软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.005400]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于编码转换的离散演化算法设计与应用*

贺毅朝[1], 王熙照[2], 赵书良[3], 张新禄[3]

[1](河北地质大学 信息工程学院, 河北 石家庄 050031)

[2](深圳大学 计算机与软件学院, 广东 深圳 518060)

[3](河北师范大学 数学与信息科学学院, 河北 石家庄 050024)

通讯作者: 贺毅朝, E-mail: heyichao119@163.com

摘 要: 为了利用演化算法求解离散域上的组合优化问题, 借鉴遗传算法(GA)、二进制粒子群优化(BPSO)和二进制差分演化(HBDE)中的映射方法,提出了一种基于映射变换思想设计离散演化算法的实用方法---编码转换法(ETM), 并利用一个简单有效的编码转化函数给出了求解组合优化问题的离散演化算法一般算法框架 **A-DisEA**. 为了说明 ETM 的实用性与有效性,首先基于 **A-DisEA** 给出了一个离散粒子群优化算法(DisPSO),然后分别利用 BPSO、HBDE 和 DisPSO 等求解集合联盟背包问题和折扣{0-1}背包问题,通过对计算结果的比较表明: BPSO、HBDE 和 DisPSO 的求解性能均优于 GA,这不仅说明基于 ETM 的离散演化算法在求解 KP 问题方面具有良好的性能,同时也说明利用 ETM 方法设计离散演化算法是一种简单且有效的实用方法.

关键词: 离散演化算法; 编码转换; SUKP 问题; D{0-1}KP 问题

中图法分类号: TP311

中文引用格式: 贺毅朝,王熙照,赵书良,张新禄.基于编码转换的离散演化算法设计与应用.软件学报,2018,29(9). http://www.jos.org.cn/1000-9825/5400.htm

英文引用格式: He YC, Wang XZ, Zhao SL, Zhang XL. Design and applications of discrete evolutionary algorithms based on encoding transformation. Ruan Jian Xue Bao/Journal of Software, 2018,29(9) (in Chinese). http://www.jos.org.cn/1000-9825/5400.htm

Design and Applications of Discrete Evolutionary Algorithm Based on Encoding Transformation

HE Yi-Chao[1], WANG Xi-Zhao[2], ZHAO Shu-Liang[3], ZHANG Xin-Lu[3]

[1](College of Information and Engineering, Hebei GEO University, Shijiazhuang 050031, China)

[2](College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China)

[3](College of Mathematics and Information Science, Hebei Normal University, Shijiazhuang 050024, China)

Abstract: In order to solve a combinatorial optimization problem in discrete domains by using evolutionary algorithms, draw lessons from the design concept of genetic algorithm (GA), binary particle swarm optimization (BPSO) and binary differential evolution with hybrid encoding (HBDE), we propose a simple and practical method for designing discrete evolution algorithm based on the idea of mapping transformation. This method is named Encoding Transformation Method (ETM). A framework of discrete evolution algorithm named A-DisEA is advanced by using a simple and effective encoding conversion function. For illustrating the practicability and effectiveness of ETM, a discrete particle swarm optimization (DisPSO) algorithm based on ETM is proposed. Then, we use GA, BPSO,

* 基金项目:国家自然科学基金(61503252,71371063,11471097);深圳知识创新项目基础研究项目(JCYJ20150324140036825);河北省高等学校科学研究计划项目(ZD2016005);河北省自然科学基金项目(F2016403055)

本文由演化学习专刊特约编辑俞扬教授和钱超教授推荐.

收稿时间: 2017-07-09; 修改时间: 2017-07-13; 采用时间: 2017-09-26; jos 在线出版时间: 2017-11-13

CNKI 网络优先出版: 2017-11-13 14:13:24, http://kns.cnki.net/kcms/detail/11.2560.TP.20171113.1413.007.html

HBDE and DisPSO to solve the set union knapsack problem (SUKP) and the discounted {0-1} knapsack problem (D{0-1}KP), respectively. The results show that for SUKP and D{0-1}KP, the discrete evolutionary algorithms based on ETM, i.e. BPSO, HBDE and DisPSO have better performance than GA. this indicates that the ETM is not only a feasible method, but also a very practical and efficient method.

Key words: discrete evolutionary algorithm; encoding transformation; set union knapsack problem; discounted {0-1} knapsack problem

演化算法(Evolutionary algorithms,EAs)[1]是一类群体智能算法,其主要优点是不需要计算导数和梯度,不要求目标函数具有连续性,并且算法具有内在的隐含并行性和极强的全局寻优能力.经典 EAs 有遗传算法(Genetic algorithm,GA)[2], 粒子群优化(Particle swarm optimization,PSO)[3], 差分演化(Differential evolution,DE)[4],蚁群优化(Ant colony optimization, ACO)[5]和人工鱼群(Artificial fish swarm, AFS)[6]等,已被广泛应用于数值优化、自动控制、图像处理、决策与规划、组合优化、信号处理、模式识别、人工生命、机器人学、机器学习和图像处理等领域[2-10].

近年来,由于 EAs 在求解优化问题中的成功应用,引起了人们的极大关注,纷纷通过模仿自然界中生物群体的行为或者借鉴社会活动中的某些现象,相继提出了诸如人工蜂群(Artificial bee colony, ABC)[11],萤火虫算法(Firefly algorithm,FFA)[12],烟花算法(Fireworks algorithm,FWA)[13],果蝇优化(Fruit fly optimization, FFO)[14],灰狼优化(Grey wolf optimizer, GWO)[15]、人工藻算法(Artificial algae algorithm, AAA)[16]和鸽群优化(Pigeon-inspired optimization,PIO)[17]等许多新颖演化算法. 虽然大量新算法的提出壮大了 EAs 家族, 为应用 EAs 求解不同领域中的优化问题提供了更多新的可行方法, 但是并未改变经典 EAs 所存在的缺点,即除了 GA 和 ACO 以外,大部分算法只适于求解连续域上的数值优化问题,不能被直接用于求解离散域上的组合优化问题.为了改变这样的不利现状,人们从对已有算法的改进进行尝试,提出了两种可行方法:一种方法[18,19]是改变原有算法的进化算子,重新定义使之满足离散域上的计算要求,以实现对组合优化问题的求解.如文献[18]利用置换的插入操作与互换操作重新定义 ABC 的进化算子，给出了一个求解流水调度问题的离散 ABC;文献[19]利用逻辑与、逻辑或运算取代 ABC 中的加法与减法运算，提出了一个可求解二元优化问题的离散 ABC.此方法的缺点是不能保证新算法继承原始算法的所有特性,而且不具有通用性.另一种方法[20,21]是保持原有算法的进化算子不变,利用映射将个体对应的实向量映射为一个 0-1 向量,由此实现对某些组合优化问题(例如背包问题[21]和集合覆盖问题[22])的求解.如文献[20]利用取整与模 2 运算构成的函数将一个实向量映射为 0-1 向量,提出了一个求解二元优化问题的离散 ABC;文献[21]利用分段函数将一个实向量映射为 0-1 向量,提出了一个二进制差分演化算法 HBDE.这一方法的优点是不仅能够保持原始算法的所有特性,而且具有很好的通用性.本文是对后一种方法的总结与提高,首先指出 GA、二进制粒子群优化(BPSO)[23]和具有混合编码的二进制差分演化(HBDE)[21]中编码转换思想的一致性,给出一种将实向量映射为整型向量(包括 0-1 向量)的简单方法,提出基于编码转换(ETM)设计离散演化算法(Discrete evolutionary algorithm,DisEA)的一般框架;然后基于 ETM 给出了一个离散粒子群优化算法(DisPSO),通过对集合联盟背包问题(Set union knapsack problem, SUKP)[24,25]和折扣{0-1}背包问题(Discounted {0-1} knapsack problem, D{0-1}KP)[26,27]等的求解验证利用 ETM 设计 DisEA 的实用性与有效性.

本文在第 1 节分别简述了 GA、BPSO 和 HBDE 中实现编码转换所使用的映射及其简单性质,在第 2 节中,分析了 GA、BPSO 和 HBDE 中编码转换函数的一致性, 给出了一种将实向量映射为整型向量的简单方法, 提出了基于编码转换方法 ETM 设计 DisEA 的一般思路, 并基于 ETM 给出了一个离散粒子群优化算法 DisPSO. 在第 3 节分别利用 BPSO、HBDE 和 DisPSO 求解 SUKP 和 D{0-1}KP, 通过与 GA 及其改进算法的计算结果比较验证了基于 ETM 设计的离散演化算法在求解 KP 问题方面的优越性能, 以此说明利用 ETM 设计 DisEA 的实用性与有效性. 最后, 总结全文并展望今后进一步的研究方向.

1 编码转换思想

1.1 GA中的编码转换思想

遗传算法(Genetic algorithm, GA)[2,28]是 Holland 教授于 1975 年借鉴生物进化规律提出的一种演化算法,是最早被提出的演化算法之一,已被广泛应用于数值优化、组合优化、机器学习、图像识别、神经网络和模糊控制等领域[28-30].GA 的进化过程主要由三种遗传算子:交叉算子、变异算子和选择算子实现,而且每一种遗传算子均存存多种不同的实现方式.限于篇幅有关它们的详细介绍不再赘述,请参考文献[28]中所述.

在利用 GA 求解数值优化问题时,一般个体编码采用 0-1 向量形式,这就必须将其转换为一个表示问题解向量的实数向量,由此对个体进行性能的评价和实现对问题的求解.设个体 X 的编码为 $X=[x_{11},x_{12},\ldots,x_{1k},x_{21},\ldots,x_{2k},x_{31},\ldots,x_{dk}]\in\{0,1\}^{dk}$，它所对应的解向量为 $Y=[y_1,y_2,\ldots,y_d]$，其中 $y_i\in[L_i,U_i]$，$L_i<U_i$ 且为实数,$i=1,2,\ldots,d$, d 为问题的维数.记 GA 将 dk 维 0-1 向量 X 转换为 d 维实数向量 Y 的编码转化函数为映射为 Ψ_{GA}, 则 $Y=\Psi_{GA}(X)$ 且对于 $i=1,2,\ldots,d$ 有:

$$y_i = L_i + \left(\sum_{j=1}^{k} x_{ij}\,2^{j-1}\right)\frac{U_i - L_i}{2^k - 1} \tag{1}$$

其中,$\delta_i=(U_i-L_i)/(2^k-1)$表示第 i 维分量 y_i 的编码精度.显然,Ψ_{GA} 是从离散空间 $\{0,1\}^{dk}$ 到连续空间 $\prod_{i=1}^{d}[L_i,U_i]$ 上的一个单射,当限定解向量 Y 中分量 y_i 的编码精度为 δ_i 之后,Ψ_{GA} 也可以看作是一个双射.

GA 利用遗传算子产生新个体 $X\in\{0,1\}^{dk}$,利用映射 Ψ_{GA} 求得它对应的解向量 Y,利用 Y 的目标函数值对个体 X 进行评价,并根据评价结果确定当前全局最优个体.由此可以看出:编码转换函数 Ψ_{GA} 在 GA 的个体与解向量之间起着非常重要的桥梁作用.

1.2 BPSO中的编码转换思想

微粒群优化(Particle swarm optimization,PSO)[3,31]是由 Kennedy 和 Eberhart 在 1995 提出的一种演化算法,其基本思想源于对自然界中鸟类等生物群体觅食过程的仿真研究.在 PSO 中,每个粒子(个体)具有一个飞行速度和一个位置,并用粒子位置表示待求解问题的一个潜在解.PSO 利用速度更新方程不断改变粒子的飞行速度,并根据其速度确定粒子的新位置,不断搜寻更好的结果,以实现算法的进化过程.

设粒子的速度为 $V=[v_1,v_2,\ldots,v_d]$, 位置为 $X=[x_1,x_2,\ldots,x_d]$, $P=[p_1,p_2,\ldots,p_d]$ 表示粒子的历史最好位置,$G=[g_1,g_2,\ldots,g_d]$表示 PSO 的全局最好位置,其中 $v_j\in[LV_j,UV_j]$,$LV_j<UV_j$ 且为实数, x_j, p_j, $g_j\in[LX_j,UX_j]$,$LX_j<UX_j$ 且为实数,$j=1,2,\ldots,d$, d 为问题的维数,则 PSO 依次利用(2)式与(3)式实现粒子速度与位置的更新.

$$v_j=v_j+c_1*r_1*(p_j-x_j)+c_2*r_2*(g_j-x_j) \tag{2}$$

$$x_j=v_j+x_j \tag{3}$$

其中 $j=1,2,\ldots,d$; $r_1\sim U(0,1)$与 $r_2\sim U(0,1)$为两个相互独立的随机数;c_1 和 c_2 为加速常数,在 0~2 之间取值.

为了能够利用 PSO 求解二元优化问题,Kennedy 和 Eberhart[23]于 1997 年提出了二进制微粒群算法(Binary particle swarm optimization,BPSO).在 BPSO 中,粒子的速度 $V=[v_1,v_2,\ldots,v_d]$中的分量均限定为 $v_j\in[-A,A]$,A 是一个正实数; 粒子的位置被一个 0-1 向量 $X=[x_1,x_2,\ldots,x_d]\in\{0,1\}^d$取代,粒子速度更新方程仍使用(2)式,粒子的位置更新方程改为由(4)式实现.

$$x_j=\begin{cases}0, & sig(v_j)\le r_3;\\ 1, & otherwise;\end{cases} \tag{4}$$

其中,$sig(x)=1/(1+e^{-x})$是模糊函数,$r_3\sim U(0,1)$是一个随机数.

在 BPSO 中,确定粒子的位置实质上是利用一个从连续空间$[-A,A]^d$ 上到离散空间$\{0,1\}^d$ 的映射 Ψ_{BPSO} 实现的,即 $X=\Psi_{BPSO}(V)$,其中 X 的每一维分量 x_j 由 V 的相应分量 v_j 利用(4)式来确定.显然,映射 Ψ_{BPSO} 是 BPSO 中的编

码转换函数,它是一个满射,并且在粒子的速度与位置(问题解向量)之间也起着重要的桥梁作用.

1.3 HBDE中的编码转换思想

差分演化(Differential evolution,DE)[4,32]是 Rainer Storn 和 Kenneth Price 于 1997 年为求解切比雪夫多项式而提出的一种演化算法,它也是一种基于实数编码的演化算法,非常适合于求解连续域上的最优化问题,在第一届 IEEE 演化大赛中表现超群,引起了国内外学者的普遍关注.DE 进化原理为:首先利用差异个体重组得到一个父子混合的中间个体,若其优于父代个体则替换之,否则保持父代个体不变.在 DE 的 10 种变异策略中“DE/r/1/bin”使用最多,为此下面基于此模式描述 DE 通过差异个体重组产生中间个体的方法.

设 $X_1=[x_{11},x_{12},\ldots,x_{1d}]$,$X_2=[x_{21},x_{22},\ldots,x_{2d}]$,$X_3=[x_{31},x_{32},\ldots,x_{3d}]$是 DE 种群中不同于 $X=[x_1,x_2,\ldots,x_d]$的三个互不相同的个体,$V=[v_1,v_2,\ldots,v_d]$是相应于个体 X 的中间个体,其中 $x_{1j}, x_{2j}, x_{3j}, x_j,v_j\in[L_j,U_j]$, $j=1,2,\ldots,d$,d 为问题维数,则中间个体 V 利用下面的(5)式产生.

$$v_j=\begin{cases}x_{1j}+F*(x_{2j}-x_{3j}), & if \quad r\le CR\vee j=R(i)\\ x_j, & otherwise\end{cases} \tag{5}$$

其中, $0<F<1$ 称为缩放因子; $r\sim U(0,1)$是一个随机数; CR 称为变异因子,且 $CR\in(0,1)$; $R(i)$是$[1,n]$上的一个随机正整数.

为了利用 DE 求解二元优化问题,贺毅朝等人[21,33]基于编码转换设计理念提出了具有混合编码的二进制差分演化算法 HBDE.在 HBDE 中,限定个体 X 与中间个体 V 在$[-A, A]^d$上取值, A 是一个正实数,并仍然使用(5)式产生中间个体 V.此外,引入一个 0-1 向量 $Y=[y_1,y_2,\ldots,y_d]\in\{0,1\}^d$作为个体 X 对应于求解问题的潜在解, Y 利用映射 $Y=\Psi_{HBDE}(X)$得到,其中分量 $y_j(j=1,2,\ldots,d)$与 x_j 之间的对应关系见下式.

$$y_j=\begin{cases}1, & if \quad x_j\ge 0\\ 0, & otherwise\end{cases} \tag{6}$$

易见,映射 $Y=\Psi_{HBDE}(X)$是从连续空间$[-A,A]^d$到离散空间$\{0,1\}^d$的一个满射,它将个体 X 的实数向量编码转换为 0-1 向量 Y 作为问题的潜在解,并由此利用 Y 的目标函数值来评价 X 的优劣,实现了 DE 在二元优化问题求解中的应用.

2 一种设计 DisEA 的简单方法及其应用

2.1 基于编码转换思想的DisEA设计方法

无论是 GA 中从离散空间映射到连续空间上的编码转换函数 Ψ_{GA},还是 BPSO 和 HBDE 中从连续空间映射到离散空间上的编码转换函数 Ψ_{BPSO} 和 Ψ_{HBDE},它们在算法中起到的作用是完全一致的,都是为了把个体编码转换为问题的可行解或潜在解.事实上,这种编码转换思想存在于每一个演化算法中,因为即使是仅适于求解连续域上优化问题的演化算法(如 PSO、DE、ABC 和 FFA 等),可视其编码转换函数为连续空间上的一个自同构恒等映射.

设 $\boldsymbol{E}_1$ 是演化算法 **A** 中个体编码构成的度量空间,$\boldsymbol{E}_2$ 是待求解问题的潜在解构成的度量空间,我们将利用从 $\boldsymbol{E}_1$ 到 $\boldsymbol{E}_2$ 上的一个编码转换函数 $\Psi_{\mathbf{A}}$ 对算法 **A** 离散化的方法称为编码转换法(Encoding transformation method, ETM). 一般情况下, 函数 $\Psi_{\mathbf{A}}$ 必须是一个满射.尽管理论上这样的满射有无数多个,但从实用的角度而言,简单高效且易于实现的 $\Psi_{\mathbf{A}}$ 是算法 **A** 离散化的关键所在.

为了能够利用 PSO、DE、AFS、ABC 和 FFA 等演化算法求解可行解为$\{0,1,\ldots,n-1\}^d$ (其中 $n\ge2$)中的 d 维整型向量的组合优化问题,例如可行解为$\{0,1\}^d$中的 d 维 0-1 向量的 SUKP 问题,可行解为$\{0,1,2,3\}^d$上 d 维整型向量的 D{0-1}KP 问题,下面根据已有的工作经验,借鉴 BPSO 和 HBDE 的设计思路给出一个简单通用且易于实现的编码转换函数 $\boldsymbol{\Psi}_{\mathbf{DisEA}}$.

设组合优化问题的可行解为 $Y=[y_1,y_2,\ldots,y_d]\in\{0,1,\ldots,n\text{-}1\}^d$,其中 $n\geq 2$,d 为问题的维数.设 EAs 中个体 X 的编码为 d 维实数向量 $X=[x_1,x_2,\ldots,x_d]$,$x_j\in[-A,A]$, $j=1,2,\ldots,d$,A 是一个正实数,于是定义映射 $\boldsymbol{\Psi}_{\mathbf{DisEA}}:[-A,A]^d\rightarrow\{0,1,\ldots,n\text{-}1\}^d$, $n\geq 2$, 对任意 $X\in[-A,A]^d$, $Y=\boldsymbol{\Psi}_{\mathbf{DisEA}}(X)$且满足:

$$y_j=\begin{cases}0, & if \quad x_j\in[-A,-A+2*A*\alpha_0);\\ k, & if \quad x_j\in[-A+2*A*\sum_{i=0}^{k-1}\alpha_i,-A+2*A*\sum_{i=0}^{k}\alpha_i),1\leq k\leq n-2;\\ n-1, & if \quad x_j\in[-A+2*A*\sum_{i=0}^{n-2}\alpha_i,A].\end{cases}\tag{7}$$

其中 $\alpha_j(j=0,1,\ldots,n\text{-}1)$是满足条件 $\alpha_0+\alpha_1+\ldots+\alpha_{n-1}=1$ 且 $0<\alpha_j<1$ 的实数.

容易看出,映射 $\boldsymbol{\Psi}_{\mathbf{DisEA}}$ 对每一维分量 y_j 是一个分段函数,它将区间$[-A,A]$分成 n 个小区间: $\boldsymbol{I}_0=[-A,-A+2A*\alpha_0)$, $\boldsymbol{I}_1=[-A+2A*\alpha_0,\ -A+2A*(\alpha_0+\alpha_1)),\ldots,\ \boldsymbol{I}_{n-2}=[-A+2A*(\alpha_0+\ldots+\alpha_{n-3}),\ -A+2A*(\alpha_0+\ldots+\alpha_{n-2}))$, $\boldsymbol{I}_{n-1}=[-A+2A*(\alpha_0+\ldots+\alpha_{n-2}),\ A]$, 当分量 $x_j\in\boldsymbol{I}_k$ 时 $y_j=k$.

在理论上(7)式中满足条件 $\alpha_0+\alpha_1+\ldots+\alpha_{n-1}=1$ 且 $0<\alpha_j<1$ 的实数 $\alpha_0,\alpha_1,\ldots,\alpha_{n-1}$ 有无数个, 如何选取它们的值是 $\boldsymbol{\Psi}_{\mathbf{DisEA}}$ 实现的关键.事实上,注意到分量 x_j 在$[-A,A]$上随机均匀变化,而且它对应的 y_j 的值也是一个不确定的量,因此我们可以取 $\alpha_j=1/n$ $(j=0,1,\ldots,n\text{-}1)$,于是(7)式简化为:

$$y_j=\begin{cases}0, & if \quad x_j\in[-A,-A+2*A/n);\\ k, & if \quad x_j\in[-A+2k*A/n,-A+2(k+1)*A/n),1\leq k\leq n-2;\\ n-1, & if \quad x_j\in[-A+2(n-1)*A/n,A].\end{cases}\tag{8}$$

映射 $\boldsymbol{\Psi}_{\mathbf{DisEA}}$: $[-A,A]^d\rightarrow\{0,1,\ldots,n\text{-}1\}^d$ 是一个满射,它实现了将 d 维实数向量转化为 d 维整型向量的功能,为设计 DisEA 提供了一个简单可行的编码转换方法.

显然,BPSO 中的映射 Ψ_{BPSO}、HBDE 中的映射 Ψ_{HBDE} 以及 BABC[25]中使用的编码转换函数均为 $\boldsymbol{\Psi}_{\mathbf{DisEA}}$ 的特例.为了说明如何利用 $\boldsymbol{\Psi}_{\mathbf{DisEA}}$ 基于 ETM 构造 DisEA,下面对已有演化算法 **A**,给出它的基于 ETM 构造的离散演化算法 **A-DisEA** 的一般算法框架.

算法 1. A-DisEA

Input: 组合优化问题实例 $f(Y)$,其中 $Y\in\{0,1,\ldots,n\text{-}1\}^d$; 算法 **A** 的参数.

Output: 最优解或近似最优解 $Y^*=[y_1^*,y_2^*,\ldots,y_d^*]$和 $f(Y^*)$.

1 随机产生初始种群 $\mathbf{P}(0)=\{X_i(0)=[x_{i1}(0),x_{i2}(0),\ldots,x_{id}(0)]\in[-A,A]^d|1\leq i\leq N\}$;

2 $Y_i(0)\leftarrow\boldsymbol{\Psi}_{\mathbf{DisEA}}(X_i(0))$, $1\leq i\leq N$; //计算个体 $X_i(0)$对应的解

3 根据 $f(Y_i(0))$确定 $\mathbf{P}(0)$中的最优个体 $B(0)=[b_1(0),b_2(0),\ldots,b_d(0)]$和当前最好解 $Y^*(0)$; $t\leftarrow 1$;

4 **while**($t\leq MaxIt$) **do**

5 利用算法 **A** 的进化算子产生种群 $\mathbf{P}(t)=\{X_i(t)=[x_{i1}(t),x_{i2}(t),\ldots,x_{id}(t)]\in[-A,A]^d|1\leq i\leq N\}$;

6 $Y_i(t)\leftarrow\boldsymbol{\Psi}_{\mathbf{DisEA}}(X_i(t))$, $1\leq i\leq N$; //计算个体 $X_i(t)$对应的潜在解

7 根据 $f(Y_i(t))$确定 $\mathbf{P}(0)\cup\mathbf{P}(1)\cup\ldots\cup\mathbf{P}(t)$中最优个体 $B(t)=[b_1(t),b_2(t),\ldots,b_d(t)]$和当前最好解 $Y^*(t)$;

8 $t\leftarrow t+1$;

9 **end while**

10 **return**($Y^*(MaxIt)$, $f(Y^*(MaxIt))$).

在算法 1 中,$MaxIt$ 是算法迭代次数,它一般是维数 d 的常数倍.令算法 **A** 产生新一代种群的时间复杂度为 T(**A**),计算函数值 $f(Y)$的时间复杂度为 T(f),则算法 **A-DisEA** 的时间复杂度为 $MaxIt$*(T(**A**)+O(d*N)+N*T(f)).

2.2 基于ETM的离散粒子群优化算法

BPSO、HBDE 和 BABC 的编码转换函数均是将一个实数向量转化为 0-1 向量,因此它们主要用于求解二元组合优化问题,例如 SUKP 问题、集合覆盖问题[22]和可满足性问题[34]等,并不完全适用于求解可行解为

$\{0,1,\ldots,n\text{-}1\}^d$ ($n\geq 3$)上的一个整数向量的组合优化问题(例如有界背包问题[35]).为此,下面基于 **A-DisEA** 框架,利用 PSO 给出一个基于 ETM 的离散粒子群优化算法 **DisPSO**,以阐明如何利用已有演化算法基于 ETM 构造求解此类组合优化问题的离散演化算法.

记 $V_i(t)=[v_{i1}(t),v_{i2}(t),\ldots,v_{id}(t)]\in[-A,A]^d$ 表示 DisPSO 的第 t 代种群 $\mathbf{P}(t)$中的第 i 个粒子的飞行速度,它的位置记为 $X_i(t)=[x_{i1}(t),x_{i2}(t),\ldots,x_{id}(t)]\in\{0,1,\ldots,n\text{-}1\}^d$ ($n\geq 3$),表示所求解问题的一个可行解(或潜在解).又记 $P_i(t)=[p_{i1}(t),p_{i2}(t),\ldots,p_{id}(t)]\in\{0,1,\ldots,n\text{-}1\}^d$ 为第 i 个粒子的历史最好位置,$G(t)=[g_1(t),g_2(t),\ldots,g_d(t)]\in\{0,1,\ldots,n\text{-}1\}^d$ 为 $\mathbf{P}(0)\cup\mathbf{P}(1)\cup\ldots\cup\mathbf{P}(t)$中的全局最好位置,$MaxIt$ 为算法的迭代次数,则对于最大组合优化问题 $Max\ f(X)$,$X\in\{0,1,\ldots,n\text{-}1\}^d$ ($n\geq 3$),DisPSO 的算法伪代码描述如下:

算法 2. DisPSO

Input: $Max\ f(X)$的一个实例,参数 A 的值,以及 PSO 的参数值.

Output: 最优解或近似最优解 $G(MaxIt)\in\{0,1,\ldots,n\text{-}1\}^d$ 和 $f(G(MaxIt))$.

```
1   随机产生初始种群 P(0)={V_i(0)=[v_i1(0),v_i2(0),…, v_id(0)]∈[-A,A]^d|1≤i≤N};
2   X_i(0)←Ψ_DisEA(V_i(0)), 并根据 f(X_i(0))确定 P_i(0), 1≤i≤N;
3   确定 P(0)中的全局最好位置 G(0); t←0;
4   while(t≤MaxIt) do
5       for i←1 to N do
6           for j←1 to d do     //生成第 t+1 代种群 P(t+1)
7               v_ij(t+1)←v_ij(t)+c_1*r_1*(p_ij(t)-x_ij(t))+c_2*r_2*(g_j(t)-x_ij(t));
8           end for
9           X_i(t+1)←Ψ_DisEA(V_i(t+1)), 并根据 f(X_i(t+1))确定 P_i(t+1);
10      end for
11      确定 P(0)∪P(1)∪…∪P(t+1)中的全局最好位置 G(t+1);
12      t←t+1;
13  end while
14  return(G(MaxIt), f(G(MaxIt))).
```

在算法 2 中,编码转换函数 $\boldsymbol{\Psi}_{\mathbf{DisEA}}$ 利用(7)式实现; 参数 r_1、r_2、c_1 和 c_2 与 PSO 中的含义完全相同; 参数 A 的值由 $\boldsymbol{\Psi}_{\mathbf{DisEA}}$ 的值域中 n 的大小决定, n 越大 A 的值越大.一般地,当 n=4 时可取 A=3.0.易知, DisPSO 的算法时间复杂度为 $O(MaxIt*(d+T(f))*N))$,$T(f)$表示计算函数值 $f(X)$的时间复杂度.

3 实例计算与比较

目前,经典演化算法 GA 已成为衡量新算法性能优劣的标准.因此,为了说明基于 ETM 构造的 DisEA 在求解组合优化问题时的性能优劣,我们利用 BPSO 和 HBDE 求解 SUKP 问题,利用 DisPSO 求解 D{0-1}KP 问题,通过与 GA 的计算结果比较验证它们的求解性能. 所有计算使用 Acer Aspire E1-570G 笔记本,硬件配置为 Intel(R) Core(TM)i5-3337u CPU-1.8GHz, 4GB DDR3 内存(3.82GB 可用),操作系统为 Microsoft Windows 8,用 C++语言编程实现各算法,编译环境为 Visual C++6.0.

3.1 BPSO、HBDE与GA求解SUKP的比较

SUKP[24]是 0-1 KP 的一个扩展形式.在 SUKP 中,给定一个元素集合 $U=\{1,2,\ldots,n\}$和一个项的集合 $S=\{1,2,\ldots,m\}$,其中每一个项 $i\in S$ (i=1,2,…,m)对应一个元素的非空子集 $U_i\subseteq U$,并具有一个价值 $p_i>0$;每一个元素 $j\in U$ (j=1,2,…,n)具有一个重量 $w_j>0$.对于任意非空子集 $A\subseteq S$,定义 A 的价值为 $P(A)=\sum_{i\in A}p_i$, A 的重量为 $W(A)=\sum_{j\in\bigcup_{i\in A}U_i}w_j$. SUKP 的目标是对于给定预算(或背包载重)$C$,求子集 $S^*\subseteq S$,使得在满足 $W(S^*)\leq C$ 的前提下

$P(S^*)$达到最大.

不失一般性,设 SUKP 中 p_i,w_j 和 C 均为正整数;子集族$\{U_1,U_2,\ldots,U_m\}$是 U 的一个覆盖,其中 $U_i \subset U$ (i=1,2,…,m)且 $U_i \neq \Phi$;$W(S)>C$ 且对$\forall i \in S$ 有 $\sum_{j\in U_i} w_j \leq C$.设 $Y=[y_1,y_2,\ldots,y_m] \in \{0,1\}^m$ 是一个 m 维 0-1 向量,$A_Y=\{i|y_i \in Y$ 且 $y_i=1,1\leq i\leq m\} \subseteq S$, 则 SUKP 的整数规划模型[25]为:

$$MAX \quad f(Y)=\sum_{i=1}^{m} y_i p_i \tag{9}$$

$$\text{s.t.} \quad W(A_Y)=\sum_{j\in \bigcup_{i\in A_Y} U_i} w_j \leq C \tag{10}$$

显然,任意 0-1 向量 $Y=[y_0,y_1,\ldots,y_m] \in \{0,1\}^m$ 都是 SUKP 的一个潜在解,当它满足约束条件(10)时是一个可行解,否则为一个不可行解.

文献[25]中提出的 BABC 即是一个基于 ETM 的 DisEA,通过对三类大规模 SUKP 实例的计算表明,BABC 的求解性能明显比 GA 的更优.因此,我们只需比较 BPSO、HBDE 与 GA 在求解 SUKP 问题时的性能优劣,即可判定基于 ETM 构造的离散演化算法在求解 SUKP 问题时的性能.

Table 1 Comparing with HBDE, BPSO and GA for the first kinds of SUKP Instances

表 1 利用 HBDE, BPSO 和 GA 求解第 1 类 SUKP 实例的计算结果比较

Instance	*CBEST*	Algorithm	*Best*	*Mean*	*Worst*	*StD*	*Time*
sukp 100_85	13283	HBDE	**13283**	13070.8	13003	75.25	0.179
_0.10_0.75		BPSO	13082	12979.2	12497	147.28	0.199
		GA	13044	12956.4	12596	130.66	0.112
sukp 100_85	12479	HBDE	**12479**	12201.6	12065	87.95	0.202
_0.15_0.85		BPSO	12238	12089.0	11576	128.99	0.225
		GA	12066	11546.0	11296	214.94	0.119
sukp 200_185	13405	HBDE	13402	13149.5	12563	115.67	1.450
_0.10_0.75		BPSO	13241	12831.6	11687	434.69	1.640
		GA	13064	12492.5	12596	320.03	1.013
sukp 200_185	14215	HBDE	14004	13503.9	12945	234.98	1.582
_0.15_0.85		BPSO	14044	13380.7	12782	332.27	1.737
		GA	13671	12802.9	12332	291.66	1.133
sukp 300_285	11407	HBDE	10553	10223	9812	181.49	4.828
_0.10_0.75		BPSO	10869	10371.9	9708	240.20	5.759
		GA	10553	9980.9	9640	142.97	3.608
sukp 300_285	12245	HBDE	**12245**	11212.8	10187	499.34	5.090
_0.15_0.85		BPSO	**12245**	11034.1	10056	581.86	5.812
		GA	11016	10349.8	9906	215.13	3.899
sukp 400_385	11435	HBDE	11321	10477.3	9530	397.22	11.572
_0.10_0.75		BPSO	11230	10580.0	9915	329.43	13.375
		GA	10083	9641.9	9370	168.94	9.779
sukp 400_385	10397	HBDE	**9649**	**9238.7**	**8871**	**204.474**	**12.625**
_0.15_0.85		BPSO	9990	9500.4	9081	278.67	15.076
		GA	9831	9326.8	8980	192.20	9.978
sukp 500_485	11716	HBDE	11085	**10507.6**	**10220**	**187.354**	**23.148**
_0.10_0.75		BPSO	11473	10839.5	10322	311.32	26.049
		GA	11031	10567.9	10288	123.15	18.198
sukp 500_485	9892	HBDE	9449	8935.3	8633	113.79	24.277
_0.15_0.85		BPSO	9456	9012.9	8627	197.57	29.790
		GA	9472	8692.7	8400	180.12	19.720

在 BPSO、HBDE 和 GA 中,个体编码都采用 n 维 0-1 向量形式,算法的最大迭代次数为 $MaxIt=Max\{m,n\}$,其中 m 为 SUKP 中项的个数,n 为元素个数;并且均使用文献[25]中提出的算法 S-GROA 处理求解过程中所产生的不可行解.在 HBDE 中,种群规模为 N=20,[-A,A]=[-3.0,3.0], 缩放因子 F=0.5, 变异因子 CR=0.3.在 BPSO 中,种群规模为 N=20,[-A,A]=[-5.0,5.0],加速常数为 c_1= c_2=2.0. GA 的种群规模为 N=50,其它参数设置与文献[25]中的完全相同,略之.

在表 1~表 3 中,*CBEST* 是各 SUKP 实例目前已知的最好结果,*Best* 和 *Worst* 是 HBDE、BPSO 和 GA 独立求解各实例 100 次所得结果中的最好值和最差值,*Mean* 和 *StD* 是 100 次求解结果的平均值和标准差,*Time* 是各算法独立求解每个实例一次的平均耗费时间.

Table 2 Comparing with HBDE, BPSO and GA for the second kinds of SUKP Instances

表 2 利用 HBDE, BPSO 和 GA 求解第 2 类 SUKP 实例的计算结果比较

Instance	*CBEST*	Results	*Best*	*Mean*	*Worst*	*StD*	*Time*
sukp 100_100	14044	HBDE	**13990**	**13771.1**	13681	71.53	0.202
_0.10_0.75		BPSO	**14044**	13846.1	13664	62.21	0.307
		GA	**14044**	13806.0	13587	144.91	0.129
sukp 100_100	13508	HBDE	**13508**	13377.3	12885	142.69	0.235
_0.15_0.85		BPSO	**13508**	13428.9	13104	115.87	0.270
		GA	13145	12234.8	11582	388.66	0.143
sukp 200_200	12350	HBDE	**12350**	11531.3	10898	266.66	1.633
_0.10_0.75		BPSO	12019	11344.8	10641	330.30	2.221
		GA	11656	10888.7	10337	237.85	1.106
sukp 200_200	12317	HBDE	11800	11163.8	10521	278.92	1.719
_0.15_0.85		BPSO	11821	11357.2	10607	381.88	1.922
		GA	11792	10827.5	10304	334.43	1.183
sukp 300_300	12695	HBDE	12210	12071.7	11747	87.37	5.308
_0.10_0.75		BPSO	12644	12187.6	11807	180.18	6.166
		GA	12055	11755.1	11169	144.45	3.789
sukp 300_300	11425	HBDE	10857	**9972.2**	**9447**	**183.91**	**5.681**
_0.15_0.85		BPSO	11007	10409.4	9463	304.99	6.707
		GA	10666	10099.2	9549	337.42	4.106
sukp 400_400	11490	HBDE	10847	10286.9	9894	206.51	11.583
_0.10_0.75		BPSO	11310	10600.5	10022	271.05	13.419
		GA	10570	10112.4	9786	157.89	9.187
sukp 400_400	10915	HBDE	10029	9276.4	8661	302.50	12.881
_0.15_0.85		BPSO	10404	9383.6	8597	411.48	15.103
		GA	9235	8793.8	8501	169.52	9.830
sukp 500_500	10960	HBDE	10605	10394.8	10148	108.52	26.518
_0.10_0.75		BPSO	10888	10522.4	10139	166.42	28.253
		GA	10460	10185.4	9919	114.19	20.717
sukp 500_500	10194	HBDE	9629	9233.1	8993	90.75	26.129
_0.15_0.85		BPSO	9840	9447.9	8731	202.16	32.389
		GA	9496	8882.9	8577	158.21	20.379

Table 3 Comparing with HBDE, BPSO and GA for the third kinds of SUKP Instances

表 3 利用 HBDE, BPSO and GA 求解第 3 类 SUKP 实例的计算结果比较

Instance	*CBEST*	Results	*Best*	*Mean*	*Worst*	*StD*	*Time*
sukp 85_100	12045	HBDE	**12045**	11263.4	11088	116.51	0.168
_0.10_0.75		BPSO	11710	11482.6	11174	189.17	0.190
		GA	11454	11092.7	10749	171.22	0.113
sukp 85_100	12369	HBDE	**12369**	12209.6	11595	201.33	0.193
_0.15_0.85		BPSO	**12369**	11750.7	11374	424.40	0.211
		GA	12124	11326.3	10369	417.00	0.131
sukp 185_200	13696	HBDE	13458	12836.6	12366	234.34	1.379
_0.10_0.75		BPSO	13497	12703.0	12247	382.50	1.572
		GA	12841	12236.6	11843	198.18	1.231
sukp 185_200	11298	HBDE	**11298**	10354.9	9912	240.506	1.522
_0.15_0.85		BPSO	**10920**	**10242.5**	**9783**	**373.53**	**1.732**
		GA	10920	10351.5	9832	208.08	1.204
sukp 285_300	11568	HBDE	11374	10943.4	10674	152.83	4.700
_0.10_0.75		BPSO	11538	11104.9	10419	190.23	5.612
		GA	10994	10640.1	10304	126.84	3.827
sukp 285_300	11763	HBDE	**10822**	**10080.1**	**9658**	**223.76**	**6.034**
_0.15_0.85		BPSO	11377	10529.8	9767	320.93	6.844
		GA	11093	10190.3	9737	249.76	3.990
sukp 385_400	10326	HBDE	10192	9747.0	9274	213.62	11.061
_0.10_0.75		BPSO	10252	9782.2	9089	222.64	12.985
		GA	9799	9432.8	9137	163.84	9.325
sukp 385_400	10302	HBDE	9770	9273.6	8859	167.11	12.684
_0.15_0.85		BPSO	**10302**	9131.5	8198	271.05	14.953
		GA	9173	8703.7	8342	154.15	9.911

sukp 485_500 _0.10_0.75	11037	HBDE	10835	10575.8	10098	219.49	22.211
		BPSO	10923	10461.7	9929	232.03	26.590
		GA	10311	9993.2	9799	117.73	18.708
sukp 485_500 _0.15_0.85	9964	HBDE	9396	9153.2	8753	126.20	25.060
		BPSO	9589	9180.1	8631	217.33	29.108
		GA	9329	8849.5	8586	141.84	20.129

由表 1 可以看出:从求解效果看,BPSO 求解所有实例的结果(*Best*、*Worst* 和 *Mean*)均优于 GA;除实例 sukp400_385_0.10_0.75 和 sukp500_485_0.10_0.75 以外,HBDE 求解其它实例的结果(*Best*、*Worst* 和 *Mean*)均优于 GA.从稳定性看,HBDE 的稳定性远远优于 GA 和 BPSO,GA 的略优于 BPSO.虽然 GA 的求解速度比 HBDE 和 BPSO 的快,但当实例规模不大时它们之间的差距很小.以上比较结果表明: HBDE 和 BPSO 比 GA 更适于求解 $n>m$ 的一类 SUKP 实例.

由表 2 可以看出:从求解效果看,BPSO 求解所有实例的结果(*Best*、*Worst* 和 *Mean*)均优于 GA; 除实例 sukp100_100_0.10_0.75 和 sukp300_300_0.15_0.85 之外,HBDE 求解其它实例的结果(*Best*、*Worst* 和 *Mean*)均优于 GA.从稳定性看,BPSO 和 HBDE 的算法稳定性远远优于 GA.从求解速度看,GA 的速度最快,HBDE 和 BPSO 的速度虽然略慢,但是与 GA 的差距不大.以上比较结果表明:HBDE 和 BPSO 比 GA 更适于求解 $n=m$ 的一类 SUKP 实例.

由表 3 可以看出:从求解效果看,除了实例 sukp185_200_0.15_0.85 之外,BPSO 求解其它实例的结果(*Best*、*Worst* 和 *Mean*)均优于 GA;除了实例 sukp285_300_0.15_0.85 以外,HBDE 求解其它实例的结果(*Best*、*Worst* 和 *Mean*)均优于 GA.从稳定性看, HBDE 的稳定性远远优于 GA 和 BPSO,GA 与 BPSO 的稳定性基本相同.虽然 GA 的求解速度比 HBDE 和 BPSO 的略快,但相比后者极佳的求解结果这点差距是微不足道的.以上比较表明:HBDE 和 BPSO 均比 GA 更适于求解 $n<m$ 的一类 SUKP 实例.

通过以上比较可以看出: HBDE 和 BPSO 求解 SUKP 问题的效果明显比 GA 的更佳,稳定性也更好,而且求解速度与 GA 差别不大,说明基于 ETM 方法构造的离散演化算法 HBDE 和 BPSO 是求解 SUKP 问题的有效算法,由此也表明基于 ETM 构造离散演化算法是一种简单且有效的方法.

3.2 DisPSO、FirEGA与SecEGA求解D{0-1}KP的比较

D{0-1}KP[26]是 Guldan 于 2007 年提出的一个新颖背包问题,它的定义为: 给定 n 个均含有 3 个项(或物品)的项集, 项集 $i(0\le i\le n\text{-}1)$中含有的 3 个项分别记为 $3i$, $3i+1$,$3i+2$, 其中前两个项 $3i$ 和 $3i+1$ 具有的价值系数分别为 p_{3i} 和 p_{3i+1},具有的重量系数分别为 w_{3i} 和 w_{3i+1}; 前两个项合并在一起构成第三个项 $3i+2$,它具有的价值系数为 $p_{3i+2}=p_{3i}+p_{3i+1}$,具有的折扣重量系数为 w_{3i+2}, 满足 $w_{3i+2}<w_{3i}+w_{3i+1}$ 并且 $w_{3i}<w_{3i+2}$, $w_{3i+1}<w_{3i+2}$. 在项集 $i(0\le i\le n\text{-}1)$中, 项 $3i$, $3i+1$,$3i+2$ 中至多有一个可以被选择装入载重为 C 的背包中. D{0-1} KP 为如何选择各项装入背包使得装入背包的所有项的重量系数之和在不超过背包载重的前提下价值系数之和达到最大?

不失一般性, 设 $p_j,w_j(0\le j\le 3n\text{-}1)$和 C 均为正整数, 且 $w_{3i+2}\le C$ $(0\le i\le n\text{-}1)$, $\sum_{i=0}^{n-1} w_{3i+2} > C$. 则 D{0-1}KP 的第一数学模型[26]为:

$$MAX\ f(X)=\sum_{i=0}^{n-1}(x_{3i}p_{3i}+x_{3i+1}p_{3i+1}+x_{3i+2}p_{3i+2}) \tag{11}$$

$$\text{s.t.}\quad x_{3i}+x_{3i+1}+x_{3i+2}\le 1,\quad i=0,1,\ldots,n\text{-}1, \tag{12}$$

$$\sum_{i=0}^{n-1}(x_{3i}w_{3i}+x_{3i+1}w_{3i+1}+x_{3i+2}w_{3i+2})\le C, \tag{13}$$

$$x_{3i}, x_{3i+1}, x_{3i+2}\in\{0,1\},\quad i=0,1,\ldots,n\text{-}1. \tag{14}$$

其中, 变量 $x_j(0\le j\le 3n\text{-}1)$表示项 j 是否被装入背包中，$x_j=1$ 表示项 j 被装入了背包中, $x_j=0$ 表示项 j 没有被装入背包. 显然,任意 $3n$ 维 0-1 向量均表示 D{0-1}KP 的一个潜在解, 当它同时满足约束条件(12)和(13)时为一个可行解.

令 $X=[x_0,x_1,\ldots,x_{n-1}]\in\{0,1,2,3\}^n$ 为一个 n 维整型向量,则 D{0-1}KP 的第二数学模型[27]为:

$$MAX \quad f(X)=\sum_{i=0}^{n-1}\lceil x_i/3\rceil p_{3i+|x_i-1|} \tag{15}$$

$$\text{s.t.} \quad \sum_{i=0}^{n-1}\lceil x_i/3\rceil w_{3i+|x_i-1|}\le C \tag{16}$$

$$x_i\in\{0,1,2,3\}, \quad i=0,1,\ldots,n-1. \tag{17}$$

其中$\lceil x\rceil$为顶函数;变量 x_i ($0\le i\le n-1$)表示项集 i 中是否存在项被装入了背包中, $x_i=0$ 项集 i 中没有项被装入背包, $x_i=1$ 表示项 $3i$ 被装入了背包中, $x_i=2$ 表示项 $3i+1$ 被装入了背包中, $x_i=3$ 表示项 $3i+2$ 被装入了背包中. 任意 n 维整型向量均表示 D{0-1}KP 的一个潜在解,当其满足约束条件(16)时即为一个可行解.

文献[27]利用 GA 求解 D{0-1}KP 问题, 分别基于不同的模型给出了求解它的两个有效算法 FirEGA 和 SecEGA.为此,下面利用 DisPSO 求解文献[27]中的四类大规模 D{0-1}KP 实例, 通过与 FirEGA 和 SecEGA 的计算结果进行比较验证 DisPSO 的求解性能.在利用 DisPSO 求解时使用第二数学模型,即个体的编码为 $\{0,1,2,3\}^n$ 上一个 n 维整型向量;种群规模和迭代次数与算法 FirEGA 和 SecEGA 的相同,即 $N=50$ 与 $MaxIt=3n$,n 为项集的个数;取$[-A,A]=[-3.0,3.0]$,加速常数为 $c_1=c_2=0.5$，并利用文献[27]中提出的 NROA 算法处理不可行解. FirEGA 和 SecEGA 的其它参数设置与处理不可行解的方法请参考文献[27],不再赘述.

在表 4~表 7 中,给出了所有实例的最优值 *OPT*, DisPSO、FirEGA 和 SecEGA 独立求解各实例 100 次的计算结果中的最好值 *Best* 和最差值 *Worst*, 各实例 100 次独立计算结果的数学期望 *Mean* 与标准差 *StD*, 以及各算法独立求解每个实例一次的平均耗费时间 *Time*.

Table 4 Comparison of calculation results of DisPSO, FirEGA and SecEGA for solving UDKP1~UDKP10

表 4 DisPSO, FirEGA 和 SecEGA 求解实例 UDKP1~UDKP10 的计算结果比较

Instance	*OPT*	Algorithm	*Best*	*Mean*	*Worst*	*StD*	*Time*
UDKP1	85740	DisPSO	85558	85262.3	84958	178.38	0.142
		FirEGA	80650	79103.2	77935	690.01	0.152
		SecEGA	78287	76807.2	75156	798.95	0.133
UDKP2	163744	DisPSO	161901	161280	160789	241.34	0.642
		FirEGA	155039	151662	149875	1044.95	0.596
		SecEGA	148043	145548	143833	883.43	0.543
UDKP3	269393	DisPSO	267142	266423	266679	172.71	1.327
		FirEGA	246698	240886	237980	1491.97	1.381
		SecEGA	228823	225492	222486	1353.58	1.161
UDKP4	347599	DisPSO	343990	343535	343058	196.82	2.518
		FirEGA	321605	317319	314486	1426.85	2.436
		SecEGA	305796	299978	297606	1435.46	2.173
UDKP5	442644	DisPSO	436132	435409	434796	266.93	3.894
		FirEGA	405409	399620	395367	1692.23	3.827
		SecEGA	376147	370808	367574	1611.71	3.360
UDKP6	536578	DisPSO	530009	529308	528743	247.34	5.579
		FirEGA	486556	478726	474015	2233.61	5.517
		SecEGA	447438	442499	438809	1765.28	4.811
UDKP7	635860	DisPSO	629658	629020	628539	251.35	7.828
		FirEGA	568119	560948	556938	2441.80	7.691
		SecEGA	529753	521401	518407	1813.04	6.522
UDKP8	650206	DisPSO	640262	639534	638519	285.05	11.844
		FirEGA	590137	585286	580684	2078.87	10.090
		SecEGA	550645	546678	543836	1449.36	9.109
UDKP9	718532	DisPSO	709112	708396	707786	257.29	15.653
		FirEGA	655172	649636	645012	2023.64	13.130
		SecEGA	613581	602215	605835	2003.75	11.374
UDKP10	779460	DisPSO	764299	763627	762612	335.37	19.192
		FirEGA	712270	706575	701545	2013.43	15.891
		SecEGA	665459	658908	655645	1723.80	14.773

由表 4 可以看出:从求解效果看,DisPSO 求解所有实例的结果(*Best*、*Worst* 和 *Mean*)远远优于 FirEGA 和 SecEGA 的求解结果.从稳定性看,DisPSO 的 *StD* 值远比 FirEGA 和 SecEGA 的更小,因此它的算法稳定性更佳.虽然 FirEGA 和 SecEGA 的求解速度比 DisPSO 的略快,但是这点差距不足以影响 DisPSO 在求解结果方面所具有的巨大优势,因此 DisPSO 比 FirEGA 和 SecEGA 更适用于求解 UKDP 类实例.

Table 5 Comparison of calculation results of DisPSO, FirEGA and SecEGA for solving WDKP1~WDKP10

表 5 DisPSO, FirEGA 和 SecEGA 求解实例 WDKP1~WDKP10 的计算结果比较

Instance	*OPT*	Algorithm	*Best*	*Mean*	*Worst*	*StD*	*Time*
WDKP1	83098	DisPSO	82976	82918.3	82825	32.25	0.181
		FirEGA	82803	82693.2	82592	52.04	0.157
		SecEGA	80014	79021.8	78096	473.67	0.140
WDKP2	138215	DisPSO	137912	137836	137747	27.48	0.746
		FirEGA	137704	137584	137356	63.23	0.653
		SecEGA	133315	132276	131337	415.62	0.562
WDKP3	256616	DisPSO	256097	255902	255774	65.78	1.469
		FirEGA	254240	253657	253307	173.01	1.366
		SecEGA	238331	235721	234025	873.58	1.242
WDKP4	315657	DisPSO	315117	314958	314717	76.13	2.674
		FirEGA	313966	312849	311998	484.76	2.381
		SecEGA	293640	290851	288764	950.06	2.299
WDKP5	428490	DisPSO	427802	427643	427200	91.80	4.131
		FirEGA	426783	424548	423058	798.53	3.815
		SecEGA	393617	390014	387992	1059.83	3.648
WDKP6	466050	DisPSO	464964	464755	464228	142.00	6.595
		FirEGA	463185	461672	457718	1107.57	5.521
		SecEGA	429208	425112	423269	1058.37	5.405
WDKP7	547683	DisPSO	546531	546269	545886	145.74	8.350
		FirEGA	544019	541949	538126	1224.68	7.736
		SecEGA	501557	496134	493845	1230.94	6.775
WDKP8	576959	DisPSO	575543	575200	574749	155.38	11.433
		FirEGA	573427	571559	563253	1495.36	10.041
		SecEGA	530971	523203	520350	2157.09	9.805
WDKP9	650660	DisPSO	648987	648565	648085	170.38	16.989
		FirEGA	647477	644820	630086	2056.06	13.318
		SecEGA	598343	586770	583854	2315.50	11.907
WDKP10	678967	DisPSO	677398	677092	675987	232.80	19.068
		FirEGA	675452	673008	668239	1441.96	16.210
		SecEGA	620230	606215	609964	3090.86	14.810

Table 6 Comparison of calculation results of DisPSO, FirEGA and SecEGA for solving SDKP1~SDKP10

表 6 DisPSO, FirEGA 和 SecEGA 求解实例 SDKP1~SDKP10 的计算结果比较

Instance	*OPT*	Algorithm	*Best*	*Mean*	*Worst*	*StD*	*Time*
SDKP1	94459	DisPSO	94286	94235.8	94121	39.32	0.136
		FirEGA	93276	93170.8	93070	42.15	0.135
		SecEGA	89769	88831.5	87463	594.91	0.129
SDKP2	160805	DisPSO	159837	159710	159376	66.13	0.627
		FirEGA	159159	159004	158859	96.53	0.617
		SecEGA	153821	152059	150753	489.39	0.575
SDKP3	238248	DisPSO	236402	236137	235767	101.19	1.417
		FirEGA	235454	235241	235043	86.52	1.412
		SecEGA	224997	223580	221918	543.38	1. 224
SDKP4	340027	DisPSO	336890	336554	336083	163.91	2.561
		FirEGA	336440	335963	335709	122.41	2.439
		SecEGA	318510	315513	313747	851.14	2.288
SDKP5	463033	DisPSO	460167	459736	459241	172.32	3.761
		FirEGA	452900	447587	444255	1974.99	3.579
		SecEGA	421108	416964	413933	1291.65	3.299
SDKP6	466097	DisPSO	460857	460415	459619	214.88	6.026
		FirEGA	459443	458893	458584	162.94	5.292
		SecEGA	430738	427304	425504	1031.12	4.923
SDKP7	620446	DisPSO	615877	615223	614214	281.73	7.989
		FirEGA	599361	592279	579673	3949.03	7.718
		SecEGA	561224	556083	552007	1926.26	6.368
SDKP8	670697	DisPSO	664634	664083	663413	263.27	10.502
		FirEGA	661563	660104	659928	426.06	10.242
		SecEGA	611644	606263	603774	1446.94	9.770

SDKP9	739121	DisPSO	731491	730837	729863	360.97	13.646
		FirEGA	729135	727544	727064	343.67	12.131
		SecEGA	674885	667900	664580	1614.04	11.086
SDKP10	765317	DisPSO	**755954**	755228	754456	319.49	17.134
		FirEGA	756205	753394	750757	985.46	15.572
		SecEGA	708935	695557	691994	2956.08	14.360

由表 5 可以看出:从求解效果看,除了 SDKP10 的 *Best* 以外,DisPSO 求解所有实例的结果(*Best*、*Worst* 和 *Mean*)均远远优于 FirEGA 和 SecEGA 的结果.从稳定性看, DisPSO 的算法稳定性最好,FirEGA 只是对部分实例而言稳定性较好,SecEGA 的稳定性极差.从求解速度看,虽然 FirEGA 和 SecEGA 比 DisPSO 的速度快,但是差距不大,因此 DisPSO 比 FirEGA 和 SecEGA 更适用于求解 SKDP 类实例.

由表 6 可以看出:从求解效果看,DisPSO 求解所有实例的结果(*Best*、*Worst* 和 *Mean*)远远优于 FirEGA 和 SecEGA 的求解结果.从稳定性看, DisPSO 的稳定性明显比 FirEGA 和 SecEGA 的更优,并且随着实例规模的增大,这种优势变得愈加明显.虽然 DisPSO 的求解速度比 FirEGA 和 SecEGA 的略慢,但是这点差距不足以撼动 DisPSO 在求解结果方面的明显优势,因此 DisPSO 比 FirEGA 和 SecEGA 更适用于求解 WKDP 类实例,.

Table 7 Comparison of calculation results of DisPSO, FirEGA and SecEGA for solving IDKP1~IDKP10

表 7 DisPSO, FirEGA 和 SecEGA 求解实例 IDKP1~IDKP10 的计算结果比较

Instance	*OPT*	Algorithm	*Best*	*Mean*	*Worst*	*StD*	*Time*
IDKP1	70106	DisPSO	**70106**	70083.4	70037	19.45	0.153
		FirEGA	**70106**	70074.4	70022	23.23	0.137
		SecEGA	68663	67999.8	67369	328.44	0.144
IDKP2	118268	DisPSO	**118268**	118240	118232	13.41	0.705
		FirEGA	118169	117869	117625	102.60	0.533
		SecEGA	114434	113385	112307	446.67	0.639
IDKP3	234804	DisPSO	234784	234740	234459	40.09	1.504
		FirEGA	234607	233997	233666	175.42	1.224
		SecEGA	220096	217982	216313	835.83	1.289
IDKP4	282591	DisPSO	**282591**	282554	282320	40.23	2.784
		FirEGA	282148	280695	278881	827.63	2.286
		SecEGA	263238	260425	258922	933.40	2.359
IDKP5	335584	DisPSO	**335584**	335529	335204	71.36	4.489
		FirEGA	335004	333484	329621	1173.90	4.338
		SecEGA	309573	306878	304881	907.19	3.868
IDKP6	452463	DisPSO	452433	452298	451773	122.48	6.107
		FirEGA	451799	449863	446704	1161.52	4.979
		SecEGA	414090	411367	408788	1099.31	5.391
IDKP7	489149	DisPSO	489137	489011	488543	98.42	8.645
		FirEGA	488009	485592	476385	2294.28	7.162
		SecEGA	451528	444316	442133	1280.31	7.576
IDKP8	533841	DisPSO	533836	533703	533357	103.21	11.983
		FirEGA	533035	529984	514196	2308.11	9.014
		SecEGA	490494	481831	478035	2215.66	9.571
IDKP9	528144	DisPSO	528133	527962	527598	125.30	14.716
		FirEGA	526410	523982	511651	2216.13	11.776
		SecEGA	489661	477001	471848	3656.22	12.907
IDKP10	581244	DisPSO	581237	581032	580583	136.14	18.423
		FirEGA	579220	576772	568903	1905.18	13.573
		SecEGA	535541	521604	516445	4265.07	15.026

由表 7 可以看出:从求解效果看,DisPSO 求解所有实例的结果(*Best*、*Worst* 和 *Mean*)远远优于 FirEGA 和 SecEGA 的求解结果,而且 DisPSO 能够求得 IDKP1,IDKP2,IDKP4 和 IDKP5 的最优值.从稳定性看,DisPSO 的算法稳定性远远比 FirEGA 和 SecEGA 的更优,而且随着实例规模的增大,这种优势变得愈加突出.虽然 DisPSO 的求解速度比 FirEGA 和 SecEGA 的略慢,但是对其在求解结果方面的优势影响不大,因此 DisPSO 比 FirEGA 和 SecEGA 更适用于求解 IKDP 类实例,.

以上比较结果表明: DisPSO 比 FirEGA 和 SecEGA 更适于求解 D{0-1}KP 问题,这说明基于 ETM 方法构造的离散演化算法 DisPSO 不仅简单、易于实现,而且求解 D{0-1}KP 问题的效果极佳,由此进一步表明基于 ETM 方法构造 DisEA 不仅是可行的,而且是高效的.

4 结束语

本文在分析 GA、BPSO 和 HBDE 中所使用的编码转换函数的共性基础上,提出了一种基于编码转换思想设计离散演化算法的实用方法:编码转换法(ETM),在给出一个简单且具有普遍适用性的编码转换函数 $\boldsymbol{\Psi}_{\mathbf{DisEA}}$ 的基础上,提出了基于 ETM 设计离散演化算法的一般算法框架 **A-DisEA**. 为了说明 ETM 的简单性与实用性,先基于 **A-DisEA** 一般框架给出了一个离散粒子群优化算法 DisPSO,然后分别利用 BPSO、HBDE 和 DisPSO 求解 SUKP 问题和 D{0-1}KP 问题,通过与 GA 及其改进算法的比较验证了这些算法在求解 KP 问题方面的优越性能,由此说明对已有 EAs(例如 PSO、DE 和 ABC 等)使用 ETM 方法基于 **A-DisEA** 一般框架构造相应的离散演化算法是一种简单有效的实用方法.

当前,存在许多组合优化问题的可行解表示为$\{0,1,\ldots,n\text{-}1\}^{\mathrm{d}}$ ($n\geq 4$)上的一个 d 维整数向量更适宜,例如有界背包问题[35]和广义二次多背包问题[36]等, DisPSO(或 DisDE、DisABC)求解这些问题的效果如何?还是一个有待于今后探讨的问题.此外,萤火虫算法(FFA)[12]、烟花算法(FWA)[13]、果蝇优化(FFO)[14]、灰狼优化(GWO)[15]、人工藻算法(AAA)[16]和鸽群优化(PIO)[17]等新被提出的这些算法主要用于求解数值优化问题,如何利用它们高效求解组合优化问题是一个有意义的研究问题;为此,今后将研究它们基于 ETM 构造的离散演化算法在求解组合优化问题方面的优劣.由于 $\boldsymbol{\Psi}_{\mathbf{DisEA}}$ 在 ETM 方法中的重要地位,深入探讨简单易行、效果极佳且具有普遍适用性的新编码转换函数是一个值得进一步研究的问题.

References:

[1] Kenneth A. De Jong. Evolutionary Computation--A Unified Approach. Cambridge: MIT Press, 2016.

[2] Mitchell M. An Introduction to Genetic Algorithms. Cambridge: MIT Press, 1996.

[3] Poli R, Kennedy J, and Blackwell T. Particle swarm optimization: An overview. Swarm Intell, 2007, 1(1): 33-57.

[4] Storn R, Price K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997, 11(4): 341-359.

[5] Dorigo M, and Stützle T. Ant Colony Optimization. Cambridge. MA: MIT Press, 2004.

[6] Mehdi Neshat,Ghodrat Sepidnam,Mehdi Sargolzaei, Adel Najaran Toosi. Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. Artificial Intelligence Review, 2014, 42(4): 965-997.

[7] X.Yao, Chen G L, Xu X H, Liu Y. A survey of evolutionary algorithms. Chinese Journal of Computers, 1995, 18(9): 694-706.

[8] Wang L. Intelligent optimization algorithms with applications. Beijing: Tsinghua University Press, 2001.

[9] Gong MG, Jiao LC, Yang DD, Ma WP. Research on evolutionary multi-objective optimization algorithms. Journal of Software, 2009, 20(2): 271−289.

[10] Coello C A C. An Introduction to Evolutionary Algorithms and Their Applications. Advanced Distributed Systems. Springer Berlin Heidelberg, 2005: 425-442.

[11] Dervis Karaboga, Barbaros Basturk. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization, 2007, 39(3): 459-471.

[12] Yang, X.-S. Firefly algorithms for multimodal optimization. In Stochastic algorithms: Foundations and applications, SAGA 2009. Lecture notes in computer sciences. 2009, 5792: 169-178.

[13] Tan Y, Zhu Y. Fireworks algorithm for optimization. In: Tan Y, Shi Y, Tan KC, editors. ICSI 2010. LNCS, vol. 6145. Heidelberg: Springer. 2010, 355–64.

[14] Wen-Tsao Pan. A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example. Knowledge-Based Systems, 2012, 26(2): 69-74.

[15] Seyedali Mirjalili, Seyed Mohammad Mirjalili, Andrew Lewis. Grey Wolf Optimizer. Advances in Engineering Software, 2014, 69(1): 46-61.

[16] Sait Ali Uymaz, Gulay Tezel, Esra Yel. Artificial algae algorithm (AAA) for nonlinear global optimization. Applied Soft Computing, 2015,31(C): 153-171.

[17] DUAN Haibin, YE Fei. Progresses in Pigeon-inspired Optimization Algorithms. Journal of Beijing University of Technolory (Natural Sciences Edition), 2017,43(1): 1-7.

[18] William M. Fatih Tasgetiren,Quan-Ke Pan,P.N. Suganthan,Angela H-L Chen. A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. Information Sciences, 2011,181(16):3459-3475.

[19] Dongli Jia,Xintao Duan,Muhammad Khurram Khan. Binary Artificial Bee Colony optimization using bitwise operation. Computers & Industrial Engineering, 2014,76(C): 360–365.

[20] Mustafa Servet Kiran. The continuous artificial bee colony algorithm for binary optimization. Applied Soft Computing. 2015,33 (C): 15-23.

[21] He Yichao, Wang Xizhao, Kou Yingzhan. A Binary Differential Evolution Algorithm with Hybrid Encoding. Journal of Computer Research and Development. 2007, 44(9):1476-1484.

[22] Yang Yu, Xin Yao,Zhi-Hua Zhou.On the approximation ability of evolutionary optimization with application to minimum set cover. Artificial Intelligence, 2012, 180–181(2):20–33.

[23] Kennedy J, Eberhart R C, A discrete binary version of the particle swarm algorithm. In: Proceedings of 1997 Conference on System, Man, and Cybernetices, 4104-4109, 1997.

[24] Ashwin Arulselvan. A note on the set union knapsack problem. Discrete Applied Mathematics, 2014, 169(41):214-218.

[25] Y. He, H. Xie, T. Wong, X. Wang, A novel binary artificial bee colony algorithm for the set-union knapsack problem, Future Generation Computer Systems, 2018, 78(1): 77-86.

[26] Aiying Rong, José Rui Figueira, Kathrin Klamroth. Dynamic programming based algorithms for the discounted {0-1} knapsack problem. Applied Mathematics and Computation. 2012, 218(12): 6921- 6933.

[27] HE Yi-Chao, WANG Xi-Zhao, LI Win-Bin, ZHANG Xin-Lu, CHEN Yi-Ying. Research on Genetic Algorithms for the Discounted {0-1} Knapsack Problem, Chinese Journal of Computers, 2016, 39(12): 2614-2630.

[28] Chen GL, Wang XF, Zhuang ZD, Wang DS. Genetic algorithms and its applications. Beijing: Science Press, 2003.

[29] Deng L, Zhao J, Wang X. Genetic algorithm solution of network coding optimization. Journal of Software, 2009, 20(8): 2269−2279. http://www.jos.org.cn/1000-9825/3370.htm.

[30] Wang Z, Fan XY, Zou YG, Chen X. Genetic algorithm based multiple faults localization technique. Ruan Jian Xue Bao/Journal of Software, 2016, 27(4): 879−900 (in Chinese). http://www.jos.org.cn/1000-9825/4970.htm.

[31] Zeng Jian-chao, Jie Jing, Cui Zhi-hua. Particle swarm optimization algorithm. Beijing: Science Press, 2004.

[32] Kenneth V Price, RainerM Storn, Jouni A Lampinen. Differential evolution: A practical approach to global optimization. Springer, 2005.

[33] He YC, Wang XZ, Li WB, Zhao SL. Exact algorithms and evolutionary algorithms for randomized time-varying knapsack problem. Ruan Jian Xue Bao/Journal of Software, 2017, 28(2):185−202 (in Chinese). http://www.jos.org.cn/1000-9825/4937.

[34] Yousef Kilani. Comparing the performance of the genetic and local search algorithms for solving the satisfiability problems. Applied Soft Computing, 2010, 10 (1): 198–207.

[35] Hans Kellerer, Ulrich Pferschy, David Pisinger. Knapsack Problems. Berlin: Springer, 2004.

[36] Chen Y, Hao J K. Memetic Search for the Generalized Quadratic Multiple Knapsack Problem. IEEE Transactions on Evolutionary Computation, 2016, 20(6): 908-923.

附中文参考文献:

[7] 姚新,陈国良,徐惠敏,刘勇.进化算法研究进展.计算机学报,1995,18(js), 694-706.

[8] 王凌.智能优化算法及其应用.北京:清华大学出版社,2001.

[9] 公茂果,焦李成,杨咚咚,马文萍.进化多目标优化算法研究.软件学报,2009, 20(2): 271-289.

[17] 段海滨,叶飞.鸽群优化算法研究进展.北京工业大学学报(自然科学版), 2017,43(1): 1-7.

[21] 贺毅朝,王熙照,寇应展.一种具有混合编码的二进制差分演化算法.计算机研究与发展, 2007, 44(9): 1476-1484.

[27] 贺毅朝,王熙照,李文斌,张新禄,陈嶷瑛.基于遗传算法求解折扣{0-1}背包问题的研究.计算机学报, 2016, 39(12): 2614-2630.

[28] 陈国良,王熙法,庄镇泉,王东生.遗传算法及其应用.北京:人民邮电出版社.2003:1-162.

[29] 邓亮,赵进,王新.基于遗传算法的网络编码优化.软件学报,2009,20(8):2269-2279.

[30] 王赞,樊向宇,邹雨果,陈翔.一种基于遗传算法的多缺陷定位方法.软件学报,2016,27(4):879-900.

[31] 曾建潮,介婧,崔志华.微粒群算法.北京:科学出版社,2004.

[33] 贺毅朝,王熙照,李文斌,赵书良.求解随机时变背包问题的精确算法与进化算法.软件学报,2017,28(2):185-202..