元胞微粒群算法及其在多维背包问题中的应用①

刘 勇[1,2]，马 良[1]
（1. 上海理工大学管理学院，上海 200093；2. 盐城工学院基础教学部，盐城 224051）

摘要： 针对离散微粒群算法早熟收敛问题，基于元胞自动机的原理和离散微粒群算法，提出一种元胞微粒群算法. 将元胞及其邻居引入到算法中来保持种群的多样性，利用元胞的演化规则进行局部优化，避免算法陷入局部极值. 通过对典型多维背包问题的仿真实验和与其他算法的比较，表明本算法可行有效，有良好的全局优化能力.

关键词： 元胞自动机；离散微粒群算法；多维背包问题；优化
中图分类号： O22 **文献标识码：** A **文章编号：** 1007－9807(2011)01－0086－11

0 引 言

微粒群优化算法 PSO(particle swarm optimization) 是由 Kennedy 和 Eberhart 于 1995 年提出的，是基于对鸟群社会行为模拟的演化算法[1-2]. PSO 算法已经被成功应用于函数优化、神经网络训练、模糊系统控制以及其他工程领域. 但 PSO 算法大部分是针对求解连续空间的优化问题，如何将其改进并应用于求解离散空间的优化问题，特别是组合优化中的 NP 问题，是 PSO 算法的一个改进与应用研究方向[3]. 1997 年，Kennedy 和 Eberhart 首先提出了二进制微粒群优化算法[4] (Binary Particle Swarm Optimization，BPSO)，Clerc 针对旅行商问题(Traveling Salesman Problem，TSP) 提出的 TSP-DPSO 算法[5,3]，Salman 等利用连续 PSO 算法求解分布式计算机任务分配问题[6]. 总的来看，对离散微粒群算法研究的还很少，改进离散微粒群算法的性能，尤其是解决其易陷入局部极值、早熟收敛和收敛速度慢是关注的重点[3].

元胞自动机由 Von Neuman 提出[7]，是一个时间、空间、状态都离散的动力学系统，已成为以离散性为特点描述复杂行为的一种具有广阔发展前景的方法[8]. 元胞相互离散，构成一个元胞空间，元胞空间内的每个元胞的变化取决于其周围半径为 r 邻居中的元胞，它们遵循同样的演化规则，作同步局部更新. 大量元胞通过简单的相互作用来模拟复杂系统的演化. 元胞自动机已在混沌与分形、图像处理、智能材料、机器学习等方面有着广泛的应用.

本文基于元胞自动机的原理和二进制微粒群算法的特点，提出一种元胞微粒群算法 CPSO (Cellular Particle Swarm Optimization)，将元胞自动机的元胞、元胞空间及元胞邻居与微粒群算法搜索空间相结合，利用元胞及其邻居来丰富微粒的行为，保持种群的多样性，同时按元胞的演化规则作局部优化，避免算法早熟收敛，快速搜寻到全局最优解. 通过对多维背包问题的标准测试库的实验，表明算法的高效性.

1 元胞微粒群算法

在 Kennedy 和 Eberhart 提出的二进制微粒群算法 BPSO[4] 中，微粒被表达为一组 0、1 构成的二进制向量，微粒速度被 Sigmodal 函数转换成 [0, 1] 内，并和随机数比较，用来确定微粒位置是取 0 还是 1. 该算法简洁，易于实现，但在演化过程中

① 收稿日期：2009－01－11；修订日期：2009－10－19.
基金项目：国家自然科学基金资助项目(70871081)；上海市重点学科建设项目资助(S30504).
作者简介：刘 勇(1982—)，江苏人，博士生. Email: liuyong.seu@163.com

种群易陷入局部最优,群体多样性降低,导致早熟收敛现象的出现,而且这种缺陷不会随着演化次数的增加而改变.

元胞微粒群算法将元胞自动机的思想引入到 BPSO 算法中,有助于保持种群的多样性,避免早熟现象产生,使微粒群在搜索过程中保持进一步寻优能力.

1.1　元胞自动机原理

元胞自动机最基本的组成包括元胞、元胞空间、邻居及规则四部分. 简单讲,元胞自动机可以视为由一个元胞空间和定义于该空间的变换函数所组成,可模拟复杂结构和过程的模型.

从集合论角度元胞自动机有着严格地描述和定义[9]:

设 d 代表空间维数,k 代表元胞的状态,并在一个有限集合 S 中取值,r 代表元胞的邻居半径. Z 是整数集,表示一维空间,t 代表时间. 为叙述和理解上简单起见,在一维空间上考虑元胞自动机,即假定 $d=1$. 那么整个元胞空间就是在一维空间,将整数集 Z 上的状态集 S 的分布,记为 S^Z. 元胞自动机的动态演化就是在时间上状态组合的变化,可以记为

$$F:\ S_t^Z \rightarrow S_{t+1}^Z$$

这个动态演化又由各个元胞的局部演化规则 f 所决定的. 这个局部函数 f 通常又常常被称为局部规则. 对于一维空间,元胞及其邻居可以记为 S^{2r+1},局部函数则可以记为

$$f:\ S_t^{2r+1} \rightarrow S_{t+1}$$

对于局部规则 f 来讲,函数的输入、输出集均为有限集合,实际上. 它是一个有限的参照表. 对元胞空间内的元胞,独立施加上述局部函数,则可得到全局的演化

$$F(c_{t+1}^i) = f(c_t^{i-r},\cdots,c_t^i,\cdots,c_t^{i+r})$$

c_t^i 表示在位置 i 处的元胞,至此,就得到了一个元胞自动机模型.

标准的元胞自动机是一个四元组:

$$A = (L_d, S, N, f)$$

式中 A 代表一个元胞自动机系统; L 表示元胞空间,d 是一正整数,表示元胞自动机内元胞空间的维数; S 是元胞的有限的、离散的状态集合; N 表示一个所有邻域内元胞的组合(包括中心元胞),即包含 n 个不同元胞状态的一个空间矢量,记为

$$N = (S_1, S_2, \cdots, S_n)$$

n 是元胞的邻居个数. $s_i \in Z$(整数集合),$i \in \{1,\cdots,n\}$; f 表示将 S^n 映射到 S 上的一个局部转换函数. 所有的元胞位于 d 维空间上,其位置可用一个 d 元的整数矩阵 Z^d 来确定.

1.2　多维背包问题的元胞微粒群算法描述

多维背包问题是组合优化中一个经典的 NP 难题,有着许多重要的实际应用背景,如资源分配、投资决策、货物装载、材料切割等问题. 同时由于多维背包问题结构简洁,可作为子问题为研究更复杂问题奠定理论基础,有很高的理论研究价值[10].

多维背包问题的数学模型为

$$\text{Max}\quad f(x_1,x_2,\cdots,x_n) = \sum_{j=1}^{n} c_j x_j$$

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leqslant b_i \quad (i=1,2,\cdots,m) \\ x_j \in \{0,1\} \quad (j=1,2,\cdots,n) \end{cases}$$

其中,n 为物品的编号,m 为资源的编号; c_j 为第 j 个物品的受益量; b_i 为第 i 种资源的预算; a_{ij} 为第 j 个物品占用第 i 种资源的量; x_j 为 0 - 1 决策变量(当物品 j 被选择时 $x_j=1$,否则 $x_j=0$).

对于多维背包问题,其可行域集合为 n 维欧氏空间,可将其中的任一元素视为元胞,而所有元素构成元胞空间. 结合多维背包问题,给出元胞微粒群算法的具体描述.

定义1　设物品选择结果的集合 $C=(c_1,c_2,\cdots,c_i,\cdots,c_n)$,其中 $c_i \in \{0,1\}$,$c_i=1$ 表示选中物品 i,$c_i=0$ 表示未选中物品 i. C 中 c_i 任意取值的排序组合的集合为元胞空间,可表示为 $L = \{CellX = (c_1,c_2,\cdots,c_i,\cdots,c_n) \mid c_i \in \{0,1\}\}$,每个组合 $CellX$ 为元胞.

定义2　Moore 邻居类型:

$$N_{moore} = \{CellY \mid diff(CellY - CellX) \leqslant r,\ CellX, CellY \in L\}$$

其中,$diff(CellY-CellX)$ ② $\leqslant r$ 为两个组合排序的

② $diff(CellY-CellX)$ 解释如下:
设 $CellX=(c_1,c_2,\cdots,c_i,\cdots,c_n)$,$c_i \in \{0,1\}$,$CellY=(c_1',c_2',\cdots,c_i',\cdots,c_n^i)$,$c_i' \in \{0,1\}$,先将 $CellY$ 和 $CellX$ 作异或运算,"异或"的意思是判断两个相应的位值是否为异,为异就取 1,否则就取 0,再把异或运算结果的每一位值相加得到的值为差异度.

差异，若无差异为 0，有差异时，最小为 1. r 为差异度，本文 r 取 1.

定义 3 扩展 Moore 邻居类型:

$$N'_{Moore} = \{ CellY \mid diff(CellY - CellX) \leqslant R, \quad CellX, CellY \in L\}$$

其中，$diff(CellY - CellX) \leqslant R$ 为两个组合排序的差异，若无差异为 0，有差异时，最小为 2. R 为差异度，本文 R 取 2.

定义 4 微粒的位置和速度更新方程

设解空间的维数为 d，第 i 个微粒在进化的第 k 代时位置为 $x_i^k = (x_{i1}^k, x_{i2}^k, \cdots, x_{id}^k)^{\mathrm{T}}$，微粒已搜索过的最好位置为 $p_i^k = (p_{i1}^k, p_{i2}^k, \cdots, p_{id}^k)^{\mathrm{T}}$，速度为 $v_i^k = (v_{i1}^k, v_{i2}^k, \cdots, v_{id}^k)^{\mathrm{T}}$，整个种群经过的最好位置为 $g_i^k = (g_{i1}^k, g_{i2}^k, \cdots, g_{id}^k)^{\mathrm{T}}$. 每个微粒根据如下公式更新速度和位置[11-12]

$$V_{ij}^{k+1} = wv_{ij}^k + c_1R_1(p_{ij}^k - x_{ij}^k) + c_2R_2(g_{ij}^k - x_{ij}^k)$$

$$v_{ij}^{k+1} = \begin{cases} v_{\max} & v_{ij}^{k+1} > v_{\max} \\ v_{ij}^{k+1} & v_{\min} \leqslant v_{ij}^{k+1} \leqslant v_{\max} \\ v_{\min} & v_{ij}^{k+1} \leqslant v_{\min} \end{cases}$$

$$x_{ij}^{k+1} = \begin{cases} 1 & R_3 < \mathrm{sig}(v_{ij}^{k+1}) \\ 0 & \text{否则} \end{cases}$$

其中，w 为惯性权重，起着平衡全局和局部搜索的作用; c_1、c_2 为学习因子，c_1 反映了微粒飞行过程中所记忆的最好位置对微粒飞行速度的影响，被称为“认知系数”，则反映了整个微粒群所记忆的最好位置对飞行速度的影响[11]，又称为“社会学习系数”; R_1，R_2 和 R_3 为 [0,1] 随机数，$\mathrm{sig}(x)$ 是一个模糊函数，其定义为 $\mathrm{sig}(x) = 1/(1 + e^{-x})$. v_{ij}^{k+1} 决定了微粒的位置分量取 0 或 1 的概率，即以 $\mathrm{sig}(v_{ij}^{k+1})$ 的概率取 1，以 $1 - \mathrm{sig}(v_{ij}^{k+1})$ 的概率取 0[11-12].

定义 5 适应值

将约束优化问题转化为无约束问题，一般可采用罚函数的方法，采用较简单的和形式的罚函数，令罚函数为

$$h = \sum_{i=1}^{m} \left| \min\left\{0, b_i - \sum_{j=1}^{n} a_{ij}x_j\right\} \right|$$

适应值为

$$f - Mh = \sum_{j=1}^{n} c_jx_j - M\sum_{i=1}^{m} \left| \min\left\{0, b_i - \sum_{j=1}^{n} a_{ij}x_j\right\} \right|$$

其中，M 为充分大的正数.

定义 6 元胞演化规则

依据元胞邻居的定义计算其邻居的目标解，比较元胞和其邻居的差异，选择最好的目标解.

元胞微粒群算法主要流程如下:

步骤 1 确定种群规模 N，惯性权重因子 w，学习因子 c_1 和 c_2，$nc \leftarrow 0$; (nc 为迭代步数或搜索次数);

步骤 2 初始化所有微粒的位置和速度:

每个微粒的位置由 $x_{ij}^0 = \begin{cases} 0 & R(0,1) < 0.5 \\ 1 & \text{否则} \end{cases}$ 随机生成.

每个微粒的速度由 $v_{ij}^0 = v_{\min} + R(0,1)(v_{\max} - v_{\min})$ 随机生成.

其中，$i = 1,2,\cdots,N$，$j = 1,2,\cdots,n$，$R(0,1)$ 表示 [0,1] 随机数，$v_{\max}$ 和 $v_{\min}$ 表示最大速度和最小速度.

步骤 3 计算每个微粒的适应值，记录当前的最好解;

步骤 4 按元胞邻居的定义，在邻居范围内演化，记录最好解;

步骤 5 比较更新每个微粒的最好位置和整个种群的最好位置;

步骤 6 置 $nc \leftarrow nc + 1$;

步骤 7 若 $nc <$ 预定的迭代次数，更新每个微粒的位置和速度; 转步骤 3;

步骤 8 输出目前的最好解.

2 数值试验

为验证算法的有效性，选用“http://elib.zib.de/pub/mp-testdata/ip/sac94-suite/index.html”公布的典型的测试多维背包问题的算例. 实验所用硬件为 Pentium3.4GHz，1GRAM，软件为 Windows XP 和 Matlab.

2.1 参数设置

在元胞微粒群算法中，主要有 7 个参数需要设置，即种群规模 N，惯性权重因子 w，学习因子 c_1、c_2，最大速度 $v_{\max}$，最小速度 $v_{\min}$ 和邻居类型.

Shi 和 Eberhart 认为，微粒群算法优化结果对

种群大小不敏感[13]. 针对种群规模 N 的一些实验表明,单纯依靠提高种群规模并不能明显提高算法性能,但在很大程度上会影响计算速度. 根据本文的实验经验,将 N 设置为 $N = 100$.

在离散二进制微粒群算法中,v_{ij} 表示一个概率,即微粒的每一维分量的取值以 $\mathrm{sig}(v_{ij})$ 的概率取 1,而以 $1 - \mathrm{sig}(v_{ij})$ 的概率取 0. v_{ij} 越大,微粒位置取 1 的概率越大,反之越小. 在传统微粒群算法中,为防止微粒远离搜索空间,将 v_{ij} 限制在 $[v_{\min}, v_{\max}]$ 之间. 在离散微粒群算法中,这种限制也存在,表示算法所允许的概率范围. 经过大量实验分析,本文将 $v_{\max}$ 和 $v_{\min}$ 分别设置为 4 和 -4,搜索效果较好,同时也与文献[14]的结论一致,即为防止 sig 函数饱和,通常将 v_{ij} 限制在 $[-4, 4]$ 之间.

目前对离散微粒群算法中惯性权重因子 w,学习因子 c_1、c_2 设置问题的研究还很欠缺,如何取值无法给出统一的标准,缺少实用性指导原则,对这些可调参数值的选取比较通用的方法是根据实验而定. 在实验中发现,当 $w = 1$,$c_1 = c_2 = 2$ 算法性能较优.

最后讨论邻居类型的设置问题,邻居类型有 Moore 邻居和扩展 Moore 邻居两种. 因为算法在规模差不多的算例上的测试效果基本相同,故将测试算例分为 4 组,即 m 和 n 都较小的为一组; m 较大而 n 较小的为一组; m 较小而 n 较大的为一组; m 和 n 都较大的为一组. 每组中选取一个代表算例进行分析,其中,weing6. dat 为第一组中的算例,pet3. dat 为第 2 组中的算例,weing7. dat 为第 3 组中的算例,weish22. dat 为第四组中的算例. 图 1 给出了不同邻居类型对算法优化过程的影响.

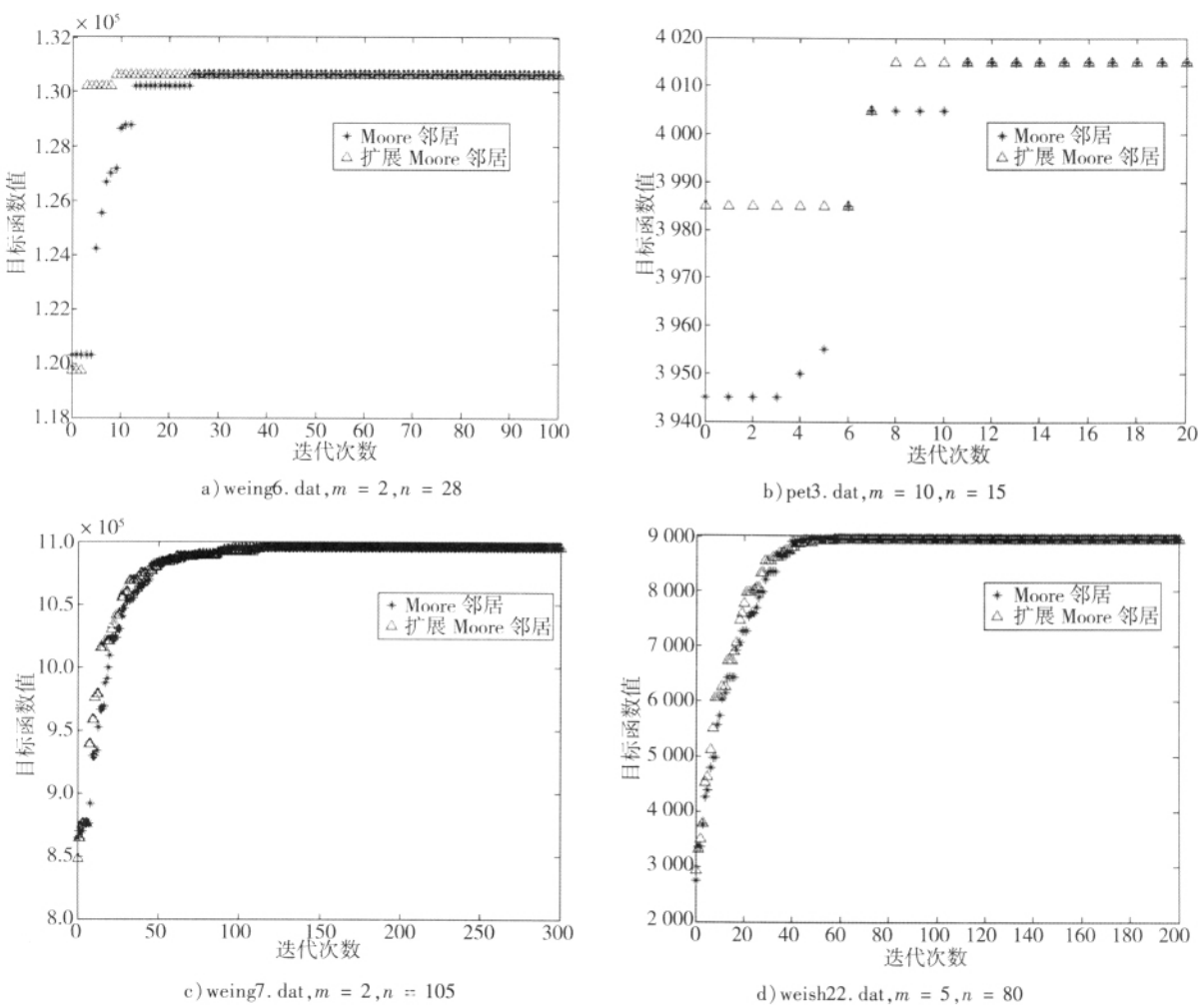


图 1　邻居类型对优化过程的影响

Fig. 1 Experimental results of optimization process under different neighbor type

从图 1 可以发现，无论采用 *Moore* 邻居还是扩展 *Moore* 邻居，算法均可达到全局最优，但两者具体的优化过程不同. 对小规模问题，若采用扩展 *Moore* 邻居，算法搜索多样性显著增强，获取全局最优解概率更大，易于搜索到最优解. 对大规模问题，若采用扩展 *Moore* 邻居，算法搜索多样性的优势相对不够明显，但计算量显著增加，因为问题规模越大，在邻居半径增大的情况下，算法需要更多的搜索时间即需要计算更多解的目标函数值，同时由于约束条件的增加，得到较当前更优的解的可能性会减少. 采用 *Moore* 邻居，在小规模问题中部分算例优势不够明显，但随着问题规模的增大，结合求解质量和计算时间两个方面考虑，综合性能较优. 选择邻居类型的关键是找到一个平衡点，使得算法既能达到全局最优，又能有较快的优化速度，邻居类型应根据具体问题而定.

2.2 测试结果及分析

标准多维背包问题测试库中，共 55 个算例，元胞微粒群算法对每个实例均能达到最优解，表 1 是元胞微粒群算法在测试每一个多维背包时，记录达到最优解时的迭代次数和采用的邻居类型.

表 1 多维背包问题元胞微粒群算法(CPSO) 计算结果

Table 1 Computational results of MKP of CPSO

测试数据	*m*	*n*	已知最优解	元胞微粒群算法	迭代次数	邻居类型
weing1. dat	2	28	141 278	141 278	18	扩展 Moore 邻居
weing2. dat	2	28	130 883	130 883	9	扩展 Moore 邻居
weing3. dat	2	28	95 677	95 677	13	扩展 Moore 邻居
weing4. dat	2	28	119 337	119 337	15	扩展 Moore 邻居
weing5. dat	2	28	98 796	98 796	18	扩展 Moore 邻居
weing6. dat	2	28	130 623	130 623	11	扩展 Moore 邻居
pb4. dat	2	29	95 168	95 168	14	扩展 Moore 邻居
weing7. dat	2	105	1 095 445	1 095 445	108	Moore 邻居
weing8. dat	2	105	624 319	624 319	101	Moore 邻居
hp1. dat	4	28	3 418	3 418	9	扩展 Moore 邻居
pb1. dat	4	27	3 090	3 090	17	扩展 Moore 邻居
pb2. dat	4	34	3 186	3 186	21	扩展 Moore 邻居
hp2. dat	4	35	3 186	3 186	28	扩展 Moore 邻居
weish01. dat	5	30	4 554	4 554	18	扩展 Moore 邻居
weish02. dat	5	30	4 536	4 536	27	扩展 Moore 邻居
weish03. dat	5	30	4 115	4 115	8	扩展 Moore 邻居
weish04. dat	5	30	4 561	4 561	13	扩展 Moore 邻居
weish05. dat	5	30	4 514	4 514	25	扩展 Moore 邻居
pet6. dat	5	39	10 618	10 618	32	扩展 Moore 邻居
weish06. dat	5	40	5 557	5 557	33	扩展 Moore 邻居
weish07. dat	5	40	5 567	5 567	17	扩展 Moore 邻居
weish08. dat	5	40	5 605	5 605	18	扩展 Moore 邻居
weish09. dat	5	40	5 246	5 246	12	扩展 Moore 邻居
weish10. dat	5	50	6 339	6 339	22	扩展 Moore 邻居

续表 1

Table 1 Continue

测试数据	m	n	已知最优解	元胞微粒群算法	迭代次数	邻居类型
weish11. dat	5	50	5 643	5 643	19	扩展 Moore 邻居
weish12. dat	5	50	6 339	6 339	19	扩展 Moore 邻居
weish13. dat	5	50	6 159	6 159	33	扩展 Moore 邻居
pet7. dat	5	50	16 537	16 537	30	扩展 Moore 邻居
weish14. dat	5	60	6 954	6 954	37	扩展 Moore 邻居
weish15. dat	5	60	7 486	7 486	27	扩展 Moore 邻居
weish16. dat	5	60	7 289	7 289	37	扩展 Moore 邻居
weish17. dat	5	60	8 633	8 633	27	扩展 Moore 邻居
weish18. dat	5	70	9 580	9 580	61	Moore 邻居
weish19. dat	5	70	7 698	7 698	84	Moore 邻居
weish20. dat	5	70	9 450	9 450	103	Moore 邻居
weish21. dat	5	70	9 074	9 074	98	Moore 邻居
weish22. dat	5	80	8 947	8 947	87	Moore 邻居
weish23. dat	5	80	8 344	8 344	106	Moore 邻居
weish24. dat	5	80	10 220	10 220	107	Moore 邻居
weish25. dat	5	80	9 939	9 939	117	Moore 邻居
weish26. dat	5	90	9 584	9 584	55	Moore 邻居
weish27. dat	5	90	9 819	9 819	93	Moore 邻居
weish28. dat	5	90	9 492	9 492	79	Moore 邻居
weish29. dat	5	90	9 410	9 410	62	Moore 邻居
weish30. dat	5	90	11 191	11 191	77	Moore 邻居
pet2. dat	10	10	87 061	87 061	3	扩展 Moore 邻居
pet3. dat	10	15	4 015	4 015	11	扩展 Moore 邻居
pet4. dat	10	20	6 120	6 120	13	扩展 Moore 邻居
flei. dat	10	20	2 139	2 139	11	扩展 Moore 邻居
pb5. dat	10	20	2 139	2 139	13	扩展 Moore 邻居
pet5. dat	10	28	12 400	12 400	34	扩展 Moore 邻居
pb7. dat	30	37	1 035	1 035	32	Moore 邻居
pb6. dat	30	40	776	776	19	Moore 邻居
sent01. dat	30	60	7 772	7 772	49	Moore 邻居
sent02. dat	30	60	8 722	8 722	70	Moore 邻居

为进一步验证算法的性能,将算法与二进制微粒群算法和文献[5]的禁忌搜索算法比较. 元胞微粒群算法和二进制微粒群算法参数设置同前所述,为方便起见,元胞邻居均采用 Moore 邻居,禁忌搜索算法参数设置见文献[5]. 每个算例独立运行20次,记录 20 次实验中获优次数、最优解、最劣解和平均解,实验结果如表2所示. 图2是元胞微粒群算法和二进制微粒群算法部分算例寻优效果对比图.

表 2　元胞微粒群算法(CPSO)和二进制微粒群算法(BPSO)、禁忌搜索算法(TS)性能比较

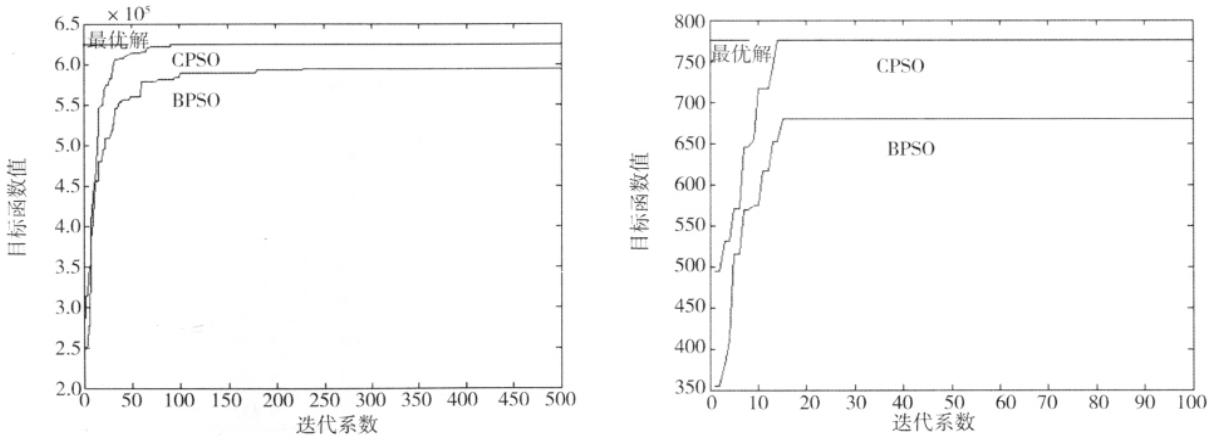
Table 2 Comparison results of CPSO, BPSO and TS

测试数据	m	n	最优解	算法	最大迭代次数	获优次数	最优解	最劣解	平均解
weing4. dat	2	28	119 337	BPSO	80	10/20	119 337	115 831	118 110
				TS	112	20/20	119 337	119 337	119 337
				CPSO	80	20/20	119 337	119 337	119 337
weing6. dat	2	28	130 623	BPSO	100	7/20	130 623	130 233	130 370
				TS	112	11/20	130 623	129 272	130 367.7
				CPSO	100	20/20	130 623	130 623	130 623
weing8. dat	2	105	624 319	BPSO	500	0/20	596 790	561 228	579 930
				TS	420	12/20	624 319	621 086	623 025.8
				CPSO	500	15/20	624 319	606 029	623 220
weish09. dat	5	40	5 246	BPSO	100	10/20	5 246	5 168	5 223.5
				TS	200	20/20	5 246	5 246	5 246
				CPSO	100	20/20	5 246	5 246	5 246
pet7. dat	5	50	16 537	BPSO	100	0/20	16 458	16 188	16 324
				TS	200	17/20	16 537	16 524	16 535
				CPSO	100	18/20	16 537	16 465	16 533
weish11. dat	5	50	5 643	BPSO	100	0/20	5 624	5 425	5 517.8
				TS	200	20/20	5 643	5 643	5 643
				CPSO	100	19/20	5 643	5 639	5 642.8
weish12. dat	5	50	6 339	BPSO	100	0/20	6 318	6 072	6 240.7
				TS	200	17/20	6 339	6 338	6 338.85
				CPSO	100	20/20	6 339	6 339	6 339
weish13. dat	5	50	6 159	BPSO	100	0/20	6 140	5 925	6 052.6
				TS	200	18/20	6 159	6 147	6 157.8
				CPSO	100	19/20	6 159	6 050	6 153.6
weish14. dat	5	60	6 954	BPSO	150	0/20	6 923	6 732	6 819.9
				TS	240	18/20	6 954	6 921	6 950.8
				CPSO	150	18/20	6 954	6 923	6 950.9
weish15. dat	5	60	7 486	BPSO	100	0/20	7 449	7 128	7 318.9
				TS	240	20/20	7 486	7 486	7 486
				CPSO	100	20/20	7 486	7 486	7 486
weish16. dat	5	60	7 289	BPSO	100	0/20	7 269	7 021	7 124.5
				TS	240	20/20	7 289	7 289	7 289
				CPSO	100	19/20	7 289	7 288	7 288.9
weish17. dat	5	60	8 633	BPSO	150	0/20	8 618	8 519	8 577.6
				TS	240	19/20	8 633	8 624	8 632.55
				CPSO	150	19/20	8 633	8 624	8 632.55

续表 2

Table 2 Continue

测试数据	m	n	最优解	算法	最大迭代次数	获优次数	最优解	最劣解	平均解
weish18. dat	5	70	9 580	BPSO	200	0/20	9 549	9 364	9 454. 9
				TS	280	19/20	9 580	9 573	9 579. 65
				CPSO	200	17/20	9 580	9 565	9 577. 8
weish19. dat	5	70	7 698	BPSO	200	0/20	7 683	7 340	7 546. 4
				TS	280	20/20	7 698	7 698	7 698
				CPSO	200	17/20	7 698	7 683	7 695. 8
weish20. dat	5	70	9 450	BPSO	200	0/20	9 408	9 154	9 308. 9
				TS	280	17/20	9 450	9 430	9 447
				CPSO	200	18/20	9 450	9 433	9 448. 3
weish25. dat	5	80	9 939	BPSO	200	0/20	9 801	9 553	9 678. 5
				TS	320	17/20	9 939	9 936	9 938. 55
				CPSO	200	16/20	9 939	9 923	9 937. 1
weish26. dat	5	90	9 584	BPSO	200	0/20	9 283	8 870	9 083. 1
				TS	360	19/20	9 584	9 542	9 581. 9
				CPSO	200	15/20	9 584	9 543	9 577. 8
weish27. dat	5	90	9 819	BPSO	200	0/20	9 551	9 071	9 307. 3
				TS	360	19/20	9 819	9 816	9 818. 85
				CPSO	200	20/20	9 819	9 819	9 819
weish28. dat	5	90	9 492	BPSO	200	0/20	9 345	8 877	9 048. 3
				TS	360	13/20	9 492	9 469	9 483. 5
				CPSO	200	15/20	9 492	9 438	9 482. 7
weish29. dat	5	90	9 410	BPSO	200	0/20	9 287	8 720	9 026
				TS	360	20/20	9 410	9 410	9 410
				CPSO	200	19/20	9 410	9 394	9 409. 2
weish30. dat	5	90	11 191	BPSO	400	0/20	11 098	10 787	10 941
				TS	360	19/20	11 191	11 182	11 190. 5
				CPSO	400	13/20	11 191	11 182	11 190
pet3. dat	10	15	4 015	BPSO	30	14/20	4 015	4 005	4 012
				TS	60	19/20	4 015	4 005	4 014
				CPSO	30	20/20	4 015	4 015	4 015
pet5. dat	10	28	12 400	BPSO	100	3/20	12 400	12 330	12 376
				TS	112	18/20	12 400	12 370	12 398
				CPSO	100	19/20	12 400	12 380	12 399
pb7. dat	30	37	1 035	BPSO	100	0/20	1 034	936	1 009. 5
				TS	148	20/20	1 035	1 035	1 035
				CPSO	100	19/20	1 035	1 034	1 034. 95
pb6. dat	30	40	776	BPSO	100	7/20	776	679	724. 5
				TS	160	20/20	776	776	776
				CPSO	100	20/20	776	776	776



a) weing8. dat($m = 2, n = 105$)　　b) pb6. dat ($m = 30, n = 40$)

图 2　元胞微粒群算法和二进制微粒群算法部分算例寻优效果对比

Fig. 2 Comparison results of CPSO and BPSO

由表 2 的对比数据可以看出，不论获优次数、最优解的值还是最劣解的值、平均解的值元胞微粒算法均远远优于二进制微粒群算法. 二进制微粒群算法很多情况下没有搜索到最优解或者获优次数很少，极易陷入局部最优而停滞不前. 根据微粒群算法原理[1,2,4]，问题的最优解往往在当前最优解的周围，因此需要作进一步局部搜索. 二进制微粒群算法搜索能力较弱，一旦陷入局部极值就无法摆脱，在增加迭代次数的情况下也无法避免. 从图 2 可以看出，元胞微粒算法不仅在最终优化结果上，而且在收敛速度上相比于二进制微粒群算法都有显著的改善，可以更快的达到全局最优.

元胞微粒群算法是将元胞自动机原理和微粒群算法相融合的一种优化算法. 在微粒群算法中，微粒之间进行优化信息的交流与共享，不断调整自己的位置，向最优解方向搜索[16]. 而基于元胞自动机原理，重复简单的运算规则能描述复杂行为，反复进行局部的交互作用能实现全局优化. 分布在元胞空间中的微粒在寻优的同时，按照元胞演化规则在邻居范围内作同步更新，避免算法早熟收敛，增强全局优化能力; 每个微粒都具有感知能力[17]，通过优化信息共享机制调整搜索方向，逐渐逼近全局最优解. 在元胞微粒算法中利用元胞及其邻居来保持种群的多样性，增强算法搜索的多样性; 按演化规则寻优也有助于微粒发现较其目前个体最优更优的位置，从而提高其搜索速度，若发现较当前全局最优更优的位置，有利于提高整个种群的搜索速度. 元胞微粒算法性能与禁忌搜索算法相当，部分算例的获优次数等指标优于禁忌搜索算法，除个别算例外，最大迭代次数要少于禁忌搜索算法.

3　结束语

本文将元胞自动机和离散微粒群算法有机结合，提出一种元胞微粒群算法. 多维背包问题的测试结果表明，元胞微粒群算法克服了离散微粒群算法易陷入局部极值的问题，能以较快的速度达到全局最优，具有较高的全局寻优性能.

参 考 文 献:

[1] Kennedy J, Eberhart R C. Particle swarm optimization [C] // Proceedings of the IEEE International Conference on Neural Networks (Perth) , IV. Piscataway, NJ, USA: IEEE Service Center, 1995: 1942 – 1948.

[2] Eberhart R C, Kennedy J. A new optimizer using particle swarm theory [C] // The 6th Int'l Symposium on Micro Machine and Human Science. Nagoya, Japan, 1995: 39 – 43.

[3] 沈林成，霍霄华，牛轶峰. 离散粒子群优化算法研究现状综述 [J]. 系统工程与电子技术，2008，30(10) : 1986 – 1990，1994.

Shen Lincheng, Huo Xiaohua, Niu Yifeng. Survey of discrete particle swarm optimization algorithm [J]. Systems Engineering and Electronics, 2008, 30(10): 1986 – 1990, 1994. (in Chinese)

[4] Kennedy J, Eberhart R C. A discrete binary particle swarm optimization [C]//Proceedings of 1997 Conference on System, Man, and Cybernetics. Piscataway, NJ, USA: IEEE Service Center, 1997: 4104 – 4109.

[5] Onwubolu GC, Babu BV. New Optimization Techniques in Engineering [M]. Berlin: Springer-Verlag, 2004.

[6] SalmanA, AhmanI, Al-MadaniS. Particle swarm optimization for task assignment problem [J]. Microprocessors and Microsystems, 2002, 26, 363 – 371.

[7] Von Neumann J, Burks A W. Theory of Self Reproducing Automata [M]. Urbana: University of Illinois Press, 1966.

[8] 张永安, 白学志. 复杂系统研究的重要工具——细胞自动机及其应用 [J]. 自然杂志, 1997, 192 – 195, 196.
Zhang Yongan, Bai Xuezhi. A important tool for complex system research: Cellular automata and its application [J]. Nature Magazine, 1997, 192 – 195, 196. (in Chinese)

[9] 谢惠明. 复杂性与动力系统 [M]. 上海: 上海科技教育出版社, 1994.
Xie Huiming. Complexity and Dynamical Systems [M]. Shanghai: Shanghai Technology and Education Press, 1994. (in Chinese)

[10] 马　良, 王龙德. 背包问题的蚂蚁优化算法 [J]. 计算机应用, 2001, 21(8): 4 – 5.
Ma Liang, Wang Longde. Ant optimization algorithm for knapsack problem [J]. Computer Application, 2001, 21(8): 4 – 5. (in Chinese)

[11] 曾建潮, 介　婧, 崔志华. 微粒群算法 [M]. 北京: 科学出版社, 2004.
Zeng Jianchao, Jie Jing, Cui Zhihua. Particle Swarm Optimization [M]. Beijing: Science Press, 2004. (in Chinese)

[12] 贺毅朝, 王彦祺, 刘建芹. 一种适于求解离散问题的二进制粒子群优化算法 [J]. 计算机应用与软件, 2007, 24(1): 157 – 159.
He Yichao, Wang Yanqi, Liu Jianqin. A new binary particle swarm optimization for solving discrete problems [J]. Computer Applications and Software, 2007, 24(1): 157 – 159. (in Chinese)

[13] Shi Y, Eberhart R C. Empirical study of particle swarm optimization [C]//Proceedings of the 1999 Congress on Evolutionary Computation. Piscataway, NJ, USA: IEEE Service Center, 1999: 1945 – 1950.

[14] 潘全科, 王　凌, 高　亮. 离散微粒群优化算法的研究进展 [J]. 控制与决策, 2009, 24(10): 1441 – 1449.
Pan Quanke, Wang Ling, Gao Liang. The-state-of-art discrete particle swarm optimization algorithms [J]. Control and Decision, 2009, 24(10): 1441 – 1449. (in Chinese)

[15] 贺一, 邱玉辉, 刘光远, 等. 多维背包问题的禁忌搜索求解 [J]. 计算机科学, 2006, 33(9): 169 – 172.
He Yi, Qiu Yuhui, Liu Guangyuan, et al. A Tabu Search algorithm for the multidimensional knapsack problems [J]. Computer Science, 2006, 33(9): 169 – 172. (in Chinese)

[16] 韩　毅, 唐加福, 牟立峰, 等. 粒子群算法求解无能力约束生产批量计划问题 [J]. 管理科学学报, 2008, 11(5): 33 – 40.
Han Yi, Tang Jiafu, Mu Lifeng, et al. Particle swarm optimization algorithm for solving incapacitated multilevel lot-sizing problems [J]. Journal of Management Sciences in China, 2008, 11(5): 33 – 40. (in Chinese)

[17] 肖人彬, 陶振武. 群集智能研究进展 [J]. 管理科学学报, 2007, 10(3): 80 – 96.
Xiao Renbin, Tao Zhenwu. Research progress of swarm intelligence [J]. Journal of Management Sciences in China, 2007, 10(3): 80 – 96. (in Chinese)

Cellular particle swarm optimization algorithm and its application to multi-dimensional knapsack problem

LIU Yong[1,2], *MA Liang*[1]

1. School of Management, University of Shanghai for Science and Technology, Shanghai 200093, China;
2. Department of Fundamental Teaching, Yancheng Institute of Technology, Yancheng 224051, China

Abstract: Aiming at the premature convergence problem in discrete particle swarm optimization algorithm, a

novel cellular particle swarm optimization algorithm is proposed, which is based on the principles of cellular automata and discrete particle swarm optimization algorithm. Cellular and its neighbor are introduced into the algorithm to maintain the swarm's diversity and the algorithm uses evolutionary rule of cellular in local optimization to avoid local optima. Simulated tests of multi-dimensional knapsack problem and comparisons with other algorithms show the algorithm is feasible and effective and the algorithm has strong global optimization ability.

Key words: cellular automata; particle swarm optimization algorithm; multi-dimensional knapsack problem; optimization

(上接第 68 页)

定理 3 的证明

依据文中的定义,社会最优下的社会福利与分权供应链时的社会福利之间的比值为

$$\rho^d = \frac{W(Q^w)}{W(Q^d)} = \frac{\int_0^{Q^w} p\mathrm{d}x - cQ^w}{\int_0^{Q^d} p\mathrm{d}x - cQ^d}$$

$$= \frac{\int_0^{Q^I} \overline{F}(x)\,\mathrm{d}x + \int_{Q^I}^{Q^w} \overline{F}(x)\,\mathrm{d}x - rQ^w}{\int_0^{Q^d} \overline{F}(x)\,\mathrm{d}x - rQ^d} \qquad (A17)$$

已知$\rho^d \geqslant 1$,$\int_0^{Q^d} \overline{F}(x)\,\mathrm{d}x \geqslant Q^d \overline{F}(Q^d)$,并且类似定理 1 中的技术处理,将$\overline{F}(Q^w)$可以表示成$h(Q)$的指数函数[14],不等式(A17)放松为

$$\rho^d \leqslant \frac{g(Q^d)}{1-g(Q^d)} \frac{r(\overline{F}(Q^d)/r)^{1/g(Q^d)} - \overline{F}(Q^d)}{\overline{F}(Q^d) - r} \qquad (A18)$$

固定$g(Q^d)$,上述不等式的右端项是关于$\overline{F}(Q^d)$的增函数. 因此,根据表达式(A13),分两种情形进行讨论:

(1) 当 $g(Q^d) \in [n(1-r)/(n+2), (n+1-\sqrt{(n-1)^2+4nr})/2]$ 时,$\overline{F}(Q^d) \leqslant 1$,(A18)放松为$\rho^d \leqslant \frac{g(Q^d)}{1-g(Q^d)} \frac{r^{1-1/g(Q^d)} - 1}{1-r}$. 可以验证其右端项是关于$g(Q^d)$的减函数,故当$g(Q^d)$趋向$k = n(1-r)/(n+2)$时,上界可以进一步放松为$\rho^d \leqslant \frac{k}{1-k} \frac{r^{1-1/k} - 1}{1-r}$. 并且当$r \to 1$时,$k \to 0$,$\rho^d \to \frac{n}{n+2}(e^{1+2/n} - 1)$.

(2) 当$0 \leqslant g(Q^d) \leqslant n(1-r)/(n+2)$时,$\overline{F}(Q^d) \leqslant nr/(n-(n+2)g(Q^d))$,(A18)放松为$\rho^d \leqslant \frac{n}{(n+2)} \frac{(n/(n-(n+2)g(Q^d)))^{1/g(Q^d)-1} - 1}{1-g(Q^d)}$. 可以验证其右端项是关于$g(Q^d)$的增函数,故当$g(Q^d)$趋向$k = n(1-r)/(n+2)$时,上界可以进一步放松为$\rho^d \leqslant \frac{n}{n+2} \frac{(n/(n-(n+2)k))^{1/k-1} - 1}{1-k}$. 并且当$r \to 1$时,$k \to 0$,$\rho^d \to \frac{n}{n+2}(e^{1+2/n} - 1)$.

综合上述两种情形,定理 3 得证.