

引用格式:张海南,游晓明,刘升. 动态调度策略与竞争机制融合的蚁群优化算法[J]. 微电子学与计算机,2019,36(7):36-42.
[ZHANG H N, YOU X M, LIU C. Ant colony optimization algorithm based on dynamic scheduling strategy and competition mechanism[J]. Microelectronics & Computer, 2019, 36(7): 36-42.]

动态调度策略与竞争机制融合的蚁群优化算法

张海南,游晓明,刘 升

(上海工程技术大学 电子电气工程学院,上海 201620)

摘 要: 针对蚁群算法收敛速度较慢与易陷入局部最优的问题,提出了动态调度策略与竞争机制融合的蚁群优化算法. 该算法重点引入调度策略,随迭代时期的变化,通过反馈系数将路径信息实时反馈给调度算子,引导蚂蚁动态选择路径,在广阔的空间中充分探索最优解,避免蚁群陷入局部最优. 另将蚁群分为双子群竞争搜索最优解,并给予不同的激励,从而平衡算法的多样性与收敛速度. 通过 14 个经典旅行商问题(Traveling Salesman Problem)实例进行验证,该算法能以较少的迭代次数取得最优解或接近最优解,表明算法的有效性及其优越性.

关键词: 动态调度策略;竞争机制;反馈系数;调度算子;激励函数;蚁群优化算法

中图分类号: TP301

文献标识码: A

文章编号: 1000-7180(2019)07-0036-07

Ant colony optimization algorithm based on dynamic scheduling strategy and competition mechanism

ZHANG Hai-nan, YOU Xiao-ming, LIU Sheng

(College of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China)

Abstract: Aiming at the problem that the ant colony algorithm has slow convergence speed and easy to fall into local optimum, an ant colony optimization algorithm combining dynamic scheduling strategy and competition mechanism is proposed. The algorithm focuses on introducing the scheduling strategy. The algorithm focuses on introducing the scheduling strategy, with the change of the iteration period, the path information is fed back to the scheduling operator in real time through the feedback coefficient, guiding the ants to dynamically select the path, and fully exploring the optimal solution in the broad space to avoid the ant colony falling into the local optimum. In addition, the ant colony is divided into two sub-group competitive search optimal solutions, and different incentives are given to balance the diversity and convergence speed of the algorithm. The algorithm is verified by 14 examples of Traveling Salesman Problem, the algorithm can obtain the optimal solution or the near optimal solution with fewer iterations, indicating the effectiveness and superiority of the algorithm.

Key words: dynamic scheduling strategy; competition mechanism; feedback coefficient; scheduling operator; incentive function; ant colony optimization algorithm

1 引言

蚁群算法(ACO)是由意大利学者 Dorigo 等^[1]

在对自然界真实蚁群觅食行为研究的基础上,提出的一种具有高度创新性的启发式算法,具有天然的并行性、较强的鲁棒性,易与其他优化方法结合. 它

的应用已经从 TSP 问题扩展到组合优化领域的各个方面,包括二次分配问题、机器人路径规划^[2]、网络路由选择问题^[3]等。但蚁群算法存在搜索时间长,易陷入局部最优解等不足,从而影响了 ACO 算法的实际应用。

针对如何平衡解的多样性和收敛速度两者之间的矛盾这一问题,学者们纷纷提出了各种改进策略。文献[4]提出遗传增强蚁群优化算法,维护历史最优解,将蚂蚁搜索解与遗传算法搜索解合并选择当前最优解,防止蚁群优化陷入局部最优,仿真实验表明算法的有效性;但存在的问题是实验测试数据集较少,算法性能测试不够全面。文献[5]提出一种新信息素挥发因子的自适应动态更新策略,对信息素进行动态更新,并采用一种新的量子旋转门对量子概率幅值的收敛趋势进行改变,实验证明算法有较快的收敛速度;但该算法只与传统蚁群算法进行比较,没有与更多的改进算法作对比,有一定的不足。文献[6]选择以信息素矩阵进行交互的策略,建立了并行蚁群算法模型,实验表明该模型可以有效地提高算法性能;但存在的问题是在面对较大规模的 TSP 问题时,通信损耗占运算时间比值稍大。文献[7]提出一种改进信息素二次更新局部优化算法,对当前最优解的子路径进行筛选,对其进行信息素的二次更新,另外提出随机插入法对局部最优解中城市的排序进行调整,跳出局部最优;但存在的问题是随机插入法本身具有随机性,算法跳出局部最优的能力较弱。文献[8]提出一种局部重启策略,在动态 TSP 中的节点替换期间,重置空间从而能够有效地搜索最优解;但该算法的计算复杂度较高,不能够快速收敛到最优解,耗时较长。文献[9]将粒子群算法和蚁群算法相结合,对参数进行匹配,筛选出最佳参数组合,再寻找最优解;问题在于算法的时间复杂度较高,计算成本较大。文献[10]提出了动态调整选择策略的改进蚁群算法,通过适当刺激蚂蚁尝试具有较弱信息素的解,以提高所得解的全局性,但迭代次数较多,收敛速度较慢。

针对以上蚁群算法中存在的问题,本文提出了动态调度策略与竞争机制融合的蚁群算法。

2 基本蚁群优化算法

20 世纪 90 年代 Dorigo 根据自然界蚂蚁的行为提出了 Ant System(AS),由于最初用以求解 TSP 问题,故这里结合求解 TSP 问题来介绍基本 ACO 数学模型。

设拟求解的 TSP 中共有 n 座城市,在算法的初始时刻,将 m 只蚂蚁随机的放到 n 座城市,同时,将每只蚂蚁的禁忌表的第一个元素设置为它当前所在的城市。此时各路径上的信息素量相等,记为 $\tau_{ij} = c$ (c 为一较小的常数)。接下来,每只蚂蚁根据路径上残留的信息素量和启发式信息(两城市间的距离)独立地选择下一座城市,在时刻 t ,蚂蚁 k 从城市 i 转移到城市 j 的概率 $p_{ij}^k(t)$ 为

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{s \in J_k(i)} [\tau_{is}(t)]^\alpha [\eta_{is}(t)]^\beta}, & j \in J_k(i) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

式(1)中, $J_k(i) = \{1, 2, \dots, n\} - \text{tabu}_k$ 表示蚂蚁 k 下一步允许选择的的城市集合。列表 tabu_k 记录了蚂蚁 k 当前走过的城市,即禁忌城市。当所有 n 座城市都加入到 tabu_k 中时,蚂蚁 k 便完成了一次周游,此时蚂蚁 k 所走过的路径便是 TSP 的一个可行解。式(1)中, $\eta_{ij}(t)$ 是一个启发式因子,表示蚂蚁从城市 i 转移到城市 j 的期望程度。在 AS 算法中, $\eta_{ij}(t)$ 通常取城市 i 与城市 j 之间距离的倒数。 α 和 β 分别表示信息素和启发式因子的相对重要程度。

当所有蚂蚁完成一次周游后,各路径上的信息素根据式(2)、式(3)更新:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij} \quad (2)$$

$$\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k \quad (3)$$

式中, ρ ($0 < \rho < 1$) 表示路径上信息素的蒸发系数; $1 - \rho$ 表示信息素的残留系数; $\Delta\tau_{ij}$ 表示本次迭代结束后边 ij (城市 i 到城市 j) 上信息素的增量, $\Delta\tau_{ij}^k$ 表示第 k 只蚂蚁在本次迭代中留在边 ij 上的信息素量。如果蚂蚁 k 没有经过边 ij , 则 $\Delta\tau_{ij}^k$ 的值为 0。即 $\Delta\tau_{ij}^k$ 表示为

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k} & \text{if the } k \text{ ant used edge}(i, j) \text{ in its tour,} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

式中, Q 表示信息素强度; L_k 为第 k 只蚂蚁在当前循环中所走路径长度。

3 动态调度策略与竞争机制融合的蚁群优化算法

3.1 动态调度策略

在地铁交通系统中,通过对列车的运行速度,停靠站时间以及各线路的乘客数量进行实时监测,对

列车进行全面的调度控制^[11]. 例如在早晚高峰时期,通过监测乘客数量的动态变化,加开列车班次,缩短列车停靠站时间,来缓解地铁交通线路压力. 蚁群是一个动态集群,每只蚂蚁通过信息素和启发式信息的引导选择节点,构建一条完整的路径,因此具有一定的随机性. 通过实验研究发现,在算法的迭代前期,蚂蚁在选择路径时更容易受启发式信息的影响,表明节点间长度较短的路径被选择的概率较大. 在小规模的 TSP 问题中,节点数量较少,短路径长度在全局最优路径组合中占比较大;在中大规模 TSP 问题中,节点数量多,优先选择节点间的短路径,最终得出的路径往往不是全局最优路径组合,即局部最优路径.

如何引导蚂蚁在迭代期内合理选择路径,避免陷入局部最优,是亟待解决的问题. 受地铁交通中列车调度的启发,本文提出了一种动态调度改进策略. 通过设置反馈系数收集 (i, j) 边上的信息,包括路径长度和走过的蚂蚁数量,并将信息反馈给调度算子,调度算子根据当前的迭代期,引导蚂蚁合理选择 (i, j) 边. 调度策略结构如图 1 所示.

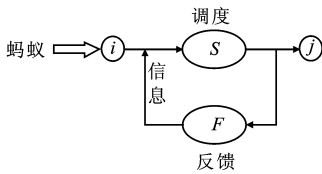


图 1 调度结构图

(1) 设置反馈系数:

$$F = \sum_{a \in A(i, j)} \frac{d(a) - d_{\text{mid}}}{d_{\text{max}} - d_{\text{min}}} \quad (5)$$

式中, $A(i, j)$ 表示选择 (i, j) 边所有蚂蚁集合; d_{max} 和 d_{min} 分别表示节点之间的路径长度的最大值和最小值; d_{mid} 表示两者的均值.

反馈系数 F 是调度算子 S 的一个重要参数,对当前节点 i 和下一节点 j 的动态变化进行监测,收集路径上的信息,将信息实时反馈给调度算子 S .

(2) 设置调度算子:

$$S = 1 - \lg \frac{2NC}{NC_{\text{max}}} \cdot F \quad (6)$$

式中, NC 为当前迭代次数; NC_{max} 为算法最大迭代次数.

调度算子 S 依据 F 反馈的信息,调度蚂蚁选择下一节点 j ,并随着迭代时期的变化,动态调整蚂蚁选择路径的概率.

本文构建的调度策略,作用于整个迭代周期,通

过引导蚂蚁合理选择路径,不仅可以避免陷入局部最优,使得算法过早收敛,又能够提高解的质量. 在迭代前期,适当的调度蚂蚁选择较长路径,能够使蚂蚁充分探索解空间,寻找全局最优解;在迭代后期,调度蚂蚁相对集中地选择短路径,使其快速收敛到最优解. 全面分析可知,动态调度策略在蚁群寻优的过程中,具有一定的积极作用.

3.2 双子群竞争机制

为使算法能够更快的寻找到全局最优解,本文建立了双子群竞争机制. 将蚁群分为数量相等的 X_1 、 X_2 两个子群并行搜索解空间,当满足一定条件时,将各自找到的最优解比较排名(Rank),得出优胜解. 竞争获胜子群(Winning Subgroup, WS)在下次 Rank 前继续进行搜索,失败子群(Failure Subgroup, FS)启用跟随策略,向 WS 学习,自适应调整搜索最优解的方向,提高 FS 的竞争力. 在下次竞争时,如果两个子群的排名发生变化,启动激励机制,给予 WS 正向激励,给予 FS 反向激励,竞争机制流程图如图 2 所示.

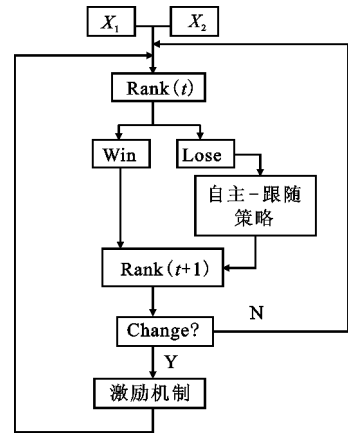


图 2 竞争机制流程图

3.2.1 自主-跟随搜索策略

X_1 、 X_2 两个子群并行搜索一定代数后,当满足 $NC \% D = 0$ 时(设 $D = 9$,若城市规模大于 200,设 $D = 8$),对 X_1 、 X_2 找到的最优解进行排名. FS 启动跟随策略,在下次迭代时,对选择的节点 j 位置进行更新. 位置更新如式(7).

$$\begin{aligned} \varphi_{i,j}(t+1) = & \varphi_{i,j}(t) + \omega(\varphi_{\text{rand}} - \varphi_{i,j}(t)) \\ & + \lambda(\varphi_{\text{best}} - \varphi_{i,j}(t)) \end{aligned} \quad (7)$$

式中, $\varphi_{i,j}(t)$ 为当前位置; φ_{rand} 为随机位置; φ_{best} 为 WS 最优位置,其中 ω 、 λ 分别为自主搜索权重和跟随搜索权重.

自主搜索权重 ω 和跟随搜索权重 λ 设定如式

(8)、(9) 所示:

$$\omega = (\omega_{\max} - \omega_{\min}) \exp((NC_{\max} - NC) \wedge (r - 1) / NC_{\max} \wedge r) + \omega_{\min} \quad (8)$$

式中, $\omega_{\max} = 1.2$; $\omega_{\min} = 0.3$; r 为控制因子, 设 $r = 6$

$$\lambda = \begin{cases} \lambda \cdot \frac{NC_{\max} + NC}{NC_{\max}} & \lambda < \lambda_{\max} \\ \lambda_{\max} & \lambda \geq \lambda_{\max} \end{cases} \quad (9)$$

式中, $\lambda_{\max} = 0.7$.

在 Rank 后, FS 按照公式(7)进行位置更新, ω 和 λ 共同作用, 调整 FS 的搜索方向, 若 λ 偏大, 则 FS 向着 WS 方向进行搜索; 若 ω 偏大, 则更倾向于自主寻优. 随着迭代次数的增加, 本文将 ω 和 λ 设定为自适应权重, 合理调整搜索的倾向程度, 提高 FS 的竞争力.

3.2.2 动态激励模型

在第 n 次竞争之后 ($n > 1$), 若排名发生变化, 启动激励机制. 为了衡量两个子群的竞争差距, 设定质量因子 G 如下式所示:

$$G = \frac{|L_{1\text{best}} - L_{2\text{best}}|}{m \cdot \max\{L_{\max} - L_{\text{aver}}, L_{\text{aver}} - L_{\min}\}} \quad (10)$$

式中, $L_{1\text{best}}$ 为 X_1 当前迭代最好解; $L_{2\text{best}}$ 为 X_2 当前迭代最好解; $L_{\max}$ 为双子群在本次竞争中搜索的最差解; $L_{\min}$ 为双子群在本次竞争中搜索的最好解; m 为蚂蚁数量.

衡量得出两个子群解的质量差距, 做出相应的反馈, 定义激励函数模型如下:

$$M = \begin{cases} M_1 & G \in [0, \frac{\delta}{h}) \\ M_2 & G \in [\frac{\delta}{h}, \delta] \end{cases} \quad (11)$$

式中, δ 为衡量因子; h 为常数, M 初始值: $M = 2m / \sum_{k=1}^m L_k$; $M_1 = 0.6M$; $M_2 = 0.8M$. 根据 X_1 、 X_2 各自的最优解求解出质量因子, 选择合适的激励值.

基于激励机制的信息素更新公式如下:

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij} \pm M \quad (12)$$

式中, 正向激励为 $+M$; 反向激励为 $-M$. 其中, 当前最优路径在 (i, j) 边上的信息素增量 $\Delta\tau_{ij}$ 为:

$$\Delta\tau_{ij} = \begin{cases} 1/L_{gb} & \text{if } (i, j) \in \text{global best road} \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

通过双子群竞争机制, 利用自主-跟随搜索策略调整 FS 的搜索方向, 既达到了向 WS 学习的目的, 又保留自主搜索到最优解的可能性, 加强蚂蚁的探索能力, 从而增加算法的多样性. 当排名发生变化, 通

过质量因子衡量两个子群当前最优解的差距, 给予合适的激励, 强化最优路径, 加快算法的收敛速度.

3.3 算法流程

Algorithm: DSACO-CM for the TSP

1. Procedure DSACO-CM()

2. Begin:

3. initialize the pheromone and the parameters;

4. while(termination condition not met do)

5. for iter1←1 to nc do

6. for i←1 to m do # mas ants' number

7. for j←2 tondo # nas cities' number

8. Construct ant solutions

9. end for

10. end for

11. if nc%D=0

12. Adopt competitive mechanism

13. judge winner and loser

14. loser←auto-follow strategy

15. Update pheromone←incentive mechanism

16. else

17. Update pheromone

18. end if

19. end for

20. end while

4 仿真实验

为了验证本文算法(DSACO-CM)的有效性, 选取 TSPLIB 标准库中 14 个经典实例进行实验仿真. 硬件环境为酷睿 i5-7400M, 内存为 8GB DDR3, 硬盘为 1TB, 操作系统采用 Windows 10, 通过 MATLAB R2016a 软件对小规模以及中大规模的 TSP 城市都进行了较为详细的对比实验.

本文选取 ACS 算法和文献[12]中 PSO-M-ACO 算法进行对比实验, ACS 是蚁群算法中应用最为广泛的算法之一, 并且在 ACO 算法中, 是改进较为出色的算法. PSO-M-ACO 算法是粒子群与蚁群结合的改进算法, 算法性能优越. 各算法的参数设置如表 1, ACS 参数设置参考文献[1], PSO-M-ACO 参数设置参考文献[12].

表 1 参数设置

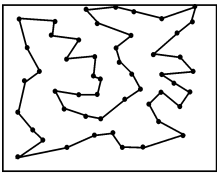
参数名	DSACO-CM	PSO-M-ACO	ACS
α	1	1.5	1
β	5	2	4
ρ	0.3	0.7	0.3
Q	6	—	—
D	[8, 9]	—	—
δ	0.6%	—	—
h	3	—	—
$NC_{\max}$	2 000	2 000	2 000
ω	—	0.9	—
m	50	—	50

本文从两个方面来说明 DSACO-CM 算法体现出的效果,一方面从解的多样性角度,在相同的迭代次数下,比较三种算法找到的最优解大小以及标准最优解的偏差值;另一方面从收敛速度进行对比,即找到最优解的时间消耗.除了进行对比实验外,本文还将 DSACO-CM 算法与不加动态调度策略的算法(ACO-CM)以及 ACS 算法进行对比实验,探讨动态调度策略对本文算法性能的影响.

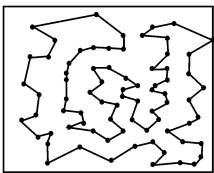
4.1 问题求解及对比

为了较好的比较算法的性能,从解的质量、收敛速度、平均解这三个方面进行比较,分别用 DSACO-CM、PSO-M-ACO、ACS 对 TSPLIB 中 14 个典型实例进行 20 次实验,实验结果如表 2.在表 2 中,标准最优解为 TSPLIB 实例中当前最优路径长度,最优解为 20 次实验所求得的最优解,平均解为 20 次实验所求得最优解的平均值,偏差值为最优解与当前最优解的偏差程度,迭代次数为算法取得最优解所需要的迭代次数.

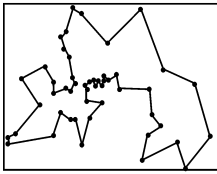
根据表 2 实验所得数据,选取几个实例进行详细说明.对于 kroA200;DSACO-CM 算法找到最优解 29 368,PSO-M-ACO 算法和 ACS 算法都没有找到最优解,且 DSACO-CM 算法最大迭代次数为 405,远小于其他两种算法,表明本文算法寻得最优解的质量和收敛速度都相对较好;对于 kroB150;DSACO-CM 算法和 PSO-M-ACO 算法都能找到最优解,但 DSACO-CM 在 162 代找到最优解,收敛速度较快,而 PSO-M-ACO 在 505 代找到最优解,收敛速度较慢.对于 Lin318;DSACO-CM、PSO-M-ACO 和 ACS 找到的最优解与标准解虽然都有误差,但相比发现,DSACO-CM 与最优解的偏差值最小,且以较小的迭代次数搜索到最优解,优于所对比的算法.综上分析,DSACO-CM 跳出局部最优能力较强,算法的收敛速度相对较快.图 3 为 DSACO-CM 在 14 个典型 TSPLIB 实例的最优路径图,各 TSP 问题的标准最优解标注图下.



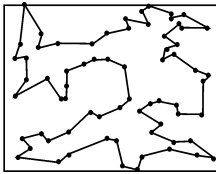
(a) Eil51:426



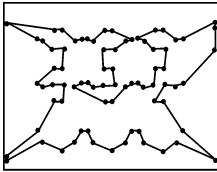
(b) Eil76:538



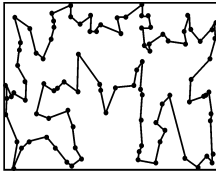
(c) Berlin52:7542



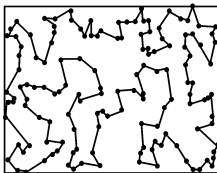
(d) St70:675



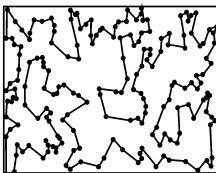
(e) Pr76:108204



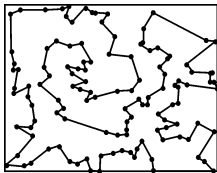
(f) kroA100:21282



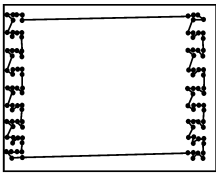
(g) kroA150:26588



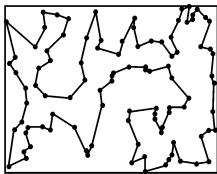
(h) kroA200:29368



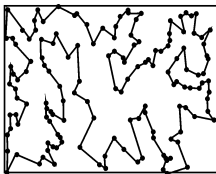
(i) Ch130:6118



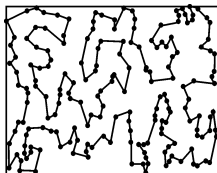
(j) Pr107:44303



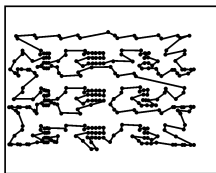
(k) kroB100:22141



(l) kroB150:26130



(m) kroB200:29505



(n) Lin318:42494

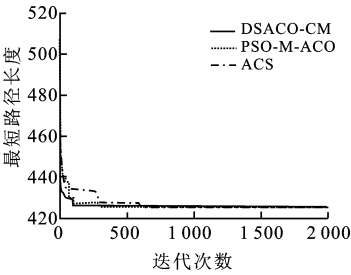
图 3 最短路径

4.2 收敛速度对比

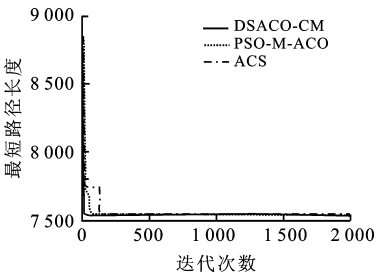
为更加直观地对比各个算法的性能,在 14 个 TSPLIB 实例中选取不同规模的城市如:Eil51、Berlin52、kroB150、Ch130、和 kroA200,比较算法的收敛速度,如图 4 所示.

表 2 算法性能对比

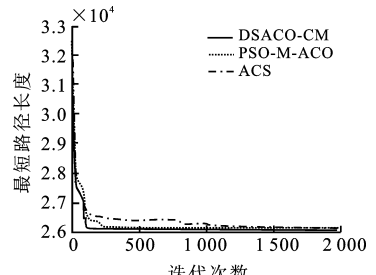
TSP 实例	标准解	算法	最优解	平均解	偏差/%	迭代次数
Eil51	426	DSACO-CM	426	427.7	0.00	94
		PSO-MACO	426	428.1	0.00	264
		ACS	426	428.3	0.00	598
Eil76	538	DSACO-CM	538	540.5	0.00	411
		PSO-MACO	538	548.5	0.00	607
		ACS	541	544.5	0.56	364
Berlin52	7 542	DSACO-CM	7 542	7 565.2	0.00	30
		PSO-MACO	7 542	7 591.7	0.00	53
		ACS	7 542	7 601.3	0.00	136
St70	675	DSACO-CM	675	680.0	0.00	532
		PSO-MACO	675	684.5	0.00	634
		ACS	677	692.1	0.29	710
Pr76	108 159	DSACO-CM	108 204	109 706.5	0.04	490
		PSO-MACO	108 473	111 456.3	0.29	875
		ACS	109 866	111 363.4	1.57	384
Pr107	44 303	DSACO-CM	44 303	44 601.0	0.00	580
		PSO-MACO	44 326	44 694.5	0.05	738
		ACS	44 497	44 784.0	0.44	1 145
kroA100	21 282	DSACO-CM	21 282	21 304.0	0.00	189
		PSO-MACO	21 282	21 994.0	0.00	347
		ACS	21 282	22 407.8	0.00	885
kroA150	26 524	DSACO-CM	26 588	26 840.2	0.24	341
		PSO-MACO	26 648	26 963.1	0.46	583
		ACS	26 740	27 045.6	0.81	1 252
kroA200	29 368	DSACO-CM	29 368	29558.2	0.00	405
		PSO-MACO	29 410	30 015.4	0.14	530
		ACS	29 594	30 205.3	0.76	1 231
kroB100	22 141	DSACO-CM	22 141	22 234.7	0.00	279
		PSO-MACO	22 184	22 308.5	0.19	546
		ACS	22 304	22 358.0	0.73	932
kroB150	26 130	DSACO-CM	26 130	26 647.2	0.00	162
		PSO-MACO	26 130	26 678.3	0.00	505
		ACS	26 161	26 587.0	0.12	1 022
kroB200	29 437	DSACO-CM	29 505	29 956.2	0.23	651
		PSO-MACO	29 872	30 165.2	1.47	763
		ACS	29 624	29 996.5	0.63	1 069
Ch130	6110	DSACO-CM	6 118	6 199.6	0.13	208
		PSO-MACO	6 151	6 204.3	0.67	646
		ACS	6 195	6 306.0	1.39	1 653
Lin318	42 029	DSACO-CM	42 494	42 940.1	1.10	507
		PSO-MACO	43 581	44 027.8	3.69	981
		ACS	45 725	46 297.1	8.79	1 421



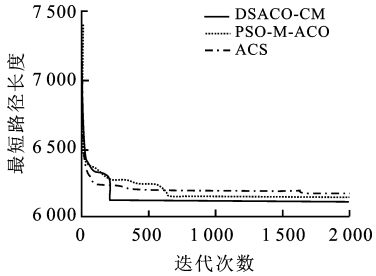
(a) Eil51



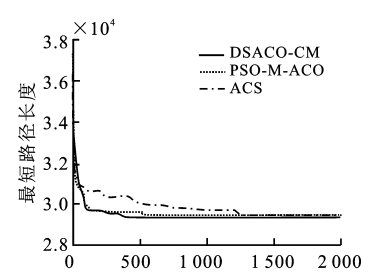
(b) Berlin52



(c) kroB150



(d) Ch130



(e) kroA200

图 4 收敛速度对比

在图 4 中,对 Eil51、Berlin52 分析可知,DSACO-CM、PSO-M-ACO、和 ACO 的最优解都等于标准最优解,但是 DSACO-CM 的收敛速度优于另外两个算法;对于 kroA200 和 kroB150,DSACO-CM 能够以最少的迭代次数搜索到标准最优解;对于 Ch130 实例,三种算法虽都没有找到最优解,但 DSACO-CM 寻得最优解为 6118,与标准解偏差 0.13,PSO-M-ACO 寻得最优解 6151,与标准解偏差 0.67,ACS 寻得解的偏差值最大,且比较三种算法的迭代次数,本文算法迭代次数最少。

结合表 2 可以看出在大多数 TSP 问题上, DSACO-CM 能够以最少的迭代次数找到最优解, 虽然在 st70、pr107、kroB200 和 Lin318 实例中找到最优解所需要的迭代次数较多, 但找到的最优解均优于 PS0-M-ACO 和 ACS. 因此, 从整体上看, DSACO-CM 具有较高的收敛速度, 同时解的多样性相对较好.

4.3 动态调度策略对算法性能的影响

由于在中小规模的 TSP 问题中, 本文算法几乎都能找到最优解, 因此本文为了分析动态调度策略对本文算法性能的影响, 选取几个中大规模城市进行对比实验, 如图 5 和图 6 所示.

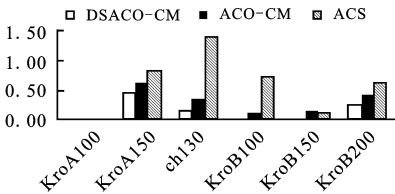


图 5 最优解偏差

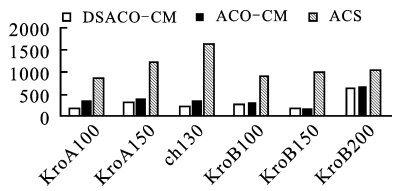


图 6 最大迭代次数

由图分析可知, 在中大规模的 TSP 问题中, 本文算法搜索到的最优解偏差小于 ACO-CM 和 ACS, 并且最大迭代次数小于 ACO-CM 算法和 ACS 算法, 表明动态调度策略提高了本文算法的性能, 解的质量和算法的收敛速度都有一个明显的提升.

5 结束语

在深入研究蚁群优化算法之后, 本文提出了动态调度策略与竞争机制融合的蚁群优化算法. 将调度策略融入到蚁群算法中, 在迭代的不同时期, 依据路径信息的变化, 动态引导蚂蚁选择路径, 从而激发蚁群的探索能力. 并且通过双子群竞争机制, 给予两个子群正向与反向激励, 寻优的效率提高. 经多次在不同规模的 TSP 问题上实验, 表明了本文算法是可行有效的, 对比改进的蚁群算法和传统的蚁群算法, 求解精度和收敛速度都有所提升. 下一步工作是继续研究竞争机制在多种群算法的应用, 优化调度策略, 从而进一步提高算法的性能.

参考文献:

[1] DORIGO M, MANIEZZO V, COLORNI A. Ant system: optimization by a colony of cooperating agents [J]. IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics A Publication of the IEEE Systems Man & Cybernetics Society, 1996, 26(1):29-31.

[2] 赵娟平, 高宪文, 刘金刚, 等. 移动机器人路径规划的参数模糊自适应窗口蚁群优化算法[J]. 控制与决策, 2011, 26(7):1096-1100.

[3] MAHMOOD K, KAMAL S, KHAN H M. Dynamic optimization of network routing problem through ant colony optimization (ACO)[J]. IEEE Transactions on Cybernetics, 2012, 28(5):629-640.

[4] 梁豪默, 王智学, 刘艺. 遗传增强蚁群优化算法[J]. 微电子学与计算机, 2018, 35(8):101-104.

[5] 万正宜, 彭玉旭. 求解旅行商问题的改进型量子蚁群算法[J]. 计算机工程与应用, 2016, 52(22):59-63.

[6] 马继辉, 余明捷, 陈鑫杰, 等. 基于 Hama 并行蚁群算法模型及 TSP 应用研究[J]. 交通运输系统工程与信息, 2016, 16(3):168-173.

[7] 许凯波, 鲁海燕, 程毕芸, 等. 求解 TSP 的改进信息素二次更新与局部优化蚁群算法[J]. 计算机应用, 2017, 37(6):1686-1691.

[8] PRAKASAM A, SAVARIMUTHU N. Novel local restart strategies with hyper-populated ant colonies for dynamic optimization problems[J]. Neural Computing & Applications, 2018(1):1-14.

[9] GÓMEZCABRERO D, RANASINGHE D N. Fine-tuning the ant colony system algorithm through particle swarm optimization[J]. Cancer Research, 2018, 25(25):73-77.

[10] 郑松, 侯迪波, 周泽魁. 动态调整选择策略的改进蚁群算法[J]. 控制与决策, 2008, 23(2):225-228.

[11] 曹弋, 杨忠振, 左忠义. 基于地铁停车换乘的区域差异性停车收费策略[J]. 交通运输系统工程与信息, 2017, 17(3):12-18.

[12] ELLOUMI W, ABED H E, ABRAHAM A, et al. A comparative study of the improvement of performance using a PSO modified by ACO applied to TSP[J]. Applied Soft Computing, 2014, 25(C):234-241.

作者简介:

张海南 男, (1994-), 硕士研究生. 研究方向为智能算法、最优化问题、嵌入式系统. E-mail: zhn2368@163. com.

游晓明 女, (1963-), 博士, 教授. 研究方向为智能系统、分布式并行处理、进化算法.

刘 升 男, (1966-), 博士, 教授. 研究方向为量子启发式进化算法、分布式并行处理、进化算法.