文章编号: 1001-0920(2019)11-2459-10 **DOI:** 10.13195/j.kzyjc.2018.0298

基于虚拟蚂蚁的局部优化蚁群算法

李 俊†, 周 虎, 李 波

(1. 武汉科技大学 计算机科学与技术学院，武汉 430065；
2. 智能信息处理与实时工业系统湖北省重点实验室，武汉 430065)

摘 要: 蚁群算法在解决一些NPC (Non-deterministic polynomial complete) 问题时具有较大的优势,但也存在一些不足,如收敛精度低、收敛速度慢等. 为了平衡收敛精度与收敛速度之间的矛盾,提出一种基于虚拟蚂蚁的局部优化蚁群算法. 该算法通过降低重复计算资源的比例来提高计算资源的利用率,从而提升较少迭代次数时的精度. 对单位信息素和全局更新策略进行调整,使之与所提出的算法匹配. 同时,增加两点局部优化算子 —— 点交换和交叉去除,加快收敛速度,进一步提高解的精度. 通过约束局部优化算子的参数,减少局部优化的计算量,使整体算法的复杂度与基本蚁群算法大致相当. 从最终的实验数据可以得出,所提出的算法在较少迭代次数的情况下可以得出较高的精度,在收敛速度与收敛精度之间实现较好的平衡.

关键词: 虚拟蚂蚁；交叉去除；点交换；单位信息素；平衡矛盾；蚁群算法

中图分类号: TP301.6；TP18 **文献标志码:** A

Local optimization ACO based on virtual ant colony algorithm

LI Jun†, ZHOU Hu, LI Bo

(1. College of Computer Science and Technology，Wuhan University of Science and Technology，Wuhan 430065，China；2. Hubei Province Key Laboratory of Intelligent Infomation Processing and Real-time Industrial System，Wuhan 430065，China)

Abstract: The ant colony algorithm has great advantages in solving some non-deterministic polynomial complete (NPC) problems, but there are some shortages, such as low convergence precision and slow convergence speed. In order to balance convergence accuracy and convergence speed, a local ant colony optimization algorithm based on virtual ants is proposed in this paper. This algorithm improves the efficiency of computing resources by reducing the proportion of repeated computing resources, thus improving the accuracy of less iterations. The unit pheromone and the global updating strategy are adjusted to make the method match. At the same time, the two-point local optimization operator-point exchange and cross removal are added to accelerate the convergence rate and further improve the precision of the solution. By constraining the parameters of local optimization operators , the computation of local optimization is reduced, and the complexity of the whole algorithm is approximately equal to that of the basic ant colony algorithm. From the final experimental data, it can be concluded that the algorithm can obtain higher precision with less iterative times, and a good balance is made between the convergence speed and the convergence precision.

Keywords: virtual ants; cross removal; point exchange; unit pheromone; equilibrium contradiction; ant colony algorithm

0 引 言

蚁群算法最先由意大利学者Dorigo等[1-2]提出,是一种用于寻找优化路径的概率型算法. 该算法具有鲁棒性强[3]、并行计算[4]和启发式搜索等特征,已广泛应用于电路布局[5]、车辆调度[6]、多维背包(MKT)[7]、IP回溯(IPTBK)[8]、旅行商[9-10]等问题中. 但是,蚁群算法也存在一些问题,如收敛精度低、收敛速度慢、容易停滞等问题.

针对这些问题,学者们主要从蚁群算法双要素(蚂蚁、信息素)及其他改进元素等方面对蚁群算法进行改进. 文献[11]采用动态蚂蚁数量解决了收敛速度慢的缺点,即每一次迭代,蚂蚁的数量都在一个范围内取一个随机数,然后进行搜索. 结合该算法,采用动态蚂蚁数量的方法可以快速跳出局部最优解,从而很好地收敛. 文献[12]对蚂蚁进行分类,不同类型的蚂蚁处理不同的工作,通过多类蚂蚁之间的共同协调

收稿日期: 2018-03-15；**修回日期:** 2018-05-23.

基金项目: 国家自然科学基金项目(61572381)；武汉科技大学智能信息处理与实时工业系统湖北省重点实验室基金项目(znxx2018QN06).

责任编委: 魏秀琨.

†通讯作者. E-mail: lijun@wust.edu.cn.

以及相互影响,确认更优路径. 文献[13]对蚁群算法信息素更新规则进行研究,提出了一个基于迭代思想的信息素更新规则. 对信息残留因子进行实验,确定在新的信息素更新规则下信息素挥发系数的最佳合理值,较好地解决了蚁群算法收敛速度慢的问题. 文献[14]提出了基于方向信息素协调的蚁群算法,通过定义方向信息素,增大选择较短路径的概率,同时引入全局选择策略,增加算法的鲁棒性. 文献[15]提出了NACA算法,引入差分演化算法进行局部信息素更新,将全局信息素灾变与轮盘赌算子结合起来以提高求解精度. 文献[16]引入精英策略,只对前n条路径进行信息素的更新,减少不必要的信息素. 文献[17]将城市点按照k-means算法聚类,分类后求各个部分的路径再合并成整个路径.

显然,上述算法主要从收敛精度或者收敛速度一个方面解决问题,因而存在一些弊端. 如:文献[14]提出的基于方向信息素协调蚁群算法的收敛速度提升并不明显;文献[15]提出的新型蚁群算法(NACA)其算法精度的提升幅度不大;文献[16]由于每次迭代时进行蚂蚁路径的排序,而排序的时间复杂度较高,导致整体算法的时间复杂度较高;文献[17]聚类连接的过程时间复杂度较高.

针对收敛精度与收敛速度之间平衡的问题,本文提出虚拟蚂蚁的局部优化蚁群算法,主要思想是:1)当算法接近收敛时会有较多的蚂蚁走之前找到的最优路径,在不影响或较少影响当前最优路径权重的同时,增加蚂蚁扩展寻路的能力,减少寻找重复路径的概率,从而提升计算资源的利用率;2)通过局部优化加快收敛,提升算法的精度.

1 基本蚁群算法

基本蚁群算法是Dorigo等[1]提出的蚁群算法(ACS). 原理是蚂蚁在寻找食物时会留下信息素,信息素可以引导其他蚂蚁的选择,浓度越高的路径,选择的概率越大. 随着时间的推移,信息素会逐渐挥发,留下来的将是路径较优的线路. ACS算法的大致流程如下:

For $k=1$: Iter % Iter为迭代次数
  初始化蚂蚁的位置和信息素
 For $i=1$: m %m为蚂蚁只数
  蚂蚁按照状态转移规则选择路径
  局部更新信息素
 End
全局更新信息素
End.

信息素的更新规则依据下式进行:

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij}(k). \quad (1)$$

其中:τ_{ij}为t时刻点i到点j之间的信息素量,ρ为信息素挥发系数,$\sum_{k=1}^{m}\Delta\tau(k)$为经过路径$ij$的蚂蚁留下来的信息量之和. $\Delta\tau_{ij}$的计算公式为

$$\Delta\tau_{ij}=\begin{cases}Q/L_{\rm Best},\ j\in\text{allowed};\\ 0,\ \text{otherwise}.\end{cases} \quad (2)$$

第k号蚂蚁的路径选择概率公式为

$$P_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum\tau_{ij}^{\alpha}\eta_{ij}^{\beta}},\ j\in\text{allowed};\\ 0,\ \text{otherwise}.\end{cases} \quad (3)$$

2 虚拟蚂蚁的局部优化蚁群算法(VLACO)

本文提出的算法,即虚拟蚂蚁的局部优化蚁群算法(VLACO),通过优化计算资源的利用率来提升部分计算性能. “虚拟”蚂蚁是指那些走当前最优路径的蚂蚁. 该部分蚂蚁较多时,会减少总体蚂蚁的探索次数. 降低“虚拟”蚂蚁的比例是本文算法的主要优化点:让一部分“虚拟”蚂蚁走当前最优路径,给之后的蚂蚁一定的指引;让一部分“真实”蚂蚁进行探索,增加搜索次数,从而提升计算资源的利用率. 次要优化为局部优化和全局优化. 局部优化的主要目的是在迭代中期加快收敛速度;全局优化的作用是对当前最优路径进行加强,从而加快收敛速度.

2.1 虚拟蚂蚁优化

蚁群算法在求解精度和收敛速度上有个矛盾点:较快的收敛速度和较高的精度无法同时得到保证. 为了平衡这一矛盾,本文提出虚拟蚂蚁的概念.

在运用ACO解决TSP问题时发现,迭代一段时间后,蚂蚁在选择路径时,会更有可能走上次的最优路径,即找到的路径会重复,浪费了计算资源. 为了更好地描述该现象,做出如下定义:

定义1 在不考虑其他任何局部路径调整的情况下,如果m只蚂蚁共迭代了N次进行路线的寻找,则计算资源总量量化为$\mathrm{TCR}=m\times N$.

定义2 若某只蚂蚁在寻找路线时,找到的路线为上次的最优路线,则记为重复寻找. m只蚂蚁在N次迭代中的全部重复寻找的数量之和记为重复计算资源,用符号表示为RCR.

在计算资源总量TCR不变的前提下,重复计算资源RCR越大,资源的利用率越小. 在保证当前最优路径权重的基础上,适当减少RCR,便可让更多的计算资源得到合理的利用,增大找到更优解的概率.

记 $P_{\max_i}$ 为点 i 到其他可达点的“最大概率”, P_{total_i} 为除去“最大概率”的概率和. a 为比例系数, x 为待求变量, 则有

$$\frac{P_{\max_i}}{P_{\max_i}+P_{\text{total}_i}}\times a=\frac{P_{\max_i}\times x}{P_{\max_i}\times x+P_{\text{total}_i}}, \tag{4}$$

$$x=\frac{a\times P_{\text{total}_i}}{P_{\text{total}_i}+P_{\max_i}-a\times P_{\max_i}}. \tag{5}$$

a 的取值与城市数有关, 计算公式为

$$a=\sqrt[N]{W}, \tag{6}$$

其中 W 表示原先该走上次最优路线的蚂蚁走其他路线的比例.

算法描述如下:

Step 1: 计算当前的 $P_{\max_i}$ 和 P_{total_i}, 并按式(5)和(6)求出 x.

Step 2: 计算新的 $P_{\text{sum}}(\text{新})=P_{\max_i}\times x+P_{\text{total}_i}$, 当前的 $P_{\max_i}$ 值变为 $P_{\max_i}\times x$ (只计算用, 不更新全局的信息素对应的概率).

Step 3: 其他各点概率不变, 用轮盘赌算法找出下一个选择的节点 next.

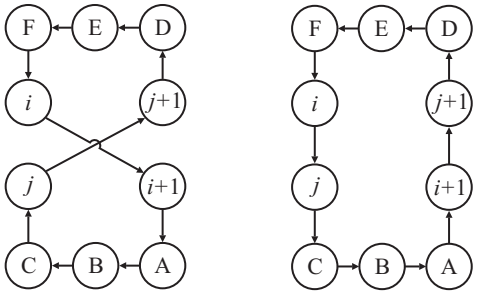
Step 4: 如果路径 (i,next) 不在最优路径上, 则找到最优路径中节点 i 之后的节点 j, 将路径 (i,j) 增加信息素 $\Delta\tau_{ij}\times\dfrac{P_{\max_i}(\text{原})}{P_{\text{sum}}(\text{原})\times N}$; 否则, 什么都不做.

Step 5: 路径 (i,next) 的信息素按正常策略更新.

2.2　局部优化

2.2.1　交叉去除

在寻找路径的过程中会出现如图 1(a) 所示的交叉情况, 而仅依靠信息素较难将交叉去除, 因此需要增加优化步骤, 进行交叉的去除. 文献 [18] 提出了一种交叉去除的思路, 本文将其改进后运用于 TSP 问题中.



(a) 有交叉　　(b) 交叉去除

图 1　交叉去除原理

图 1(a) 为有交叉的情况, 节点的序列为 $i\to i+1\to \text{A}\to \text{B}\to \text{C}\to j\to j+1\to \text{D}\to \text{E}\to \text{F}\to i$. 经优化后, 变成图 1(b) 的情况, 节点序列为 $i\to j\to \text{C}\to \text{B}\to \text{A}\to i+1\to j+1\to \text{D}\to \text{E}\to \text{F}\to i$.

文献 [18] 只是提出了节点顺序的改变, 并未详细说明如何进行交叉的判断, 本文将交叉的判断进行说明.

对于向量 AB, 其他点相对于 AB 的位置有两种情况: 在 AB 的左侧和 AB 的右侧.

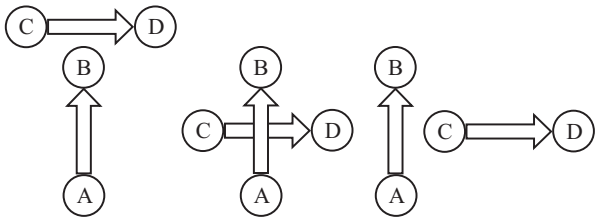
定义 3　若 C 点在以向量 AB 的 A 为原点, 顺时针转过 $(0°,180°)$, 则 C 点在向量 AB 的右侧; 顺时针转过 $(180°,360°)$, 则 C 点在向量 AB 的左侧.

已知向量 AB(x,y), 向量 AM$(y,-x)$, C 为任意一点, 若向量 AC·AM(内积) >0, 则点 C 在向量 AB 的右侧; 若向量 AC·AM(内积) <0, 则点 C 在向量 AB 的左侧.

证明　因为向量 AC·AM(内积) >0, 所以 $\angle\text{MAC}\in[0°,90°)$. 即 $\angle\text{BAC}\in(0°,180°)$, C 在以向量 AB 的 A 为原点, 顺时针转过 $(0°,-180°)$ 的区域内. 所以向量 AC·AM(内积) >0, 于是点 C 在向量 AB 的右侧.

向量 AC· AM(内积) <0 的情况同理可证. □

平面内向量 AB 与 CD 的关系如图 2 所示. 当点 A、点 B 在向量 CD 两侧且点 C、点 D 也在 AB 两侧时, AB 与 CD 相交, 否则, AB 与 CD 不相交. 算法描述中的交叉判断根据此原理进行.



(a) 位置 1　　(b) 位置 2　　(c) 位置 3

图 2　位置情况

算法描述如下.

Step 1: 在当前最优路径上, 选择两条不同的路径段 $(i,i+1)$ 和路径段 $(j,j+1)$.

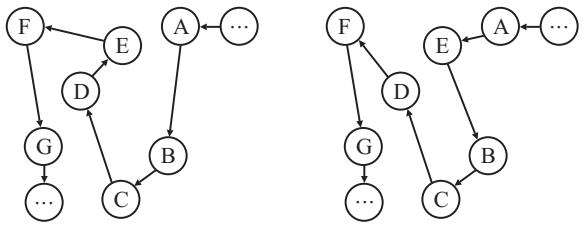
Step 2: 根据定义 3 判断是否交叉, 若交叉, 则执行下一步; 否则, 跳到 Step 4.

Step 3: 交换点 $i+1$ 与点 j; 反转原先 $i+1$ 到 j 的序列 ABC; 衰减 $(i,i+1)$ 和 $(j,j+1)$ 的信息素, 增加 (i,j) 和 $(i+i,j+1)$ 的信息素.

Step 4: 若选择了所有路段, 则结束; 否则, 跳到 Step 1.

2.2.2　点 交 换

在迭代一定次数后, 信息素的累积量较多, 例如图 3(a) 中, 若多次迭代都经过了 AB 路线, 则选择 AE 路线的概率就会小得多. 在比较次优解和最优解的过程中发现, 有些次优解是没有交叉的, 只是与该点相近的某个点进行了交换, 从而达到最优, 如图 3(b) 所示.



(a) 交换前 (b) 交换后

图 3 点交换示意

图3(a)中的序列为$\cdots \to \mathrm{A} \to \mathrm{B} \to \mathrm{C} \to \mathrm{D} \to \mathrm{E} \to \mathrm{F} \to \mathrm{G} \to \cdots$,交换后变为$\cdots \to \mathrm{A} \to \mathrm{E} \to \mathrm{B} \to \mathrm{C} \to \mathrm{D} \to \mathrm{F} \to \mathrm{G} \to \cdots$.

算法描述如下:

Step 1: 选择两个不相同的点i和点j,将节点i置于节点j之前,计算交换后的路径长度L_{after}.

Step 2: 如果L_{after}比之前路径长度小,则保留交换后的路径信息;否则,进行下一步.

Step 3: 是否全部节点均遍历完成,若是,则结束;否则,执行Step 1.

局部优化的两个算子增加了一定的计算复杂性,出于求解速度方面考虑,有两点平衡的方法:一是在迭代初期,交叉的概率较大,此时路径与最优路径相差较大,会使得两种局部优化的最内层循环执行次数较多,此阶段(如迭代前10次)不进行局部优化;二是优化具有确定性,即相同的输入具有相同的输出,只有在路径有变动的情况下才进行局部优化.

2.3 全局优化

2.3.1 单位信息素调整

蚁群算法的单位信息素$\Delta\tau$的更新公式按照式(2)进行. 这样更新信息素,收敛速度较慢,应该加强上次最优路径的权重. 即对于找到的处于上次最优路径的部分路径,信息素应进行加强,而不在上次最优路径上的部分路径正常更新. 这样收敛速度会得到提升. 现对信息素公式作如下调整:

$$\Delta\tau_{ij}=\begin{cases}\dfrac{D_{ij}\times\gamma_1}{L_{\mathrm{best}}},\ i、j在上次最优路径上;\\ \dfrac{D_{ij}\times\gamma_2}{L_{\mathrm{best}}},\ i、j不在上次最优路径上.\end{cases}\tag{7}$$

其中:D_{ij}为节点i到节点j之间的距离;L_{best}为上次最优路径长度;γ_1、γ_2为参数,一般设定γ_1比γ_2大,同时γ_1随迭代次数在(min, max)之间线性改变.

2.3.2 全局更新策略调整

每迭代完一次后,会找到一条最优的路线,如果信息素不做修正,则实际上本次找到的m条路径的权重基本相当. 因为每条线路在留下信息素时是按照相同的策略,所以最优路径的信息素应当加强. 迭代初期,信息素的“累计量”较少,较小的增强会造成较大的影响,而迭代后期恰好相反. 增强量应与迭代次数成正相关. 城市的数量在此也有一定的影响,城市规模越大,相同量的信息素的影响会越小. 增强量与城市也应成正相关. 单位信息素的参数是符合该需求的,因此,最优路径信息素公式更新为

$$\tau_{ij}=\begin{cases}\tau_{ij}+\Delta\tau_{ij},\ i、j在最优路径上;\\ \tau_{ij},\ i、j不在最优路径上.\end{cases}\tag{8}$$

其中$\Delta\tau_{ij}$按式(7)计算.

2.4 整体算法

蚁群算法的模型有多种,如蚁周模型(即每只蚂蚁完成所有节点的遍历后更新信息素)、蚁量模型(即每只蚂蚁选择一个节点便更新一个路径的信息素). 本文算法选择的是蚁量模型,信息素和概率均进行了修改,使之与本文算法相匹配. 信息素使用式(8)进行更新,概率公式增加一个偏移量e(自然对数的底). 概率公式为

$$P_{ij}^{k}=\begin{cases}\dfrac{(\tau_{ij}+\mathrm{e})^{\alpha}\times\eta_{ij}^{\beta}}{\sum(\tau_{ij}+\mathrm{e})^{\alpha}\times\eta_{ij}^{\beta}},\ j\in\mathrm{allowed};\\ 0,\ \mathrm{otherwise}.\end{cases}\tag{9}$$

算法流程见图4.

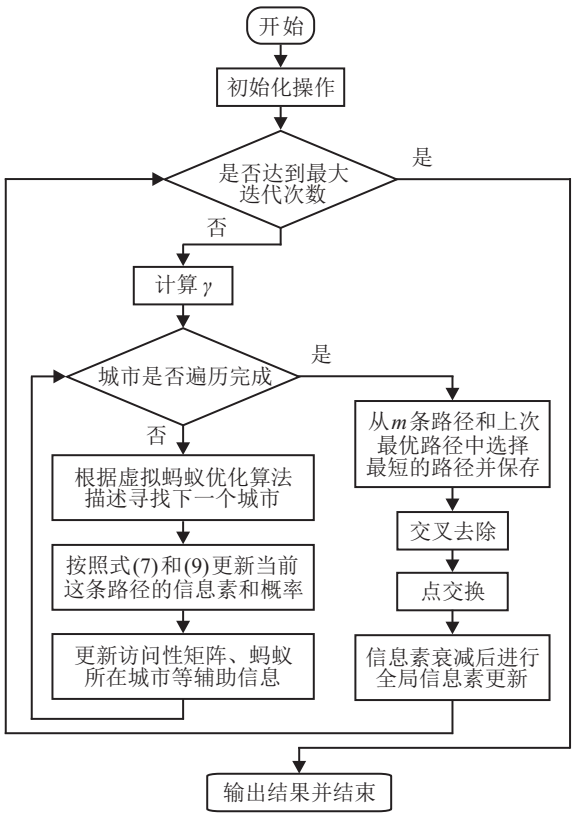


图 4 完整算法流程

3 算法分析

本文的实验环境为Windows 1064位系统, CPU i5-3230 M 2.6 GHz.

3.1 虚拟蚂蚁优化分析

3.1.1 (0-1)对比

本文算法的核心点在于虚拟蚂蚁优化, 因此, 将该部分单独拿出来进行对比实验, 以验证优化效果. 实验中的公共参数设定为$\alpha = 2, \beta = 3, \rho = 0.382$, 迭代次数为200, 蚂蚁只数$m$等于城市数$n$, 虚拟蚂蚁优化中的参数$W = 0.4$. 从TSPLIB[19]中选取30个数据集, 共进行20组独立实验, 实验结果见表1. 其中: TCR和RCR的定义见虚拟蚂蚁优化的说明部分, min.err为20组实验中的最优结果与已知解的误差, avg.err为20组结果的平均值与已知解的误差.

表 1　虚拟蚂蚁优化对比实验结果

数据集	已知解	TCR	优化前			优化后		
			min.err/%	avg.err/%	RCR	min.err/%	avg.err/%	RCR
burma 14	31	2 800	−0.39	0.05	1 315	−0.39	**−0.13**	562
ulysses 22	74	4 400	1.77	**1.87**	1 919	1.77	1.88	204
Oliver 30	424	6 000	−0.06	1.03	3 901	−0.06	**0.71**	1 480
Oliver 31	15 377	6 200	0.00	1.28	2 294	0.00	**0.99**	832
att 48	33 522	9 600	0.26	3.28	5 610	0.26	**1.85**	1 994
eil 51	426	10 200	**1.14**	3.81	5 963	1.47	**3.03**	1 952
berlin 52	7 542	10 400	0.03	4.17	5 810	0.03	**3.57**	2 143
st 70	675	14 000	1.30	5.31	6742	**1.04**	**3.68**	1 947
eil 76	538	15 200	**2.97**	6.60	8 850	3.20	**4.75**	2 607
pr 76	108 159	15 200	**1.61**	5.87	8 390	2.02	**4.42**	2 670
rat 99	1 211	19 800	4.83	8.10	9 889	**1.82**	**5.57**	3 195
kroA100	21 282	20 000	2.17	6.78	8325	**0.18**	**3.25**	1 852
kroB100	22 141	20 000	3.61	7.23	8 993	**0.79**	**3.00**	2 051
kroC100	20 749	20 000	1.93	5.84	9 626	**0.81**	**3.18**	1 321
kroD100	21 294	20 000	2.69	6.30	7 460	**1.56**	**3.18**	1 530
kroE100	22 068	20 000	2.69	6.12	9 618	**0.56**	**3.42**	1 147
eil101	629	20 200	7.35	9.60	9 416	**4.55**	**7.12**	2 382
lin105	14 379	21 000	3.61	8.51	9 611	**0.35**	**3.22**	2 229
pr107	44 303	21 400	0.30	3.04	7 510	**0.20**	**1.84**	1 539
pr124	59 030	24 800	2.67	7.66	14 235	**0.73**	**3.27**	3 792
bier127	118 282	25 400	3.03	9.24	5 632	**1.79**	**5.45**	1 677
ch130	6 110	26 000	3.69	9.70	7 312	**3.25**	**5.79**	1 267
pr136	96 772	27 200	5.55	9.26	6 663	**4.27**	**7.58**	1 846
gr137	699	27 400	5.04	10.66	7 965	**2.08**	**8.43**	1 884
pr144	58 537	28 800	1.06	4.06	6 354	**0.23**	**2.69**	1 961
ch150	6 528	30 000	3.92	8.96	8 540	**2.27**	**5.46**	1 457
kroA150	26 524	30 000	6.29	10.25	8 387	**4.06**	**6.91**	1 041
kroB150	26130	30 000	5.79	10.62	8 569	**3.93**	**6.72**	931
pr152	73 682	30 400	2.58	6.45	6 630	**1.92**	**4.08**	1 128
rat195	2 323	39 000	9.31	12.39	7 036	**5.26**	**8.84**	1 771

由表1中可以看出, 增加了虚拟蚂蚁优化的20组的平均值误差和最优值误差在绝大多数数据上均优于未加虚拟蚂蚁优化的平均误差和最优值误差, 精度的提升幅度也较为可观. 例如对于数据集kroE100, 不进行虚拟蚂蚁优化的平均误差为6.12 %, 最优值误差为2.69 %; 而增加了虚拟蚂蚁优化后, 平均误差降为3.42 %, 最优误差降为0.56 %, 下降幅度分别为44.12 %和79.18 %.

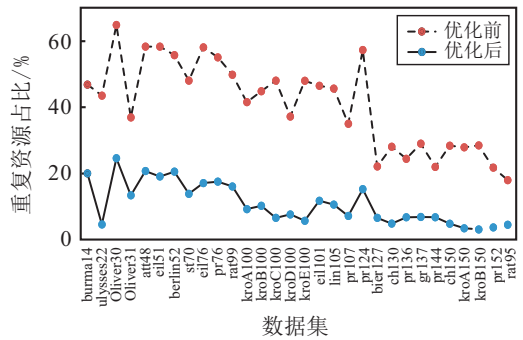


图 5　重复计算资源对比

由图5可以看到, 增加了虚拟蚂蚁优化后重复计算资源的占比会小很多, 在一定程度上提高了计算资源的利用率.

3.1.2 参数对比

虚拟蚂蚁优化的关键在于相对性参数W的设置, 参数W可通过实验数据来设置. 为了较为客观地展现虚拟优化的效果, 实验设置的组数较多. 本实验从TSPLIB中选择30组数据集(att 48, berlin 52, bier 127, burma 14, ch 130, ch 150, eil 101, eil 51, eil 76, gr 137, kroA 100, kroA 150, kroB 100, kroB 150, kroC 100, kroD 100, kroE 100, lin 105, Oliver 30,

Oliver 31, pr 107, pr 124, pr 136, pr 144, pr 152, pr 76, rat 195, rat 99, st 70, ulysses 22), 以步长为0.01对$W(0.01\sim0.99)$参数进行实验,每个数据集独立进行20次,统计30组数据集的最优路径长度之和与30组数据的平均路径之和,然后进行分析. 实验的公共参数为$m=n,\alpha=2,\beta=3,\rho=0.328,\text{Iter}=200$. 图6给出了$W$参数影响曲线,其中

$$\text{Err.avg}=\sum_{i=1}^{30}\text{数据集}i\text{的20组独立实验的平均误差},$$

$$\text{Err.min}=\sum_{i=1}^{30}\text{数据集}i\text{的20组独立实验的最优值误差}.$$

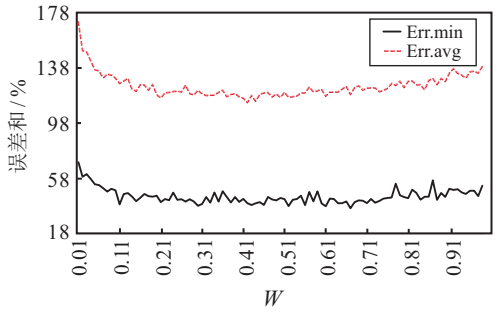


图 6 W 参数影响曲线

由图6可以看出:W参数从0.01变化到0.30左右时,各误差有较为明显的下降,此时可理解为“真实蚂蚁”的数量在增加,因而增加了探索的机会;W参数在0.30～0.75区间时,随着W参数的增长,各误差在小范围内波动,进行探索的“真实蚂蚁”增多时,其留下的信息素会影响后面蚂蚁的寻路,增加干扰,在此范围内干扰较少;当W大于0.75时,各误差有上升的趋势,在此范围的干扰较大. 由此可知,W参数在0.30～0.75之间时优化效果较好.

3.2 优化策略分析

本文算法提出了5点优化策略,为了验证各个优化策略的影响,进行5组独立的实验. 每组实验分别执行20次. 5组实验所使用的公共参数如表2所示.

表 2 公共参数

蚂蚁只数m	ρ	α	β	迭代次数Iter
100	0.382	2	3	100

实验使用的数据集为TSPLIB数据集中的oliver 30、eil 51、st 70、eil 76、lin105、kroA150、kroB150、kroA200、ts 225、gil 262,共10个数据集. 每组实验的编号和相关说明如表3所示. 实验的结果和相关的图示见表4和图7.

表 3 实验说明

编号	优化策略	参数说明
A	全局更新(单位信息素用式(2),Q改为D_{ij})	对于γ_1,$(\min,\max)=(0.4\times N^{1.1}/2, N^{1.1}/2)$
B	全局更新+单位信息素	$\gamma_2=1$,其他同上
C	全局更新+单位信息素+虚拟蚂蚁	$W=0.4$,其他同上
D	全局更新+单位信息素+虚拟蚂蚁+交叉去除	衰减与ρ相同,增加率为120%,其他同上
E	全局更新+单位信息素+虚拟蚂蚁+交叉去除+点交换	同上

表 4 单因子对比数据

数据集	已知最优解	编号	误差/%	平均误差/%	最优解	平均值
oliver 30	423.74	A	0.66	2.83	426.54	435.73
		B	**0.00**	0.83	423.74	427.27
		C	**0.00**	0.58	423.74	426.19
		D	**0.00**	0.76	423.74	426.95
		E	**0.00**	**0.14**	423.74	424.35
eil 51	426.00	A	7.18	14.40	456.60	487.33
		B	1.08	3.69	430.61	441.71
		C	1.04	2.69	430.45	437.44
		D	**0.67**	2.37	428.87	436.11
		E	**0.67**	**2.11**	428.87	434.97
st 70	675.00	A	13.36	17.07	765.20	790.20
		B	2.04	4.75	688.78	707.09
		C	**0.71**	4.08	679.76	702.53
		D	1.12	**3.21**	682.53	696.66
		E	1.37	3.59	684.24	699.24
eil 76	538.00	A	9.21	13.62	587.54	611.27
		B	2.78	5.18	552.98	565.87
		C	1.68	4.23	547.06	560.76
		D	1.98	4.13	548.66	560.20
		E	**1.65**	**3.45**	546.87	556.56
lin 105	14 379.00	A	9.23	16.83	15 706.63	16 799.24
		B	0.35	5.32	14 428.75	15 144.47
		C	0.76	2.83	14 488.54	14 786.55
		D	0.95	3.05	14 515.59	14 817.23
		E	**0.03**	**1.70**	14 383.00	14 623.74
kroA 150	26 524.00	A	17.83	23.08	31 252.63	32 645.88
		B	4.41	8.55	27 694.10	28 791.38
		C	4.01	6.66	27 586.86	28 291.47
		D	**3.11**	6.39	27 349.67	28 217.83
		E	3.19	**4.69**	27 368.97	27 768.99
kroB 150	14 381.00	A	18.45	22.82	30 951.52	32 093.47
		B	4.51	9.20	27 307.66	28 533.39
		C	2.79	7.63	26 859.71	28 124.27
		D	3.68	6.45	27 091.46	27 816.26
		E	**1.41**	**4.20**	26 497.92	27 226.55
kroA 200	29 368.00	A	20.68	28.62	35 441.07	37 773.54
		B	5.66	11.67	31 031.62	32 794.17
		C	3.22	9.57	30 312.29	32 177.23
		D	3.96	7.28	30 531.96	31 506.14
		E	**2.31**	**5.18**	30 045.41	30 887.93
ts 225	126 643.00	A	14.66	19.61	145 207.40	151 474.28
		B	1.74	3.54	128 845.50	131 121.66
		C	1.57	4.87	128 637.10	132 816.58
		D	2.22	4.08	129 448.30	131 815.06
		E	**0.25**	**2.66**	126 964.50	130 014.77
gil 262	2 378.00	A	27.45	31.75	3 030.72	3 132.99
		B	12.23	17.90	2 668.84	2 803.56
		C	8.33	14.99	2 576.09	2 734.48
		D	**5.14**	9.69	2 500.15	2 608.32
		E	5.16	**7.11**	2 500.80	2 547.19

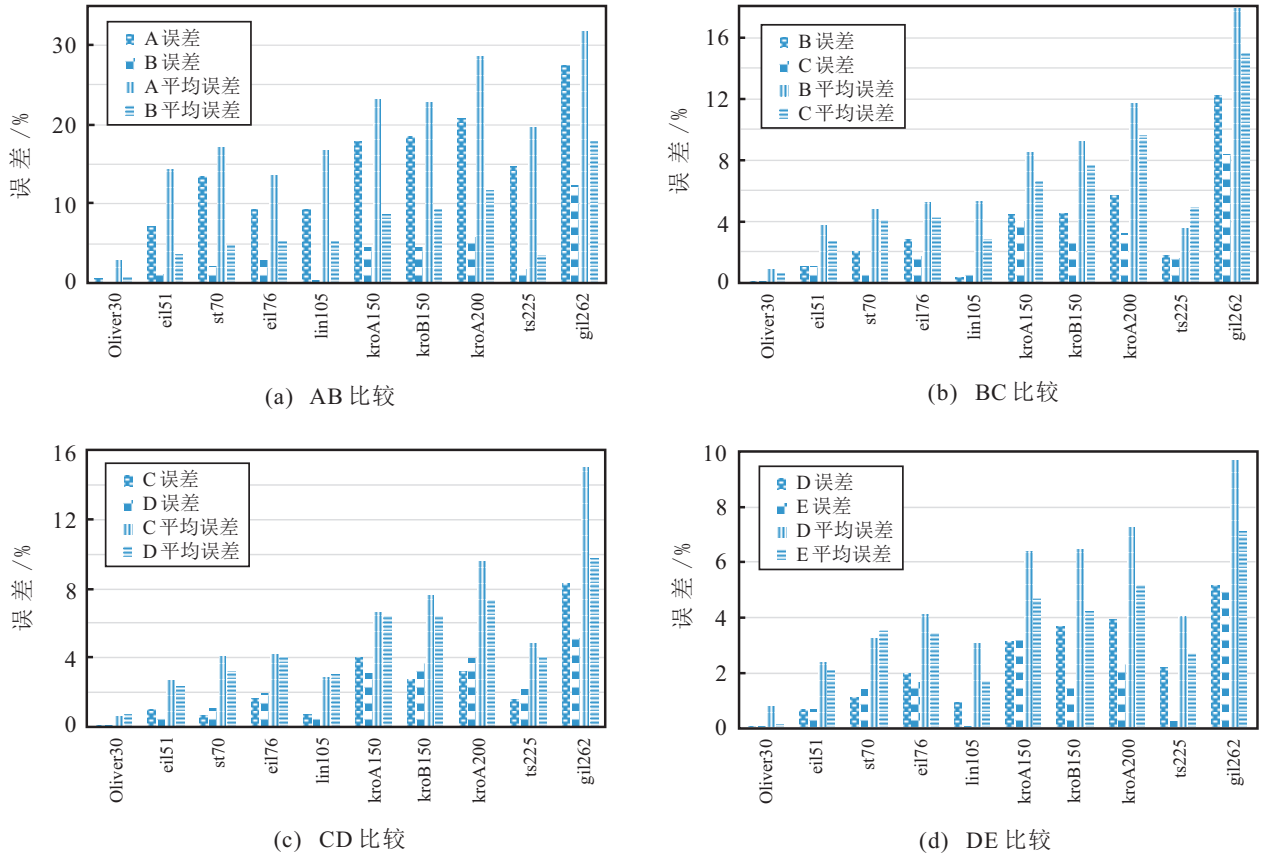


图 7　单因子对比

单独将 A 和 B 做比较, 由表 4 和图 7(a) 中可以看出, 单位信息素的优化策略增加后, 无论是最优值还是平均值, 其效果均得到较大幅度的提升. 例如, ts225 数据集的最优值误差从 14.66 % 降为 1.74 %, 平均误差从 19.61 % 降为 3.54 %, 说明单位信息素优化获得了较好的效果. 单独将 B 和 C 做比较, 由表 4 和图 7(b) 中可以看出, 对于数据集 oliver 30、eil 51、st 70、eil 76、kroA 150、kroB 150、kroA 200 和 gil 262, 在最优值误差和平均值误差方面, 增加虚拟蚂蚁优化后的效果比没有虚拟蚂蚁优化的效果好. 从整体上看, 虚拟蚂蚁优化效果显著.

单独比较 C 和 D, 结合表 4 和图 7(c), 在 105 个点以下的数据集中, C 的最优值误差效果比 D 要好. 在全局优化、单位信息素优化和虚拟蚂蚁优化的基础上, 增加交叉去除优化在较小规模数据集中看不出优劣, 随着点的增多还是具有较大的优势. 例如 200 个城市的 kroA200, 平均误差从 9.57 % 降为 7.28 %. 单独比较 D 和 E, 结合表 4 和图 7(d), 在 70 个点以下 D 和 E 在最优值误差和平均值误差方面相差不大; 从 105 个点开始, 增加点交换的 E 在两方面均比没有增加点交换的 D 要好很多. 如 kroA200, 平均误差从 7.28 % 降为 5.18 %, 下降幅度高达 28.8 %.

5 组实验中, 将全局优化作为比较的基础, 其他 4 个优化点分别累加优化策略后相互比较, 从中可知, 单位信息素优化、虚拟蚂蚁优化、交叉去除优化和点交叉优化均获得了较好的效果. 5 种优化结合后的实验效果在较少蚂蚁只数 (100 只), 和较小迭代次数 (100 次) 的情况下已经具有较高的精度, 例如 225 个点的数据集最优误差和平均误差为 0.25 % 和 2.66 %, 效果较为可观.

3.3　收敛速度分析

为了确定本文算法的收敛情况, 从 TSPLIB 库中选择 14 组数据集进行实验, 实验时设定最大迭代次数为 2 000, 如果在 n 次迭代后的 500 次结果都不改变, 则将 n 记为收敛时的迭代次数, 否则记 2 000 为收敛时迭代次数. 实验的参数设置与 E 组实验相同. 具体收敛情况如表 5 所示, 其中 avg.N 表示平均收敛次数, min.N 表示最小收敛次数.

由表 5 中可以看出, 本文算法在这些数据集中最大平均收敛次数不超过 800, 有些数据集不超过 200, 而最小收敛次数大多不超过 100. 同时, 结果的精度也是十分可观的, 9 个数据集的最优误差均小于 1 %. 对于 ts 225, 基本已经没有误差了. 说明本文算法在保证精度的同时, 也保证了较快的收敛速度.

表5 收敛情况

数据集	已知最优解	误差/%	平均误差/%	最优解	平均值	avg.N	min.N
att 48	33 522.00	0.20	0.78	33 588.34	33 782.75	113	13
eil 51	426.00	0.67	1.14	428.87	430.87	346	49
berlin 52	7 542.00	0.03	3.17	7 544.37	7 781.40	57	14
st 70	675.00	0.31	2.00	677.11	688.49	152	26
rat 99	1 211.00	0.83	2.66	1 221.01	1 243.27	395	37
kroA 100	21 282.00	0.12	3.08	21 307.42	21 937.60	175	28
pr 144	58 537.00	1.17	2.75	59 223.28	60 147.19	378	62
eil 101	629.00	3.27	4.73	649.57	658.76	365	26
lin 105	14 379.00	0.03	1.74	14 383.00	14 629.16	182	31
kroA 200	29 368.00	2.16	4.22	30 001.46	30 607.00	554	67
kroB 200	29 437.00	2.69	4.51	30 228.82	30 764.02	520	105
ts 225	126 643.00	0.00	1.36	126 645.90	128 371.51	309	40
pr 226	80 369.00	0.55	3.34	80 810.01	83 053.49	552	37
a 280	2 579.00	2.03	6.25	2 631.34	2 740.12	797	203

3.4 算法复杂度分析

蚁群算法中影响算法复杂度的因子主要有蚂蚁只数M、城市数量N、迭代次数Iter. 蚁群算法在根据概率选择城市时一般会使用轮盘赌算法,遍历法的轮盘赌复杂度为$O(n)$,基本蚁群算法的时间复杂度为$O(\text{Iter}{\cdot}M{\cdot}N^2)$. 本文算法中选择城市时使用的也是轮盘赌算法,只是将其中的一些参数进行了调整,该部分复杂度不变. 局部优化的两点与全局信息素调整属于同一层级. 点交换的时间复杂度为$O(N^2)$. 对于交叉去除,交叉的判断为$O(1)$,去除单个交叉的过程涉及到序列反转,近似为$O(N)$,因此,最坏情况下的复杂度为$O(N^3)$,最好情况下(没有交叉)复杂度为$O(N^2)$. 算法最坏情况下的时间复杂度为$O\{\text{Iter}\cdot(M\cdot N^2+N^3)\}$,算法最好情况下的时间复杂度为$O\{\text{Iter}\cdot(M\cdot N^2)\}$.

实际上,由于控制初期不进行局部优化,且每次优化内层执行次数较少,从虚拟蚂蚁优化分析中可知,重复资源计算率基本小于20 %,即有80 %以上的情况没有进行局部优化. 从整体上看本文算法的时间复杂度与基本蚁群算法的时间复杂度基本相当.

4 与其他算法的对比分析

为了验证本文算法与其他算法之间的优劣,分别进行数组实验对比分析. 考虑到算法的实现过程因人而异,对实验结果精度和复杂度等的影响,这里尽可能从文献中获取算法提出者的实验数据. 蚁群算法中的影响因素较多,其中蚂蚁的只数、迭代次数和实验的组数直接影响实验的计算次数,这些参数都遵循文献设定的参数. 而其他的参数,如α、β、ρ等参数不影响计算次数,但应该有适用于相应算法的最优组合. 文献中的参数对其算法而言是一个相对较好的参数,本文算法的这些参数也根据实验的具体情况设置. 以下实验中的误差等于(实验最优值 – 已知最优值)/已知最优值,平均误差等于(平均值 – 已知最优值)/已知最优值.

4.1 与文献[14]对比

文献[14]是将迭代参数设置为较大值,统计几乎收敛的情况下的实验效果,参数设置为$m=n,\alpha=1,\beta=5,\rho=0.9,\varepsilon=1.5\,n,Q=100$. 本实验中的参数设置为$m=n$,其他参数与E组实验相同,迭代次数设定最大迭代次数2 000. 如果迭代n_1次后的500次的结果均为n_1次时的结果,则将n_1记为收敛时的迭代次数,否则将2 000记为收敛时的次数. 实验的统计结果如表6所示.

表6 与文献[14]的对比实验数据

数据集	已知最优解	算法	误差/%	平均误差/%	最优值	平均值	平均迭代次数
city 30	423.74	文献[14]	0.04	0.21	423.91	424.64	497
		VLACO	**0.00**	0.32	423.74	425.11	**190**
eil 51	426.00	文献[14]	0.88	2.63	429.74	437.20	1 078
		VLACO	**0.70**	1.57	428.98	432.71	**348**
st 70	675.00	文献[14]	1.08	2.21	682.31	689.92	932
		VLACO	**0.31**	3.20	677.11	696.63	**230**
eil 76	538.00	文献[14]	—	—	—	—	970
		VLACO	1.64	3.55	546.83	557.07	**279**

由实验结果可以看出,在实验条件基本相同的情况下,本文算法(VLACO)收敛时的迭代次数远小于文献[14]中的,最优误差也优于文献[14]. 文献[14]对于oliver30数据集在迭代近1 000次都无法找到已知最优解,可见,该文献在求解精度方面较差. 对于数据集st70,本实验的最优值误差为0.31 %,平均迭代次数为230;文献[14]的最优值误差为1.08 %,迭代次数为932,迭代次数远大于230. 可见本文算法较文献[14]的算法在精度和收敛速度等方面都要好.

4.2 与文献[19]对比

文献[19]设定的参数为$m=100,\rho=0.95,\alpha=1,\beta=2$,最大迭代次数Iter = 100,每个数据集进行5次独立重复实验,进行统计. 本实验中公共参数与文献[19]相同,其他参数与E组实验相同. 实验的统计结果如表7所示.

表 7　与文献[19]的对比实验数据

数据集	已知最优解	本文算法		文献[19]	
		误差/%	最优	误差/%	最优
att 48	33 522	**0.20**	**33 588.34**	4.96	35 185
berlin 52	7 542	**0.03**	**7 544.37**	1.59	7 662
st 70	675	**1.54**	**685.4**	8.59	733
pr 76	108159	**1.42**	**109 689.6**	9.92	118 892
rat 99	1 211	**2.67**	**1 243.29**	10.57	1 339
kroA 100	21 282	**0.06**	**21 294.4**	8.97	23 190
eil 101	629	**4.15**	**655.08**	12.24	706
lin 105	14 379	**0.03**	**14 383**	5.20	15 126
pr 107	44 303	**0.42**	**44 488.68**	8.44	48 040
pr 124	59 030	**1.37**	**59 836.67**	4.07	61 430
bier 127	118 282	**1.30**	**119 822.1**	5.89	125 245
pr 136	96 772	**4.20**	**100 833.3**	17.80	113 993
pr 144	58 537	**0.65**	**58 915.98**	3.13	60 370
kroA 150	26 524	**3.24**	**27 382.25**	14.28	30 312
eil 51	426	**0.83**	**429.53**	5.40	449
pr 152	73 682	**1.73**	**74 958.57**	11.92	82 462
eil 76	538	**3.32**	**555.87**	6.51	573
rat 195	2 323	**3.28**	**2 399.3**	10.55	2 568
kroA 200	29 368	**3.82**	**30 490.16**	17.58	34 530
ts225	126 643	**1.52**	**128 566.2**	6.51	134 886
pr 226	80 369	**2.53**	**82 403.16**	9.94	88 355
gil 262	2 378	**6.53**	**2 533.17**	18.92	2 828
pr 264	49 135	**4.77**	**51 478.3**	12.44	55 247
a 280	2 579	**8.02**	**2 785.82**	28.54	3 315
pr 299	48 191	**5.80**	**50 984.65**	19.39	57 533

通过与文献[19]的对比可以看出,在蚂蚁只数与迭代次数相同的条件下,本文算法在25个数据集中的最优解均好于文献[19]. 在只迭代100次的情况下,25组中最优误差在2%以下的有13组,而文献[19]最优误差在2%以下的只有一组. 误差在1%以内的有6组,文献[19]则一组都没有. 可见本文算法较文献[19]具有较大的优势.

4.3 与文献[15]对比

为了更全面地比较本文算法在精度方面的情况,与更多的算法进行比较. 实验数据部分来自文献[15],因当时的已知最优解与现在的有差别,故只取数据中的最优值和平均值,而最优误差和平均误差均根据目前已知最优解计算所得. 文献[15]的参数设定见表8,本算法的蚂蚁只数(m)和迭代次数(Iter)与文献[15]相同,其他参数设定与E组实验相同. 具体实验的对比结果如表9所示.

表 8　文献[15]的参数

m	α	β	γ	ρ	F	Q	Iter
50	1	5	4	0.4	0.5	100	500

表 9　与文献[15]的对比实验数据

数据集	已知最优解	算法	最优解	平均值	最优误差/%	平均误差/%
eil 51	426.00	ACA	468.00	501.00	9.86	17.61
		ACGA	463.00	499.00	8.69	17.14
		HGI-ACGA	453.00	488.00	6.34	14.55
		NACA	440.00	458.00	3.29	7.51
		VLACO	428.98	434.56	**0.70**	**2.01**
berlin 52	7 542.00	ACA	8 145.00	8 823.00	8.00	16.98
		ACGA	7 820.00	8 201.00	3.69	8.74
		HGI-ACGA	7 769.00	7 989.00	3.01	5.93
		NACA	7 602.00	7 892.00	0.80	4.64
		VLACO	7 544.37	7 782.20	**0.03**	**3.18**
st 70	675.00	ACA	784.00	885.00	16.15	31.11
		ACGA	762.00	814.00	12.89	20.59
		HGI-ACGA	736.00	789.00	9.04	16.89
		NACA	712.00	768.00	5.48	13.78
		VLACO	685.83	696.90	**1.60**	**3.24**
eil 76	538.00	ACA	605.00	666.00	12.45	23.79
		ACGA	602.00	662.00	11.90	23.05
		HGI-ACGA	586.00	640.00	8.92	18.96
		NACA	575.00	620.00	6.88	15.24
		VLACO	549.46	555.66	**2.13**	**3.28**
pr 107	44 303.00	ACA	47 860.00	49 845.00	8.03	12.51
		ACGA	47 589.00	49 363.00	7.42	11.42
		HGI-ACGA	46 659.00	48 989.00	5.32	10.58
		NACA	46 640.00	48 890.00	5.28	10.35
		VLACO	44 346.19	44 999.80	**0.10**	**1.57**

从表9中可以看出,在5组数据集中,本文算法(VLACO)在最优误差、平均误差方面均优于文献[15]和文献[20]提出的算法. 说明本文算法在求解精度方面具有较大优势. 对于求解速度方面,由于文献未公布相关数据,不好进行比较. 从理论上分析,本文算法的时间复杂度与基本蚁群算法的复杂度大致相当,文献[15]的算法引入了差分演化的部分概念,使得信息素的更新公式的复杂度从$O(1)$提升为$O(M)$,局部更新从$O(N)$提升为$O(M\cdot N)$. 文献[15]的时间复杂度比基本蚁群算法稍高. 本文算法与基本蚁群算法复杂度基本相当,求解速度应该会在同一数量级上. 总之,本文算法优于对比文献中的算法.

5 结　论

在蚁群算法研究的基础上,本文提出了一种基于虚拟蚂蚁的局部优化蚁群算法,该算法提升了部分计算资源的利用率,在收敛速度与结果的精度之间作出了较好的平衡. 通过单因子优化策略影响实验、收敛速度分析实验和与其他同类型的算法的比较实验可以得出,本文算法(VLACO)在结果的精确度和收敛速度等方面都取得了较好的效果.

参考文献(References)

[1] 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.

[2] Bonabeau E, Dorigo M, Theraulaz G. Swarm intelligence: From natural to artificial systems[M]. New York: Oxford University Press, Inc, 1999: 1-24.

[3] 孙彬, 孙俊清, 刘凤连, 等. 基于蚁群算法的鲁棒离散泊位分配问题的研究[C]. 中国控制会议, 北京, 2010: 1727-1732.
(Sun B, Sun J Q, Liu F L, et al. On robust discrete berth allocation based on ant colony algorithm[C]. Chinese Control Conf. Beijing, 2010: 1727-1732.)

[4] 黎自强, 田茁君, 王奕首, 等. 求解平衡约束圆形Packing问题的快速启发式并行蚁群算法[J]. 计算机研究与发展, 2012, 49(9): 1899-1909.
(Li Z Q, Tian Z J, Wang Y S, et al. A fast heuristic parallel ant colony algorithm for circles Packing problem with the equilibrium constraints[J]. J of Computer Research and Development, 2012, 49(9): 1899-1909.)

[5] 王晴. 基于蚁群算法的电路故障诊断技术研究[D]. 武汉: 华中科技大学电子信息与通信学院, 2007: 1-48.
(Wang Q. Research on circuit fault diagnosis based on ant colony algorithm[D]. Wuhan: School of Electronic Information and Communications, Huazhong University of Science and Technology, 2007: 1-48.)

[6] Lai M, Tong X. A metaheuristic method for vehicle routing problem based on improved ant colony optimization and Tabu search[J]. J of Industrial & Management Optimization, 2017, 8(2): 469-484.

[7] Hoos H H. Taming the complexity monster or: How I learned to stop worrying and love hard problems[J]. Acm Sigevolution, 2017, 9(2): 11.

[8] Wang P, Lin H T, Wang T S. An improved ant colony system algorithm for solving the IP traceback problem[J]. Information Sciences, 2016, 326(C): 172-187.

[9] Mavrovouniotis M, Muller F M, Yang S. Ant colony optimization with local search for dynamic traveling salesman problems[J]. IEEE Trans on Cybernetics, 2017, 47(7): 1743-1756.

[10] Yang J, Ding R, Zhang Y, et al. An improved ant colony optimization (I-ACO) method for the quasi travelling salesman problem (Quasi-TSP)[J]. Int J of Geographical Information Science, 2015, 29(9): 1534-1551.

[11] 杨再甫, 黄友锐, 曲立国, 等. TSP的改进蚁群算法求解及其仿真研究[J]. 合肥工业大学学报: 自然科学版, 2014, 37(8): 928-932.
(Yang Z F, Huang Y R, Qu L G, et al. Solution of TSP based on improved ant colony algorithm and its simulation[J]. J of Hefei University of Technology: Science and Technology, 2014, 37(8): 928-932.)

[12] 杜鹏桢, 唐振民, 孙研. 一种面向对象的多角色蚁群算法及其TSP问题求解[J]. 控制与决策, 2014, 29(10): 1729-1736.
(Du P Z, Tang Z M, Sun Y. An object-oriented multi-role ant colony optimization algorithm for solving TSP problem[J]. Control and Decision, 2014, 29(10): 1729-1736.)

[13] 张于贤, 丁修坤, 薛殿春, 等. 求解旅行商问题的改进蚁群算法研究[J]. 计算机工程与科学, 2017, 39(8): 1576-1580.
(Zhang Y X, Ding X K, Xue D C, et al. An improved ant colony algorithm for traveling salesman problem[J]. Computer Engineering and Science, 2017, 39(8): 1576-1580.)

[14] 孟祥萍, 片兆宇, 沈中玉, 等. 基于方向信息素协调的蚁群算法[J]. 控制与决策, 2013, 28(5): 782-786.
(Meng X P, Pian Z Y, Shen Z Y, et al. Ant algorithm based on direction-coordinating[J]. Control and Decision, 2013, 28(5): 782-786.)

[15] 张弛, 涂立, 王加阳. 新型蚁群算法在TSP问题中的应用[J]. 中南大学学报: 自然科学版, 2015, 46(8): 2944-2949.
(Zhang C, Tu L, Wang J Y. Application of self-adaptive ant colony optimization in TSP[J]. J of Central South University: Science and Technology, 2015, 46(8): 2944-2949.)

[16] Bullnheimer B, Hartl R F, Strauss C. A new rank based version of the ant system — A computational study[J]. Central European J of Operations Research, 1999, 7(1): 25-38.

[17] Feng H M, Liao K L. Hybrid evolutionary fuzzy learning scheme in the applications of traveling salesman problems[J]. Information Sciences, 2014, 270: 204-225.

[18] Junqiang W, Aijia O. A hybrid algorithm of ACO and delete-cross method for TSP[C]. Int Conf on Industrial Control and Electronics Engineering. Xi'an, 2012: 1694-1696.

[19] TSPLIB. TSPLIB[EB/OL]. [2017-10-20]. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.

[20] 姜坤霖, 李美安, 张宏伟. 面向旅行商问题的蚁群算法改进[J]. 计算机应用, 2015, 35(S2): 114-117.
(Jiang K L, Li M A, Zhang H W. Improved ant colony algorithm for travelling salesman problem[J]. J of Computer Applications, 2015, 35(S2): 114-117.)

[21] 徐金荣, 李允, 刘海涛, 等. 一种求解TSP的混合遗传蚁群算法[J]. 计算机应用, 2008, 28(8): 2084-2087.
(Xu J R, Li Y, Liu H T, et al. Hybrid genetic ant colony algorithm for traveling salesman problem[J]. J of Computer Applications, 2008, 28(8): 2084-2087.)

作者简介

李俊(1978−), 男, 副教授, 博士, 从事智能计算等研究, E-mail: lijun@wust.edu.cn;

周虎(1994−), 男, 硕士生, 从事智能计算的研究, E-mail: 1073280512@qq.com;

李波(1975−), 男, 教授, 博士, 从事智能计算、机器学习等研究, E-mail: liberol@126.com.

(责任编辑: 李君玲)