多维背包问题的二进制蚂蚁算法①

孔 民, 田 澎, 李相勇
(上海交通大学管理学院, 上海, 中国 200052)

摘要: 针对著名的多维背包问题 (MKP), 在蚁群优化系统高维立方体结构的基础上, 提出了一种二进制蚂蚁算法 (BAS). 与其他求解 MKP问题的蚂蚁算法不同, BAS根据二进制解的结构设计了特殊的信息素放置方式, 同时在算法的迭代过程中允许非可行解的产生, 并通过基于问题特征信息的修改算子修复每次迭代所产生的非可行解. BAS算法采用了特殊的信息素更新规则, 使得各个选择路径上的信息素可以直接作为选择概率, 同时, 为了避免算法陷入早熟, BAS设计了简单的局部搜索法, 并根据算法所处的不同收敛状况, 采用了不同的信息素更新规划和信息素重新初始化的方法. 针对 MKP基准问题的实验结果表明, BAS具有超越其他蚂蚁算法的求解结果, 其求解不同基准测试问题的能力表明了 BAS具有解决超大规模 MKP问题的潜力.

关键词: 蚁群优化; 二进制蚂蚁算法; 组合优化; 多维背包问题
中图分类号: O22 文献标识码: A 文章编号: 1007- 9807(2009)02- 0044- 10

0 引 言

多维背包问题 (MKP)是个著名的 NP- 难问题, 可以用数学公式描述如下

$$\max \quad f = \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} r_{ij} x_j \leqslant b_i, \quad i = 1, \cdots, m \tag{2}$$

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

公式(2)中包含了 m 个约束, 其中每一个约束条件被称为一个背包约束, 因此多维背包问题也被称为 m - 维背包问题. 设 $\boldsymbol{I} = \{1, \cdots, m\}$, $\boldsymbol{J} = \{1, \cdots, n\}$, 对于所有的 $i \in \boldsymbol{I}$ 有 $b_i \geqslant 0$, 并且对于所有的 $i \in \boldsymbol{I}$, $j \in \boldsymbol{J}$ 有 $r_{ij} \geqslant 0$ 一个定义完好的多维背包问题对于所有的 $i \in \boldsymbol{I}$, $j \in \boldsymbol{J}$, 有 $p_j > 0$ 和 $r_{ij} \leqslant b_i < \sum_{j=1}^{n} r_{ij}$.

多维背包问题可以看作为具有 m 个资源和 n 个项目的资源分配问题. 每个资源 $i \in \boldsymbol{I}$ 的总预算为 b_i, 而每个项目 j 具有的收益为 p_j, 但要消耗一定数量的各种资源, 对于每个资源 i, 项目 j 对于该资源的消耗量为 r_{ij}. 多维背包问题要解决的就是在有限的资源约束条件下选择一定数量的项目, 使得收益最大化.

由于其简单的结构可以用来充分探索组合优化问题的多种性质, 而且许多复杂的优化问题可以转化为一系列的背包问题, 多维背包问题成为了被广泛和深入研究的离散规划问题[1]. 多维背包问题在现实生活中也有广泛的应用背景, 许多实际的应用可以描述为多维背包问题. 例如, 在分布式计算机系统中的处理和数据库的分配问题[2], 项目选择和货物装配问题[3], 以及库存压缩问题[4] 等等.

蚁群优化算法 (ACO) 是近期开发出来的, 以群体智能为基础的现代启发式算法[5, 6]. 自诞生以来, ACO已经被成功地应用到多种 NP- 难的组合优化问题[7]. 例如旅行商问题[8], 二项式分配问题[9, 10], 以及车辆路径问题[11] 等等. ACO 的主人思想是模仿自然界中蚂蚁在路径上留下信息素

① 收稿日期: 2005- 12- 12; 修订日期: 2006- 11- 24.
基金项目: 十一五国家科技支撑计划资助项目 (2006BAF01A44).
作者简介: 孔 民 (1969—), 男, 上海人, 博士, 高级顾问. Email: shanghaikongmin@ 263.net

痕迹的方式进行间接的信息交流. 这种信息素痕迹被认为是种分布式的信息, 反映了人工蚂蚁在解决问题时所积累下来的经验.

针对 MKP 的应用问题, Leguizamon 和 Michalewicz[12], Fidanova[13] 以及 Alata等[14] 提出了基于 ACO 的 3种不同的蚂蚁算法. 这 3种算法的主要区别在于信息素的放置方式. Leguizamon和 Michalewicz将信息素放置在每一个项目变量上; Fidanova将信息素放置在已经成功选择的项目变量之间; 而 Alaya等则将信息素放置在每一对项目变量之间. 这 3种算法共同的特点是都采用了动态的局部启发式值以及在每一个解的构筑过程中利用约束条件来指导蚂蚁的路径探索以保证解的可行性.

与上述 3种蚂蚁算法不同, 本文提出的 BAS 针对 MKP二进制解的结构设计了特殊的信息素放置方式, 在解的构筑过程中不采用局部启发式值的信息, 并且允许非可行解的产生, 由此避免了在每一次迭代过程中对局部启发式值的计算以及对约束条件的计算和判断, 从而节省了整个算法的计算时间. 同时, 实验结果表明, BAS在提高运算速度的前提下, 仍然能够求得好于其他蚁群优化算法的解.

1　BAS应用到MKP

BAS是在蚁群优化算法高维立方体结构(HCF)[15] 的基础上, 利用背包问题特征信息发展而来的, 解决约束优化问题的二进制蚂蚁算法. BAS沿袭了蚁群优化算法的传统结构, 其算法流程见图 1. 与以往蚁群优化算法不同的是, BAS增加了修改算子以修复非可行解, 将局部搜索, 信息素重新初始化等元素融合进了算法过程, 并根据算法的收敛状况采用了不同的信息素更新策略. 以下将详细描述 BAS的算法流程.

1.1　解的表达方式和构筑过程

BAS的算法模型是专门为二进制的解空间设计的. 在 BAS中, 潜在的解 $x = \{x_1, \cdots, x_n\} \in \{0, 1\}^n$ 可以用图 2表示. 在应用到 MKP问题时, $x_j = 0$表示项目 j没有被选取, 而 $x_j = 1$表示项目 j被选中. 按照图 2所表示的这样的解有可能是非可行解. 非可行解指的是至少一个背包约束未被满足, 即对于某个 $i \in I$, 有 $\sum_{j=1}^{n} r_{ij} x_j > b_i$.

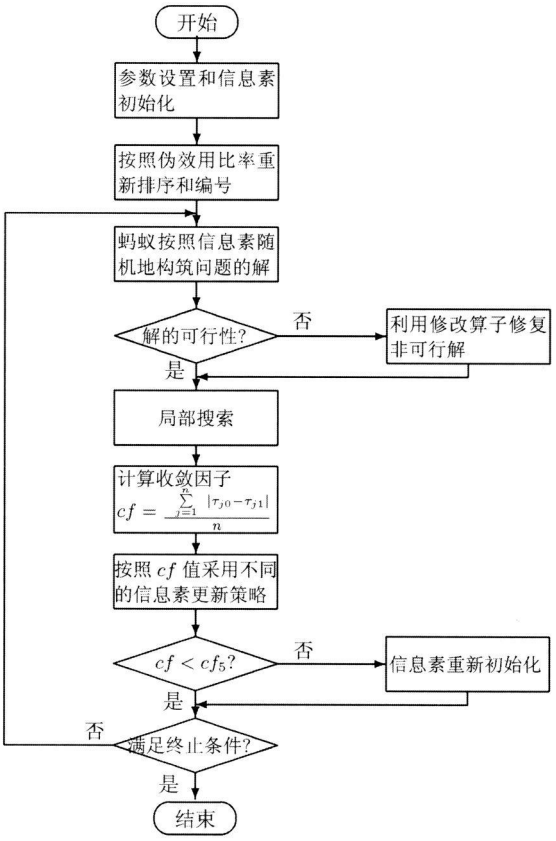

图 1　BAS 的算法流程

Fig. 1 Algorithm process of BAS

j	1	2	3	4	5	…	$n-1$	n
x_j	0	1	0	0	1	…	0	1

图 2　二进制蚂蚁算法解的结构

Fig. 2 Solution structure of binary ant algorithm

为了说明 BAS在 MKP问题中解的构筑过程, 本文将 MKP 问题描述成如图 3所示的权重图 $G = (C, L, \Gamma)$, 其中 C 表示各个项目变量 $j = 1, \cdots, n$, 以及 $n+1$(表示结束点)的集合. 相邻两个项目变量 j和 $j+1$之间有上下两条路径, 分别称为 j_0和 j_1, L 是这些路径的集合. Γ 是各条路径上信息素的集合, 路径上的信息素用 τ_{js}, $s \in \{0, 1\}$ 表示.

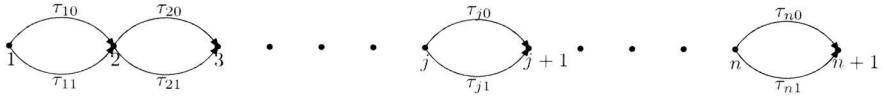

图 3　BAS中蚂蚁的行走路线图

Fig. 3 Routing diagram of ants in BAS

在 BAS中, 一群数量为 n_a 的人工蚂蚁通过共同协作来寻求问题的解, 每个人工蚂蚁通过在 MKP表示图上从节点 1顺序走到节点 $n+1$ 来完成一个解的构筑过程. 在解的构筑过程中, 人工蚂蚁在项目变量 j 表示的点上, 根据上下两条路径上信息素的分布情况, 随机地选择一条路径到达下一个节点 $j+1$. 选择上面的路径表示 $x_j = 0$, 选择下面的路径则表示 $x_j = 1$. 其选择概率为

$$p_{js}(t) = \frac{\tau_{js}(t)}{\tau_{j0}(t) + \tau_{j1}(t)}, \quad j = 1, \cdots, n, \quad s \in \{0, 1\} \tag{4}$$

其中 t 表示算法的迭代次数. 算法初始时, 所有路径上的信息素都被设置为初始值 $\tau_0 = 0.5$, 因此蚂蚁具有相同的概率选择上下两条路径. 随着算法的运行, 路径上的信息素分布将发生变化, 某些路径上的信息素会得到增强, 而另外一些路径上的信息素将逐渐地减弱, 因此蚂蚁的路径选择将逐渐地更具有倾向性. 由于 BAS采用的信息素更新法则能够保证每一对路径 $j0$ 和 $j1$ 上的信息素具有 $0 < \tau_{j0}, \tau_{j1} < 1$ 以及 $\tau_{j0} + \tau_{j1} = 1$ 的性质, 也就是说, 各个路径上的信息素可以直接代表选择概率, 则公式 (4) 所表示的选择概率可以简化为:

$$p_{js}(t) = \tau_{js}(t), \quad j = 1, \cdots, n, \quad s \in \{0, 1\} \tag{5}$$

1.2 伪效用比率和修改算子

在所有的蚂蚁完成了一次解的构筑过程后, BAS检查在这一次迭代过程中产生的解的可行性, 并利用修改算子将非可行解转化为可行解, 使之满足各个约束条件. 本文所采用的修改算子是基于伪效用比率的贪婪性算法.

1.2.1 伪效用比率 (pseudo utility ratio)

在算法的初始化部分, BAS根据伪效用比率的递减次序将各个项目变量进行重新排序和编号. 利用 Pirkul[16] 介绍的替代对偶方法 (surrogate duality approach) 来求解伪效用比率. 下面简要介绍这一方法.

为了求得伪效用比率, 首先将原 MKP问题转化为 MKP问题的替代松弛问题. MKP问题的替代松弛问题可以定义为

$$\max f = \sum_{j=1}^{n} p_j x_j \tag{6}$$

$$\text{s.t.} \sum_{i=1}^{n} \left(\sum_{i=1}^{m} \omega_i r_{ij} \right) x_j \leqslant \sum_{i=1}^{m} \omega_i b_i \tag{7}$$

$$x_j \in \{0, 1\}, \quad j = 1, 2, \cdots, n \tag{8}$$

其中 $\omega = \{\omega_1, \cdots, \omega_m\}$ 称为替代权重 (surrogate weight), 通常取正的实数. 比较简单的求解替代权重的方式是求解原 MKP问题的 LP松弛问题, 并把求得的对偶问题的变量值作为替代权重的值. 也就是说, 替代权重 ω_i 可以看成是 MKP问题经过 LP松弛后的第 i 个约束条件的影子价格. LP算法现在已经非常成熟, 利用 Matlab中的 linprog 函数在几秒钟之内就能够得到替代权重的值.

在求得替代权重以后, 可以通过下式来得到各个项目变量的伪效用比率 u_j

$$u_j = \frac{p_j}{\sum_{i=1}^{m} \omega_i r_{ij}} \tag{9}$$

1.2.2 修改算子

本文采用了 Chu和 Beasley[17] 所介绍的修改算子, 是在利用伪效用比率将各个项目变量重新排序的基础上所设计的贪婪性算法. 图 4描述了这种修改算子的伪代码.

```
输入一个非可行解 X = {x_1, ..., x_n}
计算 R_i = Σ_{j=1}^{n} r_ij x_j, ∀ i ∈ I
j = n
While(R_i > b_i, for any i ∈ I) do /* 丢弃步骤 */
if x_j = 1 then
x_j = 0, R_i = R_i - r_ij, ∀ i ∈ I
end if
j = j - 1;
endwhile
for j = 1 to n do /* 增加步骤 */
if x_j = 0 and R_i + r_ij < b_i, ∀ i ∈ I then
x_j = 1; R_i = R_i + r_ij, ∀ i ∈ I;
end if
end for
输出可行解 X
```

图 4　修改算子的伪代码

Fig. 4 Pseudo code of repair operator

修改算子包括两个步骤. 第 1个步骤称为丢弃步骤, 按照伪效用比率 u_j 的递增顺序检查非可行解中的所有项目变量. 如果项目变量的取值为 1并且目前的解未能满足约束条件, 则丢弃该项目变量, 即将该项目变量赋值为 0, 并释放其占用的资源. 重复这一过程直至得到可行解. 第 2个步骤称为增加步骤, 是第 1个步骤的逆过程, 按照 u_j

的递减顺序检查所有的项目变量. 如果项目变量的取值为 0并且增加这一项目变量并不违反约束条件的话, 则将这一项目变量赋值为 1, 并改变剩余的资源数量值. 重复这一过程直至所有的项目变量都被检查到.

1.3　局部搜索法

局部搜索法在蚁群优化算法中得到了广泛的应用, 并已经被证明是非常有效的改进解的质量的方法[5, 6]. 在 MKP的实验过程中, 设计了非常简单有效的局部搜索算法: 4变量随机翻转法. 所谓的 4变量随机翻转法就是随机地选取 4个项目变量进行翻转, 检查得到的解的可行性, 必要的话采用修改算子进行修复. 检验翻转后的解是否好于原来的解, 如果是的话, 则取代原来的解, 否则, 恢复为原来的解继续下一次翻转, 直到找到更优的解, 或者是总的尝试次数达到 1 000次. BAS在每一次迭代中进行针对全局历史最优解 S^{gb} 和本次迭代最优解 S^{ib} 的 4变量随机翻转法.

1.4　信息素的更新过程

BAS根据算法所处的不同收敛状况采用不同的信息素更新策略. 算法的收敛状况采用收敛因子 cf 进行监测, 定义该收敛因子为

$$cf = \frac{\sum_{j=1}^{n} | \tau_{j0} - \tau_{j1} |}{n} \tag{10}$$

根据 cf 的定义可以看出, 在算法开始时, 由于所有选择路径上的信息素值为 $\tau_0 = 0.5$, 此时 $cf = 0$ 而当算法趋向于收敛到某局部最优解时, 每一对选择路径上的信息素将会趋向于强弱分明, 使得 $| \tau_{j0} - \tau_{j1} | \to 1$, 此时 $cf \to 1$ 因此, BAS算法的收敛过程就相当于 cf 从 0变化到 1的过程.

BAS中信息素的更新过程包括两个部分: 首先是信息素的蒸发过程, 所有路径上的信息素都按照下面的规则进行蒸发

$$\tau_{js}(t+1) \leftarrow (1-\rho)\tau_{js}(t), \quad j = 1, \cdots, n, \ s \in \{0, 1\} \tag{11}$$

其中, ρ 表示信息素的蒸发系数; 随后是信息素的强化过程

$$\tau_{js}(t+1) \leftarrow \rho \sum_{x \in S_{upd} | js \in x} w_x, \quad j = 1, \cdots, n, \ s \in \{0, 1\} \tag{12}$$

其中, S_{upd} 表示需要进行强化的解的集合, $w_x \in (0, 1)$ 是各个解在强化过程中所占的比重, 满足 $\sum_{x \in S_{upd}} w_x = 1$, $js \in x$ 表示解 x 在其构造过程中选择了路径 js

S_{upd} 包含了 3个元素, 它们是:

1) 全局历史最优解 S^{gb}　算法开始后所发现的最优解.

2) 本次迭代最优解 S^{ib}　在本次迭代过程中产生的最优解.

3) 重新初始化后的最优解 S^{rb}　从上一次信息素重新初始化以后所找到的最优解.

BAS根据收敛因子 cf 的大小判断算法的收敛状况, 并据此采用不同的 w_x 组合进行信息素的更新, 以及决定是否进行信息素的重新初始化. 其具体策略如表 1所示. BAS在 cf 的各个阶段采用了 5种不同的信息素更新策略. 表中 w_{ib}, w_{rb}, 和 w_{gb} 分别表示解 S^{ib}, S^{rb} 以及 S^{gb} 在信息素强化过程中的权重, 并且在每个组合中都满足 $w_{ib} + w_{rb} + w_{gb} = 1$ cf_i ($i = 1, \cdots, 5$) 是划分各个阶段的阈值参数.

表 1　信息素更新策略

Table 1 Pheromone update strategy

	w_{ib}	w_{rb}	w_{gb}
$cf < cf_1$	1	0	0
$cf \in [cf_1, cf_2)$	2/3	1/3	0
$cf \in [cf_2, cf_3)$	1/3	2/3	0
$cf \in [cf_3, cf_4)$	0	1	0
$cf \in [cf_4, cf_5)$	0	0	1

当 $cf \geqslant cf_5$ 时, BAS将执行信息素重新初始化. 即将所有的信息素都重新赋值为 τ_0, 并立即进行一次只针对 S^{gb} 的信息素强化过程, 使得算法在探索 S^{gb} 附近搜索空间的同时, 仍然保证具有一定的概率发掘其他更好的解.

1.5　算法中止条件

上述的迭代过程将重复运行, 直至达到某个中止条件, 诸如得到某个可接受的解, 或者算法的迭代次数达到了预定的最大值. 本文采用的算法中止条件是: 如果算法发现了最优解, 或者是总的迭代次数达到了 3 000次.

1.6　BAS与其他蚂蚁算法的区别

如同引言中所述, 在 BAS之前, 已经有一些

蚂蚁算法被成功地应用到了 MKP上[12~14]. BAS与这些蚂蚁算法的主要区别表现为如下 3点.

1)BAS针对二进制解的结构设计了简单有效的信息素放置方法, 而其他的蚂蚁算法都考虑了项目变量之间的相互关系而采用了比较复杂的信息素放置方法.

2) 其他蚂蚁算法都采用了动态启发式值的方法,即在每一次迭代过程中都重新计算各个连接上的启发式值, 以指导人工蚂蚁路径选择的过程. 而BAS在路径选择中并不考虑局部启发式值的信息.

3) 其他蚂蚁算法在蚂蚁路径 (解) 的构筑过程中, 通过计算下一个项目变量是否满足约束条件来指导蚂蚁的路径选择行为, 从而避免产生非可行解. BAS则在解的构筑过程中, 允许非可行解的出现, 然后利用基于问题特征信息的修改算子将非可行解转化为可行解.

由于 BAS摒弃了动态的局部启发式值以及约束条件计算和判断这些非常耗时的计算过程, 因此相对于其他的蚂蚁算法来说, BAS在每一次迭代过程的计算时间得到了缩减. 通过算法的复杂性分析可以看到 BAS在计算速度上的明显优势. 虽然使用修改算子会增加一些计算时间, 但是通过上述对修改算子的描述, 可以知道该修改算子的计算复杂性为 $O(mn)$, 且每一次 BAS迭代过程的计算复杂性亦为 $O(mn)$. 而其他那些需要计算动态的局部启发式值和判断约束条件的蚂蚁算法, 其每一次算法迭代的计算复杂性最少为 $O(mn\ \log mn)$.

2　BAS的收敛性能和参数设置

有效的现代启发式算法应该具有迅速找到最优解的能力, 同时应该具有处理不同特征的问题例子的鲁棒性. 本节将给出在 BAS的算法过程中采用信息素作为选择概率的理论根据, 并讨论 BAS的收敛性能以及针对不同问题的通用参数设置方法.

2 1　信息素作为选择概率

引理 **1**　对于 BAS中任意选择路径上的信息素值 τ_{js}, 满足

$$0 < \tau_{js}(t) < 1 \qquad (13)$$

证明　从上一节中所描述的信息素更新规则可以看出, 所有选择路径上的信息素在任何时候都为正值, 即 $\tau_{js}(t) > 0$ 同时, 由于 $\sum_{x\in S_{upd}} w_x = 1$ 以及 $\tau_{js}(0) < 1$, 可以通过公式 (11) 和 (12) 计算任意路径 js 上信息素的最大可能值为

$$\begin{aligned}\tau_{js}^{max}(t) &\leqslant (1-\rho)\tau_{js}^{max}(t-1)+\rho\\ &\leqslant (1-\rho)^t\tau_{js}(0)+\sum_{i=1}^{t}(1-\rho)^{i-1}\rho\\ &= (1-\rho)^t\tau_{js}(0)+1-(1-\rho)^t\\ &< (1-\rho)^t+1-(1-\rho)^t=1 \qquad (14)\end{aligned}$$

证毕.

定理 **1**　在 BAS的整个算法过程中, 各对选择路径上的信息素可以直接作为选择概率.

证明　初始时, 由于 $\tau_{js}(0)=\tau_0=0.5$, 因此满足上述定理. 在随后的任意迭代时刻 t, 只要证明对于任意一个节点 j 在 $\tau_{j0}(t-1)+\tau_{j1}\times(t-1)=1$成立的条件下, $\tau_{j0}(t)+\tau_{j1}(t)=1$, $0<\tau_{js}(t)<1$ 依然成立, 则上述定理成立.

从引理 1可以看出, 对于任何一个选择路径 (j, s), 存在: $0<\tau_{js}(t)<1$. 在经过信息素更新过程后, 所有选择路径上的信息素都进行了蒸发过程, 但是, 对于任意一个节点 j在针对任意一个 $x\in S_{upd}$ 的信息素强化过程中, 有且只有一个 $j0$ 或者 $j1$得到了信息素的强化. 因此, 对于任意一对信息素 τ_{j0} 和 τ_{j1}, 有:

$$\begin{aligned}\tau_{j0}(t)+\tau_{j1}(t) &= (1-\rho)\tau_{j0}(t-1)+\rho\sum_{x\in S_{upd}|j0\in x} w_x +\\ &\quad (1-\rho)\tau_{j1}(t-1)+\rho\sum_{x\in S_{upd}|j1\in x} w_x\\ &= (1-\rho)(\tau_{j0}(t-1)+\tau_{j1}(t-1))+\\ &\quad \rho\sum_{x\in S_{upd}} w_x\\ &= 1-\rho+\rho=1 \qquad \text{证毕.}\end{aligned}$$

2 2　关于收敛速度的讨论

蚁群优化算法的显著特点之一就是能很快地发现较为满意的解[5, 6], 换句话说, 就是能够很快地进入局部最优解. 在 BAS的实验过程中也同样发现了这个特点. 那么, 关于加快蚁群优化算法收敛速度的主要问题就是如何跳出局部最优解, 或者说是如何增加发现更优解的概率.

在讨论如何跳出局部最优解之前, 在此需要作必要的假设, 使后面的讨论更为简单和直接. 假设 BAS一旦进入了某个局部最优解 S^{local}, 经过一定步数的迭代后, 所有属于 S^{local} 选择路径上的信

息素值都将近似地等于接近于 1的较大的值 τ_{max}, 而其他路径选择上的信息素值近似地等于较小的值 τ_{min}, 其中 $\tau_{max} + \tau_{min} = 1$ 这个假设是合理的, 因为按照 BAS的信息素更新过程, 所有属于 S^{local}的路径选择在每一次迭代过程中都将得到信息素的强化, 而其他的路径选择都只进行蒸发过程, 所以他们的信息素将线性地增强或减弱, 尤其在经过相当步数的迭代, 当 $\tau_{max} \rightarrow 1$时, 这个特点更为明显. 在 BAS的实验过程中也观察到了这种现象.

假设离 S^{local}最近的更优的解 S^* 与 S^{local}之间的距离为 l, 即 $|S^* - S^{local}| = l$, 也就是说 S^{local}需要经过特定的 l 次的翻转②才能发现 S^*. 则, 根据 BAS解的构筑过程, 可以得到从 S^{local}发现 S^* 的概率为

$$p^* = \tau_{max}^{n-l}\tau_{min}^{l} \tag{15}$$

可以通过求解上式的一次偏微分方程 $\partial p^* / \partial \tau_{max} = 0$来求得 p^* 的最大值. 求解结果表明, 当 $\tau_{max} = (n - l)/n$和 $\tau_{min} = l/n$时, p^* 达到最大值为

$$p^*_{max} = \frac{(n-l)^{n-l} l^l}{n^n} \tag{16}$$

不幸的是, 这一概率值非常的小, 甚至小于采用枚举法的概率 $p_e = \frac{1}{C_n^l} = \frac{(n-l)! \cdot l!}{n!}$. 举例来说, 当 $n = 100$, $l = 1$时, $p^*_{max} = 0.0037$, $p_e = 0.01$. 而当 l 增大时, 这两个概率之间的相对差距就更大了, 例如, 在 $l = 2$时, p^*_{max}和 p_e的值分别为 5.523 5E - 5和 2.020 2E - 4.

上述结果表明, 如果 BAS在进入某局部最优解之后继续其迭代过程的话, 将很难跳出早熟的陷阱. 因此, BAS在监测算法的收敛程度的同时, 引入了局部搜索法, 信息素重新初始化, 以及采用不同权重组合的解进行信息素强化等方法来增加解的多样性, 以避免在某局部最优解附近做长时间低效率的运算.

2.3　参数设置

BAS是相当简单易用的算法程序, 在整个算法中只有 4个参数需要设置, 即蚂蚁数量 n_a, 信息素初始值 τ_0, 收敛因子阈值参数 φ_i, 以及信息素蒸发系数 ρ. 下面将讨论这 4个参数的设置.

针对 BAS的一些初始实验结果表明, 参数 n_a 对算法的最终运算结果并不具有很大的影响, 但在很大程度上会影响计算速度. 直观地, 太大的 n_a 会在每一次迭代过程中进行太多的解的构筑过程以及目标函数的评估, 但对于算法效果的提高却相对有限; 而太小的 n_a 会降低算法搜索的多样性, 从而使算法过早地进入早熟. 通常, n_a 的设置与问题例子的特征有关, 好的 n_a 的设置必须考虑计算速度与求解质量之间的平衡. 根据本文的实验, 以及 Stützle和 Hoos对 MMAS算法的研究结果[18], 本文将 n_a 设置为: $n_a = n$, 其中, n 是问题例子的维度.

本文将信息素的初始值 τ_0 设置为 0.5, 目的是为了在算法初始时各对路径具有相同的选择概率. 但是, 如果根据问题例子的特征信息, 使得某些路径在算法初始时就得到较大的选择概率, 在 BAS中也是可行的.

收敛因子阈值参数 φ_i 决定了信息素强化策略的各个阶段, 而且 φ_5 决定了信息素重新初始化的阈值. 在试验中我们发现, φ_i 的取值对于算法的运算效果影响较大. 如何取值还无法给出统一的标准, 本文根据实验经验设置为: $\varphi_i = [0.3, 0.5, 0.7, 0.9, 0.95]$.

最后, 讨论一下参数 ρ的设置. 蒸发系数 ρ 决定了收敛到局部最优解的速度. 较大的 ρ会使得算法很快地收敛到局部最优解, 但解的质量往往较差; 而太小的 ρ会使得算法很难收敛到比较理想的解. 因此, 参数 ρ设置的关键就是要找到合理的平衡点, 使得算法既能充分探索现有的最优解, 也能不断地发掘新的有潜力的解. 目前来说, 对于参数 ρ的设置还没有充分的理论分析依据, 本文将在下一节中通过实验的方法来确定 ρ的最佳设置.

3　实验结果

本文采用 OR-Library中的 MKP基准问题对 BAS算法进行了测试. 测试集的名称是 5.100和 10.100, 分别是包含有 30个算例的具有 5个约束条件、

② 一次翻转指的是二进制解的某一个二进制位由 0转化为 1或者是由 1转化为 0.

100个项目变量和 10个约束条件、100个项目变量的基准问题测试集. 在所有的实验中, BAS的通用参数设置如下: $n_a = 100$, $\tau_0 = 0.5$, $cf = [0.3\ 0.5\ 0.7\ 0.9\ 0.95]$, 以及最大迭代次数 $t_{max} = 3\,000$.

3.1 参数 ρ的实验设置

测试集 5.100和 10.100中各包含了 30个例子. 这 30个例子按照 $\alpha = b_i / \sum_{j=1}^{n} r_{ij} = 1/4$, $\alpha = 1/2$以及 $\alpha = 3/4$分为 3个部分各 10个例子. 本实验从每一个部分中抽取一个例子作为参数 ρ的测试集. 所选择的例子为: 5.100.0.5, 100.10.5, 100.20.10.100.0.10.100.10以及 10.100.20.

对于每一个测试例子, BAS计算 ρ从 0.01变化到 0.5的实验结果, ρ的变化步长设置为 0.01. 图 5显示的是对于每一个例子运算 30次后, 得到的最优解与已知最优解之间目标函数值的平均相对误差.

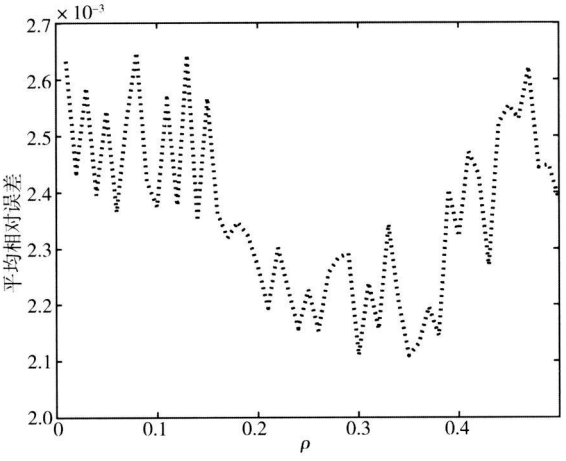

图 5　参数 ρ的实验结果

Fig. 5 Experimental results of parameter ρ

从图 5可以看出, ρ的变化对于测试结果的影响并不是很大. ρ = 0.3和 ρ = 0.35是可以选择的比较好的设置, 在此后的所有试验中, 本文都将采用 ρ = 0.3.

3.2 同其他蚁群优化算法的比较试验

本文将 BAS与其他的 ACO算法进行了比较. 这些算法来自于 Leguizamon 和 Michalewicz[12], Fidanova[13], 以及 Alaya等[14].

表 2显示了基准测试集 5.100的运算结果. 对于每一个算例, 表中记录了从 OR-library中得到的已知最优目标函数值, Leguizamon 和 Michalewicz算法[12], Fidanova算法[13], Alaya等算法[14]得到的最优值和平均值, 以及 BAS在运行 30次后得到的最优值和平均值. 从结果可以看出, BAS算法所得到的平均值都超过了所有的其他 ACO算法. 在所有的 30个测试例子中, 除了例子 100.5.14以外, BAS发现了其他 29个例子的已知最优解.

表 3显示了基准测试集 10.100的运算结果. 对于每一个测试算例, 表中记录了从 OR-library中得到的已知最优目标函数值, Leguizamon 和 Michalewicz算法得到的最优值和平均值[12], Alaya等. 算法得到的最优值和平均值[14], 以及 BAS运行 30次后得到的最优值和平均值. 在这些算例中, BAS同样取得了非常好的结果, 其平均值都超越了其他的所有算法. 在所有 30个算例中, 除了例子 100.10.13和 100.10.27以外, BAS发现了其余 28个算例的最优解.

4　结束语

本文针对著名的多维背包问题提出了基于蚁群优化系统高维立方体结构的二进制蚂蚁算法 BAS与先前的针对 MKP问题的蚁群优化算法不同, BAS采用了针对二进制解的结构所设计的特殊的信息素放置方法; 在算法的迭代过程中摒弃了动态的局部启发式值的计算以及约束条件的限制, 允许非可行解的产生, 从而避免了在每一次迭代中进行动态启发式值和约束判定计算所需的庞大计算量. BAS参数设置的讨论表明, BAS中所使用的信息素可以直接表示为路径选择的概率. 同时, 本文引入了局部搜索法, 信息素重新初始化, 使用不同的解进行信息素的强化等方法, 使得 BAS能够尽快地跳出局部最优. BAS不仅提高了 ACO算法在解决 MKP问题时的运算速度, 而且其运算结果也较为令人满意. 试验结果表明, 在所有的较大规模的基准问题例子中, BAS的求解结果都显著地超过了其他类型的 ACO算法.

背包问题是种特殊的 0-1规划问题, BAS在 MKP问题上的成功应用, 说明了这种针对二进制解的结构设计的信息素放置方法可以应用到其他的 0-1规划问题中. 如何设计和充分利用问题的特征信息是今后研究所要着重解决的问题.

表 2　MKP 基准测试集 5.100的运算结果

Table 2 Experimental results on MKP benchmark problems 5.100 instances

N°	最优目标函数值	Leguizamon和 Michalewicz		Fidanova Alaya等			BASMKP	
		最优值	平均值	最优值		平均值	最优值	平均值
00	24381	24381	24331	23984	24381	24342	24381	24381
01	24274	24274	24245	24145	24274	24247	24274	24274
02	23551	23551	23527	23523	23551	23529	23551	23539.3
03	23534	23527	23463	22874	23534	23462	23534	23527.8
04	23991	23991	23949	23751	23991	23946	23991	23977.67
05	24613	24613	24563	24601	24613	24587	24613	24613
06	25591	25591	25504	25293	25591	25512	25591	25591
07	23410	23410	23361	23204	23410	23371	23410	23410
08	24216	24204	24173	23762	24216	24172	24216	24212.4
09	24411	24411	24326	24255	24411	24356	24411	24408.6
10	42757			42705	42757	42704	42757	42735.9
11	42545			42445	42510	42456	42545	42505.37
12	41968			41581	41967	41934	41968	41966.77
13	45090			44911	45071	45056	45090	45076.07
14	42218			42025	42218	42194	42198	42197.87
15	42927			42671	42927	42911	42927	42927
16	42009			41776	42009	41977	42009	42009
17	45020			44671	45010	44971	45020	45019.5
18	43441			43122	43441	43356	43441	43439
19	44554			44471	44554	44506	44554	44554
20	59822			59798	59822	59821	59822	59822
21	62081			61821	62081	62010	62081	62015.3
22	59802			59694	59802	59759	59802	59774.93
23	60479			60479	60479	60428	60479	60477.03
24	61091			60954	61091	61072	61091	61063.4
25	58959			58695	58959	58945	58959	58959
26	61538			61406	61538	61514	61538	61532.53
27	61520			61520	61520	61492	61520	61506.87
28	59453			59121	59453	59436	59453	59453
29	59965			59864	59965	59958	59965	59963.33

表 3 MKP 基准测试集 10. 100的运算结果.

Table 3 Experimental results on MKP benchmark problems 10 100 instances

N°	最优目标函数值	Leguizamon和 Michalewicz		Fidanova Alaya等		BASMKP	
		最优值	平均值	最优值	平均值	最优值	平均值
00	23064	23057	22996	23064	23016	23064	23058. 87
01	22801	22801	22672	22801	22714	22801	22776. 33
02	22131	22131	21980	22131	22034	22131	22116. 17
03	22772	22772	22631	22717	22634	22772	22766. 6
04	22751	22654	22578	22654	22547	22751	22707. 8
05	22777	22652	22565	22716	22602	22777	22733. 53
06	21875	21875	21758	21875	21777	21875	21861. 93
07	22635	22551	22519	22551	22453	22635	22635
08	22511	22418	22292	22511	22351	22511	22433. 83
09	22702	22702	22588	22702	22591	22702	22702
10	41395			41395	41329	41395	41387. 33
11	42344			42344	42214	42344	42282. 83
12	42401			42401	42300	42401	42367
13	45624			45624	45461	45598	45561. 73
14	41884			41884	41739	41884	41846. 5
15	42995			42995	42909	42995	42991. 4
16	43559			43553	43464	43559	43535. 8
17	42970			42970	42903	42970	42962. 8
18	42212			42212	42146	42212	42212
19	41207			41207	41067	41207	41172
20	57375			57375	57318	57375	57361. 43
21	58978			58978	58889	58978	58970. 67
22	58391			58391	58333	58391	58382. 37
23	61966			61966	61885	61966	61905. 67
24	60803			60803	60798	60803	60803
25	61437			61437	61293	61437	61377. 2
26	56377			56377	56324	56377	56352
27	59391			59391	59339	59366	59366
28	60205			60205	60146	60205	60171. 5
29	60633			60633	60605	60633	60633

参 考 文 献:

[1] Martello S, Toth P. Knapsack Problems: Algorithms and Computer Implementations[M]. Chichester-New York: John Wiley & Sons, 1990.

[2] Gavish B, Pirkul H. Allocation of databases and processors in a distributed computing system[A]. In: Akoka J. Management of Distributed Data Processing[M]. Amsterdam: North-Holland, 1982. 215— 231.

[3] Shih W. A branch and bound method for the multiconstraint zero-one knapsack problem[J]. Journal of The Operational Research Society, 1979, 30(4): 369—378.
[4] Gilmore P C, Gomory R E. The theory and computation of knapsack functions[J]. Operation Research, 1966, 14(6): 1045—1074.
[5] Bonabeau E, Dorigo M, Theraulaz G. Swarm Intelligence: From Natural to Artificial Systems[M]. New York Oxford: Oxford University Press, 1999.
[6] Dorigo M, Stützle T. Ant Colony Optimization[M]. Cambridge, Massachusetts, London, England: The MIT Press, 2004.
[7]马　良, 项培军. 蚂蚁算法在组合优化中的应用[J]. 管理科学学报, 2001, 4(2): 32—37.
Ma Liang, Xiang Pei-jun. Applications of the ant algorithm to combinatorial optimization[J]. Journal of Management Sciences in China, 2001, 4(2): 32—37. (in Chinese)
[8] Dorigo M, Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem[J]. IEEE Trans. Evol. Comput, 1997, 1(1): 53—66.
[9] Gambardella L M, Taillard E, Dorigo M. Ant colonies for the quadratic assignment problem[J]. J. Oper. Res. Soc., 1999, 50(2): 167—176.
[10] Maniezzo V, Colorni A. The ant system applied to the quadratic assignment problem[J]. IEEE Trans. Data Knowl. Eng., 1999, 11(5): 769—778.
[11] Gambardella L M, Taillard E D, Agazzi G. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows[A]. In: Corne D, Dorigo M, Glover F. New Ideas in Optimization[M]. London: McGraw-Hill, 1999. 63—76.
[12] Leguizamon G, Michalewicz Z. A New Version of Ant System for Subset Problem[C]. IEEE Congress on Evolutionary Computation, Washington D. C., 1999, 2: 1459—1464.
[13] Fidanova S. Evolutionary Algorithm for Multidimensional Knapsack Problem[M]. Granada: PPSN VIII-Workshop, 2002.
[14] Alaya I, Solnon C, Gheira K. Ant Algorithm for the Multi-dimensional Knapsack Problem[C]. International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2004), 2004, October: 63—72.
[15] Blum C, Dorigo M. The hyper-cube framework for ant colony optimization[J]. IEEE Transactions on Man, Systems and Cybernetics — Part B, 2004, 34(2): 1161—1172.
[16] Pirkul H. A heuristic solution procedure for the multiconstraint zero-one knapsack problem[J]. Naval Research Logistics, 1987, 34(2): 161—172.
[17] Chu P C, Beasley J E. A genetic algorithm for the multidimentional knapsack problem[J]. Journal of heuristic, 1998, 4(1): 63—86.
[18] StÄutzle T, Hoos H. MAX-MIN ant system[J]. Future Generation Computer Systems Journal, 2000, 16(8): 889—914.

Binary ant system for multidimensional knapsack problem

KONG Min, TIAN Peng, LI Xiang-yong
School of Management, Shanghai Jiaotong University, Shanghai 200052, China

Abstract This paper proposes a binary ant system (BAS), an improved hyper cube framework of ant colony optimization (ACO) applied to multidimensional knapsack problem (MKP). Different to other Ant Systems applied to MKP, BAS designs a special pheromone laying method based on the binary solution structure, admits infeasible solution during the solution construction procedure, and uses a problem specific repairing operator to repair those infeasible solutions generated in each iteration. BAS uses a special pheromone update rule, in order to use the pheromone directly as the selecting probability during the solution construction procedure. To avoid premature, BAS designs a simple local search method, and incorporates different pheromone updating strategy and a pheromone re-initialization procedure depending on the convergence status of the algorithm. Experimental results from various benchmark problems of MKP show the advantage of BAS over other ant systems, and indicates the potential of BAS in dealing with larger size MKP problems.

Key words ant colony optimization; binary ant system; combinatorial optimization; multidimensional knapsack problem