基于多点速度向量和自适应速度值的离散二进制粒子群算法改进

沈佳杰 江 红 王 肃
(华东师范大学信息科学技术学院 上海 200241)

摘 要 针对标准的离散二进制粒子群算法在高维环境下迭代速度慢和易早熟的缺点,通过引入多点速度向量和自适应的速度计算方法,提出一个多点基于速度向量和自适应速度值的改进的自适应离散二进制粒子群算法,通过理论推导改进的离散粒子运算法可有效提高离散差分进化算法对于复杂问题先的全局最优值搜索能力和离散粒子群算法对于复杂优化问题的收敛速度。实验验证了理论推导的结果。

关键词 离散问题优化,粒子群算法,多点速度向量,自适应速度值
中图法分类号 TP18 **文献标识码** A

Improved Binary Particle Swarm Optimization Algorithm Based on Multi Velocity Vector and Adaptive Speed Value

SHEN Jia-jie JIANG Hong WANG Su
(School of Information Science and Technology, East China Normal University, Shanghai 200241, China)

Abstract Aiming to easy to prematurity and low iteration speed problem of the standard binary particle swarm optimization (BPSO) algorithm in high dimension environment, using the the multi velocity vector and adaptive speed value, an improved discrete binary particles swarm optimization based on multi velocity vector and adaptive speed value was proposed. Though theoretical derivation, the correctness of improved discrete particle swarm optimization algorithm was proofed. The correctness of the theoretical derivation was verified by the experiment.

Keywords Discrete problem optimization, Particle swarm optimization, Multi velocity vector, Adaptive speed value

标准粒子群算法(particles swarm optimization algorithm, PSO)最早是由 Kennedy, Eberhart 在 1995 年提出的[1],其因优化问题的良好性能和可扩展性而被广泛应用于各个不同的领域,与此同时也不少学者对于标准的粒子群算法的改进进行着讨论,如动态改变惯性系数权值[2,3]和引入种群熵概念[4]以及引入梯度计算[5]等方法。在 1997 年两名原作者又将标准的粒子群算法进行改进,提出了一个离散二进制粒子群算法[6](binary particles swarm optimization algorithm, BPSO),在此以后又有许多的学者对粒子群算法进行了改进,使其可以更好地应用到离散的条件下,如二次分配问题[7]、零空闲流水线调度问题[8]、车辆路径问题[9]、指派问题[10]、最小生成树问题[11]等很多方面。对于这一类问题还可以参考文献[12]。

本文通过引入多点向量和自适应速度值,提出基于多点向量和自适应速度值的粒子群算法,并通过理论推导证明改进的二进制粒子群算法较标准的二进制粒子群算法在问题最优值搜索上有了一定的改进,实验验证了理论推导的结论。

本文第 1 节主要介绍标准粒子群算法和几种经典的离散粒子群算法;第 2 节主要介绍本文中改进的粒子群算法的定义、算法步骤以及相关性质;第 3 节主要介绍本文中实验,并对实验的数据进行展示;最后是本文的结论。

1 标准粒子群算法和几种常见离散粒子群算法

1.1 标准粒子群算法

假设有若干个粒子组成的群体,对 n 维的空间进行搜索,对于每一个粒子来说都要考虑自己的历史适应度最好的点和群体内适应度最好的点,在其基础上考虑自己下一步位置取值。第 i 个粒子的速度 X_i 和位置 V_i 表示如下:

$$\begin{cases} X_i=(x_{i1},x_{i2},\cdots,x_{iD}) \\ V_i=(v_{i1},v_{i2},\cdots,v_{iD}) \end{cases} (1\leqslant i\leqslant n) \tag{1}$$

式中,n 为种群(swarm)的大小,D 为解空间维数 。

第 i 个粒子位置的局部最优值为:

$$p_i=(x_{i1},x_{i2},\cdots,x_{in}) \tag{2}$$

式中,$x_{i1},x_{i2},\cdots,x_{in}$ 为当前粒子的历史最优值点的坐标。

粒子位置的全局最优值为:

$$p_{gi}=(x_{gi1},x_{gi2},\cdots,x_{gin}) \tag{3}$$

式中,$x_{gi1},x_{gi2},\cdots,x_{gin}$ 是所有粒子历史最优值点的坐标。

根据以上的定义,第 i 个粒子的速度和位置更新公式如

本文受国家 863 基金项目(2013AA01A211)资助。

沈佳杰(1989—),男,硕士生,主要研究方向为软件应用技术;江 红(1969—),女,副教授,主要研究方向为软件应用技术;王 肃(1980—),女,讲师,主要研究方向为信息系统、智能优化。

下：

$$v_{id}^{k+1}=\omega v_{id}^{k}+c_1\xi(p_{id}^{k}-x_{id}^{k})+c_2\eta(p_{gd}^{k}-x_{id}^{k}) \tag{4}$$

$$x_{id}^{k+1}=x_{id}^{k}+v_{id}^{k+1}$$

式中，ω 为惯性系数，c_1、c_2 为学习因子，其取值通常是 2，ξ、η 为两个在(0,1)均匀分布的随机数，ξ、$\eta\in\cup(0,1)$。

标准 PSO 算法流程如下：

步骤 1　初始化种群的位置以及速度信息。

步骤 2　根据式(2)和式(3)计算粒子位置的局部最优值和全局最优值。

步骤 3　根据式(4)，更新粒子的速度和位置。

步骤 4　如果迭代步骤数超过迭代步骤数上限或最优值被找到，转到步骤 5，否则跳转到步骤 2。

步骤 5　输出最优值。

大量的实验以及论文可以证明，以上标准的粒子群算法可以有效地找到各类目标函数的最优值。

1.2　离散粒子群算法

1.2.1　二进制粒子群算法 BPSO

二进制粒子群算法 BPSO(binary PSO)是 1997 年由 Kennedy 博士和 Eberhart 博士针对于 0-1 规划问题首先提出的[6]。在 BPSO 中，个体是二进制串，而每一维由 0-1 组成，其速度表示为：

$$v_{id}^{k+1}=\omega v_{id}^{k}+c_1\xi_1(p_{id}^{k}-x_{id}^{k})+c_2\xi_2(p_{gd}^{k}-x_{id}^{k}) \tag{5}$$

式中，x_{id}^{k} 为在第 k 步时第 i 个粒子的第 d 维取值。

二进制粒子群算法的位置公式如下：

$$x_{id}^{k}=\begin{cases}1, & rand(0,1)<Sig(v_{id}^{k})\\ 0, & \text{else}\end{cases} \tag{6}$$

式中，$rand(0,1)$为(0,1)之间的均匀分布的随机数，$sig(x)=\frac{1}{1+e^{-x}}$，$v_{id}^{k}$ 为在第 k 步时第 i 个粒子的第 d 维取值。

1.2.2　将粒子群算法直接用于离散问题求解

由于 BPSO 在许多的场合下，并不十分有效，有部分学者直接将连续的粒子群算法应用到不同的问题场景中，如整数规划问题[15]、静态任务分配问题[16]。通过实验证明，改进的 PSO 较经典的遗传算法可以更有效地解决不同离散优化问题。

1.2.3　重定义算法算子

另一些学者在基本的粒子群的框架之下，对标准粒子群算法中的算子进行重新定义，其中比较有代表性是将 TSP-DPSO 用于 TSP 问题的求解上，在该算法中将粒子的位置用所有城市的序列表示，则所有的排列就构成了粒子的搜索空间。该算法引入了交换子和交换序列的概念，其交换算子的定义如下：

$$S=Swap(i,j) \tag{7}$$

是指交换粒子第 i 个和第 j 元素的位置，一个交换子集则是指一组以特定顺序排列的交换子。粒子的速度则是指粒子为达到最优解而需要对当前位置执行的基本交换序。基于此概念，TSP-DPSO 算法对粒子群中的加减法等操作算子进行了重新定义，其更新公式如下：

$$V_{t+1}=c_1V_t\oplus c_2(P_{i,t}-X_t)\oplus c_3(P_{g,t}-X_t) \tag{8}$$

$$X_{t+1}=X_t+V_{t+1} \tag{9}$$

式中，粒子的位置和速度分别为 X_t 和 V_t；c_1、c_2 和 c_3 是随机生成的学习因子；$P_{i,t}$和 $P_{g,t}$则是粒子的局部最优值以及全局最优值。

TSP-DPSO 详细的过程还可以参考文献[13]。

2　本文改进的离散粒子群算法

2.1　本文中的假设与定义

本文中改进的离散二进制粒子群算法的假设以及定义如下：

假设 1　每一个粒子的局部最优点的领域内都有以一定概率存在的全局最优值点。

假设 2　二进制离散粒子算法粒子群初始时，粒子群的粒子点是均匀分布的。

假设 3　对于目标问题的最优值点，每一维取 0 或 1 的概率是相同的。

定义 1　对于每一个粒子点轨迹最优值点，存在一个局部最优值点 p_{id}^{k}，且对于每一个局部最优值点，存在一个向量指向这个局部最优值点，此点称为局部最优值加速度，记为：

$$a_{id}^{k}=c_i\xi_i^{k}(p_{id}^{k}-x_{id}^{k}) \tag{10}$$

式中，p_{id}^{k}为第 k 步第 i 个粒子的局部最优值，c_i 为学习因子，ξ_i^{k} 为[0,1]之间随机数。

定义 2　对于每一个粒子的每一维，存在一个速度值，这个速度值与迭代的步骤数成正比，称为自适应速度值，记为：

$$ch_{id}=c_2\xi_{id}st_{id}(\bar{x}_{id}-x_{id})(1-sig(fit_i)) \tag{11}$$

式中，x_{id}为第 i 个粒子第 k 维的取值情况，$\bar{x}_{id}$ 为 x_{id} 的取反(即 x_{id}取 1 时、取 0，即 x_{id}取 0 时、取 1)，ξ_{id}为一个在(0,1)之间的随机数，st_{id}为第 i 个粒子第 d 维未发生改变的步骤数，fit_i 为第 i 粒子当前的适应度值。

由以上的定义可知，如果第 i 粒子的第 k 维为 0，以上函数为正，而当第 i 粒子的第 k 维为 1 时，以上函数为负，所以上式可以变形为：

$$\begin{cases}-c_2\xi_{id}st_{id}(1-sig(fit_i)), x_{id}=1\\ c_2\xi_{id}st_{id}(1-sig(fit_i)), x_{id}=0\end{cases} \tag{12}$$

因此，本文中的算法的速度更新公式等于对于各个局部最优值加速度与自适应速度值的叠加，记为：

$$\begin{aligned}v_{id}^{k+1}&=\omega v_{id}^{k}+\sum_{i=1}^{n}a_{id}^{k}+c_2\eta_{id}st_{id}^{k}(\bar{x}_{id}^{k}-x_{id}^{k})(1-sig(fit_i))\\&=\omega v_{id}^{k}+\sum_{i=1}^{n}c_i\xi_i^{k}(p_{id}^{k}-x_{id}^{k})+c_2\eta_{id}st_{id}^{k}(\bar{x}_{id}^{k}-x_{id}^{k})(1-\\&\quad sig(fit_i))\end{aligned} \tag{13}$$

定义 3　问题空间可行域状态点与不可行域状态点的比值，称为问题可行域比值，记为：

$$rd_i=\frac{Fr_i}{Ir_i} \tag{14}$$

式中，Fr_i 为第 i 个问题可行域包含的离散点的数量，Ir_i 为第 i 个目标问题包含的离散点的数量。

2.2　算法描述

本文中改进的离散二进制粒子群算法的步骤如下：

步骤 1　初始种群的位置以及速度信息。

步骤 2　根据式(2)和式(11)，更新每一个粒子点的局部最优值点信息和每一个粒子每一维的迭代步数信息以及自适应速度值。

步骤 3　根据式(13)和式(4)，更新所有的粒子点的速度以及位置信息。

步骤 4　如果迭代步骤数超过迭代步骤数上限或最优值被找到，转到步骤 5，否则返回步骤 2。

步骤 5　输出最优值。

2.3　本文中的算法的定理以及性质证明

本文中改进的离散二进制粒子群算法性质定理证明如下。

定理 1　若所有的初始粒子点都落在不可行域内，标准的粒子群算法开始时将是盲目搜索。

证明：由于初始情况下，所有的粒子都在不可行域内，因此在初始阶段每一个粒子的局部最优值和所有粒子的全局最优值的收益值都为 0，所以速度更新式(4)变为：

$$v_{id}^{k+1}=\omega v_{id}^{k}+c_1\xi_1(p_{id}^{1}-x_{id}^{k})+c_2\xi_2(p_{gd}^{1}-x_{id}^{k}) \tag{15}$$

式中，p_{id}^{1} 为初始化时第 i 个局部最优值点，p_{gd}^{1} 为初始化时全局的全局最优值点。

又因为 p_{id}^{1}、p_{gd}^{1} 和 x_{id}^{1} 在初始化时是随机生成的，而随机生成的数据点都在不可行域内，所以 v_{id}^{k+1} 为一个盲目搜索的随机值，进而根据其速度值更新的粒子群中粒子的位置信息也是一个盲目搜索的随机值。

推论 1　在问题的可行域比值较小和空间较大时，标准的离散二进制粒子群算法易陷入盲目搜索和局部最优。

证明：可行域比值较小和空间较大时，粒子的局部最优值可能落在不可行域内；由于不可行域内的函数值为 0，因此速度更新公式变成：

$$v_{id}^{k+1}=\omega v_{id}^{k}+c_1\xi_1(p_{id}^{1}-x_{id}^{k})+c_2\xi_2(p_{gd}^{k}-x_{id}^{k}) \tag{16}$$

式中，p_{id}^{1} 为第 i 个粒子第 1 步时的局部最优值。

由于 p_{id}^{1} 是初始化时的随机值，因此无法为速度的更新提供依据，由于 p_{gd}^{k} 可能是在一个局部最优值点的领域内，因此标准的离散二进制粒子群算法在问题的可行域比值较小和空间较大时，易陷入盲目搜索和局部最优。

定理 2　改进的离散二进制粒子群算法在第 k 步有一个粒子 i 陷入局部最优值时，必然存在一个 n，使得粒子 i 在 $k+n$ 步时至少有一维翻转，且该粒子的适应度值越大，n 的取值也越大。

证明：

由于在第 k 步粒子 i 陷入了局部最优值其位置信息 x_{id}，因此 st_{id} 的值会不断增加，又因为自适应速度值的计算式(11)，在 st_i 无穷大时，其对于当前粒子 i 该位取反的概率无穷大，所以必然存在一个 n，使得粒子 i 在 $k+n$ 步时至少有一维翻转。

又因为在式(11)中，存在一个因子 $1-sig(fit_i)$ 使当前适应度的一个值域为[0.5,1)的减函数，所以粒子的适应度越高，自适应速度值越小，需要经过的步骤数 n 越大。

推论 2　在粒子点数量和迭代步骤数足够的情况下，本文中改进的离散二进制粒子群算法可以找到全局最优值点。

证明：由定理 2 可知，本文中的算法的粒子点由于不会陷入自适应速度值，因此也不会陷入问题的局部最优值，又因为本文中的算法速度更新公式中位置更新速度是由所有的粒子最优值位置组成的，所以在粒子数足够的情况下，必然有一个粒子的最优值在全局最优值的邻域内，因此改进的离散二进制粒子群算法可以在粒子点数量和迭代步骤数足够的情况下，找到全局最优值点。

3　实验

本文中使用经典的 0-1 背包问题进行测试[14]，本文对于物品数为 10、20、50、60、100、500、1000 这 8 种不同的测试问题进行 10 次独立的实验，表 1 中展示了本文中实验使用的问题规模为(10,20,50,60)的 4 个测试用例 1－4，问题规模为(100,200,500,1000)的问题用例 5－8 中的测试数据由机器随机生成。表 2 中主要展示了在不同问题用例下，标准的离散二进制优化算法和本文中改进的离散二进制算法在十次算法迭代中所找到的最优解、平均解和最差解和各个不同的解之间的方差以及找到最优解时最小的迭代步骤数、最大的迭代步骤数、平均的迭代步骤数以及所有最优值之间的标准差。

其中 SPO 表示标准的离散粒子群算法，IPO 表示本文中改进的离散二进制粒子群算法。

表 1　本文中实验测试用例实例编号

	背包容量	物品总数	物品重量	物品价值
1	269	10	{95,4,60,32,23,72,80,6,265,46}	{55,10,47,5,4,50,8,61,85,87}
2	878	20	{92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58}	{44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63}
3	1000	50	{80,82,85,70,72,70,66,50,55,25,50,55,40,48,50,32,22,60,30,32,40,38,35,32,25,28,30,22,50,30,45,30,60,50,20,65,20,25,30,10,20,25,15,10,10,10,4,4,2,1}	{220,208,198,192,180,180,165,162,160,158,155,130,125,122,120,118,115,110,105,101,100,100,98,96,95,90,88,82,80,77,75,73,72,70,69,66,65,63,60,58,56,50,30,20,15,10,8,5,3,1}
4	2400	60	{135,133,130,11,128,123,20,75,9,66,105,43,18,5,37,90,22,85,9,80,70,17,60,35,57,35,61,40,8,50,32,40,72,35,100,2,7,19,28,10,22,27,30,88,91,47,68,108,10,12,43,11,20,37,17,4,3,21,10,67}	{350,310,300,295,290,287,283,280,272,270,265,251,230,220,215,212,207,203,202,200,198,196,190,182,181,175,160,155,154,140,132,125,110,105,101,92,83,77,75,73,72,70,69,66,60,58,45,40,38,36,33,31,27,23,20,19,10,9,4,1}

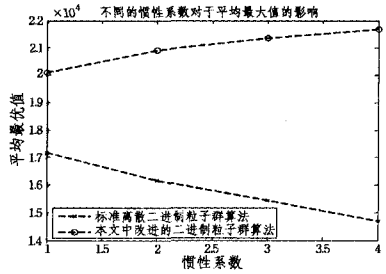
如表 2 所列，在问题规模比较小的情况下(如问题实例 1、问题实例 2)，标准的离散二进制粒子群算法和本文中改进的离散二进制粒子群算法都可以在较短迭代步骤数内找到几乎相同的最优值，而随着问题规模的变大，两种算法之间可以找到最优值的差距变大，甚至在部分情况下(如问题实例 7、问题实例 8)最优值的差距达到了将近一倍。

表 2　标准的粒子群算法和本文中的算法在算法迭代终止时的性能对比

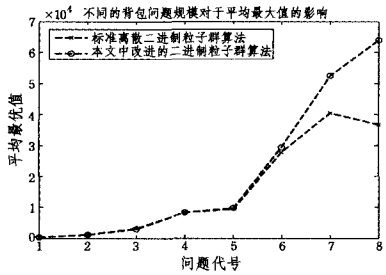
问题实例	问题规模	使用算法	惯性系数	最优解	最差解	平均解	最小迭代数	最大迭代数	平均迭代数	标准差
1	10	SPSO	0.8	269	269	269	1	479	86.2	0
			0.7	269	269	269	3	144	52	0
			0.6	269	269	269	2	137	34.8	0
			0.5	269	269	269	3	94	31.1	0
		IPO	0.8	269	269	269	5	249	105.6	0
			0.7	269	269	269	2	128	21.6	0
			0.6	269	269	269	5	111	52.7	0
			0.5	269	269	269	1	99	31.5	0
2	20	SPSO	0.8	1024	1024	1024	11	57	28.9	0
			0.7	1024	1024	1024	15	72	36	0
			0.6	1024	1024	1024	4	47	17.6	0
			0.5	1024	1024	1024	15	120	56.4	0
		IPSO	0.8	1024	1024	1024	36	344	110.8	0
			0.7	1024	1024	1024	16	117	48.4	0
			0.6	1024	1024	1024	6	173	52.3	0
			0.5	1024	1024	1024	15	140	62.6	0
3	50	SPSO	0.8	3099	3077	3077	186	949	591.4	5.989806
			0.7	3092	3072	3089.1	199	897	632.9	6.201254
			0.6	3094	3072	3083.7	441	973	757.5	7.933053
			0.5	3086	3061	3084.4	574	994	825.7	7.015063
		IPSO	0.8	3063	3019	3078.9	492	998	731.7	15.46322
			0.7	3076	3036	3056.7	305	997	726.3	12.8759
			0.6	3084	3028	3062.3	208	994	694.3	16.53313
			0.5	3092	3054	3071.4	274	989	707.6	12.38458
4	60	SPSO	0.8	8362	8329	8354.3	62	977	375.4	10.11105
			0.7	8362	8350	8356.4	120	964	521.2	4.501851
			0.6	8362	8291	8331.9	406	963	643.9	22.14322
			0.5	8345	8258	8310	108	983	653.1	29.86265
		IPSO	0.8	8356	8326	8350.3	35	873	461.2	9.775821
			0.7	8362	8325	8351.4	84	957	565	10.30857
			0.6	8362	8303	8348.3	65	792	323.6	19.28183
			0.5	8362	8326	8349.1	89	878	321.8	13.39527
5	100	SPSO	0.8	10480	9953	10216.7	806	998	919.5	171.1478
			0.7	9839	9309	9647.8	562	992	823.4	189.5467
			0.6	9573	8583	9128	805	914	866.8	304.1812
			0.5	8734	8030	8358.4	674	989	889.8	248.4777
		IPSO	0.8	10293	8465	9477.5	653	998	859	522.4626
			0.7	9944	8755	9535.6	536	947	789.7	414.2627
			0.6	10380	9442	9860.7	736	987	885.4	326.3025
			0.5	10401	9494	9929.2	690	998	854.5	305.0395
6	200	SPSO	0.8	29248	28693	28979.6	704	995	919.3	163.6495
			0.7	28497	27508	28000.2	706	986	822.7	322.3772
			0.6	28132	26926	27180.2	485	998	854.4	349.36
			0.5	27215	26054	26467.5	366	996	826.3	423.3982
		IPSO	0.8	29484	28527	29060.8	806	996	917.7	330.5806
			0.7	29727	28165	29228.5	660	989	896.1	429.024
			0.6	29981	29077	29614.3	634	994	904.4	245.9241
			0.5	30167	29372	29731.6	621	977	871.3	256.4446
7	500	SPSO	0.8	47226	44394	45777.7	744	1000	906.7	1020.09
			0.7	42758	40943	42037.5	696	955	821.6	609.9776
			0.6	39978	37158	38167.4	423	983	746.8	885.3597
			0.5	37263	34756	35888.4	169	981	603.9	875.7888
		IPSO	0.8	52046	48358	50260.6	894	1000	966.3	1100.259
			0.7	53368	51420	52424.4	832	999	963.5	763.5581
			0.6	55122	51599	53219.1	880	999	963.6	1203.791
			0.5	54984	52839	53759.6	829	998	963.7	613.0578
8	1000	SPSO	0.8	41769	37445	39597.5	12	976	424.1	1421.92
			0.7	39810	34712	36865.4	9	993	269.2	1639.529
			0.6	38364	34167	36318.5	5	223	64.3	1368.788
			0.5	37049	31292	34165.5	5	537	101.1	2048.9
		IPSO	0.8	61939	53944	59320.8	949	999	984.3	2397.612
			0.7	65484	60075	63474.4	929	1000	983.6	1598.109
			0.6	68092	63177	65469.4	963	999	985.2	1599.737
			0.5	69125	65391	67324.5	971	1000	988	1191.27

图 1(a)展示了不同的惯性系数的情况下，标准的离散二进制粒子群算法和本文中的离散二进制粒子群算法迭代终止时，不同惯性系数与目标函数平均最优值之间的关系；图 1(b)展示了在不同的问题规模的情况下，标准的离散二进制粒

子群算法和本文中的离散二进制粒子群算法迭代终止时，不同问题规模与目标函数平均最优值之间的关系。



(a) 不同的惯性系数情况下，两种离散二进制粒子群算法迭代终止时最优值之间的关系

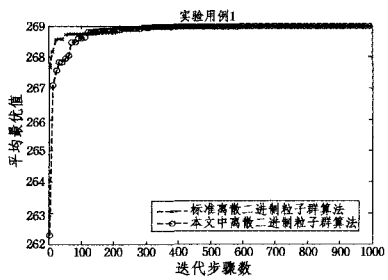


(b) 不同的问题规模情况下，两种离散二进制粒子群算法迭代终止时最优值之间的关系

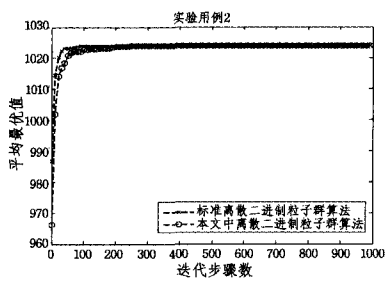
图 1　不同的条件下离散二进制粒子群算法终止时，两种离散二进制粒子群算法的平均最优值

由图 1(a)可知，在不同的惯性系数条件下，本文中改进的离散二进制粒子群算法较标准的离散二进制粒子群算法可以找到更好的最优解；由图 1(b)可知，在问题规模比较小的情况下，标准的离散二进制粒子群算法和本文中的离散二进制粒子群算法都可以在迭代终止时，找到几乎相同的最优解，随着问题规模的不断扩大，改进的离散二进制粒子群算法相对于标准的离散二进制粒子群算法的最优值的差距逐渐扩大，在问题规模最大的第 8 个问题实例中，本文中的离散二进制粒子群算法的最优值甚至达到了标准离散二进制粒子群算法的两倍。

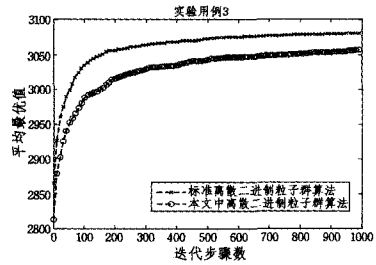
图 2 中展示了问题实例 1—8 中，标准的离散二进制粒子群算法和本文中的离散二进制粒子群算法其迭代步骤数与平均最优值的关系。



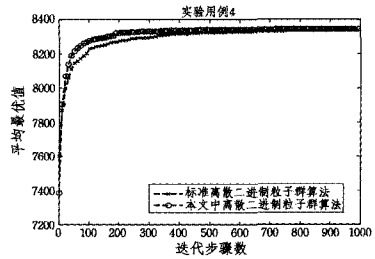
(a) 问题实例 1



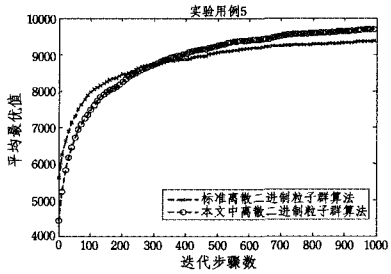
(b) 问题实例 2



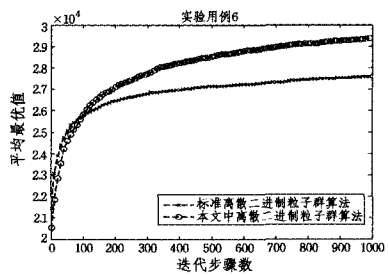
(c) 问题实例 3



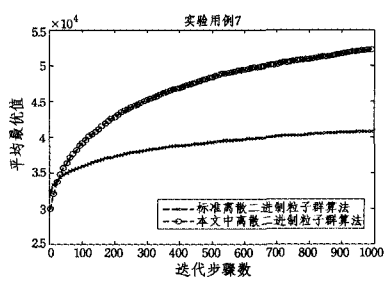
(d) 问题实例 4



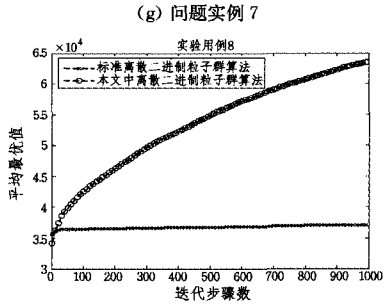
(e) 问题实例 5



(f) 问题实例 6



(g) 问题实例 7



(h) 问题实例 8

图 2　两种不同的离散二进制粒子群算法迭代步骤与平均最优值之间的关系

如图 2 所示，在不同的问题规模下，本文中改进的离散粒子群算法有较快的迭代速度，且随着问题求解难度的升高，本文中改进的离散二进制粒子群算法较标准的离散二进制粒子群算法的效果的差距也逐渐体现出来。在问题规模较小的情况下，如图 2(a)(b)(c)(d)所示，标准的离散二进制粒子群算法和本文中改进的离散二进制粒子群算法都可以在较短的迭代步骤中找到较好的最优解，但是随着问题规模的增加，改进的粒子群算法相对于粒子群算法的优势开始逐渐体现出来，如图 2(g)(h)中改进的离散二进制粒子群算法在迭代步骤数足够的情况下，改进的离散二进制粒子群算法找到的最优值甚至是标准的离散二进制粒子群算法的两倍。

结束语 本文中展示了一个改进的二进制离散粒子群算法。通过理论证明，改进的算法较原算法有更高的收敛速度以及最优值。通过实验对经典的离散优化问题背包问题进行的测试验证了本文中改进的二进制离散粒子群算法相对于标准的二进制离散粒子群算法具有较快的收敛速度，能在相同的迭代步骤数内找到更好的目标函数最优值。

由于实验有限，对本文中的算法通用性的讨论依然不足，能否找到一个在各种实验问题情况下都可以找出一个比较通用的高效离散粒子群算法，依然是一个值得研究的问题。

参 考 文 献

[1] Kennedy J, Eberhart R C. Particle swarm optimization[C]// Proceedings of IEEE International Conference on Neural Networks. Piscataway New Jersey: Institute of Electrical and Electronics Engineers, Inc. 1995: 1942-1948

[2] 张顶学，关治洪，刘新芝. 一种动态改变惯性权重的自适应粒子群算法 [J]. 控制与决策，2008，23(11)：1253-1257

[3] 王丽，王晓凯. 一种非线性改变惯性权重的粒子群算法 [J]. 计算机工程与应用，2007，43(4)：47-48

[4] 段晓东，高红霞，刘向东，等. 一种基于种群熵的自适应粒子群算法[J]. 计算机工程，2007，33(18)：222-223

[5] 王俊伟，汪定伟. 一种带有梯度加速的粒子群算法[J]. 控制与决策，2004，19(11)：1298-1300

[6] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm[C]//Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. Orlando: IEEE Inc, 1997: 4104-4108

[7] 钟一文，蔡荣英. 求解二次分配问题的离散粒子群优化算法[J]. 自动化学报，2007，33(8)：871-874

[8] 潘全科，王凌，赵保华. 解决零空闲流水线调度问题的离散粒子群算法[J]. 控制与决策，2008，23(2)：190-194

[9] 魏明，靳文舟. 求解车辆路径问题的离散粒子群算法[J]. 计算机科学，2010，37(4)：187-191

[10] 孙晓雅，林焰. 一种新的离散粒子群算法在指派问题中的应用[J]. 计算机应用研究，2009，26(11)：4091-4093

[11] 郭文忠，陈国龙. 一种求解多目标最小生成树问题的有效离散粒子群优化算法[J]. 模式识别与人工智能，2009，22(4)：597-604

[12] 张长胜，孙吉贵，欧阳丹彤. 一种自适应离散粒子群算法及其应用研究[J]. 电子学报，2009，37(2)：299-304

[13] Clerc, Maurice. Discrete particle swarm optimization, illustrated by the traveling salesman problem[M]. New optimization techniques in engineering. Berlin Heidelberg: Springer, 2004: 219-239

[14] 胡中华，赵敏. 引入侦查子群的蚁群算法求解 0/1 背包问题[J]. 贵州师范大学学报：自然科学版，2009，27(3)：82-88

[15] Parsopoulos K E, Vrahatis M N. Recent approaches to global optimization problems through particle swarm optimization[J]. Natural computing, 2002, 1(2/3): 235-306

[16] Salman A, Ahmad I, Al-Madani S. Particle swarm optimization for task assignment problem[J]. Microprocessors and Microsystems, 2002, 26(8): 363-371

(上接第 97 页)

ceptor. revive<1。我们还验证了当列车实时速度大于速度阈值时，控制器向感应器、接收器发送休眠信号，感应器和接收器必须在 1s 内进入休眠状态，以实现我们低功耗的目的；接收器收到触发信号必须在 1s 向控制器发送确认信号；感应器器收到触发信号必须在 2s 内向控制器发送确认信号，如图 9 所示。

```
acceptor.wait -->controller.work1<1
Property is satisfied.
sensor.wait -->controller.work<2
Property is satisfied.
controller.send --> sensor.revive and acceptor.revive<1
Property is satisfied.
controller.sendc --> sensor.sleeps and acceptor.sleepa<1
Property is satisfied.
```

图 9 系统实时性验证结果

结束语 列车运行过程中对攀车人员的实时感知，可以有效缩减列车进港时对列车车厢的排查时间，能够有效提高港口作业的效率，对于实现港口信息化、自动化和智能化都起到重要作用。文中设计的 3SHHR 系统实现了对重载列车的全程追踪式感知和监控，并基于速度触发条件，实现了感知系统的低功耗目标。自动机模型从系统行为逻辑正确性和时间约束两方面入手，对 3SHHR 系统进行精准刻画。基于 UppAal 工具的验证实验表明，3SHHR 系统满足状态可达性、安全性(无死锁)、活性和实时性等物联网系统的典型关键性质。为进一步正确设计 3SHHR 系统模块、详细电路和编码以及测试用例等奠定了理论基础。

参 考 文 献

[1] 姜子英，程建平，刘森林，等. 我国煤电的外部成本初步研究[J]. 煤炭学报，2008，35(11)：1325-1328

[2] 陈雍君，周磊山，余吉安. 重载铁路列车运行调整计划的序优化策略研究[J]. 铁道学报，2013，35(1)：1-7

[3] 孙景昊，白秋果，刘杰民. 港口铁水联运信息协同平台关键技术研究[R]. 北戴河：东北大学，2012

[4] Aho A V, Ullman J D. Fundations of computer science[M]. New York: Computer Science Press, 1992

[5] Lee E A. Cyber-Physical Systems-Are Computing Foundations Adequate? [C]//Position Paper for NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap. 2006

[6] Alur R, Dill D. A theory of timed automata [J]. Theoretical Computer Science, 1994, 126: 183-235

[7] 谭国真，孙景昊，王宝财，等. 时变网络中国邮路问题的时间自动机模型[J]. 软件学报，2011，22(6)：1267-1280

[8] Behrmann G, David A, Larsen K G. A tutorial on UppAal: Formal methods for the design of real-time systems [M]. Berlin Heidelberg: Springer, 2004: 200-236

[9] 李力行，金芝，李戈. 基于时间自动机的物联网服务建模与验证[J]. 计算机学报，2011，34(8)：1365-1377