应度为 241 的解 a

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	1	6	7	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

(c)适应度为 241 的解 b

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

(d)真实解

图 1 适应度相同时的不同解比较

4 用于数独求解的几何粒子群优化算法设计

4.1 几何粒子群优化算法求解数独步骤

几何粒子群优化算法求解数独的步骤如下：

第 1 步 读入数据：读入原数独文件，数独中的空位用数字 0 表示。

第 2 步 构造初始粒子：针对每一个粒子的每一行随机生成从数字 1 到 9 的排列，将数独该行中原有给定的数字从排列中剔除，之后将剩余的排列依次填入数独中。这样就在满足给定数字位置不变的情况下，满足了行排列的约束。

第 3 步 适应度函数：计算粒子的行、列和九宫格秩的总和，当适应度函数值为 243 时，该粒子就是数独的最优解。

第 4 步 几何交叉：将粒子的当前值、个体历史最优值、种群历史最优值视为 3 个父代，根据 3 个父代的权重随机产生掩码，然后进行几何交叉，几何交叉的结果便是下一代粒子。

第 5 步 几何变异：对于每个粒子的每一行，首先产生 0~1 的随机数字，根据变异频率来决定是否发生变异。当发生变异时，随机选取粒子每行中非固定的 2 个数字进行交换。

第 6 步 交换调整：根据第 3 节的分析改进算法，当粒子不是数独的最优解，但是比较接近时，可以通过交换其中不满秩的两列中的数字来得到最优解，若不能交换则选择其他的不满秩的两列，交换过的数字不再交换，当这些重复数字都没法交换时，跳出整个交换过程。

几何粒子群优化算法流程如图 2 所示。

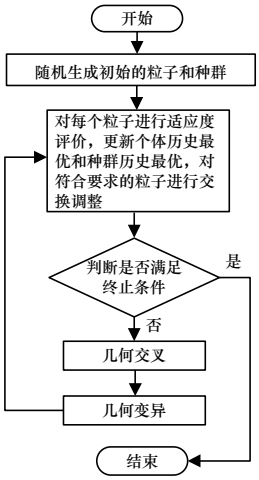


图 2 几何粒子群优化算法流程

4.2 求解结果与分析

依据以上算法的思想，用 C++进行编程，通过求解不同难度的数独问题，充分理解参数变化与算法性能之间的关系。影响该算法性能的主要参数变量有 6 个：几何交叉中的 3 个父代的权重 $w_a, w_b, w_c > 0$，其中，w_a 为当前粒子值的权重；w_b 为个体历史最优值的权重；w_c 为种群历史最优值的权重。它们满足 $w_a + w_b + w_c = 1$，变异频率 Mutation，种群大小 Population 和进化次数 Generation。

将 w_a 和 w_b 从 0.0、0.2、0.4、0.6、0.8 和 1.0 取值搭配组合，而一旦前 2 个权重取定了，第 3 个权重也就定了。变异频率分别取 0.0、0.3、0.7。种群大小 Population 分别取 50、200、500。进化次数 Generation 固定为 2 000 代。针对图 1 中的数独，通过大量实验得到如表 1~表 3 的结果。其中，表 1~表 3 为种群大小为 200 时在 50 次运算中的最佳适应度平均值。

表 1 变异频率为 0 时的最佳适应度平均值

w_b	w_a					
	0.0	0.2	0.4	0.6	0.8	1.0
1.0	215	–	–	–	–	–
0.8	237	236	–	–	–	–
0.6	230	237	237	–	–	–
0.4	226	227	237	238	–	–
0.2	225	225	229	230	240	–
0.0	209	223	226	228	228	211

表 2 变异频率为 0.3 时的最佳适应度平均值

w_b	w_a					
	0.0	0.2	0.4	0.6	0.8	1.0
1.0	238	–	–	–	–	–
0.8	238	–	–	–	–	–
0.6	239	–	240	–	–	–
0.4	239	238	241	240	–	–
0.2	237	240	240	239	239	–
0.0	213	230	233	231	230	219

表 3 变异频率为 0.7 时的最佳适应度平均值

w_b	w_a					
	0.0	0.2	0.4	0.6	0.8	1.0
1.0	232	–	–	–	–	–
0.8	238	–	–	–	–	–
0.6	231	–	242	–	–	–
0.4	228	236	241	237	–	–
0.2	224	235	237	234	230	–
0.0	214	238	233	228	225	215

实验表明，当 $w_a \in (0.35, 0.45)$，$w_b \in (0.3, 0.5)$，$w_c \in (0.02, 0.13)$ 时，算法运行的结果较好，求解成功率高。当变异频率较大时，算法运行结果较好，但是若变异频率过大，算法收敛速度较慢，所以，变异频率 Mutation $\in (0.4, 0.7)$ 时，算法效果较

(下转第 175 页)