河北地質大學
Hebei GEO University

背包问题及其演化算法求解

贺 毅 朝

heyichao@hgu.edu.cn

CONTENTS

01 背包问题(KP)介绍

02 利用EAs求解KP问题

03 在研KP问题

背包问题(KP)介绍

■背包问题(Knapsack Problem，KP)是NP完全问题，也是一类重要的组合优化问题，在工业、经济、通信、金融与计算机等领域的资源分配、资金预算、投资决策、装载问题、整数规划、分布式系统与密码系统中具有重要的理论和应用价值.

■KP家族成员众多，除了最经典的0-1 背包问题(0-1KP)之外，还有许多0-1KP的经典扩展形式：如有界背包问题(BKP)、无界背包问题(UKP)、多维背包问题(MdKP)、多背包问题(MKP)、多选择背包问题(MCKP)、二次背包问题(QKP)、最大最小背包问题(MmKP)、集合联盟背包问题(SUKP)、多目标背包问题(MOKP)和在线背包问题(OLKP)等。

■近年来，新的KP问题不断涌现，例如动态背包问题(dynamic knapsack problem, DKP)、多重二次背包问题(quadratic multiple knapsack problem, QMKP)、多选择多维背包问题(multiple-choice multidimensional knapsack problem, MMKP)、折扣0-1背包问题(discounted {0-1} knapsack problems, D{0-1}KP)、具有单连续变量的背包问题(Knapsack Problem with a single Continuous variable, KPC)、具有惩罚的背包问题 (penalized knapsack problem ，PKP)和分离约束背包问题(Disjunctively Constrained Knapsack Problem, DCKP)等，新KP的约束条件呈现多和难的趋势。

0-1背包问题(0-1KP)

物品序号	1	2	3	…	…	n
价值	p_1	p_2	p_3			p_n
重量	w_1	w_2	w_3			w_n

背包载重为C. 为了避免平凡实例，一般均假设$w_j \leq C$且$\sum_{j=1}^{n} w_j > C$

0-1KP的数学模型为：

$$\max \ f(Y) = \sum_{j=1}^{n} p_j y_j$$

$$\text{s.t.} \ \sum_{j=1}^{n} w_j y_j \leqslant C$$

$Y=[y_1,y_2,\ldots,y_n] \in \{0,1\}^n$, 当物品$j$ (项j) 被装入背包时$y_j=1$,否则 $y_j=0$. 任意一个0-1向量是0-1 KP的一个潜在解, 只有满足约束条件时才是一个可行解, 否则称为不可行解.

有界背包问题(bounded knapsack problem,BKP)

物品序号	1	2	3	…	…	n
价值	p_1	p_2	p_3			p_n
重量	w_1	w_2	w_3			w_n
物品个数	b_1	b_2	b_3			b_n

背包载重为C. BKP的数学模型为：

$$\max\ f(Y)=\sum_{j=1}^{n}p_j y_j$$

$$\text{s.t.}\ \sum_{j=1}^{n}w_j y_j \leqslant C$$

$Y=[y_1,y_2,\ldots,y_n]$, $y_j\in\{0,1,\ldots,b_j\}$,$1\leq j\leq n$; 当$y_j\neq 0$时表示有y_j个物品j被装入背包中,否则 $y_j=0$. 任意一个整型向量是BKP的一个潜在解, 只有满足约束条件时才是一个可行解, 否则称为不可行解. 0-1KP是BKP在$b_j=1$ $(1\leq j\leq n)$时的特例, BKP的求解难度比0-1KP大.

多维背包问题(multidimensional knapsack problem, MdKP)

物品序号	1	2	3	…	…	n
价值	p_1	p_2	p_3			p_n
重量	w_{11}	w_{21}	w_{31}			w_{n1}
	w_{12}	w_{22}	w_{32}			w_{n2}
	…	…	…			…
	w_{1d}	w_{2d}	w_{3d}			w_{nd}

背包有d个重量约束$C_1, C_2, \ldots, C_d$. w_{ji}和C_i满足 $w_{ji} \leqslant C_i$,

$\sum_{j=1}^{n} w_{ji} > C_i$, $j=1,2,\ldots,n$, $i=1,2,\ldots,d$.

MdKP的数学模型为：

$$\max\ f(Y)=\sum_{j=1}^{n}p_j y_j$$

$$\text{s.t.}\ \sum_{j=1}^{n}w_{ji}y_j\leqslant C_i,\qquad i=1,2,...,d$$

$y_j\in\{0,1\}$, j=1,2,…,n, 其中p_j是项j的价值, w_{ji} (i=1,2,…,d)是项j的d个重量限制; y_j=1当且仅当项j被装入背包. 显然, 0-1 KP是MdKP在d=1时的特例, MdKP的求解难度比0-1KP大.

MdKP举例:

物品序号	1	2	3	4	5	6	7	
价值	$p_1=5$	$p_2=3$	$p_3=2$	$p_4=6$	$p_5=4$	$p_6=7$	$p_7=9$	
重量	$w_{11}=3$	$w_{21}=2$	$w_{31}=6$	$w_{41}=6$	$w_{51}=5$	$w_{61}=1$	$w_{71}=4$	$C_1=19$
	$w_{12}=2$	$w_{22}=5$	$w_{32}=1$	$w_{42}=7$	$w_{52}=11$	$w_{62}=5$	$w_{72}=3$	$C_2=22$
	$w_{13}=1$	$w_{23}=7$	$w_{33}=5$	$w_{43}=10$	$w_{53}=6$	$w_{63}=9$	$w_{73}=4$	$C_3=30$
	$w_{14}=7$	$w_{24}=1$	$w_{34}=3$	$w_{44}=2$	$w_{54}=3$	$w_{64}=4$	$w_{74}=5$	$C_4=15$

Y_1=[1, 0, 0, 1, 1, 0, 1]是一个不可行解； Y_2=[0, 1, 1, 0, 0, 1, 1]是一个可行解, 价值和为21;
Y_3=[1, 1, 1, 0, 0,1, 0]是一个可行解, 价值和为17.

二次背包问题(quadratic knapsack problem, QKP)

物品序号	1	2	3	…	…	n
价值	p_1	p_2	p_3	…	…	p_n
重量	w_1	w_2	w_3	…	…	w_n

物品序号	1	2	3	4	…	n
联合价值	0	p_{12}	p_{13}	p_{14}	…	p_{1n}
		0	p_{23}	p_{24}	…	p_{2n}
			0	p_{34}		p_{3n}
				…	…	…
						0

背包的载重为C.

QKP的数学模型如下:

$$\max\ f(Y)=\sum_{j=1}^{n}p_j y_j+\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(p_{ij}y_i y_j)$$

$$\text{s.t.}\ \sum_{j=1}^{n}w_j y_j\leqslant C$$

其中, $y_j\in\{0,1\}$, $j=1,2,\ldots,n$; $y_j=1$当且仅当项j被装入了背包中. p_i为项i自身的价值, p_{ij}为两个不同项i和$j(1\leq i\neq j\leq n)$被同时装入背包时所产生的联合价值. 0-1KP是QKP当所有$p_{ij}\equiv 0$时的特例.

集合联盟背包问题(Set-Union Knapsack Problem, SUKP)

给定一个元素集U={1,2,..., n}和一个项集S={1,2,...,m}, 每一个项$i \in S$ (i = 1,2,...,m)具有一个价值$p_i > 0$并对应一个元素的子集$U_i \subseteq U$. 每一个元素$j \in U$ (j =1,2,..., n)具有一个重量 $w_i > 0$, 背包的载重为C. 对于任意非空子集 $A \subseteq S$ ，定义A的价值和重量分别为 $P(A) = \sum_{i \in A} p_i$ 和 $W(A) = \sum_{j \in \bigcup_{i \in A} U_i} w_j$. SUKP的目地是求 $S^* \subseteq S$ 使得在满足$W(S^*) \leq C$的前提下$P(S^*)$最大.

SUKP的数学模型为：

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

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

其中，$Y=[y_1,y_2,\ldots,y_m]\in\{0,1\}^m$，$y_i=1$当且仅当项$i$被装入了背包中．$A_Y=\{i|y_i\in Y\wedge y_i=1,1\le i\le m\}$是由向量$Y$确定的$S$的一个子集。

SUKP举例：

元素集U={1,2,3,4,5,6,7,8}，w1=4，w2=3，w3=7，w4=11，w5=9，w6=5，w7=1，w8=15;
项集S={1,2,3,4,5,6}，p1=6，p2=7，p3=2，p4=11，p5=8，p6=3；且
$U1$={1,3,5,7}，$U2$={1,2,7,8}，$U3$={2,4,6,8}，$U4$={2,3,4}，$U5$={2,5,8}，$U6$={1,8}.
C= 41.

	u1	u2	u3	u4	u5	u6	u7	u8
s1	1	0	1	0	1	0	1	0
s2	1	1	0	0	0	0	1	1
s3	0	1	0	1	0	1	0	1
s4	0	1	1	1	0	0	0	0
s5	0	1	0	0	1	0	0	1
s6	1	0	0	0	0	0	0	1

Y_1=[1, 1, 0, 1, 1, 0]为不可行解，Y_2=[1, 1, 0, 0, 1, 0]为可行解.

折扣0-1背包问题(discounted {0-1} knapsack problems, D{0-1}KP)

D{0-1}KP(2007年)是将“折扣”思想引入KP而提出的一个新KP问题，可用于商贸经营、项目筛选和预算控制等领域. D{0-1}KP的一般描述为: 给定n个均含有3个项的项集,项集$i(0\leq i\leq n\text{-}1)$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$w_{3i+2}<w_{3i}+w_{3i+1}$并且$w_{3i}<w_{3i+2}$,$w_{3i+1}<w_{3i+2}$.在项集$i(0\leq i\leq n\text{-}1)$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.如何选择各项装入背包,使得它们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?

项集	0	1	2	…	…	n-1
项	0，1，2	3，4，5	6，7，8			3n-3,3n-2,3n-1
价值	p_0,p_1,p_2	p_3,p_4,p_5	p_6,p_7,p_8			$p_{3n-3},p_{3n-2},p_{3n-1}$
重量	w_0,w_1,w_2	w_3,w_4,w_5	w_6,w_7,w_8			$w_{3n-3},w_{3n-2},w_{3n-1}$

在项集$i(0\le i\le n-1)$中的3个项满足$p_{3i+2}=p_{3i}+p_{3i+1}$, $w_{3i+2}<w_{3i}+w_{3i+1}$且$w_{3i}<w_{3i+2}$, $w_{3i+1}<w_{3i+2}$. 对于每个项集$i(0\le i\le n-1)$，项$3i$, $3i+1$, $3i+2$中最多只能有一个项被装入背包中.

背包的载重为C.

D{0-1}KP的数学模型如下:

$$\max\ f(Y)=\sum_{i=0}^{n-1}(y_{3i}p_{3i}+y_{3i+1}p_{3i+1}+y_{3i+2}p_{3i+2})$$

$$\text{s.t. } y_{3i}+y_{3i+1}+y_{3i+2}\leq 1,\ i=0,1,\ldots,n\text{-}1$$

$$\sum_{i=0}^{n-1}(y_{3i}w_{3i}+y_{3i+1}w_{3i+1}+y_{3i+2}w_{3i+2})\leqslant C$$

$$y_{3i},y_{3i+1},y_{3i+2}\in\{0,1\},\quad i=0,1,\ldots,n-1$$

其中, $Y=[y_0,y_1,\ldots,y_{3n-1}]\in\{0,1\}^{3n}$, $y_i=1$表示项i被装入了背包中, $y_i=0$表示项i未被装入背包中, $0\leq i\leq 3n\text{-}1$.

D{0-1}KP的另一数学模型如下：

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

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

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

$X=[x_0,x_1,\ldots,x_{n-1}]\in\{0,1,2,3\}^n$为一个$n$维整型向量，其中$x_i$ $(0\le i\le n\text{-}1)$表示项集i中是否存在项被装入了背包中，即$x_i=0$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.

具有单连续变量的背包问题(Knapsack Problem with a single Continuous variable, KPC)

给定n个物品集合$N=\{1,2,\ldots,n\}$和一个基本载重为C的背包，其中物品$j\in N$具有价值p_j和重量w_j，背包的可变载重S是在区间$[l,u]$上连续变化的一个变量，p_j，w_j和C为正有理数，S，l和u为有理数，且$l<0<u$. 给定一个正常数c作为惩罚系数，确定在S取何值以及此时如何选择物品装入背包，使得装入物品的重量之和在不超过背包载重$C+S$的前提下价值之和减去cS最大.

KPC的数学模型为：

$$\max \quad f(X,S)=\sum_{j=1}^{n} x_j p_j - cS$$

$$s.t. \quad \sum_{j=1}^{n} x_j w_j \le C+S$$

$$x_j \in \{0,1\},\ j=1,2,\ldots,n,\ S\in[l,u]$$

其中，$X=[x_1,x_2,\ldots,x_n]\in\{0,1\}^n$，$x_j=1(j=1,2,\ldots,n)$当且仅当物品$j$被装入了背包. 在KPC中背包载重不再固定不变，而是由变量S进行连续调整，当$S>0$时背包载重增加，当$S<0$时背包载重减少；同时，目标函数也不只是装入背包物品的价值之和，而是要加上了一个关于S的增量$(-cS)$。由于0-1KP是KPC在$S=0$时的特例，KPC也是一个NP完全问题，不存在多项式时间精确算法.

02
PART

利用EAs求解KP问题

演化算法简介

■演化算法(Evolutionary Algorithms, EAs)也称进化算法或群智能算法，是一类模拟自然界遗传进化规律和某些社会现象或物理现象等的仿生算法, 本质上是一类基于迭代搜索的随机近似算法. EAs具有自组织、自适应、自学习的特性，它既不需要计算目标函数的导数和梯度,也不要求目标函数具有连续性,且具有内在的隐含并行性和全局寻优能力，因此不受问题性质的限制，能够处理传统优化算法难以解决的复杂问题。

■经典的EAs有: 遗传算法(genetic algorithm, GA)、粒子群优化(particle swarm optimization, PSO)、蚁群优化(ant colony optimization, ACO)、差分演化(differential evolution, DE)、人工鱼群算法(artificial fish swarm, AFS)、人工蜂群算法(artificial bee colony, ABC)、和声搜索算法(harmony search algorithm, HSA)、混合蛙跳算法(shuffled frog leaping algorithm, SFLA)和人工免疫算法(artificial immune system, AIS)等.

■近年来，新的EAs不断被提出，例如 人工海藻算法(Artificial algae algorithm, AAA)、果蝇优化算法(Fruit fly optimization, FFO)、头脑风暴算法 (Brain storm optimization, BSO)、烟花算法(Fireworks algorithm，FWA)、灰狼优化算法(Grey wolf optimiser, GWO)、正弦余弦算法(Sine cosine algorithm，SCA)、共生生物搜索算法(Symbiotic organisms search, SOS)、基于教与学的优化算法(Teaching-learning-based optimization，TLBO)等.

演化算法一般框架

1. 随机产生初始种群**P**(0)={$X_i(0) \in \Omega \mid 1 \le i \le N$}
2. 计算$X_i(0)$的适应度， 对个体进行评价，并确定最优(次优或最差)个体
3. For *t*←1 to *MIT* Do
4. 演化算子1~*M*：利用**P**(*t*-1)中的某些个体生成新的个体
5. 计算新产生个体的适应度，对个体进行评价，确定新最优(次优或最差)个体
6. 演化算子*M*+1(选择算子)：利用某种策略从**P**(*t*-1)和新生成个体 中选择构成**P**(*t*)
7. End For
8. 输出获得的最好解，结束

利用EAs求解KP问题的方法

- 连续型EAs的离散化：方法一是改变算法原有的进化算子, 在原有EAs的框架下重新定义进化算子，使之满足离散域上的计算要求；方法二是保持原有算法不变, 利用映射将个体对应的实向量转换为一个整型 向量，实现对离散问题的求解。

- 不可行解处理与适应度计算：由于KP为约束优化问题，除非特殊的处理方法，否则EAs求解时会产生不可行解；处理不可行解的常见方法是**罚函数法**和**修复优化法**，其中修复优化法最常用。在保证每一个体均对应一个可行解的基础上，利用可行解的目标函数值作为个体的适应度。

利用连续型演化算法求解KP问题 //基于编码转换法

1. 随机产生初始种群**P**(0)={$X_i(0) \in \Omega \mid 1 \le i \le N$}, 确定$X_i(0)$ 对应的潜在解$Y_i(0)$
2. 将$Y_i(0)$修复与优化为KP的一个可行解， 并计算$f(Y_i(0))$
3. 对个体$X_i(0)$进行评价，确定最优(次优或最差)个体
4. For t←1 to MIT Do
5. 演化算子1~M：利用**P**(t-1)中的某些个体生成新的个体
6. 确定新个体对应的潜在解，并将其修复与优化为一个可行解
7. 计算可行解的目标函数值，对个体进行评价，确定新最优(次优或最差)个体
8. 演化算子M+1(选择算子)：利用某种策略从**P**(t-1)和新生成个体中选择构成**P**(t)
9. End For
10. 输出获得的最好解，结束

03
PART

在研KP问题

■MKP问题： 主要是比较GA, DisPSO, FDDE, DisABC, GTOA,DGWO,DWOA和DSCA等求解MKP的 性能优劣，确定最适于求解MKP的方法。

■KPC问题： 主要是探讨并行EAs求解KPC的效率问题。

■DCKP问题： 主要是建模、不可行解的处理方法、寻找最适于求解DCKP的EAs。

THANKS