文章编号: 1001-0920(2012)09-1325-06

简 化 蚁 群 算 法

张兆军, 冯祖仁, 陈竹青

(西安交通大学 a. 系统工程研究所, b. 制造系统工程国家重点实验室, 西安 710049)

摘 要: 针对最大最小蚂蚁系统中信息素下界难以确定以及算法性能易受同构问题影响的缺点, 提出一种简化蚁群算法. 信息素的上下界被限制在一个固定的区间内, 不随目标函数值的更新而改变; 信息素的更新量是一个与具体目标函数值无关的常数. 所提出的简化算法不仅具有强不变性和平移不变性, 而且算法的性能不受信息素下界的影响. 针对旅行商问题的仿真实验验证了改进算法的可行性和有效性.

关键词: 最大最小蚂蚁系统; 不变性; 旅行商问题

中图分类号: TP18 **文献标志码:** A

Simplified ant colony optimization algorithm

ZHANG Zhao-jun, FENG Zu-ren, CHEN Zhu-qing

(a. Systems Engineering Institute, b. State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, China. Correspondent: ZHANG Zhao-jun, E-mail: zzj921@163.com)

Abstract: Due to the shortcomings of max-min ant system(MMAS), which is difficult to decide the lower bound of pheromone trail and easy to affect by isomorphic problems, a simplified ant colony optimization(SACO) algorithm is proposed. The upper and lower bound of pheromone trail are limited to a fixed interval and can not be changed with updating the objective function value. The added pheromone trail is a constant which is independent to the function value. It is proved that the algorithm not only has the property of linear transformational invariance and translational invariance, but also its performance is not affected by the lower bound of pheromone trail. Simulation results on the traveling salesman problem show the feasibility and effectiveness of the presented algorithm.

Key words: max-min ant system; invariance; traveling salesman problem

1 引 言

蚁群优化[1](ACO)是由意大利学者Dorigo等提出的一种仿生随机优化算法, 已被广泛应用于各种难以求解的组合优化问题, 并取得了很好的效果. 蚂蚁系统(AS)[2]是第1个ACO算法, 尽管它表明了ACO算法具有鲁棒性、正反馈、分布式计算和易与其他算法结合等优点, 但它的性能却逊于其他用于旅行商问题(TSP)的经典算法. 此后学者们提出了很多改进算法, 其中应用较为广泛的是使用局部更新策略和全局更新策略的蚁群系统(ACS)[3]以及限制信息素的上下界和使用最优解更新策略的最大最小蚂蚁系统(MMAS)[4]. 另外, 我国学者也做了很多研究工作[5-7].

作为一种基于模型的优化算法, 人们不仅关注算法性能改进的研究, 而且关注算法自身性质的研究. Blum等[8]最早进行了蚁群算法不变性的研究, 并指出标准ACO算法在处理同构问题时, 其在求解性能上有较大差异. 随后, 文献[9]对同构问题进行了严格定义, 并给出了线性不变性的定义和分类, 同时证明了AS, MMAS和ACS具有弱不变性, 并构造了相应的强不变算法, 分别记为siAS, siMMAS和siACS, 同时证明了弱不变算法与强不变算法之间是函数等价的.

综上所述, 几种改进的不变ACO算法是根据问题的同构特点, 在对信息素更新模型进行相应归一化处理的基础上得到的. 这同时表明, 在信息素更新过程中由于目标函数值的变化而导致了标准ACO算法仅仅具有弱不变性. 如果信息素的定义和更新过程均与目标函数值无关, 则可以得到另外一种构造强不变蚁群算法的方法. 鉴于此, 本文在MMAS的基础上提出一种具有强不变性的蚁群算法, 简称为简化蚁群算法(SACO).

收稿日期: 2011-03-27; **修回日期:** 2011-08-09.

基金项目: 国家自然科学基金项目(60875043, 60905044); 教育部博士点基金项目(20100211110031).

作者简介: 张兆军(1981–), 男, 博士生, 从事智能优化算法、数据挖掘等研究; 冯祖仁(1953–), 男, 教授, 博士生导师, 从事机器人控制、智能信息处理等研究.

2 最大最小蚂蚁系统及基本定义

2.1 最大最小蚂蚁系统

ACO 算法利用一群人工蚂蚁通过模拟自然界真实蚂蚁群体的觅食行为求解难以求解的离散优化问题. 在众多改进 ACO 算法中, MMAS 是一种应用比较广泛的 ACO 算法.下面以 TSP 为例描述 MMAS 的基本原理.

为叙述方便, 将 TSP 问题用构造图 $G=(N,L)$ 表示. 规定 N 是城市 (节点) 集合, L 是连接任意两个城市 i 与城市 j 的路径 l_{ij} 的集合, d_{ij} 表示两个城市 i 与城市 j 之间的距离, n 表示城市的个数, m 表示 (人工) 蚂蚁的个数.

在 MMAS 中, m 只蚂蚁并行地构建 TSP 路径. 首先, 初始化参数, 其中强调的是信息素初始化为信息素的上界; 其次, 蚂蚁被分别放在随机选择出来的城市中. 在每次迭代过程中, 蚂蚁按照一个被称为随机比例规则的路径选择策略确定下一步将选择的城市. 随机比例规则如下: 假设在第 t 次迭代时蚂蚁 k 在第 i 个城市, 则选择城市 j 作为下一个访问城市的概率为

$$P_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\displaystyle\sum_{r\in N_k}[\tau_{ir}(t)]^{\alpha}[\eta_{ir}]^{\beta}}, & j\in N_k;\\ 0, & \text{其他}.\end{cases}\tag{1}$$

其中: α 和 β 是参数, $\tau_{ij}(t)$ 表示路径 l_{ij} 上的信息素, $\eta_{ij}=1/d_{ij}$ 表示路径 l_{ij} 上的启发信息, N_i^k 是与 i 相连的所有城市, r 是 N_i^k 中的某个城市.

当所有蚂蚁都获得一个完整路径后, 算法按照下式更新信息素:

$$\tau_{ij}(t+1)=[(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}^{\text{best}}(t)]_{\tau_{\min}}^{\tau_{\max}}.\tag{2}$$

其中 $[x]_b^a$ 定义为

$$[x]_b^a=\begin{cases}a, & x>a;\\ b, & x<b;\\ x, & \text{其他}.\end{cases}$$

其中: $\rho\,(0<\rho\leqslant 1)$ 表示信息素挥发系数, $\tau_{\max}$ 和 $\tau_{\min}$ 分别表示信息素的上界和下界, $\tau_{\max}=1/\rho f(s_{\text{bs}}^{\text{best}})$, $\tau_{\min}=\xi\tau_{\max}$, 参数 $0\leqslant\xi<1$ 的定义参见文献 [4]. 令 s^{best} 表示被允许释放信息素的最优解, s^{best} 的可能取值是当前迭代最优解 $s_{\text{ib}}^{\text{best}}$ 或至今迭代最优解 $s_{\text{gb}}^{\text{best}}$, $f(s^{\text{best}})$ 表示对应的路径长度. $\Delta\tau_{ij}^{\text{best}}(t)$ 表示路径 l_{ij} 上的信息素增量, 定义为

$$\Delta\tau_{ij}^{\text{best}}(t)=\begin{cases}1/f(s^{\text{best}}), & \text{边}\,(i,j)\in s^{\text{best}};\\ 0, & \text{其他}.\end{cases}\tag{3}$$

虽然 MMAS 比 AS 在求解性能上有了很大的提高, 而且文献 [9] 已经证明 MMAS 具有弱不变性, 但也存在一些不足. 首先, 信息素上界的大小取决于所求问题的最优解, 而在实际问题中是无法得到问题最优解的; 其次, 信息素下界的确定相当困难, 目前只能得到一个近似值, 而信息素下界的取值对 MMAS 的性能影响很大[10].

2.2 基本定义

文献 [9] 给出了标准 ACO 算法的不变性的完整讨论, 这里给出几个基本定义和 1 个基本假设.

定义 1 (线性变换) I 和 $\bar{I}$ 是定义在同一个解空间 S 上的两个组合优化问题, 如果对于任意 $s\in S$, 有 $\bar{f}(s)=g_1 f(s)$ 成立, 则称 $\bar{I}$ 和 I 满足线性变换关系, 记为 $\bar{I}=g_1 I$, 其中 $g_1>0$ 是一个常数, 且 $f(s)$ 和 $\bar{f}(s)$ 是问题 I 和 $\bar{I}$ 在 s 对应的目标函数值.

定义 2 (弱不变性) 设组合优化问题 I 和 $\bar{I}$ 满足线性变化关系, S_I 和 $S_{\bar{I}}$ 是由 ACO 算法 A 求得的相应的解序列, 如果两个解序列 S_I 与 $S_{\bar{I}}$ 相同, 则称算法 A 具有弱不变性, 或称算法 A 是弱不变算法.

定义 3 (强不变性) 设组合优化问题 I 和 $\bar{I}$ 满足线性变化关系, 且 ACO 算法 A 是弱不变算法, 如果在分别求解问题 I 和 $\bar{I}$ 时, 每次迭代中信息素和启发信息的值也对应相同, 则称算法 A 具有强不变性, 或称算法 A 是强不变算法.

定义 4 (线性不变性) 设组合优化问题 I 和 $\bar{I}$ 满足线性变化关系, 且 ACO 算法 A 是弱不变算法, 如果在分别求解问题 I 和 $\bar{I}$ 时, 每次迭代中信息素和启发信息值对应相同, 则称算法 A 具有线性不变性, 或称算法 A 是线性不变算法.

定义 5 (弱函数等价) 设 A 和 $\bar{\text{A}}$ 是两个 ACO 算法, I 是一个组合优化问题, S_I 和 $\bar{S}_I$ 分别表示由算法 A 和 $\bar{\text{A}}$ 求解问题 I 得到的解序列, 如果 S_I 与 $\bar{S}_I$ 相同, 则称 ACO 算法 A 与 $\bar{\text{A}}$ 是函数等价的.

定义 6 (弱平移不变性) 令 b 是一个常数, 组合优化问题 I 和 $\bar{I}$ 满足 $\bar{I}=I+b$, 如果采用 ACO 算法 A 分别求解问题 I 和 $\bar{I}$ 时, 每次迭代中解序列和信息素值对应相同, 则称算法 A 具有平移不变性, 或称算法 A 是平移不变算法.

假设 1 当采用 ACO 算法求解满足线性关系的问题 I 和 $\bar{I}$ 时, 为了产生两列相同的伪随机数, 假设伪随机数生成器按同一种方式初始化. 同理, 当用两种 ACO 算法求解同一个问题时, 也满足这种假设.

3 SACO 算法

3.1 算法路径选择策略

SACO 算法主要改进工作集中在信息素模型的简化上, 而对于路径选择策略则沿用使用较多的随机比例规则, 即式 (1).

3.2 信息素更新规则

与MMAS类似, SACO只使用迭代最优解更新信息素. 信息素的上下界被限制在一个固定的区间内, 为方便起见, 仍然记作 $[\tau_{\min}, \tau_{\max}]$, 且满足 $0<\tau_{\min}<\tau_{\max}$. 这里 $\tau_{\min}, \tau_{\max}$ 是常数, 不受目标函数值改变的影响. 信息素的更新量是与 $\tau_{\max}$ 有关的常量, 具体描述为当所有蚂蚁都获得一个完整路径后, 算法按下式更新信息素:

$$\tau_{ij}(t+1)=[(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}^{\text{best}}(t)]_{\tau_{\min}}^{\tau_{\max}}. \quad (4)$$

其中 $\Delta\tau_{ij}^{\text{best}}$ 定义为

$$\Delta\tau_{ij}^{\text{best}}(t)=\begin{cases}\rho\tau_{\max}, \text{ 边 } (i,j)\in s^{\text{best}};\\ 0, \text{ 其他.}\end{cases} \quad (5)$$

由于挥发系数 ρ 和信息素上界 $\tau_{\max}$ 都是常量, 由式 (5) 可知, 信息素的更新量也是一个常量. 对于 $\tau_{\min}$ 和 $\tau_{\max}$, 需要特别说明的一点是, 虽然没有给出具体的取值, 但下节将证明信息素上下界限制在区间 $[\tau_{\min}, \tau_{\max}]$ 所得到的SACO算法与限制在区间 $[\tilde{\tau}_{\min}, \tilde{\tau}_{\max}]$ 所得到的算法是函数等价的, 即 $\tau_{\min}, \tau_{\max}$ 的具体取值不会影响SACO算法的性能.

3.3 信息素的初始化

MMAS算法中信息素初始化为信息素的上界, 而信息素的上界与所求问题的最优解有关, 但最优解是很难得到的. 因此, 通常的做法是先用已知的解或者通过其他方法求得的解作为代替, 再在以后的迭代过程中通过 $[\tau]_{\tau_{\min}}^{\tau_{\max}}$ 算子加以修正. 然而, 在SACO算法中已经给定了 $\tau_{\max}$ 的值, 从而可将信息素初始化为 $\tau_0=\omega\cdot\tau_{\max}$. 其中 $0<\omega\leqslant 1$ 是一个常数, 当 $\omega=1$时, 初始化为信息素的上界 $\tau_{\max}$.

3.4 SACO的性质

定理 1　令 $g_1>0$ 和 $g_2>0$ 是常数, 问题 I 和 $\bar{I}$ 满足 $\bar{I}=g_1 I$, S_I 和 $S_{\bar{I}}$ 是由SACO算法得到的解序列, 对于任意 $l_{ij}\in L, \tau_{ij}(t), \bar{\tau}_{ij}(t), \eta_{ij}, \bar{\eta}_{ij}$ 分别是在 t 次迭代时SACO算法对应的信息素和启发信息的值. 若启发信息满足 $\bar{\eta}_{ij}=g_2\eta_{ij}$, 则有 $S_{\bar{I}}=S_I$, 且 $\bar{\tau}_{ij}(t)=\tau_{ij}(t)$.

证明　使用归纳法证明结论成立. 因为信息素的上、下界均是常数, 所以对于同一个算法有

$$\bar{\tau}_{\max}=\tau_{\max},\ \bar{\tau}_{\min}=\tau_{\min}.$$

又因为信息素初始化为信息素上界的常数倍, 因此有

$$\bar{\tau}_0=\omega\cdot\bar{\tau}_{\max}=\omega\cdot\tau_{\max}=\tau_0.$$

现假设命题在 t 次迭代前是成立的, 即对于任意 $l_{ij}\in L$ 有 $\bar{\tau}_{ij}(t)=\tau_{ij}(t)$ 且 $\bar{s}^k(t-1)=s^k(t-1)$, 则对于任意 $l_{ij}\in L$ 有 $\bar{p}_{ij}^k(t)=p_{ij}^k(t)$. 这是因为

$$\bar{p}_{ij}^k(t)=\frac{[\bar{\tau}_{ij}(t)]^\alpha[\bar{\eta}_{ij}]^\beta}{\sum\limits_{l\in N_i^k}[\bar{\tau}_{il}(t)]^\alpha[\bar{\eta}_{il}]^\beta}=$$

$$\frac{[\tau_{ij}(t)]^\alpha[g_2\eta_{ij}]^\beta}{\sum\limits_{l\in N_i^k}[\tau_{il}(t)]^\alpha[g_2\eta_{il}]^\beta}=\frac{[g_2]^\beta[\tau_{ij}(t)]^\alpha[\eta_{ij}]^\beta}{[g_2]^\beta\sum\limits_{l\in N_i^k}[\tau_{il}(t)]^\alpha[\eta_{il}]^\beta}=$$

$$\frac{[\tau_{ij}(t)]^\alpha[\eta_{ij}]^\beta}{\sum\limits_{l\in N_i^k}[\tau_{il}(t)]^\alpha[\eta_{il}]^\beta}=p_{ij}^k(t). \quad (6)$$

故根据假设 1, 对于任意的 $k=1,2,\cdots,m$, 有 $\bar{s}^k(t)=s^k(t)$. 又根据 $\bar{I}=g_1 I$, 从而有 $\bar{f}(\bar{s}^k(t))=g_1 f(s^k(t))$.

由上述证明过程可知, 若在 t 次迭代时满足对于任意 $l_{ij}\in L$ 有 $\bar{\tau}_{ij}(t)=\tau_{ij}(t)$ 和 $\bar{\eta}_{ij}=g_2\eta_{ij}$, 则对于任意的 $k=1,2,\cdots,m$, 结论 $\bar{s}^k(t)=s^k(t)$ 成立. 因此, 只需要证明在 t 步迭代时, 信息素按式 (4) 更新后仍然满足条件 $\bar{\tau}_{ij}(t+1)=\tau_{ij}(t+1)$.

因为对于任意的 $k=1,2,\cdots,m$, 有 $\bar{s}^k(t)=s^k(t)$, 而当前迭代最优解是其中最好的一个解, 故对当前迭代最优解也成立, 即 $\bar{s}_{\text{ib}}^{\text{best}}(t)=s_{\text{ib}}^{\text{best}}(t)$, 且有

$$\bar{f}(\bar{s}_{\text{ib}}^{\text{best}}(t-1))=g_1 f(s_{\text{ib}}^{\text{best}}(t-1)).$$

下面证明至今迭代最优解这个结论仍成立, 即

$$\bar{s}_{\text{gb}}^{\text{best}}(t)=s_{\text{gb}}^{\text{best}}(t).$$

因为 $\bar{s}_{\text{gb}}^{\text{best}}(t-1)=s_{\text{gb}}^{\text{best}}(t-1)$, 所以有 $\bar{f}(\bar{s}_{\text{gb}}^{\text{best}}(t-1))=g_1 f(s_{\text{gb}}^{\text{best}}(t-1))$. 若在 t 次迭代中SACO算法对于问题 I 找到了更好的解, 即 $f(s_{\text{ib}}^{\text{best}}(t))<f(s_{\text{gb}}^{\text{best}}(t-1))$, 则同样对于问题 $\bar{I}$ 也有 $\bar{f}(\bar{s}_{\text{ib}}^{\text{best}}(t))<\bar{f}(\bar{s}_{\text{gb}}^{\text{best}}(t-1))$. 因此可用当前迭代最优解来更新至今迭代最优解, 即

$$s_{\text{gb}}^{\text{best}}(t)=s_{\text{ib}}^{\text{best}}(t)=\bar{s}_{\text{ib}}^{\text{best}}(t)=\bar{s}_{\text{gb}}^{\text{best}}(t);$$

否则

$$s_{\text{gb}}^{\text{best}}(t)=s_{\text{gb}}^{\text{best}}(t-1)=\bar{s}_{\text{gb}}^{\text{best}}(t-1)=\bar{s}_{\text{gb}}^{\text{best}}(t).$$

根据式 (5), 信息素的更新量满足

$$\bar{\Delta}\tau_{ij}^{\text{best}}(t)=$$

$$\begin{cases}\rho\bar{\tau}_{\max}, \text{ 边 } (i,j)\in\bar{s}^{\text{best}}\\ 0, \text{ 其他}\end{cases}=$$

$$\begin{cases}\rho\tau_{\max}, \text{ 边 } (i,j)\in\bar{s}^{\text{best}}=s^{\text{best}}\\ 0, \text{ 其他}\end{cases}=$$

$$\Delta\tau_{ij}^{\text{best}}(t). \quad (7)$$

从而对于任意 $l_{ij}\in L$, 有

$$\bar{\tau}_{ij}(t+1)=$$

$$[(1-\rho)\bar{\tau}_{ij}(t)+\bar{\Delta}\tau_{ij}^{\text{best}}(t)]_{\tau_{\min}}^{\tau_{\max}}=$$

$$[(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}^{\text{best}}(t)]_{\tau_{\min}}^{\tau_{\max}}=$$

$$\tau_{ij}(t+1). \quad (8)$$

因此, 在 $t+1$ 步迭代中信息素仍然相等. 又因为启发信息始终满足条件, 所以由归纳法知结论恒成立, 即对于任意 t, 均有 $S_{\bar{I}}=S_I$, 且 $\bar{\tau}_{ij}(t)=\tau_{ij}(t)$. □

注 1　1) 这里讨论的优化问题均是最小化问题,

下面与此类似, 不再特殊说明; 2) 定理 1 说明 SACO 算法具有线性不变性.

定理 2 令 $g_1 > 0$ 是常数, 问题 I 和 $\bar{I}$ 满足 $\bar{I} = g_1 I$, 若对于任意 $l_{ij} \in L$ 启发信息的值满足 $\bar{\eta}_{ij} = \eta_{ij}$, 则 SACO 算法是强不变算法.

证明 根据强不变性定义, 只需证明在每次迭代过程中算法对问题 I 和 $\bar{I}$ 所得到解的序列以及信息素与启发信息的值相同. 由定理 1 可知, SACO 是不变算法, 又因为对于任意 $l_{ij} \in L$ 有 $\bar{\eta}_{ij} = \eta_{ij}$, 故算法 SACO 是强不变算法. □

注 2 如果算法中不考虑启发信息的影响, 则条件自然满足, 即当 $\beta = 0$ 时, SACO 算法是强不变算法.

与定理 1 的证明过程类似, 可以得到如下结论:

定理 3 令 $b, \lambda > 0$ 是常数, 问题 I 和 $\bar{I}$ 满足 $\bar{I} = I + b$, 若对于任意 $l_{ij} \in L$, 启发信息满足 $\bar{\eta}_{ij} = \lambda \eta_{ij}$, 则 SACO 算法是平移不变算法.

注 3 定理 3 说明 SACO 不仅具有线性变换不变性, 而且具有平移不变性. 然而, 前面提到的几种强不变算法仅具有线性不变性.

定理 4 令 μ 是常数, $\mu = \dfrac{\tilde{\tau}_{\max}}{\tilde{\tau}_{\min}} = \dfrac{\tau_{\max}}{\tau_{\min}}$, 信息素被限制在区间 $[\tau_{\min}, \tau_{\max}]$ 的 ACO 算法记为 SACO 1, 信息素被限制在区间 $[\tilde{\tau}_{\min}, \tilde{\tau}_{\max}]$ 的 ACO 算法记为 SACO 2, 对应的信息素和启发信息分别记为 τ_{ij}, η_{ij}, $\tilde{\tau}_{ij}, \tilde{\eta}_{ij}$, 则 SACO 1 算法与 SACO 2 算法是函数等价的.

证明 使用归纳法证明. 根据定义 5, 只需证明两个算法对同一问题构造的解序列相同即可. 因为 $\tau_{ij}(0) = m_1 \tau_{\max}$ 和 $\tilde{\tau}_{ij}(0) = m_1 \tilde{\tau}_{\max}$, 所以有 $\tilde{\tau}_{ij}(0) = \mu \tau_{ij}(0)$. 假设在 t 次迭代前, 对于任意 $l_{ij} \in L$ 有 $\tilde{s}^k(t-1) = s^k(t-1)$ 和 $\tilde{\tau}_{ij}(t) = \mu \tau_{ij}(t)$, 则与定理 1 类似地可以证明, 对于任意 $l_{ij} \in L$ 有 $\tilde{p}_{ij}^k(t) = p_{ij}^k(t)$ 成立, 且对于任意 $k = 1, 2, \cdots, m$ 有 $\tilde{s}^k(t) = s^k(t)$. 特殊地, 有 $\tilde{s}_{\text{ib}}^{\text{best}}(t) = s_{\text{ib}}^{\text{best}}(t)$.

下面只需证明在信息素更新后仍有下式成立:

$$\tilde{\tau}_{ij}(t+1) = \mu \tau_{ij}(t+1).$$

因为 $\tilde{s}_{\text{gb}}^{\text{best}}(t-1) = s_{\text{gb}}^{\text{best}}(t-1)$, 所以有 $f(\tilde{s}_{\text{gb}}^{\text{best}}(t-1)) = f(s_{\text{gb}}^{\text{best}}(t-1))$. 如果 t 次迭代中 SACO 1 算法搜索到更好的优解, 即 $f(s_{\text{ib}}^{\text{best}}(t)) < f(s_{\text{gb}}^{\text{best}}(t-1))$, 则同样对于 SACO 2 也有 $f(\tilde{s}_{\text{ib}}^{\text{best}}(t)) < f(\tilde{s}_{\text{gb}}^{\text{best}}(t-1))$. 因此可用当前迭代最优解来更新至今迭代最优解, 即

$$s_{\text{gb}}^{\text{best}}(t) = s_{\text{ib}}^{\text{best}}(t) = \tilde{s}_{\text{ib}}^{\text{best}}(t) = \tilde{s}_{\text{gb}}^{\text{best}}(t);$$

否则

$$s_{\text{gb}}^{\text{best}}(t) = s_{\text{gb}}^{\text{best}}(t-1) = \tilde{s}_{\text{gb}}^{\text{best}}(t-1) = \tilde{s}_{\text{gb}}^{\text{best}}(t).$$

根据式 (5), 信息素的更新量满足

$$\tilde{\Delta}\tau_{ij}^{\text{best}}(t) =$$

$$\begin{cases} \rho\tilde{\tau}_{\max}, \text{ 边 } (i,j) \in \tilde{s}^{\text{best}} \\ 0, \text{ 其他} \end{cases} =$$

$$\begin{cases} \mu\rho\tau_{\max}, \text{ 边 } (i,j) \in \tilde{s}^{\text{best}} = s^{\text{best}} \\ 0, \text{ 其他} \end{cases} =$$

$$\mu\Delta\tau_{ij}^{\text{best}}(t), \tag{9}$$

从而对于任意 $l_{ij} \in L$, 有

$$\begin{aligned} &\tilde{\tau}_{ij}(t+1) = \\ &[(1-\rho)\tilde{\tau}_{ij}(t) + \tilde{\Delta}\tau_{ij}^{\text{best}}(t)]_{\tau_{\min}}^{\tau_{\max}} = \\ &[(1-\rho)\mu\tau_{ij}(t) + \mu\Delta\tau_{ij}^{\text{best}}(t)]_{\tau_{\min}}^{\tau_{\max}} = \\ &\mu\tau_{ij}(t+1). \end{aligned} \tag{10}$$

因此, 在 $t+1$ 步迭代中信息素仍然满足 $\tilde{\tau}_{ij}(t+1) = \mu\tau_{ij}(t+1)$. 由归纳法可知, 对于任意 t, 均有 $S_{\bar{I}} = S_I$, 故 SACO 1 算法与 SACO 2 算法是函数等价的. □

注 4 由定理 4 可知, SACO 算法对信息素上下界的要求并不是很严格, 但为了提高算法的性能, 一般与启发信息的值保持一定的关系, 如针对 TSP 问题, 可以将信息素的值限制在区间 $[0.001, 0.999]$ 上.

4 实验结果和分析

为了考察所提出算法的可行性和有效性, 本文以 TSP 问题为例, 采用 VC + +6.0 实现 SACO 算法和 MMAS 算法. 计算机是酷睿双核, 主频 2.8 GHz, 2.0 G 内存, Windows XP 操作系统. 测试算例选自 TSPLIB, 算例中的数字表示城市数目, 括号内的数是已知最优解. 每个算例独立运行 25 次, 每次构造解的个数是 10 000n. 其中: n 是城市数目, 蚂蚁的数目 $m = n$. 用 MMAS + s_{gb} 和 SACO + s_{gb} 分别表示采用至今迭代最优解 s_{gb} 更新信息素的 MMAS 算法和 SACO 算法. MMAS 表示每隔 10 次迭代采用至今迭代最优解更新信息素, 其余均采用当前迭代最优解更新信息素. SACO 表示每隔 10 次迭代采用当前迭代最优解更新信息素, 其余均采用至今迭代最优解更新信息素. 其他参数设置如下: 对于两种 MMAS 算法, $\alpha = 1, \beta = 2, \rho = 0.02$; 而对于两种 SACO 算法, $\alpha = 1, \beta = 6, \rho = 0.004, \omega = 0.5$. 实验中分别比较了 4 种算法在 25 次实验中所得到的优解的最优值、最差值、平均值以及标准差; 同时比较了各个算法首次获得其最优解所需要的平均迭代次数、平均所需时间以及算法完成所需要的平均时间, 分别记作首遇迭代次数、首遇时间和完成时间. 实验结果如表 1 所示.

从表 1 可以看出: 在求解质量方面, SACO 算法优于 SACO + s_{gb} 算法和 MMAS 算法; MMAS 算法优于 MMAS + s_{gb}. 除算例 eil 51 和 lin 105 外, SACO + s_{gb} 算法优于 MMAS 算法; 除算例 eil 51 和 d 198 外, SACO 算法标准差的值都小于 MMAS 算法, 这说明 SACO 算

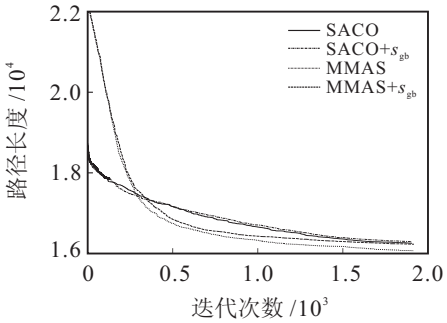
表 1　实验结果比较

算 例	算 法	最优值	最差值	平均值	标准差	首遇迭代次数	首遇时间/s	完成时间/s
eil 51(426)	MMAS $+s_{gb}$	426	435	428.52	2.82	2 105.8	2.247	10.648
	SACO$+s_{gb}$	426	431	427.36	1.38	1 353.5	1.424	10.463
	MMAS	426	428	427.16	0.62	2 328.0	3.050	10.723
	SACO	426	428	426.96	0.89	1 514.6	1.591	10.448
kroA 100(21 282)	MMAS$+s_{gb}$	21 282	21 825	21 371.80	144.12	3 679.5	15.633	42.408
	SACO$+s_{gb}$	21 282	21 379	21 301.12	42.92	3 595.1	15.530	42.908
	MMAS	21 282	21 379	21 310.48	43.88	2 757.0	12.048	43.712
	SACO	21 282	21 305	21 289.24	10.39	3391.0	14.742	43.181
lin 105(14 379)	MMAS$+s_{gb}$	14 379	14 577	14 427.4	57.51	1 378.9	6.265	44.891
	SACO$+s_{gb}$	14 379	14 434	14 385.60	18.24	1 446.9	6.531	44.414
	MMAS	14 379	14 401	14 380.76	6.09	551.0	2.558	45.161
	SACO	14 379	14 379	14 379.00	0.00	1 780.0	8.008	44.443
d 198(15 780)	MMAS$+s_{gb}$	15 888	16 229	16 040.2	104.95	6 436.1	108.759	168.546
	SACO$+s_{gb}$	15 846	16 219	15 943.68	104.50	6 626.1	112.396	169.416
	MMAS	15 956	16 042	15 979.72	17.68	5 175.6	90.110	173.747
	SACO	15 841	16 073	15 910.24	58.24	6 740.0	114.538	169.641
pr 264(49 135)	MMAS$+s_{gb}$	49 536	51 673	50 326.72	605.87	6 284.0	190.536	302.759
	SACO$+s_{gb}$	49 135	50 403	49 266.40	282.54	6 259.3	195.446	311.688
	MMAS	49 235	51 255	49 839.24	609.82	6 007.7	182.175	302.858
	SACO	49 135	50 516	49 313.80	367.60	6 691.5	209.043	311.940

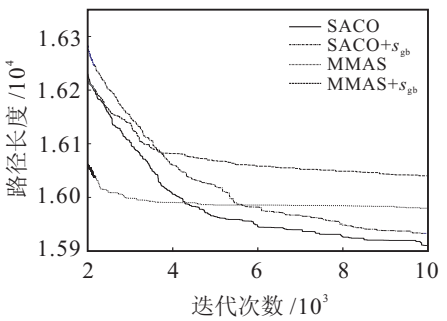
法是较为稳定的. 即使标准差差别较大的 d 198 算例, SACO 所获得的解的质量无论是最优值还是平均值均远好于 MMAS 算法. 另外, 在完成时间方面, 使用单解更新信息素的算法要快于使用交替更新策略的算法. 除算例 pr 264 外, SACO 算法快于 MMAS 算法, 但 SACO 算法首遇迭代次数高于 MMAS 算法, 这主要是由于在 SACO 算法中挥发系数 ρ 的取值较小, 而信息素又初始化为相同的值, 算法需执行较长的一段时间后, 各边上的信息素的差异才会变得明显, 即 SACO 算法的路径探索阶段需较长的时间. 为了更有效地说明算法的进化过程, 这里给出 4 种算法对于算例 d 198 的进化曲线和多样性曲线, 如图 1 和图 2 所示. 其中: 图 1(a) 为搜索前期进化曲线, 图 1(b) 为搜索后期进化曲线, 多样化的定义可参见文献 [7].

从图 1(a) 可以看出, 两种 SACO 算法的初始解的质量优于两种 MMAS 算法. 随着算法的进行, 两种 MMAS 算法对应的曲线快速下降, 而两种 SACO 算法对应的曲线下降则比较缓慢, 在大约 300 次迭代后超过 SACO 算法. 这说明 MMAS 算法的收敛速度快于 SACO 算法. 由图 2 多样性的值也可以看出, SACO 算法初始阶段多样性值低于 MMAS 算法, 但 MMAS 算法的挥发系数较大, 信息素的差异变化也较快, 算法能较快地进入 s_{gb} 区域, 即进入路径开发阶段.

从图 1(b) 可以看出, 两种 SACO 算法对应的“锯齿”较多, 说明算法仍具有较强的探索能力, 而两种 MMAS 则明显地从路径探索阶段进入路径开发阶段, 探索能力明显减弱. 在大约 5 000 次迭代以后, 两种 SACO 算法得到优解的质量超过了两种 MMAS 算法, 这说明两种 MMAS 算法已经陷入局部最优值, 而两种 SACO 算法此时仍能探索到更好的局部优解, 从而获得最终优解的质量高于 MMAS 算法. 这主要是因为 SACO 算法中所采用的信息素更新量的特殊性. 在 MMAS 中, 信息素的更新量是用局部优解的函数值作为自变量. 在初始阶段, 由于算法探索能力较强, 得到局部优解的函数值差异很大, 使得收敛速度加快, 但后期由于算法探索能力减弱, 得到局部优解的函数值差异较小, 致使更新量的差异也逐渐变小, 进而导致算法进入稳定状态, 陷入局部极值. 然而, 在 SACO 算法中, 信息素的更新量是一个固定值, 算法收敛速度恒定, 算法初期能够获得更多的有用信息, 保

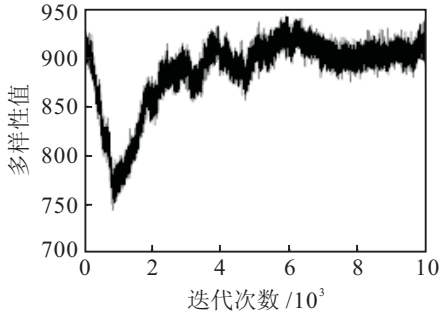


(a)　收索前期进化曲线

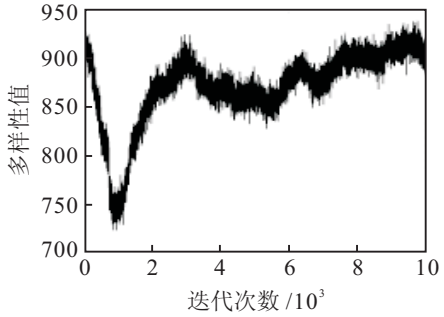


(b)　收索后期进化曲线

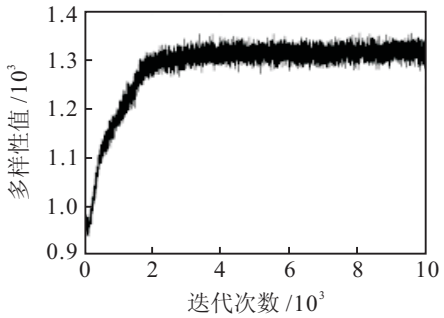
图 1　4 种算法的进化曲线



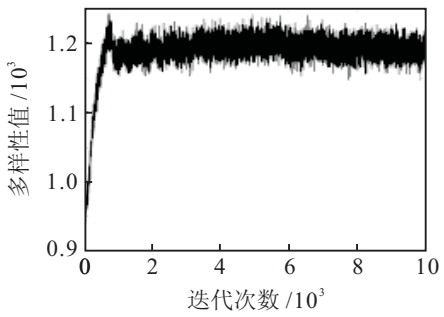
(a) SACO多样性曲线



(b) SACO+s_{gb}多样性曲线



(c) MMAS多样性曲线



(d) MMAS+s_{gb}多样性曲线

图 2 4种算法的多样性曲线

证算法后期仍具有一定的探索能力. 另外, 由图 2 可以看出, 两种MMAS算法的多样性值大于两种SACO算法, 这说明MMAS算法保持解的多样性的能力强于SACO算法, 但最终求解质量不如SACO算法高. 这说明保持算法多样性只是算法能够取得好的效果的必要条件, 这与文献[11]中的结论一致.

5 结 论

本文基于MMAS算法提出了一种简化蚁群算法. 该算法将信息素的上下界和更新量设置为与目标函数值无关的常数, 从而解决了MMAS算法中信息素下界难以确定的问题, 并证明了简化蚁群算法具有线性不变性和平移不变性. 最后以典型TSP问题为例进行了仿真实验, 实验结果表明SACO算法的求解质量高于MMAS算法, 这也说明了简化策略是可行和有效的.

参考文献(References)

[1] Dorigo M, Stützle T. Ant colony optimization[M]. Cambridge: MIT Press, 2004: 25-46.

[2] 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): 29-41.

[3] Dorigo M, Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem[J]. IEEE Trans on Evolutionary Computation, 1997, 1(1): 53-66.

[4] Stützle T, Hoos H H. Max-min ant system[J]. Future Generation Computer Systems, 2000, 16(9): 889-914.

[5] 张晓霞, 唐立新. 一种求解TSP问题的ACO&SS算法设计[J]. 控制与决策, 2008, 23(7): 762-766.
(Zhang X X, Tang L X. An ACO&SS algorithm for traveling salesman problem[J]. Control and Decision, 2008, 23(7): 762-766.)

[6] 熊伟清, 魏平. 二进制蚁群进化算法[J]. 自动化学报, 2007, 33(3):259-264.
(Xiong W Q, Wei P. Binary ant colony evolutionary algorithm[J]. Acta Automatica Sinica, 2007, 33(3): 259-264.)

[7] 黄国锐, 曹先彬, 王煦法. 基于信息素扩散的蚁群算法[J]. 电子学报, 2004, 32(5): 865-868.
(Huang G R, Cao X B, Wang X F. An ant colony optimization algorithm based on pheromone diffusion[J]. Acta Electronica Sinica, 2004, 32(5): 865-868.)

[8] Blum C, Dorigo M. The hyper-cube framework for ant colony optimization[J]. IEEE Trans on Systems, Man and Cybernetics, Part B, 2004, 34(2): 1161-1172.

[9] Birattari M, Pellegrini P, Dorigo M. On the invariance of ant colony optimization[J]. IEEE Trans on Evolutionary Computation, 2007, 11(6): 732-742.

[10] Matthews D. Improved lower limits for pheromone trails in ant colony optimization[C]. Parallel Problem Solving from Nature-PPSN X. Dortmund: Springer Press, 2008: 508-517.

[11] Ugur A, Aydin D. An interactive simulation and analysis software for solving TSP using ant colony optimization algorithms[J]. Advanced Engineering Software, 2009, 40(5): 341-349.