一种新的求解多维背包问题的分散算法*

张晓霞, 刘 哲
(辽宁科技大学 软件学院, 辽宁 鞍山 114051)

摘 要: 为了避免蚁群算法在优化搜索过程中易陷入局部最优和早熟收敛,提出一种求解多维背包问题的新型分散搜索算法。该算法是把蚁群算法的构解方法引入到分散搜索算法中,在搜索过程中,既考虑解的质量,又考虑解的分散性。同时,该分散算法还采用了动态更新参考集与阈值接收算法的阈值参数,以控制搜索空间来加快收敛速度。通过选取国际通用 MDKP 实例库中的多个实例进行测试表明,该算法可以避免陷入局部最优解,能提高全局寻优能力,其结果优于其他现有的方法,并获得了较好的结果。

关键词: 多维背包问题; 蚁群优化; 分散搜索; 参考集
中图分类号: TP18 **文献标志码**: A **文章编号**: 1001-3695(2012)05-1716-04
doi:10.3969/j.issn.1001-3695.2012.05.030

New scatter search algorithm for multidimensional knapsack problem

ZHANG Xiao-xia, LIU Zhe
(*School of Software*, *University of Science & Technology Liaoning*, *Anshan Liaoning* 114051, *China*)

Abstract: To avoid falling into local optimal solution and the premature convergence in the process of searching optimization solutions, this paper presented a new scatter search algorithm. The designed algorithm combined the solution construction mechanism of ant colony optimization(ACO) into scatter search(SS). It considered both solution quality and diversification. Simultaneously, the algorithm adopted the dynamic updating strategy and the parameter criterion of threshold accepting to accelerate the convergence towards high-quality regions of the search space. The performance of the proposed algorithm was tested on MDKP benchmark problems from the OR Library. The experimental results show that the proposed algorithm can avoid trapping in local optimal solution and enhance the ability of global optimization, and it is competitive to solve the multidimensional knapsack problem compared with the other heuristic methods in terms of solution quality.

Key words: multidimensional knapsack problem(MDKP); ant colony optimization(ACO); scatter search(SS); reference set

0 引言

多维背包问题是运筹学领域中一个经典整数规划问题,有着广泛的实际应用背景,如管理中的资源分配、资金预算[1]、分布式的数据库处理[2]等,对其求解方法的研究无论是在理论上还是实践中都具有一定的意义。

求解多维背包问题主要有精确算法和启发式算法两大类。因为精确算法的时间复杂性都是呈指数增长的,所以精确算法只能求解规模相对较小的问题,对大规模问题依赖智能优化算法解决,常见的算法有模拟退火、遗传算法、蚁群算法等。由于智能优化算法是模拟生物自然现象而提出的新型算法,具有框架灵活、与问题本身无关等优点,并获得了比较广泛的应用研究。Drexl[3]提出了模拟退火算法,实验测试一些实例能发现最优解。Li 等人[4]提出了禁忌搜索算法,并获得比较好的效果。Hanafi 等人[5]提出了一种基于振荡策略与代理约束的新型禁忌搜索算法,其实验表明,振荡策略是算法收敛性与分散性的一种有效均衡,但是解决大规模的问题计算时间比较长。Thiel 等人[6]提出标准的遗传算法(GA),该算法只能解决小规模的问题,解决大规模问题效果不好。Chu 等人[7]成功给出了 GA,该算法是限制搜索可行的范围,在合理的运行时间里,该算法优于几种启发式方法。许多研究者围绕遗传算法研究,并取得很好的效果,而分散搜索算法是在 GA 基础上发展起来的。

本文是以 MDKP 问题为研究对象,设计了一种混合分散搜索(ACO&SS)算法。该算法具有如下特点:a)ACO&SS 算法在搜索过程中,既考虑解的质量,又考虑解的分散性;b)ACO&SS 算法采用一种新型子集组合成新解的构解机制,即把蚁群算法的信息素更新技术与分散搜索的组合机制相结合而生成新的解;c)信息素更新策略是影响算法收敛速度的关键因素之一,除了采用信息素局部与全局更新外,还采用组合解公共物品临时信息素更新机制;d)根据问题特点,提出采用动态更新参考集合策略,基于阈值接收算法(TA)的局部搜索用来改进算法性能,从而加快分散搜索算法收敛速度。

1 多维背包问题的描述

MDKP 问题就是给出了 m 种资源,各种资源的最大提供数量是 $b_i(i=1,\cdots,m)$。现将 n 种物品装入背包,每个物品 j $(j=1,\cdots,n)$所需要的资源为 a_{ij},受益值是 c_j。MDKP 问题的数学模型如下:

收稿日期: 2011-09-10; **修回日期**: 2011-11-04 **基金项目**: 辽宁省教育厅资助项目(L2010196)
作者简介: 张晓霞(1966-),女,副教授,博士,主要研究方向为智能优化算法、组合优化问题(aszhangxx@163.com);刘哲,硕士研究生.

$$\text{maximize} \quad f(x)=\sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\sum_{j=1}^{n} a_{ij} x_j \leqslant b_i \quad i=1,\cdots,m \tag{2}$$

$$x_j \in \{0,1\} \quad j=1,\cdots,n \tag{3}$$

约束式(2)为背包的容量约束;约束式(3)表示 x_j 为 0-1 决策变量,当第 j 个物品放入背包时,$x_j=1$,否则 $x_j=0$。MDKP 的解实际是一个子集选取问题,就是从物品集合中选出一个合适子集,并且满足背包的能力约束,使目标函数值最大。

2　算法设计

2.1　蚁群算法的基本原理

ACO 算法[8]是模拟蚁群选择最短路径觅食行为而设计的。ACO 算法研究主要是围绕排序问题,如旅行商问题;而 MDKP 实际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排序问题考虑集合中元素之间的顺序,而子集选取问题不考虑子集元素之间顺序。ACO 算法解决子集选取问题因为不考虑子集元素之间的顺序,蚂蚁移动下一个位置与所在的当前位置无关。再有,排序问题蚁群的信息素留在连接两个点的路径上,而子集选取问题信息素留在子集的元素上[9],即用点代替排序问题的路径。MDKP 问题是根据蚁群滞留在收益大且占用资源少的物品上的信息素浓度较高,选择该物品装入背包的概率也就越大。下面介绍 ACO 算法解决 MDKP 问题的过程。

用 ACO 算法解决 MDKP 问题,将要选择的物品与它前面选择的物品无关,但与背包各种资源剩余的容量有关;选择物品时,不仅需要考虑物品的价值,还要考虑该物品占用各个资源的情况。因此,用蚁群算法解决 MDKP 问题比旅行商更复杂一些。将蚁群随机分布在物品上,每个蚂蚁模拟一个背包。蚂蚁 k 选择 j 物品, 其对应的转移概率可定义为

$$P_j^k(t)=\begin{cases}\dfrac{[\tau_j(t)]^{\alpha}[\eta_j(S^k(t))]^{\beta}}{\sum\limits_{u\notin S^k(t)}[\tau_u(t)]^{\alpha}[\eta_u(S^k(t)]^{\beta}} & j\notin S^k(t)\\ 0 & \text{otherwise}\end{cases} \tag{4}$$

其中:τ_j 表示物品 j 的信息素;α 与 β 是参数,它表示信息素与资源的相对重要程度;$\eta_j(S_k(t))$ 表示物品 j 收益值与占用资源比例关系参数,它的定义为

$$\eta_j(S^k(t))=\frac{c_j}{\sum_{i=1}^{m} a_{ij}/\gamma_i^k(t)} \tag{5}$$

其中:$\gamma_i^k(t)=b_i-\sum_{l\in S^k(t)} a_{ij}$ 为背包的剩余容量;$\sum_{l\in S^k(t)} a_{ij}$ 为已经选择物品所占用的容量;$S^k(t)$ 为已经选择的物品。在蚁群算法中,蚂蚁按式(6)选择下一个物品:

$$j=\begin{cases}\underset{l\notin S^k(t)}{\operatorname{argmax}}\{[\tau_l(t)]^{\alpha}[\eta_l(S^k(t))]^{\beta}\} & \text{if } q\leqslant q_0\\ M & \text{otherwise}\end{cases} \tag{6}$$

其中:q_0 为在[0,1]的一个参数;q 为均匀分布在[0,1]的随机变量,如果满足 $q>q_0$ 条件时,使用随机选择的方式;M 是根据式(4)概率选择规则决定的随机变量。随着搜索的进行,滞留在不同物品上的信息素浓度是不同的,蚁群就是根据信息素浓度不同不断选择物品。用 ACO 算法解决 MDKP 问题,信息素更新策略采用局部更新与全局更新两种。局部更新避免因频繁选择相同物品陷入局部最优,而全局更新是使搜索往全局最好解方向移动。蚁群算法虽然采用确定性选择与随机选择相结合的策略,当算法迭代到一定代数后,受益比较大的物品上的信息素不断加强,选择这些物品装入背包的可能性也越大,容易陷入局部最优。

2.2　分散搜索算法

分散搜索算法[10]是一种基于种群的进化算法,近几年才在组合最优化问题上取得很大进展。SS 算法原理为:首先产生初始种群,在初始种群中选择一些高质量解和比较分散的解作为参考集,再把这些解通过组合机制产生新解集合,通过局域搜索算法对新解进行改进,获得局部最优解,更新参考集,以进行新的搜索。SS 算法与 GA 最明显的区别是 GA 初始种群数目比较多,随机抽样选取而产生新的组合;而 SS 算法参考集中解的数目相对较少(20 个左右),用系统方法选择两个或多个解进行组合产生新解。因此,SS 算法既考虑解的质量,同时又考虑解的分散性。

2.3　ACO&SS 算法

SS 算法主要由初始解的生成、参考集合的生成、组合生成新解、解的改进、参考集更新五部分组成。图 1 给出了 SS 算法的结构示意,白色与黑色小圆分别代表初始解与改进解。ACO&SS 算法主要是把 ACO 构解机制嵌到 SS 算法框架结构中。因 SS 保持搜索在大范围内进行,增强了 ACO 算法跳出局部最优能力。

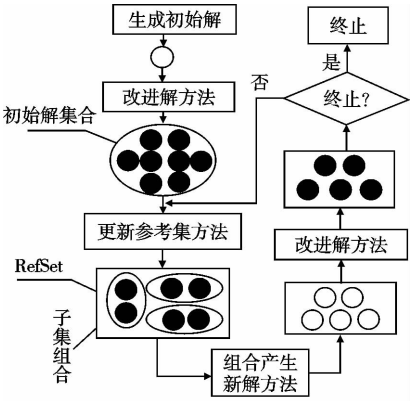

图1　分散搜索算法的结构示意

定义 1　参考集 RefSet 是一些可行解的集合,是由 RefSet_1 集合与 RefSet_2 集合组成。RefSet_1 中有 b_1 个高质量的解,RefSet_2 由 b_2 个分散性好的解组成。

定义 2　假设 $s^*(v_i^*)$ 为参考集合 RefSet 中的一个解,$s_j(v_i)$ 为非参考集合 non-RefSet 中的一个解,设 $\text{NUM}(s_j,s^*)$ 为两个解选择相同物品的数目,表示这两个解的相似度。$\text{NUM}(s_j,s^*)$ 值越大,这两个解就越相似;反之,$\text{NUM}(s_j,s^*)$ 值越小,这两个解的分散性就大。

定义 3　对于非参考集中的任意一个解 s_j,$\text{NUM}_{\max}(s_j)$ 表示该解在 RefSet 中的相似度,定义为

$$\text{NUM}_{\max}(s_j)=\max_{s^*\in \text{RefSet}}\{\text{NUM}(s_j,s^*)\} \quad s_j\in \text{non-RefSet}$$

$\text{NUM}_{\max}(s_j)$ 值最小的解也就是非参考集合中分散性最好的解。

分散搜索算法每部分根据具体情况均可作出灵活的设计,下面重点介绍 ACO&SS 算法中参考集合生成、解改进部分与参考集合更新部分,它们是算法的核心部分,也是本文的创新

部分。

2.3.1 参考集合生成方法

初始参考集合是从初始解集合中选择而得到的,本文采用 ACO 算法与启发式算法两种方法产生初始解集合。用 ACO 算法构造解,其关键是确定哪些物品装入背包、在构解过程中如何搜索到这些物品。用 ACO 算法解决 MDKP 问题时,每个蚂蚁模拟一个背包,首先确定将要装入背包的物品集合 N,在这个集合中按照式(4)和(6)的蚂蚁移动规则确定选择的物品 j,计算背包里物品所占资源的数量,重复上述过程直到没有满足资源约束条件的物品装入,并记录该解。启发式算法是随机从未选择物品集合 T 中选择物品 j,检查物品 j 是否满足资源约束条件,如果满足条件,则把该物品 j 添加到背包里,继续检查未选择物品集合 T,直到 T 集合中没有符合条件的物品而结束。

2.3.2 解的改进

参考集合中的解通过子集组合产生新解,然后用改进程序对解进一步的改进。改进程序邻域结构为插入邻域与交换邻域,对背包已经选择的物品进行添加或交换,使当前目标函数值最大。改进程序分为两个阶段:第一阶段是添加物品,把未选择的物品加入到背包中;第二阶段为交换物品,选中物品与未选中物品是否可以交换。改进程序在调用时,当前解必须是可行的。通常在进行添加与交换移动之前,需对该移动的物品是否为改进目标函数进行判断,只有目标函数改进时才能接受该移动。本研究是把阈值接收算法与邻域结合起来,用阈值控制参数控制搜索空间与移动方向。

TA 是一种邻域搜索寻优算法[11],也是对模拟退火算法(SA)的一种改进算法。TA 从某初始解 s 开始在其邻域 $N(s)$ 内搜索,采取预先设置阈值控制接受邻域移动,允许接受一定质量不高的解,用阈值控制使搜索过程向最优值方向移动,实现全局寻优机制。TA 通过阈值控制算法的搜索速度和解质量,以提高较大阈值下降速率,同时适当在小范围增加阈值,避免陷入局部最优。

2.3.3 参考集合的更新

参考集更新方法通常有静态和动态更新两种方式,本文的分散搜索算法设计采用动态更新参考集方法。根据两个策略进行更新,一个策略是考虑解的质量,另一个策略是考虑解的分散性。设 s_j 为子集组合生成的任意一个新解,s_i 是 RefSet_1 里的任意一个解,如果 $f(s_j) > \min_{i \in \mathrm{RefSet1}} \{f(s_i)\}$ 条件满足时,新解 s_j 就可以更新 RefSet_1 集合中中目标函数值最小的解;如果新解的 $\mathrm{NUM}_{\max}(s_j)$ 值比较小(也就是分散性好),并且比 RefSet_2 集合中 $\mathrm{NUM}_{\max}(s_i)$ 最大值小,该新解就添加到 RefSet_2 集合里,并从 RefSet_2 集合中删除 $\mathrm{NUM}_{\max}(s_i)$ 值最大的解。这种动态更新参考集方法既保证了参考集中解的质量,同时又考虑了解的分散性,可以加快算法的收敛速度。

2.3.4 ACO&SS 算法步骤

a)产生初始解集合,初始化一些基本参数,count = 0,k = 0,b = false。将 m 只蚂蚁随机分配到 n 个物品上,出发点放置到禁忌表 S^k,用 ACO 与启发式算法产生不同的初始解,生成初始解集合 P。

b)建立 RefSet,从 P 中选择 b_1 个较好的解加入到 RefSet_1,b_2 个较分散的解,加入到 RefSet_2,然后把这些解添加到参考集合中,RefSet = $\mathrm{RefSet}_1 \cup \mathrm{RefSet}_2$。

c)产生组合子集,每个组合子集由两个解构成,一个解从 RefSet_1 中选取,一个解从 RefSet_2 集合中选取。

d)组合产生新解。根据解的相似性产生公共物品;采用信息素临时更新机制对公共物品上的信息素进行更新;调用 ACO 算法,生成新解 s_j。

e)改进解的质量,$x = s_j$,调用 TA 对解 s_j 进行改进,$s_j = x$,$s^* = x^*$,计算解 s_j 与当前最好解 s^* 目标函数值 $f(s_j)$,$f(s^*)$。

f)参考集合更新,若参考集合已更新,转到步骤 g),否则调用 ACO 程序,产生一个新解,转到步骤 d)。

g)计算与更新参考集合的当前最好解 $f(s^*)$,b = false,count = count + 1,如果 $f(s_j) < f(s^*)$ 时,$f(s^*) = f(s_j)$,转步骤 h)。

h)重复步骤 d)~g),直到所有子集合都处理结束时,转到步骤 i)。

i)$k = k + 1$,若 $k \leqslant \mathrm{NC}_{\max}$,重复步骤 d)~h),直到满足终止条件时,程序结束。

3 实验设计与测试结果

为验证算法的性能,测试实例是从 OR-Library 实例库下载的(http://mscmga. ms. ic. ac. uk/jeb /orlib /mknapinfo. html),算法用 C++编程,并在 512 RAM 的 IBM 计算机上进行测试。测试实例的主要特征是其规模的大小、资源约束数目、物品数量。每种类型的实例由 30 个组成,根据约束条件的严厉度将实例分为 $\alpha = 0.25$、$\alpha = 0.5$ 和 $\alpha = 0.75$ 三个水平。其中,5.100 小规模实例表示包含 5 个约束与 100 个变量,该实例已经证明能获得最优,本文从每个水平选择 1 个实例作为参数测试的实例。这些实例为:5.100.01、5.100.11 和 5.100.21。许多参数都是通过这三个实例测试的。经过测试,$q_0 \in [0.40, 0.80]$,$\beta \in \{3,4,5,6\}$,$\rho \in [0.04, 0.06]$,其它参数设置为 |RefSet| = 10,$\tau_0 = 0.01$ 和最大迭代次数 $\mathrm{NC}_{\max} = 2\,000$。LINGO 优化软件线性求解器用于求解 MDKP 小规模的实例,能获得问题的最优解。然而,对于稍大规模的问题,通过 LINGO 优化软件进行求解 MDKP 需要花费大量的时间,因此有必要用智能优化算法在快速的时间内搜索到问题的近似最优解。

表 1 给出了 ACO&SS 算法与 ACO 算法的运行结果比较,同时也给出了每 10 个实例运行 30 次的目标函数值的平均值和最好解的平均偏差。从表 1 中运行结果可以得出:ACO&SS 算法运行效果比基本 ACO 好,即融合不同搜索策略的算法对解能有很好的改进效果。表 1 也给出了用 LINGO 8.0 软件的运行结果,经过测试发现用该优化软件只能解决小规模的实例,而对于问题规模比较大的实例,本文限制了运行时间,设置运行时间 10 min 为终止准则。

分析表 1 中的实验结果,得出如下结论:

a)从表中得出 ACO&SS 算法比 ACO 算法的运行结果好,但是算法因为在解组合过程中需要花费一些时间,混合分散算法运行时间比基本 ACO 算法长。

b)对于问题规模比较小的实例,如 $m = 5$,$n = 100$ 问题实例,ACO&SS 混合分散算法与 LINGO 软件都能获得最优解或

近似最优解,但是 LINGO 软件花费的平均时间比较多。

表 1　ACO&SS 算法与 ACO 算法计算结果比较

问题			ACO		ACO		ACO	
资源数	物品数	α	time	gap/%	time	gap/%	time	gap/%
5	100	0.25	7.7	0.53	12.6	0.24	56	0.24
		0.50	8.5	0.43	14.4	0.21	58	0.22
		0.75	10.4	0.39	16.5	0.19	62	0.20
5	250	0.25	35.2	0.36	36.2	0.19	600	0.21
		0.50	37.5	0.32	37.6	0.11	600	0.13
		0.75	40.4	0.26	41.4	0.06	600	0.06
10	100	0.25	9.4	1.13	19.5	0.71	351	0.82
		0.50	10.2	0.96	20.2	0.68	402	0.72
		0.75	14.4	0.73	23.8	0.35	450	0.56
10	250	0.25	36.5	0.92	67.6	0.45	600	1.62
		0.50	31.8	0.68	60.5	0.23	600	1.43
		0.75	28.4	0.46	56.2	0.14	600	1.03
30	100	0.25	20.0	1.73	36.7	1.36	600	2.00
		0.50	17.8	0.69	34.6	0.46	600	1.38
		0.75	15.3	0.54	30.5	0.27	600	0.88
30	250	0.25	51.4	1.55	115.2	0.97	600	2.11
		0.50	58.5	0.82	123.6	0.37	600	1.66
		0.75	65.7	0.61	133.5	0.30	600	1.43

c)对于问题规模比较大的实例,用 ACO&SS 算法运行结果比 LINGO 优化软件效果好,ACO&SS 算法与当前最好解的平均偏差为 0.41%,而用 LINGO 优化软件与当前最好解的平均偏差为 0.93%。

表 2 给出了 ACO&SS 算法与其他启发式方法运行结果的比较。其中 M&O 算法是由 Magazine 等人[12]提出的;V&Z 是由 Volgenant 等人[13]设计的;MKHEUR 是由 Pirkul[14]提出的;GA 代表由 Chu 等人[7]设计的遗传算法;ACO&SS 是指本文设计的混合分散算法。ACO&SS 算法在许多实例中都能获得比较好的结果,比其他已有的经典算法运行效果好。

表 2　ACO&SS 算法与其他启发式方法运行结果的比较

问题			averagegap/%				
资源数	物品数	α	M&O	V&Z	MKHEUR	GA	ACO&SS
5	100	0.25	13.69	10.30	1.59	0.99	0.24
		0.50	6.71	6.90	0.77	0.45	0.21
		0.75	5.11	5.68	0.48	0.32	0.19
5	250	0.25	6.64	5.85	0.53	0.23	0.19
		0.50	5.22	4.4	0.24	0.12	0.11
		0.75	3.56	3.59	0.16	0.08	0.06
10	100	0.25	15.88	15.55	3.43	1.56	0.71
		0.50	10.41	10.72	1.84	0.79	0.68
		0.75	6.07	5.67	1.06	0.48	0.35
10	250	0.25	11.73	10.53	1.07	0.51	0.45
		0.50	6.83	5.92	0.57	0.25	0.23
		0.75	4.42	3.77	0.33	0.15	0.14
30	100	0.25	17.39	17.21	9.02	2.91	1.36
		0.50	11.82	10.19	3.51	1.34	0.46
		0.75	6.58	5.92	2.03	0.83	0.27
30	250	0.25	13.54	12.41	3.70	1.19	0.97
		0.50	8.64	7.12	1.53	0.53	0.37
		0.75	4.49	3.91	0.84	0.31	0.30
average			8.82	8.09	1.82	0.72	0.41

4　结束语

本文对 MDKP 进行了研究,提出一种新型 ACO&SS 混合分散算法。采用该混合分散算法,避免了 ACO 算法陷入局部最优,同时提高了 SS 算法的运行性能。在算法设计中,提出采用动态更新策略,基于阈值接收算法的局部搜索来改进算法性能,从而加快算法收敛速度。通过国际通用 OR 数据实例验证可以得出以下结论:

a)ACO&SS 算法可提高全局搜索能力,在解决 MDKP 问题时具有一定的优越性,与最优解的平均偏差只有 0.41%。

b)该混合分散算法采用了动态更新参考集方法与基于阈值接收算法的局部搜索来改进算法性能,既具有精确搜索能力,又具有广域的分散搜索能力,从而提高了算法的全局搜索性能。

c)为 MDKP 问题的研究提供了一种新的有效方法,同时采用分散搜索算法为解决组合最优化问题提供了新的思路。

参考文献:

[1] LIANG R, GAO J W. Dependent-chance programming models for capital budgeting in fuzzy environments [J]. Tsinghua Science and Technology, 2008, 13(1): 117-120.

[2] DAVIS L, SAMANLIOGLU F, JIANG X C, *et al*. A heuristic approach for allocation of data to RFID tags: a data allocation knapsack problem (DAKP) [J]. Computers & Operations Research, 2012, 39(2): 93-104.

[3] DREXL A. A simulated annealing approach to the multiconstraint zero-one knapsack problem [J]. Computing, 1988, 40(1): 1-8.

[4] LI V C, CURRY G L. Solving multidimensional knapsack problems with generalized upper bound constraints using critical event tabu search [J]. Computers & Operations Research, 2005, 32(11): 825-848.

[5] HANAFI S, FREVILLE A. An efficient tabu search approach for the 0-1 multidimensional knapsack problem [J]. European Journal of Operational Research, 1998, 106(2): 659-675.

[6] THIEL J, VOSS S. Some experiences on solving multiconstraint zero-one knapsack problems with genetic algorithms [J]. INFOR, 1994, 32(4): 226-242.

[7] CHU P, BEASLEY J. A genetic algorithm for the multidimensional knapsack problem [J]. Journal of Heuristics, 1998, 4(1): 63-86.

[8] 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.

[9] LEGUIZAMON G, MICHALEWICZ Z. A new version of ant system for subset problem [C]//Proc of Congress on Evolutionary Computation. 1999: 1456-1464.

[10] MARTI M, LAGUNA M, GLOVER F. Principles of scatter search [J]. European Journal of Operational Research, 2006, 169(2): 359-372.

[11] MARIMUTHU S, PONNAMBALAM S G, JAWAHAR N. Threshold accepting and ant-colony optimization algorithms for scheduling m-machine flow shops with lot streaming [J]. Journal of materials processing technology, 2009, 209(2): 1026-1041.

[12] MAGAZINE M J, OGUZ O. A heuristic algorithm for the multidimensional zero-one knapsack problem [J]. European Journal of Operational Research, 1984, 16(3): 319-326.

[13] VOLGENANT A, ZOON J A. An improved heuristic for multidimensional 0-1 knapsack problems [J]. Journal of the Operational Research Society, 1990, 41(10): 963-970.

[14] PIRKUL H. A heuristic solution procedure for the multiconstraint zero-one knapsack problem [J]. Naval Research Logistics, 1987, 34(2): 161-172.