

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, 给出了一种基于映射变换思想设计离散演化算法(DisEA)的实用方法——编码转换法(ETM)。为了说明ETM的实用性与有效性, 首先, 基于ETM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等基于ETM构造的演化算法求解集合联盟背包问题和折扣{0-1}背包问题。通过与GA的计算结果比较指出, BPSO、HBDE和DisPSO的求解性能均优于GA, 说明基于ETM提出的DisEA在求解背包问题方面具有良好的性能。由此表明, 利用ETM方法设计DisEA是一种实用的有效方法。

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: TP181

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): 2580–2594. <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 algorithm based on encoding transformation. Ruan Jian Xue Bao/Journal of Software, 2018, 29(9): 2580–2594 (in Chinese). <http://www.jos.org.cn/1000-9825/5400.htm>

Design and Applications of Discrete Evolutionary Algorithm Based on Encoding Transformation

HE Yi-Chao¹, WANG Xi-Zhao², ZHAO Shu-Liang³, ZHANG Xin-Lu³

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, this study draws lessons from the design concept of genetic algorithm (GA), binary particle swarm optimization (BPSO) and binary differential evolution with hybrid encoding (HBDE), to propose a simple and practical method for designing discrete evolution algorithm (DisEA) based on the idea of mapping transformation. This method is named encoding transformation method (ETM). For illustrating the practicability of ETM, a discrete particle swarm optimization (DisPSO) algorithm based on ETM is presented. For showing the feasibility and effectiveness of

* 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(ZD2016005); 河北省自然科学基金(F2016403055)

Foundation item: National Natural Science Foundation of China (61503252, 71371063, 11471097); Shenzhen Science and Technology Project (JCYJ20150324140036825); Scientific Research Plan of the Higher Education Institutions of Hebei Province, China (ZD2016005); Natural Science Foundation of Hebei Province of China (F2016403055)

本文由演化学习专题特约编辑俞扬副教授、钱超副研究员推荐。

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>

ETM, GA, BPSO, HBDE and DisPSO are used 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 design of DisEA based on 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 algorithm,简称 EA)^[1]是一类群体智能算法,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 EA 有遗传算法(genetic algorithm,简称 GA)^[2]、粒子群优化(particle swarm optimization,简称 PSO)^[3]、差分演化(differential evolution,简称 DE)^[4]、蚁群优化(ant colony optimization,简称 ACO)^[5]和人工鱼群(artificial fish swarm,简称 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 EA 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,简称 ABC)^[11]、萤火虫算法(firefly algorithm,简称 FFA)^[12]、烟花算法(fireworks algorithm,简称 FWA)^[13]、果蝇优化(fruit fly optimization,简称 FFO)^[14]、灰狼优化(grey wolf optimizer,简称 GWO)^[15]、人工藻算法(artificial algae algorithm,简称 AAA)^[16]和鸽群优化(pigeon-inspired optimization,简称 PIO)^[17]等许多新颖演化算法。虽然大量新算法的提出壮大了 EA 家族,为应用 EA 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 EA 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。为了改变这种不利现状,人们对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,以实现组合优化问题的求解;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。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,简称 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,简称 SUKP)^[24,25]和折扣{0-1}背包问题(discounted {0-1} knapsack problem,简称 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组合优化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^[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 3 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在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,请见文献^[28]中所述。

利用 GA 求解数值优化问题时,一般个体编码采用 0-1 向量的形式,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 $X=[x_{11},x_{12},\dots,x_{1k},x_{21},\dots,x_{2k},x_{31},\dots,x_{dk}] \in \{0,1\}^{dk}$,它所对应的解向量为 $Y=[y_1,y_2,\dots,y_d]$,其中 $y_i \in [L_i,U_i], L_i < U_i$ 且为实数, $i=1,2,\dots,d$, 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 的编码转化函数映射为 ψ_{GA} ,则 $Y=\psi_{GA}(X)$,且对于 $i=1,2,\dots,d$ 有:

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

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} 也可以看做是一个双射.

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^[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,不断搜寻更佳的结果,以实现算法的进化过程.

设粒子的速度为 $V=[v_1,v_2,\dots,v_d]$,位置为 $X=[x_1,x_2,\dots,x_d]$, $P=[p_1,p_2,\dots,p_d]$ 表示粒子的历史最好位置, $G=[g_1,g_2,\dots,g_d]$ 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, $j=1,2,\dots,d$, d 为问题的维数,则 PSO 依次利用公式(2)与公式(3)实现粒子速度与位置的更新:

$$v_j = v_j + c_1 \times r_1 \times (p_j - x_j) + c_2 \times r_2 \times (g_j - x_j) \quad (2)$$

$$x_j = v_j + x_j \quad (3)$$

其中 $j=1,2,\dots,d; r_1 \sim U(0,1)$ 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,简称 BPSO).在 BPSO 中,粒子的速度 $V=[v_1,v_2,\dots,v_d]$ 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 $X=[x_1,x_2,\dots,x_d] \in \{0,1\}^d$ 取代,粒子速度更新方程仍使用公式(2),粒子的位置更新方程改为由公式(4)实现:

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

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 利用公式(4)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^[4,32]是 Storn 和 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 1 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/tr/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 $X_1=[x_{11},x_{12},\dots,x_{1d}], X_2=[x_{21},x_{22},\dots,x_{2d}], X_3=[x_{31},x_{32},\dots,x_{3d}]$ 是 DE 种群中不同于 $X=[x_1,x_2,\dots,x_d]$ 的 3 个互不相同的个体, $V=[v_1,v_2,\dots,v_d]$ 是相应于个体 X 的中间个体,其中 $x_{1j},x_{2j},x_{3j},v_j \in [L_j,U_j], j=1,2,\dots,d$, d 为问题维数,则中间个体 V 利用下面的公式(5)产生:

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

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, 并仍然使用公式(5)产生中间个体 V . 此外, 引入一个 0-1 向量 $Y = [y_1, y_2, \dots, y_d] \in \{0, 1\}^d$ 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 $y_j (j=1, 2, \dots, d)$ 与 x_j 之间的对应关系见下式:

$$y_j = \begin{cases} 1, & \text{if } x_j > 0 \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

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), 也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 E_1 是演化算法 A 中个体编码构成的度量空间, 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 E_1 到 E_2 上的一个编码转换函数 ψ_A 对算法 A 离散化的方法称为编码转换法(encoding transformation method, 简称 ETM). 一般情况下, 函数 ψ_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 ψ_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 $\{0, 1, \dots, n-1\}^d$ (其中 $n \geq 2$) 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 ψ_{DisEA} .

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

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

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

容易看出, 映射 ψ_{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: $I_0 = [-A, -A + 2A \times \alpha_0), I_1 = [-A + 2A \times \alpha_0, -A + 2A \times (\alpha_0 + \alpha_1)), \dots, I_{n-2} = [-A + 2A \times (\alpha_0 + \dots + \alpha_{n-3}), -A + 2A \times (\alpha_0 + \dots + \alpha_{n-2}), I_{n-1} = [-A + 2A \times (\alpha_0 + \dots + \alpha_{n-2}), A]$, 当分量 $x_j \in I_k$ 时 $y_j = k$.

在理论上, 公式(7)中满足条件 $\alpha_0 + \alpha_1 + \dots + \alpha_{n-1} = 1$ 且 $0 < \alpha_j < 1$ 的实数 $\alpha_0, \alpha_1, \dots, \alpha_{n-1}$ 有无数个, 如何选取它们的值, 是 ψ_{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 值也是一个不确定的量, 因此我们可以取 $\alpha_j = 1/n (j=0, 1, \dots, n-1)$, 于是, 公式(7)简化为

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

映射 $\psi_{DisEA}: [-A, A]^d \rightarrow \{0, 1, \dots, n-1\}^d$ 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 ψ_{DisEA} 的特例. 为了说明如何利用 ψ_{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, $Y \in \{0, 1, \dots, n-1\}^d$; 算法 A 的参数;

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

- 1 随机产生初始种群 $P(0) = \{X_i(0) = [x_{i1}(0), x_{i2}(0), \dots, x_{id}(0)] \in [-A, A]^d | 1 \leq i \leq N\}$;
- 2 $Y_i(0) \leftarrow \psi_{DisEA}(X_i(0)), 1 \leq i \leq N$; // 计算个体 $X_i(0)$ 对应的解
- 3 根据 $f(Y_i(0))$ 确定 $P(0)$ 中的最优个体 $B(0) = [b_1(0), b_2(0), \dots, b_d(0)]$ 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 $P(t) = \{X_i(t) = [x_{i1}(t), x_{i2}(t), \dots, x_{id}(t)] \in [-A, A]^d | 1 \leq i \leq N\}$;
- 6 $Y_i(t) \leftarrow \psi_{DisEA}(X_i(t)), 1 \leq i \leq N$; // 计算个体 $X_i(t)$ 对应的潜在解
- 7 根据 $f(Y_i(t))$ 确定 $P(0) \cup P(1) \cup \dots \cup P(t)$ 中最优个体 $B(t) = [b_1(t), b_2(t), \dots, b_d(t)]$ 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 $MaxIt \times (T(A) + O(d \times N) + N \times T(f))$.

2.2 基于 ETM 的离散粒子群优化算法

BPSO, HBDE 和 BABC 等 DisEA 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 $\{0, 1, \dots, n-1\}^d (n \geq 3)$ 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 $V_i(t) = [v_{i1}(t), v_{i2}(t), \dots, v_{id}(t)] \in [-A, A]^d$ 表示 DisPSO 的第 t 代种群 $P(t)$ 中的第 i 个粒子的飞行速度, 它的位置记为 $X_i(t) = [x_{i1}(t), x_{i2}(t), \dots, x_{id}(t)] \in \{0, 1, \dots, n-1\}^d (n \geq 3)$, 表示所求解问题的一个可行解 (或潜在解). 又记 $P_i(t) = [p_{i1}(t), p_{i2}(t), \dots, p_{id}(t)] \in \{0, 1, \dots, n-1\}^d$ 为第 i 个粒子的历史最好位置, $G(t) = [g_1(t), g_2(t), \dots, g_d(t)] \in \{0, 1, \dots, n-1\}^d$ 为 $P(0) \cup P(1) \cup \dots \cup P(t)$ 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 $\text{Max}f(X), X \in \{0, 1, \dots, n-1\}^d (n \geq 3)$, DisPSO 的算法伪代码描述如下:

算法 2. DisPSO.

Input: $\text{Max}f(X)$ 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 $G(MaxIt) \in \{0, 1, \dots, n-1\}^d$ 和 $f(G(MaxIt))$.

- 1 随机产生初始种群 $P(0) = \{V_i(0) = [v_{i1}(0), v_{i2}(0), \dots, v_{id}(0)] \in [-A, A]^d | 1 \leq i \leq N\}$;
- 2 $X_i(0) \leftarrow \psi_{DisEA}(V_i(0))$, 并根据 $f(X_i(0))$ 确定 $P_i(0), 1 \leq i \leq N$;
- 3 确定 $P(0)$ 中的全局最好位置 $G(0)$; $t \leftarrow 0$;
- 4 **while** ($t \leq MaxIt$) **do**
- 5 **for** $i \leftarrow 1$ **to** N **do**

```

6      for  $j \leftarrow 1$  to  $d$  do //生成第  $t+1$  代种群  $P(t+1)$ 
7           $v_{ij}(t+1) \leftarrow v_{ij}(t) + c_1 \times r_1 \times (p_{ij}(t) - x_{ij}(t)) + c_2 \times r_2 \times (g_j(t) - x_{ij}(t));$ 
8      end for
9       $X_i(t+1) \leftarrow \psi_{DisEA}(V_i(t+1))$ ,并根据  $f(X_i(t+1))$  确定  $P_i(t+1)$ ;
10     end for
11     确定  $P(0) \cup P(1) \cup \dots \cup P(t+1)$  中的全局最好位置  $G(t+1)$ ;
12      $t \leftarrow t+1$ ;
13 end while
14 return ( $G(\text{MaxIt}), f(G(\text{MaxIt}))$ ).
    
```

在算法 2 中,编码转换函数 ψ_{DisEA} 利用公式(7)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 ψ_{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 $O(\text{MaxIt} \times (d+T(f)) \times N)$, $T(f)$ 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 $U=\{1,2,\dots,n\}$ 和一个项的集合 $S=\{1,2,\dots,m\}$,其中:每一个项 $i \in S(i=1,2,\dots,m)$ 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 $j \in U (j=1,2,\dots,n)$ 具有一个重量 $w_j > 0$.对于任意的非空子集 $A \subseteq S$,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 $\{U_1, U_2, \dots, U_m\}$ 是 U 的一个覆盖,其中: $U_i \subseteq U (i=1,2,\dots,m)$ 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 $Y=[y_1, y_2, \dots, y_m] \in \{0,1\}^m$ 是一个 m 维 0-1 向量, $A_Y = \{i | y_i \in Y \text{ 且 } y_i = 1, 1 \leq i \leq m\} \subseteq S$, 则 SUKP 的整数规划模型^[25]为

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

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

显然,任意 0-1 向量 $Y=[y_0, y_1, \dots, y_m] \in \{0,1\}^m$ 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 3 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.

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 $\text{MaxIt} = \text{Max}\{m, n\}$,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 $N=50$,其他参数设置与文献[25]中的完

全相同,略。

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 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
sukp100_85_0.10_0.75	13 283	HBDE	13 283	13 070.8	13 003	75.25	0.179
		BPSO	13 082	12 979.2	12 497	147.28	0.199
		GA	13 044	12 956.4	12 596	130.66	0.112
sukp100_85_0.15_0.85	12 479	HBDE	12 479	12 201.6	12 065	87.95	0.202
		BPSO	12 238	12 089.0	11 576	128.99	0.225
		GA	12 066	11 546.0	11 296	214.94	0.119
sukp200_185_0.10_0.75	13 405	HBDE	13 402	13 149.5	12 563	115.67	1.450
		BPSO	13 241	12 831.6	11 687	434.69	1.640
		GA	13 064	12 492.5	12 596	320.03	1.013
sukp200_185_0.15_0.85	14 215	HBDE	14 004	13 503.9	12 945	234.98	1.582
		BPSO	14 044	13 380.7	12 782	332.27	1.737
		GA	13 671	12 802.9	12 332	291.66	1.133
sukp300_285_0.10_0.75	11 413	HBDE	10 553	10 223	9 812	181.49	4.828
		BPSO	10 869	10 371.9	9 708	240.20	5.759
		GA	10 553	9 980.9	9 640	142.97	3.608
sukp300_285_0.15_0.85	12 245	HBDE	12 245	11 212.8	10 187	499.34	5.090
		BPSO	12 245	11 034.1	10 056	581.86	5.812
		GA	11 016	10 349.8	9 906	215.13	3.899
sukp400_385_0.10_0.75	11 435	HBDE	11 321	10 477.3	9 530	397.22	11.572
		BPSO	11 230	10 580.0	9 915	329.43	13.375
		GA	10 083	9 641.9	9 370	168.94	9.779
sukp400_385_0.15_0.85	10 397	HBDE	9 649	9 238.7	8 871	204.474	12.625
		BPSO	9 990	9 500.4	9 081	278.67	15.076
		GA	9 831	9 326.8	8 980	192.20	9.978
sukp500_485_0.10_0.75	11 716	HBDE	11 085	10 507.6	10 220	187.354	23.148
		BPSO	11 473	10 839.5	10 322	311.32	26.049
		GA	11 031	10 567.9	10 288	123.15	18.198
sukp500_485_0.15_0.85	9 892	HBDE	9 449	8 935.3	8 633	113.79	24.277
		BPSO	9 456	9 012.9	8 627	197.57	29.790
		GA	9 472	8 692.7	8 400	180.12	19.720

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
sukp100_100_0.10_0.75	14 044	HBDE	13 990	13 771.1	13 681	71.53	0.202
		BPSO	14 044	13 846.1	13 664	62.21	0.307
		GA	14 044	13 806.0	13 587	144.91	0.129
sukp100_100_0.15_0.85	13 508	HBDE	13 508	13 377.3	12 885	142.69	0.235
		BPSO	13 508	13 428.9	13 104	115.87	0.270
		GA	13 145	12 234.8	11 582	388.66	0.143
sukp200_200_0.10_0.75	12 350	HBDE	12 350	11 531.3	10 898	266.66	1.633
		BPSO	12 019	11 344.8	10 641	330.30	2.221
		GA	11 656	10 888.7	10 337	237.85	1.106
sukp200_200_0.15_0.85	12 317	HBDE	11 800	11 163.8	10 521	278.92	1.719
		BPSO	11 821	11 357.2	10 607	381.88	1.922
		GA	11 792	10 827.5	10 304	334.43	1.183
sukp300_300_0.10_0.75	12 844	HBDE	12 210	12 071.7	11 747	87.37	5.308
		BPSO	12 644	12 187.6	11 807	180.18	6.166
		GA	12 055	11 755.1	11 169	144.45	3.789
sukp300_300_0.15_0.85	11 425	HBDE	10 857	9 972.2	9 447	183.91	5.681
		BPSO	11 007	10 409.4	9 463	304.99	6.707
		GA	10 666	10 099.2	9 549	337.42	4.106

Table 2 Comparing with HBDE, BPSO and GA for the second kinds of SUKP instances (Continued)

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

Instance	CBEST	Results	Best	Mean	Worst	StD	Time
sukp400_400_0.10_0.75	11 490	HBDE	10 847	10 286.9	9 894	206.51	11.583
		BPSO	11 310	10 600.5	10 022	271.05	13.419
		GA	10 570	10 112.4	9 786	157.89	9.187
sukp400_400_0.15_0.85	10 915	HBDE	10 029	9 276.4	8 661	302.50	12.881
		BPSO	10 404	9 383.6	8 597	411.48	15.103
		GA	9 235	8 793.8	8 501	169.52	9.830
sukp500_500_0.10_0.75	10 960	HBDE	10 605	10 394.8	10 148	108.52	26.518
		BPSO	10 888	10 522.4	10 139	166.42	28.253
		GA	10 460	10 185.4	9 919	114.19	20.717
sukp500_500_0.15_0.85	10 194	HBDE	9 629	9 233.1	8 993	90.75	26.129
		BPSO	9 840	9 447.9	8 731	202.16	32.389
		GA	9 496	8 882.9	8 577	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
sukp85_100_0.10_0.75	12 045	HBDE	12 045	11 263.4	11 088	116.51	0.168
		BPSO	11 710	11 482.6	11 174	189.17	0.190
		GA	11 454	11 092.7	10 749	171.22	0.113
sukp85_100_0.15_0.85	12 369	HBDE	12 369	12 209.6	11 595	201.33	0.193
		BPSO	12 369	11 750.7	11 374	424.40	0.211
		GA	12 124	11 326.3	10 369	417.00	0.131
sukp185_200_0.10_0.75	13 696	HBDE	13 458	12 836.6	12 366	234.34	1.379
		BPSO	13 497	12 703.0	12 247	382.50	1.572
		GA	12 841	12 236.6	11 843	198.18	1.231
sukp185_200_0.15_0.85	11 298	HBDE	11 298	10 354.9	9 912	240.506	1.522
		BPSO	10 920	10 242.5	9 783	373.53	1.732
		GA	10 920	10 351.5	9 832	208.08	1.204
sukp285_300_0.10_0.75	11 568	HBDE	11 374	10 943.4	10 674	152.83	4.700
		BPSO	11 538	11 104.9	10 419	190.23	5.612
		GA	10 994	10 640.1	10 304	126.84	3.827
sukp285_300_0.15_0.85	11 763	HBDE	10 822	10 080.1	9 658	223.76	6.034
		BPSO	11 377	10 529.8	9 767	320.93	6.844
		GA	11 093	10 190.3	9 737	249.76	3.990
sukp385_400_0.10_0.75	10 326	HBDE	10 192	9 747.0	9 274	213.62	11.061
		BPSO	10 252	9 782.2	9 089	222.64	12.985
		GA	9 799	9 432.8	9 137	163.84	9.325
sukp385_400_0.15_0.85	10 302	HBDE	9 770	9 273.6	8 859	167.11	12.684
		BPSO	10 302	9 131.5	8 198	271.05	14.953
		GA	9 173	8 703.7	8 342	154.15	9.911
sukp485_500_0.10_0.75	11 037	HBDE	10 835	10 575.8	10 098	219.49	22.211
		BPSO	10 923	10 461.7	9 929	232.03	26.590
		GA	10 311	9 993.2	9 799	117.73	18.708
sukp485_500_0.15_0.85	9 964	HBDE	9 396	9 153.2	8 753	126.20	25.060
		BPSO	9 589	9 180.1	8 631	217.33	29.108
		GA	9 329	8 849.5	8 586	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 求解其他实例的结果(*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 略优于 BPSO;
- 虽然 GA 的求解速度比 HBDE 和 BPSO 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 求解其他实例的结果(*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 求解其他实例的结果(*Best,Worst* 和 *Mean*)均优于 GA;除了实例 `sukp285_300_0.15_0.85` 以外,HBDE 求解其他实例的结果(*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,但相比后者最佳的求解结果,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.

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 Guldán 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 $i(0 \leq i \leq n-1)$ 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 3 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 $w_{3i+2} \leq w_{3i}+w_{3i+1}$ 并且 $w_{3i} < w_{3i+2}, w_{3i+1} < w_{3i+2}$.在项集 $i(0 \leq i \leq n-1)$ 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 $p_j, w_j(0 \leq j \leq 3n-1)$ 和 C 均为正整数,且 $w_{3i+2} \leq C(0 \leq i \leq n-1), \sum_{i=0}^{n-1} w_{3i+2} > C$.则 D{0-1}KP 的第一数学模型^[26]为

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

$$\text{s.t. } x_{3i}+x_{3i+1}+x_{3i+2} \leq 1, i=0, 1, \dots, n-1 \quad (12)$$

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

$$x_{3i}, x_{3i+1}, x_{3i+2} \in \{0, 1\}, i=0, 1, \dots, n-1 \quad (14)$$

其中,变量 $x_j(0 \leq j \leq 3n-1)$ 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)和约束条件(13)时为一个可行解.

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

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

$$\text{s.t. } \sum_{i=0}^{n-1} \lceil x_i / 3 \rceil w_{3i+\lfloor x_i \rfloor} \leq C \quad (16)$$

$$x_i \in \{0, 1, 2, 3\}, i=0, 1, \dots, n-1 \quad (17)$$

其中,

- $\lceil x \rceil$ 为顶函数;
- 变量 $x_i(0 \leq i \leq n-1)$ 表示项集 i 中是否存在项被装入了背包中: $x_i=0$,表示项集 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种有效算法 FirEGA 和

SecEGA.为此,下面利用 DisPSO 求解文献[27]中的 4 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 $c_1=c_2=0.5$.FirEGA 和 SecEGA 的其他参数设置请参考文献[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	85 740	DisPSO	85 558	85262.3	84 958	178.38	0.142
		FirEGA	80 650	79103.2	77 935	690.01	0.152
		SecEGA	78 287	76807.2	75 156	798.95	0.133
UDKP2	163 744	DisPSO	161 901	161 280	160 789	241.34	0.642
		FirEGA	155 039	151 662	149 875	1044.95	0.596
		SecEGA	148 043	145 548	143 833	883.43	0.543
UDKP3	269 393	DisPSO	267 142	266 423	266 679	172.71	1.327
		FirEGA	246 698	240 886	237 980	1491.97	1.381
		SecEGA	228 823	225 492	222 486	1353.58	1.161
UDKP4	347 599	DisPSO	343 990	343 535	343 058	196.82	2.518
		FirEGA	321 605	317 319	314 486	1426.85	2.436
		SecEGA	305 796	299 978	297 606	1435.46	2.173
UDKP5	442 644	DisPSO	436 132	435 409	434 796	266.93	3.894
		FirEGA	405 409	399 620	395 367	1692.23	3.827
		SecEGA	376 147	370 808	367 574	1611.71	3.360
UDKP6	536 578	DisPSO	530 009	529 308	528 743	247.34	5.579
		FirEGA	486 556	478 726	474 015	2233.61	5.517
		SecEGA	447 438	442 499	438 809	1765.28	4.811
UDKP7	635 860	DisPSO	629 658	629 020	628 539	251.35	7.828
		FirEGA	568 119	560 948	556 938	2441.80	7.691
		SecEGA	529 753	521 401	518 407	1813.04	6.522
UDKP8	650 206	DisPSO	640 262	639 534	638 519	285.05	11.844
		FirEGA	590 137	585 286	580 684	2078.87	10.090
		SecEGA	550 645	546 678	543 836	1449.36	9.109
UDKP9	718 532	DisPSO	709 112	708 396	707 786	257.29	15.653
		FirEGA	655 172	649 636	645 012	2023.64	13.130
		SecEGA	613 581	602 215	605 835	2003.75	11.374
UDKP10	779 460	DisPSO	764 299	763 627	762 612	335.37	19.192
		FirEGA	712 270	706 575	701 545	2013.43	15.891
		SecEGA	665 459	658 908	655 645	1723.80	14.773

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	83 098	DisPSO	82 976	82 918.3	82 825	32.25	0.181
		FirEGA	82 803	82 693.2	82 592	52.04	0.157
		SecEGA	80 014	79 021.8	78 096	473.67	0.140
WDKP2	138 215	DisPSO	137 912	137 836	137 747	27.48	0.746
		FirEGA	137 704	137 584	137 356	63.23	0.653
		SecEGA	133 315	132 276	131 337	415.62	0.562
WDKP3	256 616	DisPSO	256 097	255 902	255 774	65.78	1.469
		FirEGA	254 240	253 657	253 307	173.01	1.366
		SecEGA	238 331	235 721	234 025	873.58	1.242
WDKP4	315 657	DisPSO	315 117	314 958	314 717	76.13	2.674
		FirEGA	313 966	312 849	311 998	484.76	2.381
		SecEGA	293 640	290 851	288 764	950.06	2.299
WDKP5	428 490	DisPSO	427 802	427 643	427 200	91.80	4.131
		FirEGA	426 783	424 548	423 058	798.53	3.815
		SecEGA	393 617	390 014	387 992	1059.83	3.648

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

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

Instance	OPT	Algorithm	Best	Mean	Worst	StD	Time
WDKP6	466 050	DisPSO	464 964	464 755	464 228	142.00	6.595
		FirEGA	463 185	461 672	457 718	1 107.57	5.521
		SecEGA	429 208	425 112	423 269	1 058.37	5.405
WDKP7	547 683	DisPSO	546 531	546 269	545 886	145.74	8.350
		FirEGA	544 019	541 949	538 126	1 224.68	7.736
		SecEGA	501 557	496 134	493 845	1 230.94	6.775
WDKP8	576 959	DisPSO	575 543	575 200	574 749	155.38	11.433
		FirEGA	573 427	571 559	563 253	1 495.36	10.041
		SecEGA	530 971	523 203	520 350	2 157.09	9.805
WDKP9	650 660	DisPSO	648 987	648 565	648 085	170.38	16.989
		FirEGA	647 477	644 820	630 086	2 056.06	13.318
		SecEGA	598 343	586 770	583 854	2 315.50	11.907
WDKP10	678 967	DisPSO	677 398	677 092	675 987	232.80	19.068
		FirEGA	675 452	673 008	668 239	1 441.96	16.210
		SecEGA	620 230	606 215	609 964	3 090.86	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	94 459	DisPSO	94 286	94 235.8	94 121	39.32	0.136
		FirEGA	93 276	93 170.8	93 070	42.15	0.135
		SecEGA	89 769	88 831.5	87 463	594.91	0.129
SDKP2	160 805	DisPSO	159 837	159 710	159 376	66.13	0.627
		FirEGA	159 159	159 004	158 859	96.53	0.617
		SecEGA	153 821	152 059	150 753	489.39	0.575
SDKP3	238 248	DisPSO	236 402	236 137	235 767	101.19	1.417
		FirEGA	235 454	235 241	235 043	86.52	1.412
		SecEGA	224 997	223 580	221 918	543.38	1.224
SDKP4	340 027	DisPSO	336 890	336 554	336 083	163.91	2.561
		FirEGA	336 440	335 963	335 709	122.41	2.439
		SecEGA	318 510	315 513	313 747	851.14	2.288
SDKP5	463 033	DisPSO	460 167	459 736	459 241	172.32	3.761
		FirEGA	452 900	447 587	444 255	1 974.99	3.579
		SecEGA	421 108	416 964	413 933	1 291.65	3.299
SDKP6	466 097	DisPSO	460 857	460 415	459 619	214.88	6.026
		FirEGA	459 443	458 893	458 584	162.94	5.292
		SecEGA	430 738	427 304	425 504	1 031.12	4.923
SDKP7	620 446	DisPSO	615 877	615 223	614 214	281.73	7.989
		FirEGA	599 361	592 279	579 673	3 949.03	7.718
		SecEGA	561 224	556 083	552 007	1 926.26	6.368
SDKP8	670 697	DisPSO	664 634	664 083	663 413	263.27	10.502
		FirEGA	661 563	660 104	659 928	426.06	10.242
		SecEGA	611 644	606 263	603 774	1 446.94	9.770
SDKP9	739 121	DisPSO	731 491	730 837	729 863	360.97	13.646
		FirEGA	729 135	727 544	727 064	343.67	12.131
		SecEGA	674 885	667 900	664 580	1 614.04	11.086
SDKP10	765 317	DisPSO	755 954	755 228	754 456	319.49	17.134
		FirEGA	756 205	753 394	750 757	985.46	15.572
		SecEGA	708 935	695 557	691 994	2 956.08	14.360

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	70 106	DisPSO	70 106	70 083.4	70 037	19.45	0.153
		FirEGA	70 106	70 074.4	70 022	23.23	0.137
		SecEGA	68 663	67 999.8	67 369	328.44	0.144

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

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

Instance	OPT	Algorithm	Best	Mean	Worst	StD	Time
IDKP2	118 268	DisPSO	118 268	118 240	118 232	13.41	0.705
		FirEGA	118 169	117 869	117 625	102.60	0.533
		SecEGA	114 434	113 385	112 307	446.67	0.639
IDKP3	234 804	DisPSO	234 784	234 740	234 459	40.09	1.504
		FirEGA	234 607	233 997	233 666	175.42	1.224
		SecEGA	220 096	217 982	216 313	835.83	1.289
IDKP4	282 591	DisPSO	282 591	282 554	282 320	40.23	2.784
		FirEGA	282 148	280 695	278 881	827.63	2.286
		SecEGA	263 238	260 425	258 922	933.40	2.359
IDKP5	335 584	DisPSO	335 584	335 529	335 204	71.36	4.489
		FirEGA	335 004	333 484	329 621	1 173.90	4.338
		SecEGA	309 573	306 878	304 881	907.19	3.868
IDKP6	452 463	DisPSO	452 433	452 298	451 773	122.48	6.107
		FirEGA	451 799	449 863	446 704	1 161.52	4.979
		SecEGA	414 090	411 367	408 788	1 099.31	5.391
IDKP7	489 149	DisPSO	489 137	489 011	488 543	98.42	8.645
		FirEGA	488 009	485 592	476 385	2 294.28	7.162
		SecEGA	451 528	444 316	442 133	1 280.31	7.576
IDKP8	533 841	DisPSO	533 836	533 703	533 357	103.21	11.983
		FirEGA	533 035	529 984	514 196	2 308.11	9.014
		SecEGA	490 494	481 831	478 035	2 215.66	9.571
IDKP9	528 144	DisPSO	528 133	527 962	527 598	125.30	14.716
		FirEGA	526 410	523 982	511 651	2 216.13	11.776
		SecEGA	489 661	477 001	471 848	3 656.22	12.907
IDKP10	581 244	DisPSO	581 237	581 032	580 583	136.14	18.423
		FirEGA	579 220	576 772	568 903	1 905.18	13.573
		SecEGA	535 541	521 604	516 445	4 265.07	15.026

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.

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.

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(DisEA)的方法——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 ψ_{DisEA} 的基础上,提出了基于 ETM 设计 DisEA 的一般算法框架.随后,利用函数 ψ_{DisEA} ,基于 ETM 给出了一个离散粒子群优化算法 DisPSO,并分别利用 BPSO, HBDE 和 DisPSO 求解 SUKP 问题和 $D\{0-1\}$ KP 问题.通过与 GA 及其改进算法的计算结果相比较,验证了这些算法在求解组合优化问题方面的优越性能.这说明对已有 EA(例如 PSO, DE 和 ABC 等),基于 ETM 方法(使用函数 ψ_{DisEA})构造相应 DisEA 是一种可行且有效的方法.

当前,存在许多组合优化问题的可行解表示为 $\{0, 1, \dots, n-1\}^d (n \geq 4)$ 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 ψ_{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,是值得进一步研究的问题.

References:

- [1] De Jong KA. Evolutionary Computation—A Unified Approach. Cambridge: MIT Press, 2016.
- [2] Mitchell M. An Introduction to Genetic Algorithms. Cambridge: MIT Press, 1996.
- [3] Kennedy J, Eberhart RC. Particle swarm optimization. In: Proc. of the IEEE Int'l Conf. on Neural Networks. Perth: IEEE Service Center, Piscataway, 1995. 1942–1948.
- [4] Storn R, Price K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization. 1997, 11: 341–359.
- [5] Dorigo M, Stützle T. Ant Colony Optimization. Cambridge: MIT Press, 2004.
- [6] Li XL. A new intelligent optimization method—Artificial fish swarm algorithm [Ph.D. Thesis]. Hangzhou: Zhejiang University, 2003.
- [7] Yao X, Chen GL, Xu XH, Liu Y. A survey of evolutionary algorithms. Chinese Journal of Computers, 1995, 18(9): 694–706 (in Chinese with English abstract).
- [8] Wang L. Intelligent Optimization Algorithms with Applications. Beijing: Tsinghua University Press, 2001 (in Chinese).
- [9] Gong MG, Jiao LC, Yang DD, Ma WP. Research on evolutionary multi-objective optimization algorithms. Ruan Jian Xue Bao/ Journal of Software, 2009, 20(2): 271–289 (in Chinese with English abstract). <http://www.jos.org.cn/1000-9825/3483.htm>

- [10] Coello CAC. An introduction to evolutionary algorithms and their applications. In: Proc. of the Advanced Distributed Systems. Berlin, Heidelberg: Springer-Verlag, 2005. 425–442.
- [11] Karaboga D, Basturk B. 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 XS. Firefly algorithms for multimodal optimization. In: Proc. of the Stochastic Algorithms: Foundations and Applications (SAGA 2009). LNCS 5792, 2009. 169–178.
- [13] Tan Y, Zhu Y. Fireworks algorithm for optimization. In: Tan Y, Shi Y, Tan KC, eds. Proc. of the ICSI 2010. LNCS 6145, Heidelberg: Springer-Verlag, 2010. 355–364.
- [14] Pan WT. A new fruit fly optimization algorithm: Taking the financial distress model as an example. *Knowledge-Based Systems*, 2012,26(2):69–74.
- [15] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. *Advances in Engineering Software*, 2014,69(1):46–61.
- [16] Uymaz SA, Tezel G, Yel E. Artificial algae algorithm (AAA) for nonlinear global optimization. *Applied Soft Computing*, 2015, 31(C):153–171.
- [17] Duan HB, Ye F. Progresses in pigeon-inspired optimization algorithms. *Journal of Beijing University of Technology (Natural Sciences Edition)*, 2017,43(1):1–7 (in Chinese with English abstract).
- [18] Tasgetiren MF, Pan QK, Suganthan PN, Chen AHL. A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. *Information Sciences*, 2011,181(16):3459–3475.
- [19] Jia DL, Duan XT, Khan MK. Binary artificial bee colony optimization using bitwise operation. *Computers & Industrial Engineering*, 2014,76(C):360–365.
- [20] Kiran MS. The continuous artificial bee colony algorithm for binary optimization. *Applied Soft Computing*, 2015,33(C):15–23.
- [21] He YC, Wang XZ, Kou YZ. A binary differential evolution algorithm with hybrid encoding. *Journal of Computer Research and Development*, 2007,44(9):1476–1484 (in Chinese with English abstract).
- [22] Yu Y, Yao X, Zhou ZH. On the approximation ability of evolutionary optimization with application to minimum set cover. *Artificial Intelligence*, 2012,(2):20–33.
- [23] Kennedy J, Eberhart RC. A discrete binary version of the particle swarm optimization. In: Proc. of the 1997 Conf. on System, Man, and Cybernetics. 1997. 4104–4109.
- [24] Arulselvan A. A note on the set union knapsack problem. *Discrete Applied Mathematics*, 2014,169(41):214–218.
- [25] He Y, Xie H, Wong T, Wang X. A novel binary artificial bee colony algorithm for the set-union knapsack problem. In: Proc. of the Future Generation Computer Systems. 2017. <http://dx.doi.org/10.1016/j.future.2017.05.044>
- [26] Rong AY, Figueira JR, Klamroth K. Dynamic programming based algorithms for the discounted {0-1} knapsack problem. *Applied Mathematics and Computation*, 2012,218(12):6921–6933.
- [27] He YC, Wang XZ, Li WB, Zhang XL, Chen YY. Research on genetic algorithms for the discounted {0-1} knapsack problem. *Chinese Journal of Computers*, 2016,39(12):2614–2630 (in Chinese with English abstract).
- [28] Chen GL, Wang XF, Zhuang ZD, Wang DS. *Genetic Algorithms and Its Applications*. Beijing: Science Press, 2003 (in Chinese).
- [29] Deng L, Zhao J, Wang X. Genetic algorithm solution of network coding optimization. *Ruan Jian Xue Bao/Journal of Software*, 2009,20(8):2269–2279 (in Chinese with English abstract). <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 with English abstract). <http://www.jos.org.cn/1000-9825/4970.htm> [doi: 10.13328/j.cnki.jos.004970]
- [31] Zeng JC, Jie J, Cui ZH. *Particle Swarm Optimization Algorithm*. Beijing: Science Press, 2004 (in Chinese).
- [32] Price KV, Storn RM, Lampinen JA. *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, 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 with English abstract). <http://www.jos.org.cn/1000-9825/4937.htm> [doi: 10.13328/j.cnki.jos.004937]
- [34] Kilani Y. Comparing the performance of the genetic and local search algorithms for solving the satisfiability problems. *Applied Soft Computing*, 2010,10(1):198–207.

- [35] Kellerer H, Pferschy U, Pisinger D. Knapsack Problems. Berlin: Springer-Verlag, 2004.
- [36] Chen Y, Hao JK. Memetic search for the generalized quadratic multiple knapsack problem. IEEE Trans. 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,1995,18:694-706.
- [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. <http://www.jos.org.cn/1000-9825/3483.htm>
- [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,陈璇.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,王熙照,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. <http://www.jos.org.cn/1000-9825/3370.htm>
- [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. <http://www.jos.org.cn/1000-9825/4970.htm> [doi: 10.13328/j.cnki.jos.004970]
- [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. <http://www.jos.org.cn/1000-9825/4937.htm> [doi: 10.13328/j.cnki.jos.004937]



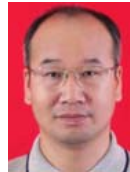
贺毅朝(1969—),男,河北晋州人,教授,CCF高级会员,主要研究领域为演化计算,组合优化算法,近似算法,群测试理论.



赵书良(1967—),男,博士,教授,博士生导师,CCF专业会员,主要研究领域为数据挖掘,大数据处理.



王熙照(1963—),男,博士,教授,博士生导师,主要研究领域为机器学习,进化计算,大数据分析.



张新禄(1968—),男,副教授,主要研究领域为智能计算,图论与组合优化,群测试理论.