

文章编号:1000-5013(2013)05-0516-05

doi:10.11830/ISSN.1000-5013.2013.05.0516

一种求解 0-1 背包问题的整数混沌粒子群优化算法

卢 璆

(华侨大学 网络与教育技术中心, 福建 厦门 361021)

摘要: 针对 0-1 背包问题(0-1 KP)的特点,以经典的速度-位移模型为基础整数编码各粒子,以混沌序列指导全局搜索,以排列的改变描述粒子的飞行.更新粒子的位置,进而提出用于求解 0-1 KP 的整数混沌粒子群优化(ICPSO)算法.该算法由于背包容量的限制,融入到编码和粒子飞行中,因而不会在进化中产生无效的粒子,从而提高了算法的求解效率.实验结果表明:ICPSO 算法简明、有效,较典型遗传算法,及粒子群算法具有更好的收敛性能和求解速度.

关键词: 粒子群优化;混沌;0-1 背包问题;遗传算法

中图分类号: TP 183

文献标志码: A

0-1 背包问题(0-1 KP)是一个典型的组合优化问题.20 世纪 50 年代末,它首先由 Dantzig^[1]提出的,其后受到了广泛的关注.这不仅因为背包问题在工业和金融投资领域能得到直接的应用^[2],而且还包括理论上的原因^[3-4].因此,研究一个高效的背包问题解法就显得尤为重要了.粒子群优化算法(particle swarm optimization,PSO)是一种基于个体进化与群体协作和竞争的随机搜索算法^[5].由于其过程简单明了、易于实现、计算效率高等特点,被公认为可以与遗传(genetic algorithm,GA)算法相媲美的高效算法,已在众多领域被广泛应用^[6-9].本文以 PSO 算法的原有速度-位移模型为基础,提出了一种用于求解 0-1 KP 的整数混沌 PSO(integer encoded chaos PSO,简称 ICPSO)算法.

1 0-1 背包问题的数学模型

0-1 背包问题可描述为:给定 n 个物品,对于第 i 个物品,其质量为 w_i ,价值为 p_i ($i=1,2,\cdots,n$),背包最大容量为 C ,问应该选择哪些物品放入背包内,以使得包中的物品总价值达到最大.

为方便描述,引入一个 0-1 变量 x_i ,其定义为

$$x_i = \begin{cases} 1, & \text{物品 } i \text{ 放入背包中,} \\ 0, & \text{物品 } i \text{ 不放入背包中.} \end{cases} \quad (1)$$

式(1)中: $i=1,2,\cdots,n$.

0-1 KP 的数学模型可表述为

$$\left. \begin{aligned} \max F &= \sum_{i=1}^{\infty} p_i x_i, \\ \text{s. t. } \sum_{i=0}^{\infty} w_i x_i &\leq C, \quad x_i = 0,1; \quad i = 1,2,\cdots,n. \end{aligned} \right\} \quad (2)$$

式(2)中:变量的取值只有 0 或 1.为不失一般性,在研究 0-1 KP 其求解方法时,还做如下假定:

1) \forall 物品 i ($i=1,2,\cdots,n$), $w_i > 0, p_i > 0$;

收稿日期: 2012-05-22

通信作者: 卢璆(1984-),女,助理工程师,主要从事智能信息处理、人工智能理论与应用的研究. E-mail:jlq@hqu.edu.cn.

基金项目: 中央高校基本科研业务费专项基金资助项目,华侨大学科研基金资助项目(11BS210)

2) \forall 物品 i ($i=1,2,\dots,n$), $w_i \leq C$;

3) $w_1 + w_2 + \dots + w_n \geq C$.

其理由是:1) 主要是为了叙述方便,以及基于现实逻辑(现实物品的质量和价值是正值)的考虑;2) 如果对于物品 i ,其质量 $w_i \geq C$,则 $x_i = 0$,可以直接排除在选择范围之内;3) 如果 $w_1 + w_2 + \dots + w_n \leq C$, $x_1 = x_2 = \dots = x_n = 1$,无需进行算法求解.

2 求解 0-1 KP 的 ICPSO 算法

2.1 粒子编码

针对 0-1 KP 的特殊性,根据式(2),采用如下编码方式:假设存在 n 个物品,其序号的集合为 $N = \{1, 2, \dots, n\}$,第 i 个粒子编码为 $X_i = (x_{i,1}, x_{i,2}, \dots, x_{i,n})$. $x_{i,j}$ 的取值按照式(1)的定义, $j=1,2,\dots,n$, $x_{i,1}w_1 + x_{i,2}w_2 + \dots + x_{i,n}w_n \leq C$. 该编码的含义是,将编码中值为 1 的物体放入背包中.

2.2 速度-位移模型

ICPSO 算法以整数编码各粒子,以混沌序列指导全局搜索,通过粒子飞行来改变粒子的排列,从而最终改变粒子的位置. 算法的进化方程为

$$v_{i,j}(t+1) = w(t) \cdot v_{i,j}(t) + c_1 \cdot r_1 \cdot \phi(p_{i,j}, x_{i,j}(t)) + c_2 \cdot r_2 \cdot \phi(g_j, x_{i,j}(t)), \quad (3)$$

$$\phi(p_{i,j}, x_{i,j}(t)) = \varphi(p_{i,j}, X_i(t)) \cdot \frac{|f(p_{i,j}) - f(x_{i,j}(t))|}{f_{\max} - f_{\min}} \cdot (b - a), \quad (4)$$

$$\phi(g_j, x_{i,j}(t)) = \varphi(g_j, x_{i,j}(t)) \cdot \frac{|f(g_j) - f(x_{i,j}(t))|}{f_{\max} - f_{\min}} \cdot (b - a), \quad (5)$$

$$\varphi(x, y) = \begin{cases} 0, & x = y, \\ 1, & x \neq y, \end{cases} \quad (6)$$

$$x_{i,j}(t+1) = \begin{cases} \text{变异}, & R_{i,j} \leq S(v_{i,j}(t)), \\ x_{i,j}(t), & R_{i,j} > S(v_{i,j}(t)), \end{cases} \quad (7)$$

$$S(v_{i,j}(t)) = \frac{1}{1 + \exp(-v_i(t))}. \quad (8)$$

式(3)中: $w(t)$ 为惯性权重. 它通过保持粒子的运动惯性,使其有扩展搜索空间的趋势. 然而,传统的线性减小的控制模式难以反映搜索空间变化的需要. 为此,采用混沌序列^[10]用于惯性权重的自适应生成,即

$$w(t+1) = u \cdot w(t)(1 - w(t)). \quad (9)$$

式(9)中: $u=4.0$; $w(0)$ 为 $(0,1)$ 上的随机数.

式(4),(5)中:函数 $f(x)$ 为每个位置适应度值; f_{\max} 和 f_{\min} 分别为所有位置取值中的最大值和最小值. 若当前位置与个体极值(或全局极值)相同,则 $\phi(p_{i,j}, x_{i,j}(t))$ (或 $\phi(g_j, x_{i,j}(t))$)为 0;否则,将它们取值限制在区间 $[a, b]$ 内,该限制操作的作用是保证速度不超过其最大取值.

式(7)中: S 函数的计算式采用式(8)^[11]. $v_{\max} \in [-6.0, 6.0]$,相应设置 $a=0$, $b=2.0$,则式(7)的含义是以概率 $1 - S(v_{i,j}(t))$ 保持不变,而以 $S(v_{i,j}(t))$ 概率发生变异. 所谓变异是指以一定的规则,改变粒子当前位置.

对于 0-1 KP 而言,各粒子编码存在潜在的背包容量的限制. 为此,设计模拟物品交换的变异规则. 当某一维度满足变异条件时,如果该维度的当前值为 0,则考虑该维度对应的物品加入背包后,是否超过背包的最大容量,如果不是,则该维度的当前值变为 1;否则,该维度为 0. 如果该维度的当前值为 1,则记背包中除该物品外已放入物品组成的集合为 N_1 ,并计算其总质量 W_1 ,记不在背包中且质量不大于 $(C - W_1)$ 物品组成的集合为 N_2 ,当 $N_2 = \emptyset$ 时,该维度值不变;否则,在集合 N_2 中选择价值最大的物品(将其对应的维度值置为 1),将本物品从背包中拿掉(置该物品的维度值为 0).

0-1 KP 中粒子位置更新算法的详细过程如下:

输入:第 i 个粒子的位置 $X_i = (x_{i,1}, x_{i,2}, \dots, x_{i,n})$,当前个体极值 $P_i = (p_{i,1}, p_{i,2}, \dots, p_{i,n})$,当前全局极值 $G = (g_1, g_2, \dots, g_n)$

输出:更新后的第 i 个粒子的位置

```
1:   For 第  $i$  个粒子的每一维  $x_{i,j}(j=1,2,\cdots,n)$ 
2:   {
3:       根据公式(3)计算每一维的速度  $v_{i,j}(t+1)$ ;
4:       产生一个随机数  $R$ ;
5:       If ( $R \leq \text{Sigmoid}(v_{i,j}(t))$ )
6:       {
7:           计算背包中其他物品的总重量  $W_1$ ;
8:           If ( $(x_{i,j}=0) \&\& (w_j \leq C-W_1)$ )  $x_{i,j}=1$ ;
9:           Else If ( $x_{i,j}=1$ )
10:          {
11:              确定背包中除该物品外其它物品组成的集合  $N_1$ ;
12:              确定不在背包中且重量不大于  $C-W_1$  物品组成的集合  $N_2$ ;
13:              If ( $N_2 \neq \emptyset$ ) 在集合  $N_2$  中选择价值最大物品,  $x_{i,j}=0$ ;
14:          }
15:      }
16:  }
17:  Return  $X_i$ ;
```

值得指出的是,由于在编码和粒子“飞行”中都将背包容量的限制考虑在内,因此本算法不会像其他求解背包问题的智能算法一样在进化过程中产生无效的编码,从而有效地提高了算法的求解效率.

2.3 适应度函数的定义

0-1 KP 的目标是使得物品总价值最大化,因此,设定粒子 $X_i=(x_{i,1},x_{i,2},\cdots,x_{i,n})$ 的适应度值为

$$F(X_i)=\sum_{j=1}^n p_j x_{i,j} \tag{10}$$

另外,对于粒子每一维的适应度值(每个物品的适应度值),采用

$$f(x_{i,j})=x_{i,j} \cdot \frac{p_j}{w_j}, \quad j=1,2,\cdots,n. \tag{11}$$

当物体 j 在背包中($x_{i,j}=1$)时,其适应度值为该物体的价值密度;当物体 j 不在背包中($x_{i,j}=0$)时,其适应度值为 0. 从而,式(4),(5)中的 f_{\max} 和 f_{\min} 分别为物体价值密度中最大者和 0.

2.4 基于 ICPSO 的 0-1 KP 求解流程

综上所述,可以总结 6 个基于 ICPSO 的 0-1 KP 的求解步骤.

- 步骤 1 初始化种群,每个粒子必须满足粒子的编码规则.
- 步骤 2 根据公式(10),(11),分别计算每个粒子的适应度值以及粒子在每一维上的适应度值.
- 步骤 3 对于每个粒子 i ,将它的适应度值与个体极值 P_i (个体所经历过的最好位置)的适应度值进行比较,若优于后者,则将它当前的位置作为个体极值.
- 步骤 4 对于每个粒子 i ,将它的适应度值与全局极值 G (群体所经历的最好位置)的适应度值进行比较,若优于后者,则将它当前的位置作为全局极值.
- 步骤 5 判断是否满足终止条件,否则继续.
- 步骤 6 对于群体中的每个粒子,依据式(3),更新每一维上的速度,并 0-1 KP 中粒子位置更新算法粒子的位置,转至步骤 2.

3 仿真实验及分析

针对 50 维的 0-1 背包问题进行仿真实验,并与 $GA^{[12]}$ 和一般 $PSO^{[13]}$ 算法进行性能比较. 首先,通过以下规则^[12]随机生成 50 维的背包问题:

- 1) 各物品的价值在区间(100, 500)中随机生成;

- 2) 各物品的质量在区间(10, 100)中随机生成;
- 3) 背包容量采用下式计算,即

$$C=r\sum_{i=1}^{50}\omega_i.$$

(12)

式(12)中: r 为区间(0.1, 0.3)上的随机数.由此规则产生的 50 维背包问题的具体数据,如表 1 所示.表 1 中:物品的总价值为 13 841,总质量为 3 429,背包的最大容量为 508.标准最优解为(1, 8, 11, 14 ,16, 18, 23, 26, 45, 47),选择物品的总质量为 507,总价值为 4 012.

表 1 随机生成的 50 维 0-1 背包问题算例

Tab.1 50-Dimension 0-1 KP Generated Randomly

物品序号	物品的价值					物品质量				
1-5	403,	280,	175,	415,	499	58,	52,	95,	73,	93
6-10	124,	191,	388,	292,	455	33,	94,	51,	87,	73
11-15	480,	360,	107,	497,	132	58,	99,	87,	56,	97
16-20	309,	173,	357,	215,	145	40,	82,	56,	83,	74
21-25	153,	454,	390,	107,	333	33,	95,	40,	47,	54
26-30	406,	243,	263,	296,	139	53,	89,	60,	69,	77
31-35	100,	234,	129,	152,	139	65,	63,	40,	24,	72
36-40	164,	131,	222,	330,	379	90,	95,	55,	79,	61
41-45	300,	475,	313,	230,	403	74,	95,	91,	41,	52
46-50	336,	379,	314,	222,	108	68,	43,	94,	89,	80

实验设置 GA 的参数为:种群数 500,交叉概率为 0.8,变异概率为 0.01,终止代数为 8 000.

经典 PSO 和 ICPSO 的参数为:种群大小为 500,终止代数为 8 000,每种算法独立运算 100 次,表 2 给出 100 次求解的相关数据统计.ICPSO 与 GA 及经典 PSO 求解 0-1 KP 的性能对比,如表 2 所示.

表 2 ICPSO 与 GA 及经典 PSO 求解 0-1 KP 的性能对比

Tab.2 Performance contrast of ICPSO, GA and traditional PSO for 0-1 KP

迭代 次数	GA 算法				经典 PSO 算法				ICPSO 算法			
	最差	最好	平均	找到最优解的次数	最差	最好	平均	找到最优解的次数	最差	最好	平均	找到最优解的次数
1 000	3 622	3 889	3 803.69	0	3 661	3 935	3 810.99	0	3 733	4 012	3 822.30	1
2 000	3 668	3 942	3 837.73	0	3 684	3 988	3 839.95	0	3 738	4 012	3 866.34	13
3 000	3 639	4 012	3 845.26	1	3 667	4 012	3 856.29	1	3 761	4 012	3 901.58	28
4 000	3 713	4 012	3 859.68	1	3 668	4 012	3 866.75	3	3 754	4 012	3 929.53	39
5 000	3 759	4 012	3 867.94	3	3 762	4 012	3 894.00	9	3 863	4 012	3 952.05	46
6 000	3 777	4 012	3 892.76	7	3 776	4 012	3 916.15	13	3 886	4 012	3 958.54	49
7 000	3 836	4 012	3 909.58	13	3 863	4 012	3 940.50	27	3 889	4 012	3 9 64.92	52
8 000	3 854	4 012	3 917.71	22	3 889	4 012	3 950.78	31	3 935	4 012	3 986.59	67

从表 2 可以看出:3 种算法都能搜索到标准最优解,但是 GA 算法在迭代 8 000 次后,只有 22 次搜索到了最优解,成功率为 22%;传统的 PSO 算法的性能优于经典的 GA 算法,当迭代次数达到 8 000 次时,搜索到最优解的次数达到 31 次,成功率为 31%;ICPSO 较传统的 PSO 算法进一步提高了求解性能,在迭代到 4 000 次时,ICPSO 就有 39 次能够找到最优解,成功率为 39%;当迭代次数达到 8 000 次时,ICPSO 搜索到最优解的次数达到 67 次,成功率为 67%.由此可见,ICPSO 对求解 0-1 背包问题具有很强的有效性和可行性,较经典的 GA 和传统的 PSO 算法,具有更高的求解精度和更快的收敛速度.

4 结论

提出了一种整数混沌粒子群优化算法(ICPSO),该算法针对 0-1 KP 的特点,在继承原有速度-位移模型的基础上,对传统 PSO 做了如下 4 点改进:

- 1) 以经典的速度-位移模型为基础,整数编码各粒子;

- 2) 以混沌序列指导全局搜索;
- 3) 以排列的改变描述粒子的飞行并更新粒子的位置;
- 4) 将背包容量的限制融入到编码和粒子飞行中,避免进化中产生无效的粒子.

测试结果表明:ICPSO 算法可用以有效求解 0-1 KP 问题,较 GA 和 PSO 算法具有更好的收敛性能和求解速度.

参考文献:

[1] DANTZIG G B. Discrete variable extremum problems[J]. Operations Reaserach,1957,5(2):266-277.

[2] 张生,魏忠华,何尚录,等. 0-1 背包问题在限额投资决策中的应用及其扰动分析[J]. 内蒙古师范大学学报:自然科学汉文版,2007,36(9):595-598.

[3] 徐光辉. 运筹学基础手册[M]. 北京:科学出版社,1999:102-188

[4] 王保仓,韦永壮,胡子濮. 基于随机背包的公钥密码[J]. 电子与信息学报,2010,32(7):1580-1584.

[5] EBERHART R,KENNEDY J. A new optimizer using particle swarm theory[C]// Proceedings of the 6th International Symposium on Micro Machine and Human Science. Los Alamitos:IEEE Press,1995:39-43.

[6] ZHU Jia-rui,JI Zhen,SHI Yu-hui ,et al. DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm[J]. IEEE Transactions on Evolutionary Computation,2011,15(5):643-658.

[7] ROUT N K,DAS D P,PANDA G. Particle swarm optimization based active noise control algorithm without second-ary path identification[J]. IEEE Transactions on Instrumentation and Measurement,2012,61(2):554-563.

[8] 胡珀,娄渊胜. 改进粒子群优化算法在服务组合中的应用[J]. 计算机工程,2011,37(17):130-132.

[9] 唐朝霞,章慧,徐冬梅. 一种改进的粒子群算法和相关反馈的图像检索[J]. 计算机科学,2011,38(10):278-280.

[10] 黄润生. 混沌及其应用[M]. 武汉:武汉大学出版社,2002:128-140.

[11] KENNEDY J,EBERHART R C. A discrete binary version of the particle swarm algorithm[C]// Proceedings of the 10th IEEE International Conference on Systems, Man and Cybernetics. Los Alamitos: IEEE Press, 1997: 4104-4109.

[12] 王小平,曹立明. 遗传算法-理论、应用与软件实现[M]. 西安:西安交通大学出版社,2002:18-50.

[13] 曾建潮,介婧,崔志华. 微粒群算法[M]. 北京:科学出版社,2004:12-18.

An Integer Chaos-Particle-Swarm-Optimization Algorithm
for 0-1 Knapsack Problem

LU Jing

(Center for Network and Education Technology, Huaqiao University, Xiamen 361021, China)

Abstract: For solving 0-1 knapsack problem (KP), an integer chaos-particle-swarm-optimization (ICPSO) algorithm is presented. In the algorithm, according to the characteristic of 0-1 KP, the classical velocity-position model is inherited; each particle is encoded with integers; a chaos sequence is employed to direct global search; and the permutation change is used to depict the flying behavior of each particle, (i. e. , the update of position). Further, because the limit of pack capacity is taken into consideration in the coding and particle flying process, no invalid particles are produced in the evolutionary process, which thereby enhances the algorithm efficiency. Experimental results demonstrate that ICPSO is simple but effective, and better than genetic algorithm and particle swarm optimization at constringency and convergence speed.

Keywords: particle swarm optimization; chaos; 0-1 knapsack problem; genetic algorithm

(责任编辑: 陈志贤 英文审校: 吴逢铁)