一种求解动态背包问题的离散粒子群优化算法

鲁江林 何中市 陈自郁
(重庆大学计算机学院 重庆 400044)

摘 要 动态背包问题(DKP)是一类经典的动态优化问题,可以用来描述许多实际的问题。迄今为止,针对动态背包问题的研究主要集中在遗传算法上,而对粒子群优化算法的研究较少。在离散粒子群优化模型的基础上,引入环境变化的探测以及环境变化后的响应机制,提出一种求解动态背包问题的离散粒子群优化算法(DSDPSO)。将该算法和现有经典的自适应原对偶遗传算法(APDGA)在两个动态背包问题上进行了对比实验,结果表明,DSDPSO算法在环境变化后能迅速地找到最优解并稳定下来,更适合于求解动态背包问题。

关键词 粒子群优化算法,动态背包问题,DSDPSO算法,集合
中图法分类号 TP18 **文献标识码** A

Discrete Particle Swarm Optimization Algorithm for Solving Dynamic Knapsack Problem

LU Jiang-lin HE Zhong-shi CHEN Zi-yu
(College of Computer Science, Chongqing University, Chongqing 400044, China)

Abstract Dynamic knapsack problem (DKP) is a kind of classic dynamic optimization problems, which can be used to describe many practical issues. So far the study of dynamic knapsack problem has mainly focused on genetic algorithm, and particle swarm optimization algorithm is of rare application. This paper proposed a discrete particle swarm optimization algorithm based on discrete particle swarm optimization model for solving dynamic knapsack problem(DSDPSO), and introduced environment change detection and post-change response mechanism. Our algorithm was compared with the existing classical adaptive primal-dual genetic algorithm(APDGA) into two dynamic knapsack problems, and the results show that the DSDPSO algorithm can rapidly find the optimal solution and remains stable after environment variation. Consequently, this algorithm is more suitable to solve dynamic knapsack problem.

Keywords Particle swarm optimization algorithm, Dynamic knapsack problem, DSDPSO algorithm, Set

1 引言

现实世界里,很多优化问题都可归结为动态优化问题(Dynamic Optimization Problems,DOPs),目标函数会随时间发生变化,要求进化算法不仅能在特定环境下找到全局最优解,而且能在动态环境中持续追踪变化中的最优解。

动态背包问题(Dynamic Knapsack Problem,DKP)是一类经典的动态优化问题,可以用来描述许多实际的问题,比如:装载问题、动态环境下的投资问题等。这类问题的最优解会随参数的变化而发生迁移,因此要求算法能够快速、有效地跟踪最优点的变化轨迹。

目前,对于0-1动态优化问题已有一定的研究。Yang[1]提出了一种求解0-1动态优化问题的原对偶遗传算法(Adaptive Primal-dual GA,APDGA),其在Collard[2]等提出的对偶遗传算法(Dual GA,DGA)中的对偶映射运算的基础上,定义了一种新的运算——原对偶(PD)映射,使低适应值染色体有机会映射为高适应值的对偶染色体,增加了种群的多样性,从而保证种群能够适应环境的变化。但PD映射在算法迭代后期,由于选择进行PD运算的染色体条数恒定,导致效果不佳。Wang[3]在此基础上提出改进了的IPDGA,在IPDGA中,种群中所有的染色体在同一基因位点上参与PD映射的机会均等,导致对偶映射改善种群多样性的能力有限;随着进化代数的增加,缺乏对种群多样性的调整,且由于对种群中所有个体均进行对偶映射,需消耗较多的计算资源,花费较长的计算时间。前述的这些研究都存在一定的不足且都是基于遗传算法而作出的。

粒子群优化算法(Particle Swarm Optimization,PSO)作为一类基于群智能的随机优化算法,被广泛应用于求解静态优化问题,并取得了很好的效果。粒子群优化算法用于求解动态连续优化问题已有一些研究成果,但用于求解动态离散优化问题的研究却较少。关于粒子群优化算法用于求解离散优化问题的研究,Kennedy和Eberhart提出了作用于离散空间的二进制粒子群优化算法(BPSO)[4]。该算法沿用了基本粒子群优化的速度更新公式,即速度仍作用于连续空间,而位

到稿日期:2011-11-12 返修日期:2012-03-03 本文受中央高校基本科研业务费科研专项(CDJZR10180005),西部交通建设科技项目(2011318740240)资助。

鲁江林(1987—),男,硕士生,主要研究方向为粒子群优化算法、机器学习,E-mail:lujianglin.mlg@gmail.com;何中市(1965—),男,教授,博士生导师,CCF会员,主要研究方向为图像处理、机器学习、自然语言处理;陈自郁(1976—),女,博士,讲师,主要研究方向为计算智能、机器学习。

置则利用 Sigmoid 函数将其离散化。BPSO 算法通过间接优化可连续变化的二进制概率来达到优化二进制变量的目的。但是，该间接优化策略根据概率而非算法本身确定的二进制变量，未能充分利用基本粒子群优化算法的性能。陈自郁等针对变长集合组合优化问题，提出了一种离散粒子群优化模型(SDPSO)[5]。该模型将集合的概念和运算引入粒子群优化中，定义了一个可变集合搜索空间，并重新定义了粒子群的位置、速度及作用于此空间的运算规则。该工作打破了常规求解离散优化问题的 0-1 编码模式，引入了集合的思想以及集合运算，但此模型不适合于求解动态离散优化问题，本文针对上述模型的不足，提出了一种求解动态背包问题的离散粒子群优化算法。

2 问题描述

2.1 背包问题

背包问题是组合优化中一个典型的 NP 完备问题，考虑有 n 个物品，其中第 i 个物品的质量为 w_i，价值为 p_i，背包可以承受的最大重量为 W，目标函数是如何装载可以使获得的总价值最大。下面给出 0-1 编码和集合两种数学模型[5]。

0-1 背包问题数学模型：

$$\begin{cases}\max f(X)=\sum_{i=1}^{n}p_i x_i \\ \text{s.t.}\ \sum_{i=1}^{n}w_i x_i\leqslant W\end{cases} \tag{1}$$

式中，$X=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}\subseteq\{0,1\}^n$。

背包问题的集合数学模型：

$$\begin{cases}\max f(X)=\sum_{i\in X}p_i \\ \text{s.t.}\ \sum_{i\in X}w_i\leqslant W\end{cases} \tag{2}$$

式中，$X\subseteq P(U)$，$P(U)$为集合 U 的幂集，$1\leqslant|X|\leqslant n$，$U=\{1,2,\cdots,n\}$。

2.2 动态背包问题

时变背包问题是一类经典的动态背包问题，动态性主要体现在允许背包的重量限制动态变化，这里重量限制用 $W(t)$ 表示，其中 t 表示迭代次数。式(3)给出了动态背包问题的集合数学模型：

$$\begin{cases}\max f(X)=\sum_{i\in X}p_i \\ \text{s.t.}\ \sum_{i\in X}w_i\leqslant W(t)\end{cases} \tag{3}$$

式中，$X\subseteq P(U)$，$1\leqslant|X|\leqslant n$，$U=\{1,2,\cdots,n\}$，$P(U)$同式(2)。

3 基于集合的离散粒子群优化模型

模型借助集合达到优化组合优化问题的目的，其主要思想是在粒子群优化算法中引入集合的概念，通过集合的运算来模拟粒子群优化算法中速度和位置的更新。下面给出了模型中的一些主要定义[5]：

定义 1 变长集合组合优化问题的集合搜索空间 Ω 为

$$\Omega=\{X|X\subseteq P(U),1\leqslant|X|\leqslant S_{\max}\} \tag{4}$$

式中，U 为原问题的全集；$P(U)$为集合 U 的幂集；$S_{\max}$为所求问题所允许的最大子集元素个数。

定义 2(加入速度 $V_i^+(t)$) 即粒子 i 在 t 时刻可增加的元素集合；

定义 3(去除速度 $V_i^-(t)$) 即粒子 i 在 t 时刻从自己的解集合中去除的元素集合；

定义 4 集合空间中的速度与位置更新公式

$$V_i^+(t+1)=V_i^+(t)\cup(\phi\otimes((Pb_i(t)\cup Lb_i(t))-X_i(t))) \tag{5}$$

$$V_i^-(t+1)=\eta\otimes(X_i(t)-(Pb_i(t)\cap Lb_i(t))) \tag{6}$$

$$X_i(t+1)=(X_i(t)\cup V_i^+(t+1))-V_i^-(t+1) \tag{7}$$

$$V_i^+(t+1)=\xi\otimes(U-(X_i(t+1)\cup Pb_i(t)\cup Lb_i(t))) \tag{8}$$

式中，“$Pb_i(t)$”表示粒子 i 在 t 时刻所得到的最佳解集合；“$Lb_i(t)$”表示 t 时刻粒子 i 的邻居粒子中最佳解集合；“$X_i(t)$”表示粒子 i 在 t 时刻得到的解集合；“$\cup$”表示集合运算中的并操作；“$-$”表示集合运算中的差操作；“$\cap$”表示集合运算中的交操作；“$\otimes$”(新增的运算)表示如下：$n\otimes A$ 表示从集合 A 中任选一个包含 n 个元素构成的子集合，不妨记为 E。这里 $n\leqslant A$，E 满足：

$$E\subseteq\{A_j\mid A_j\subseteq P(A),\mid A_j\mid=n\} \tag{9}$$

式中，A 是 Ω 空间中的集合，n 是一非负整数，$P(A)$为集合 A 的幂集。

参数定义如下：

$$\phi=\text{rand int}(0,\lfloor\alpha\times|(Pb_i(t)\cup Lb_i(t))-X_i(t)|\rfloor) \tag{10}$$

$$\eta=\text{rand int}(0,\lfloor\beta\times X_i(t)-(Pb_i(t)\cap Lb_i(t))|\rfloor) \tag{11}$$

$$\xi=\text{rand int}(0,\lfloor\gamma\times|U-(X_i(t+1)\cup Pb_i(t)\cup Lb_i(t))|\rfloor) \tag{12}$$

式中，函数 rand int(a,b)用于在整数 a、b 之间取一个随机整数；运算$\lfloor c\rfloor$表示对实数 c 向下取整。ϕ、η、ξ 均为整数，表示从集合中选择的元素数目。α、β、γ 均为(0,1)之间的任意数，用来调节 ϕ、γ、ξ 的取值范围。为了增加群的多样性，引入式(7)来确保加入一些没有包含在粒子群中的全集元素。

4 求解动态背包问题的离散粒子群优化算法

本文在上述离散粒子群优化模型的基础上，引入对环境变化的探测以及环境变化后的响应机制，提出一种求解动态背包问题的离散粒子群优化算法。环境变化的探测和环境变化后的响应机制使算法能够适应环境的动态变化。

4.1 环境变化探测机制

要追踪环境动态变化下的最优解，首先必须能对环境变化做出准确的探测。针对动态背包问题，采用两个重量约束值，一个是当前重量约束值 MAX_Mc，另一个是上一次的重量约束值 MAX_Ml。每次迭代都需要检查 ΔW 是否等于零，如果不为零，则说明环境已经改变。其中，$\Delta W=$MAX_$Mc-$MAX_Ml。

4.2 环境变化响应机制

求解动态优化问题最简单最直接的方法就是将每次环境的变化都看作是一个新问题重新求解，但对于实际问题而言，环境一般只会部分改变，很少会发生完全改变，而且即使环境发生变化，原有的信息对于新环境仍有指导作用，若对每一次变化都重新求解，这样既浪费了之前的信息和计算结果，又增加了计算时间。本文采用两方面的响应策略：其一，加入和移除速度方面的响应；其二，更新部分粒子的历史最优解。

4.2.1 加入和移除速度方面的响应

对于动态背包问题，当 $\Delta W\neq0$ 时，分两种情况：若 $\Delta W>0$，说明背包的承重量增加，此时，向背包里加入物品的个数多，

而移除物品的个数少；若 $\Delta W<0$，说明背包的承重量减少，此时，向背包里加入物品的个数少，而移除物品的个数多。加入和移除物品的个数由其生成区间决定，具体如下：在离散粒子群优化模型中，ϕ,η 分别在$[1,A]$和$[1,B]$上随机生成，其中 A,B 分别由式(13)和式(14)计算而得；在 DSDPSO 算法中，当 $\Delta W>0$ 时，ϕ 改为在$[1/2A,A]$上随机生成；而 η 改为在$[1,1/2B]$上随机生成；当 $\Delta W<0$ 时，则 ϕ 变为在区间$[1,1/2A]$上随机生成，η 变为在区间$[1/2B,B]$上随机生成；当 $\Delta W=0$，说明环境没发生变化，ϕ,η 按式(10)、式(11)计算。

A,B 的计算公式为：

$$A=\lfloor \alpha\times|(Pb_i(t)\cup Lb_i(t))-X_i(t)|\rfloor \tag{13}$$

$$B=\lfloor \beta\times|X_i(t)-(Pb_i(t)\cap Lb_i(t))|\rfloor \tag{14}$$

式中，α、β 同式(10)、式(11)。

4.2.2 更新部分粒子的历史最优解

粒子群优化算法中，各粒子记录自身历史最优解和邻居的最优解，当重量约束由大变小，即 $\Delta W<0$ 时，由于在新的重量约束下，求得的总价值永远不会大于粒子在上一个环境(即重量约束值大的环境)中记录的历史最优值，这时就需要一种策略来处理这种情况。对总重量超过当前重量约束的粒子再评价一次，在实验中评价函数采用的是贪心思想，返回的结果是不超过当前重量约束的最优解，用此解更新历史最优解。

4.3 算法描述

输入：种群大小，迭代次数，动态背包参数；

输出：选入背包的物品总价值和总重量以及物品编号。

Step1 初始化：在 Ω 上随机选择子集初始化各粒子的 X_i 和 V_i^+，V_i^-，以及初始化各粒子的邻居拓扑结构。按照优化问题来评价群中所有粒子，将当前各粒子的位置记为 Pb_i，将粒子邻居中目标值最优的个体位置记为 Lb_i，初始化 MAX_Mc 和 MAX_Ml 的值为评价函数中的重量约束值。

Step2 更新：当 $\Delta W\neq 0$ 时，则按上述讨论策略更新 ϕ,η；而当 $\Delta W=0$ 时，则按式(10)、式(11)更新。按照式(5)－式(7)分别更新各粒子的 V_i^+，V_i^- 和 X_i，再按照式(8)更新 V_i^+。

Step3 评价：按照优化问题评价群中所有粒子，比较群中每个粒子当前 X_i 的目标值与其 Pb_i 的目标值。若当前 X_i 的目标值更优，则 $Pb_i=X_i$。根据邻居结构，比较所有邻居的 Pb_i，选择最优的位置更新 Lb_i。

Step4 令 MAX_Ml=MAX_Mc，更新 MAX_Mc 为评价函数的重量约束值。

Step5 当 $\Delta W<0$ 时，对超出重量约束的粒子重新评估并以得到的解更新历史最优解。

Step6 若满足终止条件(即达到最大迭代次数)，则输出每一代的最优目标值并停止算法；否则，转向 Step2。

5 实验仿真及其结果分析

5.1 实验配置

本文选用包含 17 个物品[6]的小规模背包和包含 50 个物品[5]的大规模背包作为测试背包。小规模背包重量约束分别在{60,104}两个值和{60,80,104}三个值之间周期性震荡[7]，大规模背包重量约束在{688,986}两个值之间周期性震荡，变化周期为 T，T 分别取 50 代和 100 代，在一个运行实例中，环境改变 10 次也即重量约束值改变 10 次。背包具体参数见表 1。

表 1 背包问题参数表

背包	背包问题参数	背包属性
背包 1	背包规模	17
	物品的价值集	2 3 9 2 4 4 2 7 8 10 3 6 5 5 7 8 6
	物品的重量集	12 5 20 1 5 3 10 6 8 7 4 12 3 3 20 1 2
背包 2	背包规模	50
	物品的价值集	293 291 290 280 278 274 269 265 248 247 245 245 241 234 229 228 222 216 214 191 191 187 171 170 164 152 142 132 131 126 122 116 112 111 110 106 77 76 74 73 69 67 42 41 35 33 30 29 28 26
	物品的重量集	95 39 69 63 49 104 56 58 47 23 17 129 91 28 77 125 73 5 103 63 76 23 47 79 119 125 26 119 79 56 50 75 12 26 31 43 41 38 29 21 14 9 3 17 8 8 9 7 4 5

实验中将提出的 DSDPSO 算法同自适应原对偶遗传算法(APDGA)进行了比较，同 BPSO 的比较见文献[5]。DSDPSO 在评价函数中使用贪心思想将不可行解转化为可行解，APDGA 在评价函数中采用两种思想，一种同上述使用的贪心思想，另一种使用惩罚函数[7,9]，这样做的目的是为了对比贪心算法和惩罚函数法的优劣。对于大小背包，惩罚因子有所不同，小背包惩罚函数为：$P=2\times\Delta W$，大背包惩罚函数为：$P=4\times\Delta W$，其中 ΔW 为超出重量限制的部分。

DSDPSO 算法中粒子数 $M=20$，邻居结构采用冯诺依曼结构。APDGA 算法中种群数 $NP=128$，交叉概率 $Pc=0.6$，变异概率 $Pm=0.001$，采取的是正比选择和基因保留策略。每个实例运行 50 次，计算平均值。

5.2 性能评价指标

在这里用动态优化问题中常用的评价指标，即累积平均值(average-of-generation fitness)[8]作为算法的评估指标：

$$\overline{F}_{AOG_t}=\frac{1}{t-kT}\sum_{i=kT+1}^{t}\left[\frac{1}{50}\sum_{j=1}^{50}F_{BOG_{ij}}\right] \tag{15}$$

式中，$t=\lceil t/T\rceil-1$；$F_{BOG_{ij}}$ 是在第 j 次运行第 i 代获得的最优值；离线性能 $\overline{F}_{AOG_t}$ 是算法每一代运行 50 次平均值在每个周期内的累积平均值。在动态环境中，不仅要比较算法最后获得的最优解，还要比较在整个迭代过程中算法对最优解的跟踪能力。此外，随着迭代次数的增加，采用累积值的方法会使整个性能曲线越来越光滑。

5.3 实验结果与分析

在变化周期 $T=50$ 代和 $T=100$ 代的情况下，以累积平均值作为评价指标，比较 DSDPSO 和 APDGA 对于 2 个动态背包在 3 种情况(即小规模背包的重量约束值分别在两值和三值之间震荡以及大规模背包的重量约束值在两值间震荡)下的优化性能。图 1 和图 2 中，DSDPSO 的评价函数使用贪心思想，而 APDGA 的评价函数使用惩罚函数；图 3 和图 4 中，DSDPSO 和 APDGA 的评价函数均使用贪心思想。

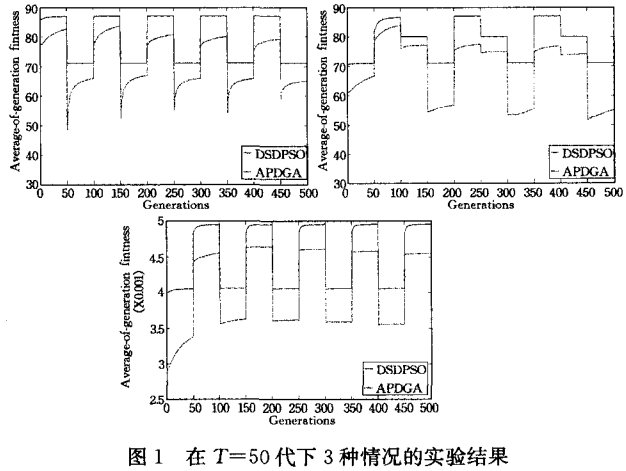

图 1 在 $T=50$ 代下 3 种情况的实验结果

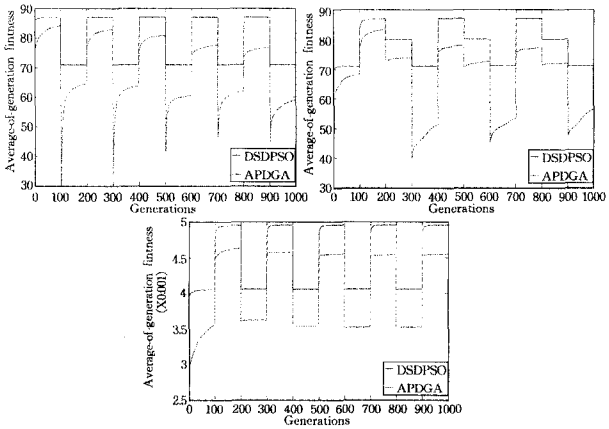

图 2　在 $T=100$ 代下 3 种情况的实验结果

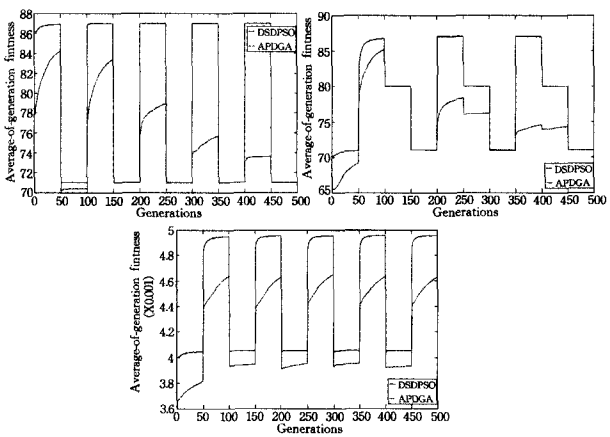

图 3　在 $T=50$ 代下 3 种情况的实验结果

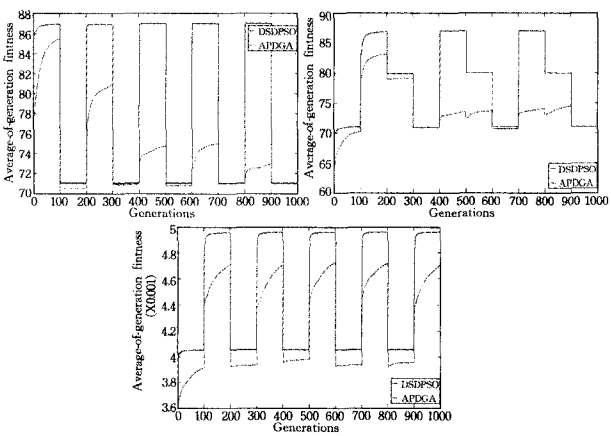

图 4　在 $T=100$ 代下 3 种情况的实验结果

从以上各图可以看出：DSDPSO 算法在新环境（即新的重量约束值）下迅速找到了此环境下的最优解并稳定在此解上；而 APDGA 算法在新环境下，除少数几个特例（图 3、图 4 中小规模背包重量约束值为 60）外，其他均没有找到当前环境下的最优解。从图 3 和图 4 的小规模背包在两值之间震荡能明显地看出，随着迭代次数的增加，APDGA 算法所找到的解逐渐变小，说明 PD 映射在算法迭代后期表现不佳。对比图 1—图 4 中 APDGA 算法所找到的解，可以看出，评价函数使用贪心思想所找到的解明显优于评价函数使用惩罚函数所找到的解。由此可得如下结论，① DSDPSO 算法在两个动态背包问题的不同变化周期均能很快地适应环境的变化，并且找到的最优解正是该重量约束值下的最优解，这符合动态环境对算法在时效性和准确性上的要求，APDGA 算法在上述方面表现不佳，主要原因是 PD 映射在算法迭代后期或环境变化强度较小时往往会失效；② 贪心思想和惩罚函数是求解背包问题的两个重要手段，相比之下，贪心思想更适合求解动态背包问题。

再从时间性能上对比这两个算法，在每个重量约束值下设定一个最优值。规定：寻优时间指算法在一个周期内达到设定的最优值所耗的时间，如果算法在整个周期内都没达到最优值，那么就把在整个周期上所耗的时间作为寻优时间。这里给出了 $T=50$ 代且评价函数均使用贪心思想的寻优时间，结果如表 2 所列。

表 2　寻优时间（单位：s）

算法 / 动态背包	DSDPSO	APDGA
小规模背包两值之间震荡	0.063457	0.133487
小规模背包三值之间震荡	0.081676	0.15517
大规模背包两值之间震荡	0.57301	0.8271

由表 2 可知，对于小规模背包，DSDPSO 算法的寻优时间比 APDGA 算法的短，约为 APDGA 算法所用时间的一半；对于大规模背包，DSDPSO 算法的寻优时间比 APDGA 算法的也短，约为 APDGA 算法所用时间的 0.7 倍。由此可见，无论是小规模背包还是大规模背包，DSDPSO 算法的寻优时间均比 APDGA 算法的优。

综上所述，DSDPSO 算法能很好地适应环境的变化，迅速找到最优解，并且正是在此重量约束下的最优解。虽然 DSDPSO 算法在每代运行时间消耗要大一点，但它能在环境变化后只经过几次迭代就找到最优解，并稳定下来，相比 APDGA 算法的数次迭代也不一定能找到最优解，其在最优解和消耗的时间总量上较优。

结束语　动态环境中的优化问题已成为进化计算的一个研究热点，而动态背包问题是一类经典的离散动态优化问题。本文提出的求解动态背包问题的离散粒子群优化算法（DSDPSO），具有很强的适应动态环境的能力，能迅速找到变化后的最优解并稳定下来。但 DSDPSO 也有不足，如集合运算耗时较长，因此，接下来的工作拟对以下两方面进行研究：其一，改进算法中集合运算方式，降低运算时间；其二，改进离散粒子群优化模型用于求解动态环境下的旅行商问题（TSP）等其他离散动态优化问题。

参 考 文 献

[1] Yang S. Non-stationary problem optimization using the primal-dual genetic algorithm[C]// Proceedings of the IEEE Congress on Evolutionary Computation. Chicago, IEEE Service Center, 2003:2246-2253

[2] Collard P, Aurand J P. DGA: An Efficient Genetic Algorithm[C]// Cohn A, ed. Proceedings of the 11th European Conference on Artificial Intelligence. 1994:487-491

[3] Wang H, Wang D. An improved primal-dual genetic algorithm for optimization in dynamic environments[C]// 13th International Conference on Neural Information Processing. Part III, 2006:836-844

[4] Kennedy J, Eberhart R. A discrete binary version of the particle

swarm algorithm [C]// Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. Piscataway: IEEE Press, 1997: 4104-4108

[5] 陈自郁,何中市,何静媛. 一种求解集合组合问题的离散粒子群优化模型[J]. 华南理工大学学报:自然科学版,2010,38(4):141-146

[6] Goldberg D E, Smith R E. Non-stationary function optimization using genetic algorithms with dominance and diploidy[C]// International Conference on Genetic Algorithms. Hillsdale: L Erlbaum Associates Inc, 1987: 59-68

[7] Simões A, Costa E. A comparative study using genetic algorithms to deal with dynamic environments [C]// Proceedings of the Sixth International Conference on Neural Networks and Genetic Algorithms. Roanne: Springer-Verlag, 2003: 203-209

[8] 闫杨,汪定伟,王大志,等. 求解动态背包问题的多智能体进化算法[J]. 东北大学学报:自然科学版,2009,30(7):948-951

[9] 王洪峰,汪定伟,刘黎黎. 求解动态优化问题的改进原对偶遗传算法[J]. 东北大学学报:自然科学版,2007,28(5):639-642

(上接第 201 页)

$\bar{v}_A(y)=1-v_A(y)\geqslant 1-\max\{v_A(x\to y),v_A(x)\}$

$=\min\{1-v_A(x\to y),1-v_A(x)\}$

$=\min\{\bar{v}_A(x\to y),\bar{v}_A(x)\}$

即 $\bar{v}_A(y)\geqslant\min\{\bar{v}_A(x\to y),\bar{v}_A(x)\}$。

故模糊集 $\bar{v}_A(x)$ 是 L 上的模糊滤子。

(2) 若模糊集 $\mu_A(x)$ 和 $\bar{v}_A(x)$ 是 L 上的模糊滤子,则 $A=\{(x,\mu_A(x),v_A(x))\mid x\in L\}$ 是 L 上的直觉模糊集,且由

① $\forall x\in L,\mu_A(x)\leqslant\mu_A(1)$,由 $\bar{v}_A(x)\leqslant\bar{v}_A(1)$ 可得,$1-v_A(x)\leqslant 1-v_A(1)$,即 $v_A(x)\geqslant v_A(1)$;

② $\forall x,y\in L,\mu_A(y)\geqslant\min\{\mu_A(x\to y),\mu_A(x)\}$;

③ $\forall x,y\in L,\bar{v}_A(y)\geqslant\min\{\bar{v}_A(x\to y),\bar{v}_A(x)\}$。

得,$1-v_A(y)\geqslant\min\{1-v_A(x\to y),1-v_A(x)\}=1-\max\{v_A(x\to y),v_A(x)\}$。

即可得,$v_A(y)\leqslant\max\{v_A(x\to y),v_A(x)\}$。

由①,②和③得,A 是 L 上的直觉模糊滤子。

推论 3 直觉模糊集 $A=\{(x,\mu_A(x),v_A(x))\mid x\in L\}$ 是 L 上的直觉模糊滤子,当且仅当 $\forall t\in[0,1]$,$(\mu_A)_t=\{x\mid x\in L,\mu_A(x)\geqslant t\}$ 和 $(\bar{v}_A)_t=\{x\mid x\in L,\bar{v}_A(x)\geqslant t\}$ 是 L 的滤子。

证明:由 μ_A 是 L 的模糊滤子,当且仅当 $\forall t\in[0,1]$,μ_A 的 t 水平截集 $(\mu_A)_t=\{x\mid x\in L,\mu_A(x)\geqslant t\}$ 是 L 的滤子和定理 9 易证。

定理 10 直觉模糊集 $A=\{(x,\mu_A(x),v_A(x))\mid x\in L\}$ 是 L 上的直觉模糊滤子,当且仅当 $A_1=\{(x,\mu_A(x),\bar{\mu}_A(x))\mid x\in L\}$ 和 $A_2=\{(x,\bar{v}_A(x),v_A(x))\mid x\in L\}$ 是 L 上的直觉模糊滤子。其中,$\bar{\mu}_A(x)=1-\mu_A(x)$,$\bar{v}_A(x)=1-v_A(x)$。

证明:(1) 若 A 是 L 上的直觉模糊滤子。$\forall x\in L,0\leqslant\mu_A(x)+\bar{\mu}_A(x)=\mu_A(x)+1-\mu_A(x)=1,0\leqslant v_A(x)+\bar{v}_A(x)=v_A(x)+1-v_A(x)=1$。则,$A_1=\{(x,\mu_A(x),\bar{\mu}_A(x))\mid x\in L\}$ 和 $A_2=\{(x,\bar{v}_A(x),v_A(x))\mid x\in L\}$ 是 L 上的直觉模糊集。$\mu_A(1)\geqslant\mu_A(x)$,$\bar{\mu}_A(1)=1-\mu_A(1)\leqslant 1-\mu_A(x)=\bar{\mu}_A(x)$,即 $\bar{\mu}_A(1)\leqslant\bar{\mu}_A(x)$。

$\forall x,y\in L,\mu_A(y)\geqslant\min\{\mu_A(x\to y),\mu_A(x)\}$

$\bar{\mu}_A(y)=1-\mu_A(y)\geqslant 1-\min\{\mu_A(x\to y),\mu_A(x)\}$

$=\max\{1-\mu_A(x\to y),1-\mu_A(x)\}$

$\geqslant 1-\min\{\mu_A(x\to y),\mu_A(x)\}$

$=\max\{1-\mu_A(x\to y),1-\mu_A(x)\}$

$=\max\{\bar{\mu}_A(x\to y),\bar{\mu}_A(x)\}$

即 $A_1=\{(x,\mu_A(x),\bar{\mu}_A(x))\mid x\in L\}$ 是 L 上的直觉模糊滤子。

同理可证,$A_2=\{(x,\bar{v}_A(x),v_A(x))\mid x\in L\}$ 也是 L 上的直觉模糊滤子。

(2) 若 A_1 和 A_2 是 L 上的直觉模糊滤子,则由 A_1 可得,$\forall x,y\in L,\mu_A(1)\geqslant\mu_A(x),\mu_A(y)\geqslant\min\{\mu_A(x\to y),\mu_A(x)\}$,由 A_2 可得,$\forall x,y\in L,v_A(1)\leqslant v_A(x),v_A(y)\leqslant\max\{v_A(x\to y),v_A(x)\}$,即直觉模糊集 $A=\{(x,\mu_A(x),v_A(x))\mid x\in L\}$ 是 L 上的直觉模糊滤子。

故,定理 10 得证。

结束语 本文将直觉模糊集和 *BL*-代数上的滤子结合起来,给出了 *BL*-代数上的直觉模糊滤子、直觉模糊格滤子、直觉模糊布尔滤子和直觉模糊蕴涵滤子的概念,讨论了 *BL*-代数上的直觉模糊滤子的若干性质,研究了 *BL*-代数中的直觉模糊滤子与直觉模糊格滤子、直觉模糊布尔滤子和直觉模糊蕴涵滤子的等价性。最后探讨了直觉模糊滤子、模糊滤子及 t 水平截集的关系,为直觉模糊集在 *BL*-代数中的进一步应用提供了丰富的理论基础。

参 考 文 献

[1] Zhu Yi-quan, Xu Yang. On filter theory of residuated lattices [J]. Information Sciences, 2010, 180: 3614-3632

[2] Zhang Xiao-hong, Bae J Y, Im D M. On fuzzy Filters and Fuzzy Ideals of BL-algebras [J]. Fuzzy Systems and Mathematics, 2006, 20(3): 8-20

[3] Yin Yun-qiang, Zhan Jian-ming. New types of fuzzy filters of BL-algebras [J]. Computers and Mathematics with Applications, 2010, 60: 2115-2125

[4] Wang Wei, Xin Xiao-long. On fuzzy filter of pseudo *BL*-algebras [J]. Fuzzy Sets and Systems, 2011, 162: 27-38

[5] Liu Lian-zhen, Li Kai-tai. Fuzzy Boolean and positive implicative filters of BL-algebras [J]. Fuzzy Sets and Systems, 2005, 152: 333-348

[6] Atanassov K. Intuitionistic fuzzy sets [J]. Fuzzy Sets and Systems, 1986, 20: 87-96

[7] Zadeh L A. Fuzzy sets [J]. Information and Control, 1965(8): 338-385

[8] 申晓勇,雷英杰,周创明,等. 基于直觉模糊集的不确定时序逻辑模型[J]. 计算机科学,2010,37(5):187-189,273

[9] 裴峥. 格蕴涵代数中的直觉模糊滤子[J]. 西华大学学报:自然科学版,2007,26(2):17-20

[10] 王伟,郭颖敏. Heyting 代数中的直觉模糊滤子[J]. 西安石油大学学报,2008,23(5):106-108

[11] Hajek P. Metamathematics of fuzzy logic [M]. Dordrecht: Kluwer Academic Publishers, 1998

[12] Zhan Jian-ming, Yu Yang. Some types of generalized fuzzy filters of BL-algebras [J]. Computers and Mathematics with Applications, 2008, 56: 1604-1616