改进帕累托算法求解超大规模多选择背包问题

杨 洋
(西华师范大学公共数学学院,四川南充 637009)

摘 要: 实际生产生活中大量多选一的问题都可以转为多选择背包问题(MCKP),但 MCKP 是一个经典的 NP 难问题,因此对于超大规模 MCKP 而言,往往只能利用粒子群算法、狼群算法、鱼群算法等群智能算法对问题进行求解. 对于群智能算法而言,高效快捷的贪心算法对于初始解的生成起着至关重要的作用. 基于凸帕累托算法(CPA),提出一种能够快速求解线性支配子集的改进帕累托算法(IPA). IPA 首先选择各类项集的质量最小项,然后计算所有物品的价值密度,最后按照价值密度从高到低选择对物品进行贪心选择,若贪心选择项的价值大于其所在项集原有选择项,则进行迭代. 仿真实验结果表明:IPA 相比于 CPA,求解速度平均提升 98.86%. 且 PSO-IPA 求解精度平均提升 28.92%.

关键词: 多选择背包问题; 贪心算法; 大数据; 帕累托前沿; 凸优化; 群智能算法; 整数优化
中图分类号: TP18; O224 **文献标识码:** A **文章编号:** 0372-2112 (2020)06-1205-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2020.06.023

Improved Pareto Algorithm for Solving Very Large Scale Multiple-Choice Knapsack Problem

YANG Yang
(*College of Mathematics Education, China West Normal University, Nanchong, Sichuan* 637009, *China*)

Abstract: In the actual production life conditions, a large number of multiple choices can be converted into a multiple-choice knapsack problem (MCKP), but MCKP is a classic NP-hard problem. Therefore, for very large scale MCKP, it is often only possible to use the particle swarm algorithm, wolf pack algorithm, fish swarm algorithm and so on to solve the problem. For swarm intelligence algorithms, efficient and fast greedy algorithms play a key role in the generation of initial solutions. Based on the convex Pareto algorithm (CPA), an improved Pareto algorithm (IPA) that can quickly get the linear programming dominated set is proposed. IPA firstly selects the minimum weight item of each set, then computes the value density of all items, and finally chooses the greedy choice of the item according to the value density from high to low. When the value of the greedy option is greater than the original selection of the item set, then IPA is iterated. The simulation results show that compared with CPA, the speed of IPA is increased by 98.86%. The PSO-IPA solution accuracy is increased by an average of 28.92%.

Key words: multiple-choice knapsack problem (MCKP); greedy algorithm; big data; Pareto front; convex optimization; swarm intelligence algorithm; integer optimization

1 引言

多选择背包问题(Multiple Choice Knapsack Problem, MCKP)作为背包问题(Knapsack Problem, KP)的一种特殊形式,在资本预算[1],项目规划[2],销售资源分配[3]等方面应用广泛. 此外,MCKP 还可以利用线性化方法[1]对非线性背包问题(Nonlinear Knapsack Problem, NLKP)进行求解[1]. 因此 MCKP 的求解算法对于诸如折扣{0-1}背包问题(Discounted {0-1} Knapsack Problem, D{0-1}KP)[4~6]、集合联盟背包问题(Set Union Knap-

收稿日期:2019-08-20;修回日期:2019-11-04;责任编辑:覃怀银
基金项目:国家自然科学基金(No. 11871059);四川省教育厅自然科学基金(No. 18ZA0469);西华师范大学校级科研团队(No. CXTD2015-4);西华师范大学英才基金(No. 17YC385);西华师范大学青年教师科研基金专项(No. 19D035)

sack Problem,SUKP)[7,8]等非线性多选择约束类问题也可以进行有效求解.

因 KP 是 NP-hard 的,自然 MCKP 也是 NP-hard 的[4],但可以在伪多项式时间内利用动态规划算法[9,10]对问题进行求解.

就求解算法而言,主要分成两个大方向.一方面,较多文献采用经典算法对问题进行求解,如:通过求解线性多选择背包问题(Linear Multiple Choice Knapsack Problem,LMCKP)[1,3,11]得到问题的上界(Upper Bound,UB)再对问题进行求解;或采用核算法[12~15]缩小问题求解规模,在保证算法求解精度的前提下,加速算法求解速度.另一方面,随着群智能算法的兴起,大量新型群智能算法被应用于求解背包问题及其子问题[11],如:狼群算法[17]、人工鱼群算法[18]、烟花算法[19]等.较多文献[20~22]也关注于利用群智能算法对 MCKP 进行快速求解.但值得注意的是,群智能算法对于超大规模问题在求解精度及求解速度上仍有很多的问题值得进一步研究,因此精度更高,速度更快的贪心算法更值得探究.不仅如此,高效精准的贪心算法对于诸如粒子群等群智能算法初始解的生成也能够起到明显改进的效果.

2 多选择背包问题模型

2.1 多选择背包问题二进制模型

记 n_i 为第 $i(1\leqslant i\leqslant m)$ 类项集中的物品个数,为简化问题求解难度,假设每个项集内物品个数相同,即 $n_p=n_q=n(1\leqslant p<q\leqslant m)$;$p_{ij}$ 为第 i 类项集中第 j 个物品的价值;w_{ij} 为第 i 类项集中第 j 个物品的质量;x_{ij} 表示第 i 类项集中第 j 个物品是否被选择;C 为背包容量.

MCKP 的二进制模型(Binary Modeling,BM)可描述如下[23]:

$$\max f(x)=\sum_{i=1}^{m}\sum_{j=1}^{n_i}p_{ij}x_{ij} \tag{1}$$

s.t.

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i}w_{ij}x_{ij}\leqslant C \tag{2}$$

$$\sum_{j=1}^{n_i}x_{ij}=1 \tag{3}$$

$$x_{ij}\in\{0,1\} \tag{4}$$

其中,$i=1,2,\cdots,m$;$j=1,2,\cdots,n$.

传统背包问题主要约束为背包载重 C,而 MCKP 与其他背包问题最大的区别便在于除了背包载重约束外,还有式(3)的约束.式(3)表示同一项集中多个物品选择其中一个物品的情况.MCKP 通过式(3)控制每个项集均被选中,且每个项集中物品数量有且仅有一个.也即是说,第 i 项集中的 n_i 个物品有且仅有一个物品被选择,从而表达出“多选择(Multiple Choice)”这一理念.关于 MCKP 模型的相关讨论可具体参考文献[1].

2.2 多选择背包问题离散模型

BM 随着项集内物品个数的增加,冗余变量过多,因此,当项集中物品个数 n 较大时,为了有效减少求解算法的空间复杂度,有必要利用整数编码的方式改进 BM,类比 BM,其离散模型(Discrete Modeling,DM)可描述如下:

$$\max f(x)=\sum_{i=1}^{m}p_{ix_i} \tag{5}$$

s.t.

$$\sum_{i=1}^{m}w_{ix_i}\leqslant C \tag{6}$$

其中,$i=1,2,\cdots,m$;$x_i=1,2,\cdots,n_i$,表示第 i 个项集所选择物品的序号.

DM 模型通过缩减冗余变量,将算法空间复杂度从 $O(nm)$ 降低到 $O(m)$,减小求解算法的空间复杂度,达到改进模型或算法的目的.对于超大规模 MCKP 问题而言,因设备内存原因,一般采用 DM 模型进行计算.

3 凸帕累托算法及其性质

随着问题规模的增加,无论是精确算法还是群智能算法,分别存在求解速度与求解精度的问题.因此,针对超大规模多选择背包问题,快速且精度可接受的算法仍值得研究.

凸帕累托算法(Convex Pareto Algorithm,CPA)因其能够精确求解 LMCKP[1,3,11],为 MCKP 提供有效上界和近似解,相对其他算法而言,结构相对简单,求解速度较快,且能保证所得结果一定是可行解,因此受到广泛应用.

相对于 MCKP,LMCKP 主要是将式(4)中的约束转化为连续变量,即:

$$x_{ij}\in[0,1] \tag{7}$$

当 $x_{ij}\in[0,1]$ 时,不难发现,LMCKP 的解空间具备凸性,因此可直接采用价值密度等方法对问题进行贪心求解[4,13,24~26].

3.1 凸帕累托算法性质

记第 i 个项集的所有物品集合为 N_i,个数为 n.

定义 1 $\forall r,s\in N_i$,若 $w_{ir}\leqslant w_{is}$,$p_{ir}\geqslant p_{is}$,则称 r 支配 s,记为 $r\succ s$.

将第 i 个项集中所有的线性非支配(Linear Programming undominated,LP-undominated)物品集合称为线性非支配子集,并记为 R_i.显然,$R_i\subset N_i$.

定义 2 $\forall s\in N_i$,$\exists r,t\in R_i$,当 $w_{ir}\leqslant w_{is}\leqslant w_{it}$,$p_{ir}\leqslant p_{is}\leqslant p_{it}$ 时,满足:

$$\begin{vmatrix} w_{is}-w_{ir} & p_{is}-p_{ir} \\ w_{it}-w_{ir} & p_{it}-p_{ir} \end{vmatrix}\leqslant 0 \tag{8}$$

则称物品 s 被物品 r,t 线性支配[4].

由定义 2 知,R_i 具备凸性.

定理 1　对 $\forall r,t\in R_i$,若 $\forall u\in R_i$,使得 $w_{ir}\leqslant w_{it}\leqslant w_{iu}$ 或 $w_{iu}\leqslant w_{ir}\leqslant w_{it}$,且 $\exists s\in N_i, s\notin R_i, w_{ir}\leqslant w_{is}\leqslant w_{it}$,则 $\exists\lambda\in[0,1]$,使得[27]:

$$\lambda w_{it}+(1-\lambda)w_{ir}=w_{is} \tag{9}$$

$$\lambda p_{it}+(1-\lambda)p_{ir}\geqslant w_{is} \tag{10}$$

3.2　凸帕累托算法模型

由式(9)、(10)可知,R_i 具备凸性,故当 $x_{ij}\in[0,1]$ 时,CPA 可对 LMCKP 进行精确求解.

对于超大规模 MCKP,虽然诸如动态规划[28~30],分支定界[28,31,32]等算法能够得到精确解,但其求解速度往往无法接受. 因此精度可接受的高效近似算法仍然值得研究. CPA 因为其在 LMCKP 方面的优势,被广泛应用于 MCKP 的近似求解方面.

CPA 将同一项集中的物品进行重新建构. 算法思想首先选择各项集中的物品质量最小项作为初始解,以确保初始解为可行解. 然后,CPA 将问题中各项集选择且仅选择一个物品这一条件进行转化,对物品进行拆解.

记第 i 个项集的线性非支配子集 R_i 中第 j 个物品 x_{ij},其中 $1\leqslant i\leqslant m, 1\leqslant j\leqslant \mathrm{card}(R_i)$,其对应的物品价值 p'_{ij} 和物品质量 w'_{ij},其计算方式为:

$$\begin{cases} p'_{ij}=p_{ij}, & \text{if}\quad j=1\\ w'_{ij}=w_{ij}, & \text{if}\quad j=1\\ p'_{ij}=p_{ij}-p_{i,j-1}, & \text{if}\quad j\neq 1\\ w'_{ij}=w_{ij}-w_{i,j-1}, & \text{if}\quad j\neq 1 \end{cases} \tag{11}$$

则问题转化为:

$$\max f(x)=\sum_{i=1}^{m}\sum_{j=2}^{\mathrm{card}(R_i)}p'_{ij}x'_{ij}+\sum_{i=1}^{m}p'_{i1} \tag{12}$$

s. t.

$$\sum_{i=1}^{m}\sum_{j=2}^{\mathrm{card}(R_i)}w'_{ij}x'_{ij}+\sum_{i=1}^{m}w'_{i1}\leqslant C \tag{13}$$

$$x'_{i,j+1}\leqslant x'_{ij} \tag{14}$$

$$x'_{ij}\in\{0,1\} \tag{15}$$

新模型去除各项集选择且仅选择一项物品(式(3))这一约束,转化为式(14),这便是 CPA 的工作原理.

3.3　凸帕累托算法

由式(11)可知,CPA 价值密度计算式如下:

$$e'_{ij}=\frac{p'_{ij}}{w'_{ij}} \tag{16}$$

也即:

$$e'_{ij}=\begin{cases}\dfrac{p_{ij}}{w_{ij}}, & \text{if}\quad j=1\\[2ex] \dfrac{p_{ij}-p_{i,j-1}}{w_{ij}-w_{i,j-1}}, & \text{if}\quad j\neq 1\end{cases} \tag{17}$$

记 MCKP 实例的价值系数矩阵为 $\boldsymbol{P}=[p_{ij}]_{m\times n_i}$,重量系数矩阵为 $\boldsymbol{W}=[w_{ij}]_{m\times n_i}$,和背包载重 C. 不失一般性,设 $p_{ij},w_{ij}(1\leqslant i\leqslant m,1\leqslant j\leqslant n_i)$ 和 C 均为正整数.

由式(12)~(17)可知,CPA 算法伪代码可描述如算法 1.

算法 1　CPA

Step1:对第 i 个项集内物体质量 W_i 及价值 P_i 按照质量大小由小到大排序.

Step2:取各项集中物品质量最小项,保证算法初始解为可行解.

Step3:利用定义 1 搜寻第 i 个项集的帕累托前沿,在帕累托前沿基础上利用式(8)搜寻线性非支配子集 R_i.

Step4:计算式(16),即求价值密度.

Step5:在不超出背包载重 C 的前提下,按照价值密度从高到低对问题进行贪心求解.

Step6:输出结果

4　改进帕累托算法及其性质

由定理 1 可知,对于 MCKP,即当 $x_{ij}\in\{0,1\}$ 时,问题不再具备凸性[24~27]. 因此,CPA 对于 MCKP 不再具有精确求解的能力. 同时,CPA 在求解帕累托前沿和线性非支配子集时流程仍较复杂,值得探索更简单快捷且高效的贪心算法.

4.1　改进帕累托算法性质

对于线性非支配子集 R_i,显然直接使用式(8)对问题进行求解较为复杂,考虑集合 R_i 内元素特性,构造新的物品价值密度排序方式.

记第 $i(1\leqslant i\leqslant m)$ 个项集第 $j(1\leqslant j\leqslant n)$ 物品的价值密度为 e_{ij},类比式(11)可知:

$$e_{ij}=\frac{p_{ij}}{w_{ij}} \tag{18}$$

不难发现,e_{ij} 相比于 e'_{ij} 计算方式更为简单快捷,因此考虑利用 e_{ij} 作为基本判断原则,通过增加其他约束达到线性非支配子集 R_i 的效果.

不妨令 $r\in R_i$. 且对 $\forall u\in R_i$,有:

$$w_{iu}\geqslant w_{ir} \tag{19}$$

定理 2　对 $\forall s\in N_i, w_{is}>w_{ir}$,若 $p_{ir}>p_{is}$,则 $s\notin R_i$.

证明　因 $w_{is}>w_{ir}$. 又 $p_{ir}>p_{is}$,故 $r\succ s$,显然 $s\notin R_i$. 证毕

定理 3　对 $\forall s,t\in N_i, w_{is}>w_{ir}, w_{it}>w_{ir}, p_{it}>p_{ir}$,若 $e_{it}>e_{is}$,且 $p_{it}\geqslant p_{is}$,则 $s\notin R_i$.

证明　下面分情况讨论.

(1)当 $w_{it}\leqslant w_{is}$ 时,因 $p_{it}\geqslant p_{is}$,则 $t\succ s$. 显然 $s\notin R_i$.

(2)当 $w_{it}>w_{is}$ 时,若 $p_{is}\leqslant p_{ir}$,则 $r\succ s$,故 $s\notin R_i$.

(3)当 $w_{it}>w_{is}$ 时,且 $p_{is}>p_{ir}, e_{it}\geqslant e_{is}$,则有:

$$p_{is}\leqslant\frac{p_{it}}{w_{it}}w_{is} \tag{20}$$

又有：

$$\frac{p_{it}}{w_{it}}w_{is}=\frac{w_{is}}{w_{it}}p_{it}<\frac{w_{is}p_{it}-w_{ir}p_{it}}{w_{it}-w_{ir}} \tag{21}$$

因 $w_{it}>w_{is}>w_{ir}$，由式(20)可知：

$$\begin{aligned}\frac{w_{is}p_{it}-w_{ir}p_{it}}{w_{it}-w_{ir}}&\leqslant\frac{w_{is}p_{it}-w_{ir}p_{it}+w_{it}p_{ir}-w_{is}p_{ir}}{w_{it}-w_{ir}}\\&=\frac{w_{is}p_{it}-w_{ir}p_{it}+w_{it}p_{ir}-w_{is}p_{ir}+w_{ir}p_{ir}-w_{ir}p_{ir}}{w_{it}-w_{ir}}\\&=\frac{w_{is}p_{it}-w_{ir}p_{it}-w_{is}p_{ir}+w_{ir}p_{ir}}{w_{it}-w_{ir}}+p_{ir}\\&=\frac{(w_{is}-w_{ir})(p_{it}-p_{ir})}{w_{it}-w_{ir}}+p_{ir}\end{aligned} \tag{22}$$

由式(20)～(22)可知，当

$$p_{ir}\leqslant p_{is}\leqslant\frac{(w_{is}-w_{ir})(p_{it}-p_{ir})}{w_{it}-w_{ir}}+p_{ir},\lambda=\frac{w_{is}-w_{ir}}{w_{it}-w_{ir}},$$

则有：

$$\begin{aligned}&\lambda p_{it}+(1-\lambda)p_{ir}\\&=\frac{w_{is}-w_{ir}}{w_{it}-w_{ir}}p_{it}+(1-\frac{w_{is}-w_{ir}}{w_{it}-w_{ir}})p_{ir}\\&=\frac{w_{is}-w_{ir}}{w_{it}-w_{ir}}p_{it}+\frac{w_{it}-w_{ir}-w_{is}+w_{ir}}{w_{it}-w_{ir}}p_{ir}\\&=\frac{w_{is}-w_{ir}}{w_{it}-w_{ir}}(p_{it}-p_{ir})+\frac{w_{it}-w_{ir}}{w_{it}-w_{ir}}p_{ir}\\&=\frac{w_{is}-w_{ir}}{w_{it}-w_{ir}}(p_{it}-p_{ir})+p_{ir}\geqslant p_{is}\end{aligned} \tag{23}$$

由式(10)、(23)可知，则 s 被 r、t 线性支配，故 $s\notin R_i$

综上，定理3得证. 证毕

4.2 改进帕累托算法

结合定理3，提出改进帕累托算法(Improved Pareto Algorithm，IPA). 其中，$\boldsymbol{P},\boldsymbol{W},C$ 参数设置与算法1相同. IPA算法伪代码可描述如算法2.

算法2 IPA

输入：$\boldsymbol{P},\boldsymbol{W},C$
输出：obj
1) $\boldsymbol{X}=\text{ones}(1,m)$；obj=0；$t=0$；
2) FOR $i\leftarrow 1:m$
3) $[a,b]=\min(\boldsymbol{W}_i)$
4) $\boldsymbol{X}_i=b$；obj=obj+$\boldsymbol{P}_{ib}$；
5) $t=t+a$；
6) END
7) $\boldsymbol{E}=\text{reshape}(\boldsymbol{P},1,[])./\text{reshape}(\boldsymbol{W},1,[])$；
$\boldsymbol{E}=[\boldsymbol{E};1:nm]$；
8) $\boldsymbol{E}=-\text{sortrows}(-\boldsymbol{E}^{\mathrm{T}},1)^{\mathrm{T}}$；
9) FOR $i\leftarrow 1:nm$
10) $k_1=\boldsymbol{E}_{2i}$；$k_2=\text{ceil}(k_1/n)$；
11) IF $\boldsymbol{P}_{k_2,\boldsymbol{X}(k_2)}<\boldsymbol{P}_{k_2,k_1-k_2\times n+n}$ &&
$t-\boldsymbol{W}_{k_2,\boldsymbol{X}(k_2)}+\boldsymbol{W}_{k_2,k_1-k_2\times n+n}\leqslant C$
12) $\boldsymbol{X}_{k_2}=k_1-k_2\times n+n$；
$o=o-\boldsymbol{P}_{k_2,\boldsymbol{X}(k_2)}+\boldsymbol{P}_{k_2,k_1-k_2\times n+n}$；
13) END
14) END
15) RETURN o

步7、8价值密度排序；步9～14贪心选择.

相对于CPA，IPA不再用对所有数据进行排序，时间复杂度从 $O(nm\log n)$ 下降至 $O(nm)$，此外，IPA不再对线性非支配子集进行求解，从而在这两方面大幅度缩减求解时间，提升求解效率.

5 实例计算与比较

为了有效展示IPA算法性能，通过实际算例对IPA和CPA进行测试. 参考文献[4]，设置四类实例，分别为：不相关实例(UnCorrelated instances，UC)，弱相关实例(Weakly Correlated instances，WC)，强相关实例(Strongly Correlated instances，SC)，逆向强相关实例(Inverse strongly Correlated instances，IC).

本文使用计算机基本配置为Intel(R) Core(TM) i7-8700 CPU@3.2GHz(12 CPUs)，16GB DDR4L(12 GB剩余)，Microsoft Windows 10家庭版. 利用MATLAB R2016a对问题进行求解绘图.

5.1 实例数据

四类实例的规模分别为 $1000\leqslant n\leqslant 10000$，$1000\leqslant m\leqslant 10000$，$n,m$ 具体数值为编号数乘以一千. 编号及参数设置如下(具体数据可通过邮箱索取).

(1) UC实例. 实例编号UC1～10，参数设置为：$w_i\in_R[1,1000]$，$p_i\in_R[1,1000]$.

(2) WC实例. 实例编号WC1～10，参数设置为：$w_i\in_R[101,1000]$，$p_i\in_R[w_i-100,w_i+100]$.

(3) SC实例. 实例编号SC1～10，参数设置为：$w_i\in_R[1,1000]$，$p_i=w_i+100$.

(4) IC实例. 实例编号IC1～10，参数设置为：$w_i\in_R[101,1000]$，$p_i=w_i-100$.

其中，$x\in_R[a,b]$ 表示变量 $x\in[a,b]$ 且 x 为实数.

实例的背包载重为 $C=\alpha\sum_{i=1}^{m}\max(\boldsymbol{W}_i)$，其中 $\boldsymbol{W}_i$ 表示 N_i 中物品的质量所构成的向量，且 $\alpha\in[0.8,0.9]$. 且 $C\geqslant\sum_{i=1}^{m}\min(\boldsymbol{W}_i)$，保证可行解.

5.2 计算结果

为了充分说明IPA算法的求解性能，除了传统CPA算法以外，还引入文献[5]中的GR-DKP(GR-DKP贪心策略为选择同一项集中价值密度最大项)和PSO-GRD-KP与IPA进行比较. 此外，为了表明IPA对于群智能算

法在初始解及贪心修复过程中的作用,将 PSO-GRDKP 中的 GR-DKP 贪心修复算法修改为 IPA,记新算法为 PSO-IPA.

通过实际计算对比,得到几类 PSO 的参数设置为:速度更新参数为 $c_1 = 1.49445$, $c_2 = 1.49445$;迭代次数 maxgen = 50,种群规模 sizepop = 50;个体速度最大最小值 $V_{max} = 125$, $V_{min} = -125$. 具体内容可参考文献[33].

分别利用 IPA 等五种算法对四类实例进行计算. 因 CPA、IPA 和 GR-DKP 三类算法均为非随机性算法,输出结果稳定,因此只记录最优解及求解时长(单位:s). 其余两类算法除最优解和求解时长外,增加平均值的记录. 求解时长为各算法 30 次独立重复试验的平均时长. 特别地,对于单次计算时长超过 1000s 的计算认为是无效计算. 此外,因利用 CPLEX 等求解器计算百万至亿级的超大规模实例均出现内存超出现象,因此无法得到算例的精确解,这更体现了快速高效的贪心算法的重要性. 为了更好的展示各算法对比通过实际计算所得结果见图 1 ~ 8.

5.3　计算结果分析

从图 1 ~ 8 中数据可知:IPA 在计算精度上不仅在能够达到 CPA 的效果,甚至有个别算例出现 IPA 结果优于 CPA 的情况. 在求解时长上,对于有效求解的 CPA 结果而言,IPA 求解速度平均提升 98.86%. 混合 IPA 的粒子群算法 PSO-IPA 求解精度平均提升 28.92%. 显然,无论从求解精度还是求解时长来看,IPA 算法均优于 CPA 算法.

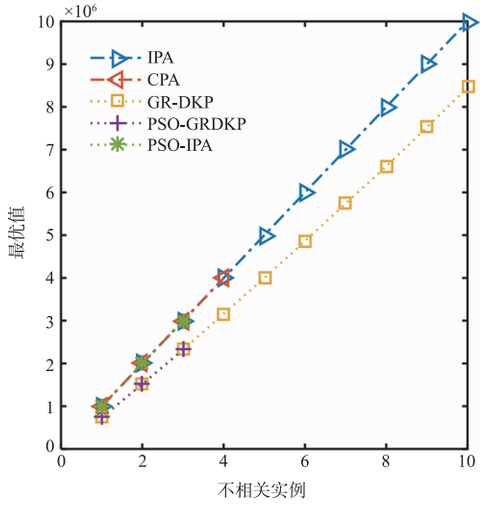

图1　五种算法求解UC结果对比图

同时,GR-DKP 相比于 IPA,除在 IC 实例中因算例个体价值密度差异较大,从而使得求解性能与 IPA 相当. 在其它类型算例中,因价值密度相对较为接近,因而

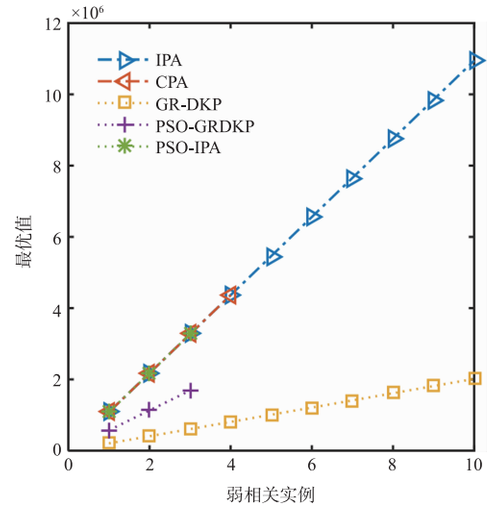

图2　五种算法求解WC结果对比图

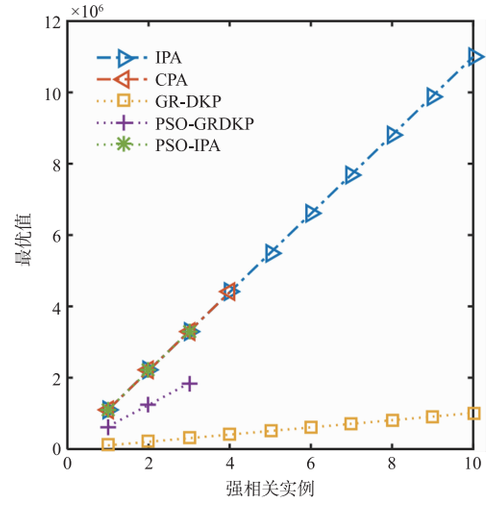

图3　五种算法求解SC结果对比图

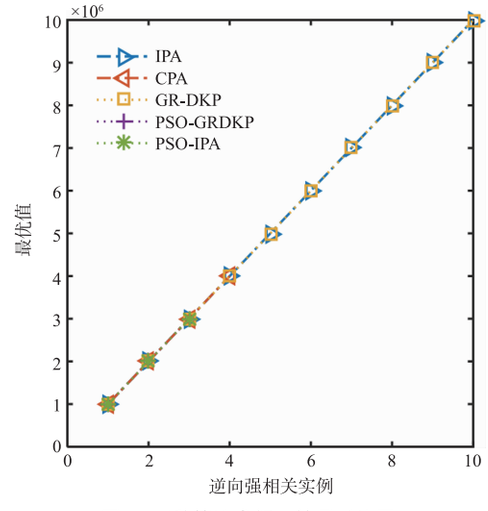

图4　五种算法求解IC结果对比图

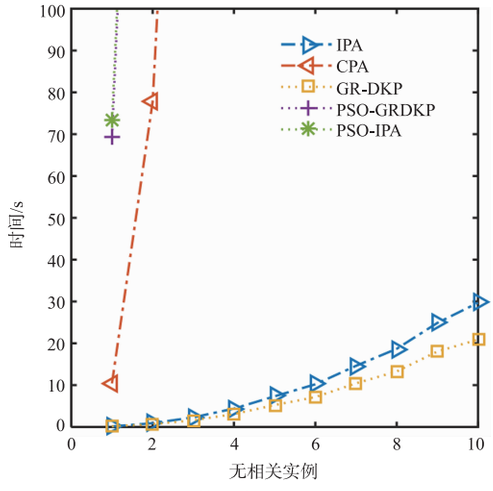

图5 五种算法求解UC速度对比图

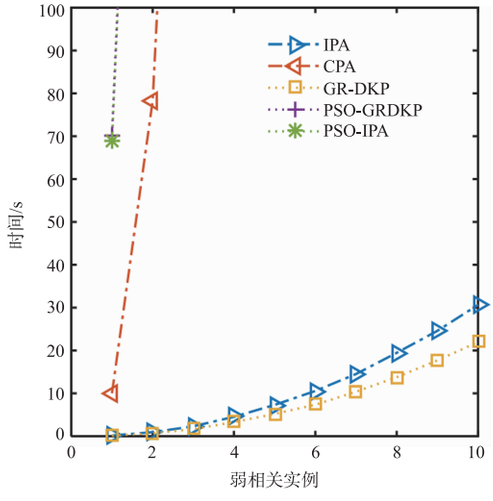

图6 五种算法求解WC速度对比图

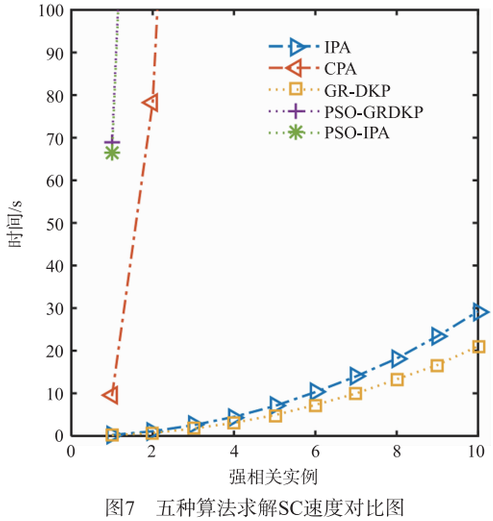

图7 五种算法求解SC速度对比图

出现 GR-DKP 求解性能较差的现象. 总体看来,GR-DKP

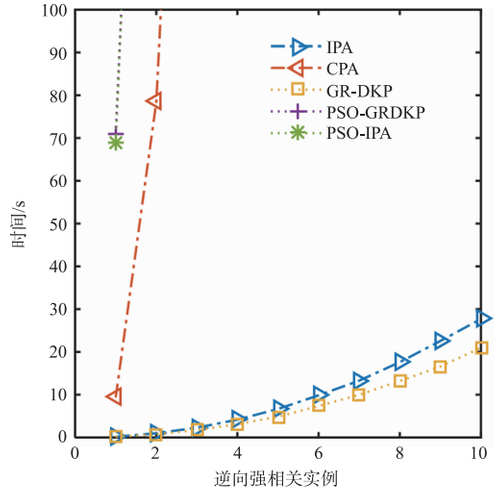

图8 五种算法求解IC速度对比图

在同一项集中物品数量较少或同一项集中物体价值密度差异较大时性能与 IPA 较为接近.

而通过 PSO-GRDKP 与 PSO-IPA 的求解对比不难发现,IPA 对于诸如 PSO 等群智能算法性能提升明显. 但同时值得注意的是,当 IPA 取值接近最优解时,单纯采取贪心修复的 PSO 算法对于局部搜索效果不佳. 接下来还可以考虑在 IPA 基础上,混合局部搜索性能更加优秀的算法对问题进行进一步求解.

6 结论

本文通过考虑线性非支配子集 R_i 的对立面判断原则,提出 IPA 算法. 显然通过定理 3 所得集合与 R_i 并不相等,但其结果更加精准. 而 IPA 因有效避免物品排序,求解帕累托子集,求解线性非支配子集等操作,大幅度缩减求解时长,快速提升求解速度. 实验结果表明,相对于 CPA,IPA 在求解时长上平均提升 98.86%. 对于定理 3 所提出的集合具备的性质及其可推广性应用将是接下来仍值得考虑的地方. 同时,也可以考虑混合局部搜索性能更强的算法对超大规模的多选择背包问题进行更好地求解.

致谢 感谢审稿专家和编辑老师为本文提出的宝贵意见. 特别鸣谢西华师范大学数学与信息学院的李军老师、潘大志老师以及上海立信会计金融学院统计与数学学院王珏钰老师对本文的指导与建议.

参考文献

[1] NAUSS R M. The 0-1 knapsack problem with multiple choice constraints[J]. European Journal of Operational Research,1978,2(2):125 – 131.

[2] BALINTFY J L,ROSS G T,SINHA P,et al. A mathemati-

cal programming system for preference and compatibility maximized menu planning and scheduling[J]. Mathematical Programming,1978,15(1):63 – 76.

[3] SINHA P, ZOLTNERS A A. Integer programming models for sales resource allocation [J]. Management Science, 1980,26(3):242 – 260.

[4] RONG A Y , FIGUEIRA J R, KLAMORTH K . Dynamic programming based algorithms for the discounted {0-1} knapsack problem[J]. Applied Mathematics and Computation,2012,218(12):6921 – 6933.

[5] HE Y C , WANG X Z , HE Y L , et al. Exact and approximate algorithms for discounted {0-1} knapsack problem [J]. Information Sciences,2016,369:634 – 647.

[6] 冯艳红,杨娟,贺毅朝,王改革. 差分进化帝王蝶优化算法求解折扣{0-1}背包问题[J]. 电子学报,2018,46(6):1343 – 1350.
FENG Y H, YANG J, HE Y C, et al. Monarch butterfly optimization algorithm with differential evolution for the discounted {0-1} knapsack problem[J]. Acta Electronica Sinica,2018,46(6):1343 – 1350. (in Chinese)

[7] HE Y C, XIE H , WONG T L , et al. A novel binary artificial bee colony algorithm for the set-union knapsack problem[J]. Future Generation Computer Systems, 2018, 78 (1):77 – 86.

[8] WEI Z Q, HAO J K. Iterated two-phase local search for the Set-Union Knapsack Problem[J]. Future Generation Computer Systems,2019,101(07):1005 – 1017.

[9] PISINGER D. A minimal algorithm for the multiple-choice knapsack problem[J]. European Journal of Operational Research,2007,83(2):394 – 410.

[10] MARTELLO S, TOTH P P. Dynamic programming and strong bounds for the 0-1 knapsack problem[J]. Management Science,1999,45(3):414 – 424.

[11] DYER M E, KAYA N, WALKER J. A branch and bound algorithm for solving the multiple-choice knapsack problem[J]. Journal of Computational and Applied Mathematics,1984,11(2):231 – 249.

[12] BALAS E, ZEMEL E. An algorithm for large zero-one knapsack problems [J]. Operations Research, 1980, 28 (5):1130 – 1154.

[13] MARTELLO S, TOTH P. A new algorithm for the 0-1 knapsack problem [J]. Management Science, 1988, 34 (5):633 – 644.

[14] MARTELLO S, TOTH P. Knapsack Problems: Algorithms and Computer Implementations[M]. New Jersey: John Wiley & Sons, Inc. 1990.

[15] PISINGER D. An expanding-core algorithm for the exact 0-1 knapsack problem[J]. European Journal of Operational Research,1995,87(1):175 – 187.

[16] 王熙照,贺毅朝. 求解背包问题的演化算法[J]. 软件学报,2017,28(01):1 – 16.
WANG X Z, HE Y C. Evolutionary algorithms for knapsack problems[J]. Journal of Software,2017,28(01):1 – 16. (in Chinese)

[17] 吴虎胜,张凤鸣,战仁军,李浩,梁晓龙. 利用改进的二进制狼群算法求解多维背包问题[J]. 系统工程与电子技术,2015,37(05):1084 – 1091.
WU H S, ZHANG F M, ZHAN R J, et al. Improved binary wolf pack algorithm for solving multidimensional knapsack problem[J]. Systems Engineering and Electronics, 2015,37(05):1084 – 1091. (in Chinese)

[18] 李迎,张璟,刘庆,张伟. 求解大规模多背包问题的高级人工鱼群算法[J]. 系统工程与电子技术,2018,40(03):710 – 716.
LI Y, ZHANG J, LIU Q, et al. Advanced artificial fish swarm algorithm for large scale multiple knapsack problem[J]. Systems Engineering and Electronics, 2018, 40 (03):710 – 716. (in Chinese)

[19] 薛俊杰,王瑛,孟祥飞,等. 二进制反向学习烟花算法求解多维背包问题[J]. 系统工程与电子技术,2017,39(02):451 – 458.
XUE J J, WANG Y, MENG X F, et al. Binary opposite back-ward learning fire works algorithm for multidimensional knapsack problem[J]. Systems Engineering and Electronics,2017,39(02):451 – 458. (in Chinese)

[20] 印桂生,崔晓晖,董宇欣,杨雪. 面向离散优化问题的改进二元粒子群算法[J]. 哈尔滨工程大学学报,2015,36(02):191 – 195.
YIN G S, CUI X H, DONG Y X, et al. An improved binary particle swarm optimization for discrete optimization problems[J]. Journal of Harbin Engineering University, 2015,36(02):191 – 195. (in Chinese)

[21] 于永新,张新荣. 基于蚁群系统的多选择背包问题优化算法[J]. 计算机工程,2003(20):75 – 76 + 84.
YU Y X, ZHANG X R. Optimization algorithm for multiple-choice knapsack problem based on ant colony system [J]. Computer Engineering, 2003 (20): 75 – 76 + 84. (in Chinese)

[22] 韩燕燕,马良,赵小强. 多选择背包问题的人工蜂群算法[J]. 计算机应用研究,2012,29(03):862 – 864.
HAN Y Y, MA L, ZHAO X Q. Artificial bee colony algorithm for multi-choice knapsack problem[J]. Application Research of Computers, 2012, 29 (03): 862 – 864. (in Chinese)

[23] IBARAKI T, HASEGAWA T, TERANAKA K, et al. The multiple-choice knapsack problem[J]. Journal of the Operations Research Society of Japan, 1978, 21 (01): 59 – 95.

[24] 华中生,张斌. 求解可分离连续凸二次背包问题的直接算法[J]. 系统工程与电子技术,2005,27(2):331 – 334.
HUA Z S,ZHANG B. Direct algorithm for separable continuous convex quadratic knapsack problem[J]. Systems Engineering and Electronics,2005,27(2):331 – 334. (in Chinese)

[25] 王粉兰. 非线性整数规划问题的若干新算法[D]. 上海:上海大学,2006.
WANG F L. New Algorithms for Nonlinear Integer Programming Problems[D]. Shanghai:Shanghai University,2006. (in Chinese)

[26] ZHENG X,SUN X L,LI D,et al. Successive convex approximations to cardinality-constrained convex programs:a piecewise-linear DC approach[J]. Computational Optimization and Applications,2014,59(1 – 2):379 – 397.

[27] SINHA P,ZOLTNERS A A. The multiple-choice knapsack problem[J]. Operations Research,1979,27(3):503 – 515.

[28] DYER M E,RIHA W O,WALKER J. A hybrid dynamic programming/branch-and-bound algorithm for the multiple-choice knapsack problem[J]. Journal of Computational and Applied Mathematics,1995,58(1):43 – 54.

[29] CLAUTIAUX F,SADYKOV R,VANDERBECK F,et al. Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem[J]. Discrete,2018,29,18 – 44.

[30] PFERSCHY U,SCATAMACCHIA R. Improved dynamic programming and approximation results for the knapsack problem with setups[J]. International Transactions in Operational Research,2017. 667 – 682.

[31] LIU G S,LI J J,YANG H D,et al. Approximate and branch-and-bound algorithms for the parallel machine scheduling problem with a single server[J]. Journal of the Operational Research Society,2019,70(9):1 – 17.

[32] SUNG C S,CHO Y K. Branch-and-bound redundancy optimization for a series system with multiple-choice constraints[J]. IEEE Transactions on Reliability,1999,48(2):108 – 117.

[33] 郁磊,史峰,王辉,等. MATLAB 智能算法 30 个案例分析[M]. 北京:北京航空航天大学出版社,2011. 130 – 136.

作者简介

杨 洋 男,1993 年 5 月出生,四川资阳人. 西华师范大学助教,主要研究方向为整数规划,图论与组合最优化.
E-mail:zhugemutian@ outlook. com