文章编号: 1001-0920(2009)05-0729-05

一种新的求解 MMKP 问题的 ACO &PR 算法

张晓霞[1,2], 唐立新[2]
(1. 辽宁科技大学 软件学院, 辽宁 鞍山 114000; 2. 东北大学 物流优化与控制研究所, 沈阳 110004)

摘 要: 针对多选择多维背包问题(MMKP)的特点,设计一种新型混合算法(ACO &PR). 该算法将线路重连算法(PR)嵌入蚁群算法(ACO),在搜索过程中既考虑解的质量,又考虑解的分散性. 线路重连算法在重连过程中,向导解的属性逐步引入起始解属性中,可快速获得该线路上的最优解. 实验结果表明,该算法优于其他现有较好的方法,获得了较好的结果.

关键词: 多选择多维背包; 蚁群算法; 线路重连算法
中图分类号: TP18 **文献标识码**: A

A new ACO &PR algorithm for multiple-choice multidimensional knapsack problem

ZHANG Xiao-xia[1,2], *TANG Li-xin*[2]
(1. The Software College, University of Science and Technology Liaoning, Anshan 114000, China; 2. Logistics Institute, Northeastern University, Shenyang 110004, China. Correspondent: ZHANG Xiao-xia, E-mail: syzhangxx@163.com)

Abstract: For the features of the multipl-choice multidimensinal knapsack problem(MMKP), a novel hybrid algorithm (ACO &PR) is designed, in which path relinking (PR) is embeded into the solution construction mechanism of ant colony optimization, and solution diversification is considered besides solution quality in the search process. In the process of the path relinking phase, the attributes of the guiding solution is introduced into the initial solution progressively to obtain the high quality solution as quickly as possible. The experimental results show that the method is very efficient and competitive to solve the MMKP compared with the better existing methods.

Key words: Multiple-choice multidimensional knapsack; Ant colony optimization; Path relinking

1 引 言

背包问题(KP)是运筹学领域中一个经典的具有 NP 难度的组合优化问题[1],为满足广泛的实际应用的需要,出现了许多变形[2]. 多维背包问题(MDKP)是一种最常用的变形,它考虑了多种资源约束[3],但物品的选择没有限制. 多选择背包问题(MCKP)是背包问题的另一种变型[4,5],它将物品分为若干类,在每类物品中只能选择一种物品装入背包. MCKP 考虑了物品选择约束,但只考虑了单一资源约束. 本文主要研究多选择多维背包(MMKP)问题,是最复杂的一类变形. MMKP 实际上是 MDKP 与 MCKP 的组合,它既考虑了多种资源的约束条件,同时又考虑了物品选择条件的限制.

许多文献是围绕单个约束进行研究,并获得了一些有效的近似算法[3],而直接研究 MMKP 的文献却相对较少. 精确算法只能求解规模较小的问题[6],而对于大规模问题则依赖启发式算法加以解决. Heranandez 等[7]提出一种将 MMKP 转换为 MDKP 的启发式算法,并获得了较高的质量解,但所消耗的时间相对多一些. Hifi 等[8]提出了解决该问题的启发式算法,该算法通过惩罚函数改变搜索轨迹,从而使算法跳出局部最优,并获得了较好的结果. 本文给出了采用蚁群算法(ACO 算法)直接解决 MMKP 问题的算法设计,包括状态转移方式和信息素更新方式. 同时针对 ACO 易于出现早熟、停滞现象,设计了一种新型混合算法(ACO &PR). 该算法既考虑了解的质量,又考虑了解的分散性,为 MMKP 的研究提供了一种新的有效方法.

收稿日期: 2008-05-16; **修回日期**: 2008-08-22.
基金项目: 国家杰出青年科学基金项目(70425003); 国家 863 计划项目(2006AA04Z174).
作者简介: 张晓霞(1967—),女,辽宁鞍山人,博士生,从事物流优化、智能优化算法等研究; 唐立新(1966—),男,黑龙江绥化人,教授,博士生导师,从事物流优化、生产调度、优化算法等研究.

2 数学描述

MMKP 给出 m 种资源,将需放入背包的物品分为相互排斥的 n 类,$I=\{1,\cdots,n\}$,物品集合为 $J=\{J_1,\cdots,J_n\}$,每类物品 $J_i(i\in I)$ 有 n_i 个不同的物品.在 J_i 类中,每个物品 $j(j\in J_i, J_i=\{1,\cdots,n_i\})$ 的单位重量收益值为 c_{ij},所需要的 k 资源为 $w_{ij}^k(k\in M, M=\{1,\cdots,m\})$.MMKP 就是从每类 $i(i\in I)$ 物品中选择一个满足资源约束的相应物品 $j(j\in J_i)$ 放入背包,使得装入物品总收益值最大.MMKP 问题的数学模型如下:

$$\text{maximize}\quad f=\sum_{i}^{n}\sum_{j}^{n_i}c_{ij}x_{ij}; \tag{1}$$

$$\sum_{i}^{n}\sum_{j}^{n_i}w_{ij}^k x_{ij}\le C^k,\ k=1,\cdots,m; \tag{2}$$

$$\sum_{j}^{n_i}x_{ij}=1,\ i=1,2,\cdots,n; \tag{3}$$

$$x_{ij}\in\{0,1\},\ i=1,2,\cdots,n,\quad j=1,2,\cdots,n_i. \tag{4}$$

约束(2)为背包容量约束,C^k 为 k 种资源的最大数量;约束(3)表示从每类物品中只能选择一个;约束(4)表示 x_{ij} 为 0-1 决策变量,当第 i 类中的第 j 个物品放入背包时,$x_{ij}=1$,否则 $x_{ij}=0$.可行解二进制编码如图 1 所示.为算法描述方便,先给出下面定义:

定义 1 用 $\sigma:=I\to J$ 映射表示每类物品与所选择物品的对应关系.$\sigma(i)=j$ 表示 i 类中物品 j 被选中,与决策变量 x_{ij} 的对应关系 $x_{ij}=1\Leftrightarrow\sigma(i)=j$.

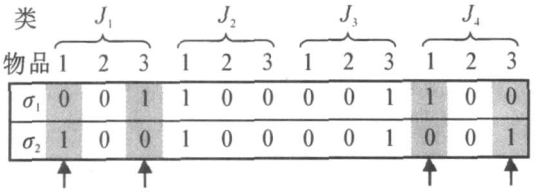

图 1 MMKP 解的表示

图 1 给出了二进制代码表示的 2 个解分别对应的映射解为 $\sigma_1=\{3,1,3,1\}$,$\sigma_2=\{1,1,3,3\}$.

定义 2 假设 σ 与 σ' 为任意 2 个解,则 $\mathrm{dist}(\sigma,\sigma')=|\{i\in I:\sigma(i)\neq\sigma'(i)\}|$ 为选择不同物品的类的数目,称为 2 个解分散度,表示这 2 个解的分散性.

$\mathrm{dist}(\sigma,\sigma')$ 值越大,表明这 2 个解的分散性越大.图 1 中 2 个解分散度为 $\mathrm{dist}(\sigma_1,\sigma_2)=2$.

3 算法设计

3.1 蚁群算法的基本原理

蚁群优化(ACO)算法是模拟蚁群觅食行为而设计的,其基本过程见文献[9,10].ACO 主要研究排序问题,如旅行商问题或二次分配问题等.而 MMKP 问题是一个子集选取问题,子集选取问题与排序问题的主要区别是前者不需考虑子集元素之间的顺序,而后者则需考虑集合之间的顺序.ACO 解决子集选取问题时,蚂蚁移动下个位置与所在的当前位置无关;对于排序问题,蚁群的信息素留在连接 2 个点的路径上,而子集选取问题信息素则留在子集的元素上,也就是用点代替排序问题的路径.下面介绍用 ACO 算法解决 MMKP 问题的过程.

将蚁群随机分布在某些类物品上,每个蚂蚁模拟一个背包.蚂蚁 k 从 i 类物品选择 j 物品,其对应的转移概率可定义为

$$p_{ij}^k=\begin{cases}\dfrac{\tau_{ij}[u_{ij}]^\beta}{\sum_{l\in J_i}\tau_{il}[u_{il}]^\beta}, & j\in J_i;\\ 0, & \text{otherwise.}\end{cases} \tag{5}$$

其中:τ_{ij} 表示第 i 类物品中 j 物品的信息素,β 为参数.$u_{ij}=c_{ij}/\sum_{k}^{m}w_{ij}^k\cdot C^k$ 表示资源的效率,在蚁群算法中,蚂蚁按下式选择下一个物品:

$$j=\begin{cases}\arg\max_{l\in J_i}\{\tau_{il}[u_{il}]^\beta\}, & q<q_0;\\ S, & \text{otherwise.}\end{cases} \tag{6}$$

其中:q_0 为一个参数,q 为均匀分布在 $[0,1]$ 的随机变量.$q>q_0$ 时,依式(5)决定随机变量 S.滞留在物品上的信息素随时间变化需要不断更新,信息素更新策略有局部和全局更新两种[10].局部更新可避免因频繁选择相同物品而陷入局部最优;全局更新可加强收益比较大物品上的信息素.

3.2 线路重连算法

线路重连算法(PR)是由 Glover 等[11]提出的,其详细介绍参见文献[12,13].PR 原理如下:首先产生初始种群,在初始种群中选择 b_1 个高质量解 RefSet_1 和 b_2 个分散性好的解 RefSet_2 作为参考集 R.从 R 中按一定规则选择 2 个解,分别作为起始解 σ_A(Initiating solution)和向导解 σ_B(Guiding solution);然后产生一条从起始解延伸到向导解的路径,以期获得质量高的解.在这条路径搜索移动过程中,向导解的属性被逐步引入起始解中,以形成一

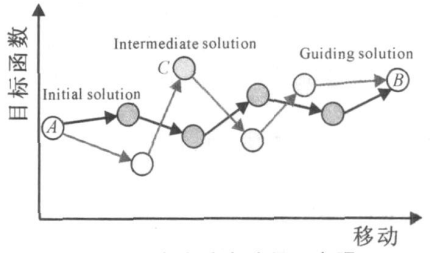

图 2 线路重连过程示意图

系列中间解(intermediate solution),用这些解按照一定准则更新参考集. 线路重连算法过程如图 2 所示.

PR 算法主要包括起始解和向导解选择规则和参考集合的更新方式两部分. 结合 MMKP 实际情况,给出下面定义.

定义 3　定义 $C^* = \{C_1, C_2\}$ 为起始解和向导解选择规则, C_1 为起始解和向导解分别从参考集 R 中选择分散度最大的解和最好解, C_2 为起始解从参考集 R 中随机选择一个解,当前最好解作为向导解.

定义 4　假设 σ_j^* 为 R 中的一个解,对于非参考集中的任意一个解 σ_j, $\mathrm{dist}_{\max}(\sigma_j)$ 定义为 $\mathrm{dist}_{\max}(\sigma_j) = \max\limits_{\sigma_j^* \in R}\{\mathrm{dist}(\sigma_j, \sigma_j^*)\}$, $\sigma_j \in$non-R,表示该解在 R 中分散度.

$\mathrm{dist}_{\max}(\sigma_j)$ 值最大的解就是非参考集合中分散性最好的解.

定义 5　定义 $R^* = \{R_1, R_2\}$ 为参考集更新准则, R_1 定义为解质量更新准则,如果满足 $f(\sigma_j) > \min\limits_{\sigma_i \in \mathrm{RefSet}_1}\{f(\sigma_i)\}$ 时,则新解 σ_j 更新 RefSet_1 集合中目标函数值最小的解; R_2 为分散度更新准则,如果满足 $\mathrm{dist}_{\max}(\sigma_j) > \min\limits_{\sigma_i \in \mathrm{RefSet}_2}\{\mathrm{dist}_{\max}(\sigma_i)\}$ 时,则 σ_j 更新 RefSet_2 集合中分散度最小的解.

下面给出 PR 算法过程,输入参数为参考集 R 和当前最好解 σ^*,函数返回值为当前最好解.

算法 1　线路重连算法

```
Path relinking(R, σ*)
{ P_size = |R|; n_iterate = 0;
{while n_iterate < P_size
{Select σ_A, σ_B          // 依 C* 选择 σ_A, σ_B
  v = dist(σ_A, σ_B); //dist 两个解分散度
  σ_C = σ_A;
While (v > 0 ‖σ_C != σ_B)
{ i = (int)(rand(1, n)// 随机产生物品类编号
  If (σ_C(i) = σ_B(i)) continue;
      Else σ_C(i) = σ_B(i);
  If σ_C is feasible
    { If (f(σ_C) > f(σ*)) f(σ*) = f(σ_C);
      If σ_C satisfies R* R = R ∪{σ_C};
      v = v - 1;}
  Else break;          // 不可行,退出循环
}
n_iterate = n_iterate + 1;}
return σ*;}            // 返回当前最好解
```

3.3　ACO &PR 算法

ACO &PR 算法是将 PR 嵌入到 ACO 算法构解机制中,该算法既考虑解的质量,同时又考虑解的分散性,从而增强了 ACO 算法跳出局部最优的能力. ACO &PR 算法主要由解的生成 Construct( ),解的修复 Repair( ) 和解的改进 3 部分组成.

解的生成 Construct( ) 采用 ACO 算法,主要有两部分,一部分是初始参考集的生成,另一部分是算法在每次迭代过程中解的生成. 产生的解有两种情况,一种是可行解,另一种是不可行解. 如果是不可行解,则首先对其修复,使其变为可行解. Construct( ) 的复杂性为 $O(\vartheta \cdot m \cdot n)$,其中 $\vartheta = \max\{n_1, \cdots, n_n\}$.

修复程序 Repair( ) 的功能是将不可行解转化为可行解,分为删除(DROP) 与添加(ADD) 物品两个阶段. DROP 阶段就是找出发生冲突最大的资源编号 k_0,然后删除 k_0 中资源权重系数最大的物品. $i_0 = \arg\max\limits_{1 \le i \le m}\{w_{ij_i}^{k_0}\}$;而 ADD 阶段就是在 i_0 类中找出满足资源约束的物品,并添加到该解中.

解的改进部分由线路重连程序 Path relinking( ) 和改进程序 Improve( ) 两部分组成. 当可行解 σ_S 生成之后,如果满足 R 更新准则 R^*,则添加到 R 中;然后调用 Path relinking( ) 对解进行改进,生成的新解再通过 Improve( ) 程序对解作进一步改进. 改进程序邻域结构为交换邻域,对每类已经选择的物品进行重新选择,使当前目标函数值最大. 改进程序规定当前解必须是可行的,且只在同类物品中进行交换. 在每类 i 物品中,寻找价值系数比较大的物品 j', $j' = \arg\max\limits_{1 \le j \le n_i}\{c_{ij} \mid c_{ij} > c_{i,\sigma_r(i)}\}$. 如果存在满足条件的物品,则用该物品替换原来解中的物品, $\sigma_r(i) = j$;否则,继续对下一个类物品进行新的搜索,直到所有类的物品处理结束. Improve( ) 程序的输入参数是一个可行解,而输出参数是一个改进的可行解,算法的复杂性为 $O(\vartheta^2 \cdot m \cdot n)$,其中 $\vartheta = \max\{n_1, \cdots, n_n\}$. ACO &SS 算法步骤如下:

算法 2(ACO &SS 算法)

Step1: 初始化一些参数: τ_0, β, $|R|$, b_1, b_2, count, $N_{\max}$.

Step2: 调用 Construct( ),产生解 σ_S. 如果解 σ_S 不可行,则调用修复程序 Repair( );否则,转 Step3.

Step3: 检查 σ_S 是否满足 R 更新条件 R^*,如果满足条件,则 $R = R \cup \{\sigma_S\}$.

Step4: 计算函数 $f(\sigma_S)$ 和 $f(\sigma^*)$,如果满足条件 $f(\sigma_S) > f(\sigma^*)$,则更新当前最好解 $\sigma^* = \sigma_S$, $f(\sigma^*) = f(\sigma_S)$;否则,转 Step5.

Step5: 改进程序, σ_r = pathrelinking(R, σ^*), σ_r^* = Improve(σ_r). 计算目标函数 $f(\sigma_r^*)$ 和 $f(\sigma^*)$,

如果满足条件 $f(\sigma_r^*) > f(\sigma^*)$,则 $\sigma^* = \sigma_r^*$, $f(\sigma^*) = f(\sigma_r^*)$;否则,转 Step6.

Step6: count = count + 1. 重复 Step2 ~ Step5,直到 count > N_{max},输出最终结果.

4 实例求解与分析

ACO &PR 算法采用 C ++ 编程,在 512 RAM, 1600 MHz CPU 速度的 IBM 计算机运行. 测试实例采用 MMKPLIB 中的典型例子,地址为 http://mscmga.ms.ic.ac.uk/jeb/orlib/MMKP.html. ACO &PR 算法中存在一些参数,这些参数值直接或间接影响算法性能. 由于 MMKP 较小规模的实例已经证明能获得最优解,一些参数以实例 I01 为例进行测试. 为测试 β 值对解的影响, β 取不同值,经 10 次仿真实验,记录每次实验在迭代不同次数时所获得的最好解的平均值. 实验测定 $\beta \in \{4,5,6\}$ 时能获得比较好的结果. 采用同样方法测试 $q_0 \in [0.50, 0.75]$, $\rho \in [0.3, 0.5]$. 其他参数设置为 $|R| = 10$, $b_1 = b_2 = 5$, $\tau_0 = 0.01$,迭代次数 $N_{max} = 1000$.

表 1 为 ACO &PR 算法与 ACO 算法的运行结果比较. 因为 ACO &PR 算法在线路重连过程中,向导解的属性逐步引入到起始解属性,可快速获得该线路上的最优解. 因此,ACO &PR 算法运行效果比 ACO 算法好,平均改进值为 2.01%. 但是,ACO &PR 算法在线路重连生成新解过程以及参考集更新等需消耗一些时间,所以运行时间比 ACO 算法长.

表 1 蚁群算法与 ACO &PR 算法寻优效果对比

实例	ACO		ACO &PR		改进值 / %
	最大值	时间 / s	最大值	时间 / s	
I01	169.00	0.02	173.00	0.03	2.31
I02	353.00	0.02	364.00	0.03	3.02
I03	1521.00	0.04	1556.00	0.07	2.25
I04	3383.20	0.10	3452.00	0.16	2.00
I05	3800.10	0.12	3905.70	0.25	2.71
I06	4723.40	0.15	4799.30	0.35	1.58
I07	23600.00	0.24	23938.24	0.54	1.41
I08	35405.00	0.30	35997.00	0.70	1.64
I09	47225.00	0.35	47928.00	1.15	1.46
I10	58824.34	0.49	59846.00	1.34	1.70

表 2 为 ACO &PR 算法与其他启发式算法的仿真结果. 表 2 中最好解从实例库 MMKPLIB[13] 中获得,Moser 算法的数据均来源于文献[14],ACO 和 ACO &PR 分别为蚁群算法和混合算法. 目前,关于 MMKP 问题的研究,只有小规模实例已经证明能获得最优解. KLMA 算法[15] 与 Moser 算法是比较经典的两个启发式算法. 许多 MMKP 问题的算法研究,除与当前最好解比较外,还与 KLMA 和 Moser 算法进行比较,以评价算法的性能. 实验表明,ACO 算法优于 Moser 方法,KLMA 算法比 ACO 算法性能好,而 ACO &PR 结果优于 KLMA 算法和 Moser 方法,获得了较好的结果. 其中 I01, I02, I05 和 I06 实例都能找到最优解,ACO &PR 算法与当前最好解平均偏差为 0.79%.

表 2 ACO &PR 算法与其他启发式算法寻优效果对比

实例	最好解	KLMA	MOSER	ACO	ACO &PR
I01	173.00	167.00	151.00	169.00	173.00
I02	364.00	354.00	291.00	353.00	364.00
I03	1602.00	1533.00	1464.00	1521.00	1556.00
I04	3597.00	3437.00	3375.00	3383.20	3452.00
I05	3905.70	3899.10	3905.70	3800.10	3905.70
I06	4799.30	4799.30	4115.20	4723.40	4799.30
I07	23983.00	23912.00	23556.00	23600.00	23938.24
I08	36007.00	35979.00	35373.00	35405.00	35997.00
I09	48048.00	47901.00	47205.00	47225.00	47928.00
I10	60176.00	59811.00	58648.00	58824.34	59846.00

根据实验的仿真结果,可得到以下结论:

1) ACO &PR 算法运行效果比 ACO 算法好,平均改进值为 2.01%.

2) ACO &PR 算法能提高全局搜索能力. 从算法的最终结果看,ACO &PR 算法在解决 MMKP 问题时具有一定的优势,与当前最好解的平均偏差只有 0.79%.

3) 起始解和向导解选择规则是 ACO &PR 算法的关键因素之一,考虑解分散度与解质量的选择规则是一种有效方式.

4) 搜索邻域空间随着问题规模的增大而增大, ACO &PR 算法搜索消耗的时间会越来越长.

5 结 论

本文以研究 MMKP 问题为例,提出了一种新型 ACO &PR 混合算法,该算法将线路重连算法嵌入蚁群算法中. 采用这种混合算法,可以避免蚁群算法陷入局部最优,提高算法的全局寻优能力. 在算法设计过程中,对起始解和向导解选择规则、参考集动态更新方式、不可行解修复策略以及解的改进策略等均进行了较为深入的讨论. 实验结果验证了所设计算法的实用性和有效性,从而为 MMKP 问题的研究提供了一种新的有效方法.

参考文献(References)

[1] Freville A. The multidimensional 0-1 knapsack problem: An overview [J]. European J of Operational

Research, 2004, 155(9): 1-21.

[2] Akbar M M, Manning E G, Shoja G C, et al. Heuristic solutions for the multiple-choice multi-dimension knapsack problem [J]. Lecture Notes in Computer Science, 2001, 2074(2): 659-668.

[3] Chu P C, Beasley J E. A genetic algorithm for the multidimensional knapsack problem[J]. J of Heuristics, 1998, 4(1): 63-86.

[4] 贺毅朝, 寇应展, 陈致明. 求解多选择背包问题的改进差分演化算法[J]. 小型微型计算机系统, 2007, 28(9): 1682-1685.
(He Y C, Kou Y Z, Chen Z M. A modified differential evolution algorithm for multiple-choice knapsack problem[J]. J of Chinese Computer Systems, 2007, 28(9): 1682-1685.)

[5] Dyer M E, Riha W O, Walker J. A hybrid dynamic programming/branch-and-bound algorithm for the multiple-choice knapsack problem [J]. J of Computational and Applied Mathematics, 1995, 58(11): 43-54.

[6] Sbihi A. A best first search exact algorithm for the multiple-choice multidimensional knapsack problem[J]. J Combinatorial Optimization, 2007, 13(4): 337-351.

[7] Hernandez R P, Dimopoulos N J. A new heuristic for solving the multichoice multidimensional knapsack problem [J]. IEEE Trans on Systems, Man and Cybernetics — Part A: Systems and Humans, 2005, 35(5): 708-717.

[8] Hifi M, Michrafy M, Sbihi A. Heuristic algorithms for the multiple-choice multidimensional knapsack problem [J]. J of the Operational Research Society, 2004, 55(12): 1323-1332.

[9] Dorigo M, Maniezzo V, Colorni A. The ant system: Optimization by a colony of cooperating agents [J]. IEEE Trans on Systems, Man and Cybernetics — Part B, 1996, 26(1): 1-13.

[10] Dorigo M, Gambardella L M. Ant colonies for the traveling salesman problem[J]. BioSystems, 1997, 43(2): 73-81.

[11] Glover F, Laguna M. Modern heuristic techniques for combinatorial problems [M]. Oxford: Blackwell Scientific Publishing, 1993: 70-150.

[12] Glover F, Laguna M, Marti R. Fundamentals of scatter search and path relinking [J]. Control and Cybernetics, 2000, 39(3): 653-684.

[13] Ho S C, Gendreau M. Path relinking for the vehicle routing problem[J]. J of Heuristics, 2006, 12(1/2): 55-72.

[14] Moser M, Jokanovic D P, Shiratori N. An algorithm for the multidimesional multiple-choice knapsack problem[J]. IEICE Trans on Fundamentals Electron, 1997, 80(3): 582-589.

[15] Khan S, Li K F, Manning E G, et al. Solving the knapsack problem for adaptive multimedia systems[J]. Stud Inform, 2002, 2(1): 154-174.

(上接第 728 页)

[11] 朱建军. 群决策中两类不确定偏好信息的集结方法研究[J]. 控制与决策, 2006, 21(8): 889-892.
(Zhu J J. Group aggregation approach of two kinds of uncertain preference information [J]. Control and Decision, 2006, 21(8): 889-892.)

[12] 肖四汉, 樊治平, 王梦光. Fuzzy 判断矩阵的一致性研究[J]. 系统工程学报, 2001, 16(2): 142-145.
(Xiao S H, Fan Z P, Wang M G. Study on consistency of fuzzy judgement matrix [J]. J of Systems Engineering, 2001, 16(2): 142-145.)

[13] Chiclana F, Herrera F, Herrera-Viedma E. Integrating three representation models in fuzzy multipurpose decision making based on fuzzy preference relations[J]. Fuzzy Sets and Systems, 1998, 97(1): 33-48.

[14] 徐泽水. 互反和互补判断矩阵的转换关系及其集成排序[J]. 系统工程与电子技术, 2002, 24(10): 60-63.
(Xu Z S. Transformation relations between reciprocal and complementary judgement matrices and their integrated prioritization approaches [J]. Systems Engineering and Electronics, 2002, 24(10): 60-63.)

[15] Aguarón J, Moreno-Jiménez J M. The geometric consistency index: Approximated thresholds [J]. European J of Operational Research 2003, 147(1): 137-145.

[16] Grawford G, Williams C. A note on the analysis of subjective judgments matrices[J]. J of Mathematical Psychology, 1985, 29(4): 387-405.

[17] 徐泽水. 区间数互补判断矩阵排序的一种实用发方法[J]. 运筹与管理, 2001, 10(1): 16-19.
(Xu Z S. A practical method for priority of interval number complementary judgement matrix [J]. Operations Research and Management Science, 2001, 10(1): 16-19.)