多解旅行商问题的小生境模因算法改进

黄鸿展（202230441138）

摘要 本文将用于解决多解旅行商问题的小生境模因算法进行改进，调整了算法的选择策略，提高了算法的搜索性能性；修改率变异率与交叉率，随适应度大小动态调整，使得交叉不但能不断开辟新的搜索空间，而且能保证子代个体的高质量。改进后算法的特点与原算法相似，采用了小生境保存技术，实现了多个最优解的并行搜索;采用自适应邻域策略来平衡探索开发;繁殖采用临界边缘感知;以及提高搜索效率的选择性局部搜索策略，其中选择搜索经调整为 or-opt，选择改用锦标赛，更好地适应离散多解的情况，交叉与变异率会随个体适应度变化而动态改变。为了评估所提出算法的性能，在对应的多解优化测试套件上进行了全面的实验，结果表明，我们的算法优于其他比较算法，优于原来的小生境模因算法。

关键词 多模态优化(MMOP)，多解旅行商问题(MSTSP)，邻域定位策略，模因算法

Modified Memetic Algorithm for Multi-Solution Traveling Salesman Problem

Hong-Zhan Huang (202230441138)

Abstract I improve the niche memetic algorithm used to solve the multi-solution traveling salesman problem, adjusts the selection strategy of the algorithm, and enhances the search performance of the algorithm; modifies the mutation rate, crossover rate and modification rate, and dynamically adjusts them according to the fitness value, so that the crossover can not only continuously open up new search spaces, but also ensure the high quality of offspring individuals. The characteristics of the improved algorithm are similar to the original algorithm. It adopts the niche reservation technology to achieve the parallel search of multiple optimal solutions; adopts the adaptive neighborhood strategy to balance exploration and exploitation; reproduction adopts critical edge perception; and the selective local search strategy to improve the search efficiency. Among them, the selection search has been adjusted to or-opt, and the selection uses the tournament, which is better adapted to the discrete multi-solution situation, and the crossover and mutation rates will change dynamically with the change of individual fitness. In order to evaluate the performance of the proposed algorithm, comprehensive experiments were conducted on the corresponding multi-solution optimization test suite. The results show that our algorithm is superior to other comparison algorithms and the original niche memetic algorithm.

Key words multimodal optimization (MMOP), multi-solution traveling salesman problem (MSTSP), neighborhood niching strategy

Ⅰ.背景介绍

旅行商问题（TRAVELING salesman problem,TSP)[1][2]，是组合优化问题之一。许多工程和学术问题都涉及到 TSP 作为子问题，在 TSP 中，一个推销员从一个城市出发，在每个城市旅行一次且只旅行一次，最后回到开始的城市，推销员的目标是找到最短的旅行路线。为了满足实际应用的不同需求，研究者提出了多种 TSP 变体，包括多目标 TSP、多售货员 TSP 等。大多数 TSP 应用程序和 TSP 变体关注于找到一个最优解决方案，而不考虑可能存在多个高质量解决方案的事实。然而，在实践中，为决策者提供多样化的最优方案同样重要，这样催生了多解 TSP(MSTSP)的研究。

由于 np 困难的性质，用确定性算法处理 MSTSP 是相当困难的。进化算法(EA)可以满足在合理的时间内找到最优解[3]，也存在 EA 应用于 MSTSPs 的相关研究。然而，当城市

规模增加时，算法未能找到任何最优解集。因此，需要一种更有效的算法来处理 MSTSP.

此外，MSTSP 问题的处理属于多模态优化(multimodal optimization, MMOP)领域。多模态优化就是一种要求找出全局最优解和全部局部极值解以备实际中使用的技术[3]，该问题广泛存在于科学研究和生产应用中。 一般的智能优化算法只能寻找到一个全局最优解，必须与拥挤模型、适应度、共享机制等小生境模型结合使用才有机会找到全部极值解。

另一个重要的问题是，在处理 MSTSPs 时应考虑种群多样性。在基于种群的算法中，由于遗传漂变和选择压力导致的多样性损失，种群最终会向一个盆地收敛[4]。遗传漂移导致种群中基因变异的消失，而选择压力则使个体的进化偏向于一小部分搜索空间，而忽略了种群多样性。

小生境遗传算法可以很好的保持种群的多样性，具有较高的全局搜索能力和收敛速度，适合用于解决多峰函数的优化问题。通过对物种竞争施加限制，可以消除遗传漂移和选择压力对整个种群的影响，不同吸引力盆地的候选者可以跨代生存。目前，流行的生态位技术包括拥挤、物种形成、聚类和邻域策略。这些小生境技术主要用于连续 MMOP，通常采用欧几里得距离作为相似性指标。

为了处理 MSTSP 问题，[5]提出了一个小生境模因算法(niche memetic algorithm, NMA)框架。为了解决 MSTSP 问题，提出的 NMA 结合了一个邻域小生境策略，可以并行搜索多个最优解。在具体框架下，还设计了四种策略来提高算法的性能，分别有助于实现探索-利用平衡、关键边缘感知、适应度评估节省和决策者友好性。

但是该方法在峰值点附近区域，仍然存在着局部搜索能力较弱和搜索速度较慢的缺陷；另外对于小生境参数的确定也没有明确的准则，只能根据经验确定；且没有统一的标准来有效度量种群的多样性，算法为了维持种群的多样性而减慢了优化收敛速度。因此，本文将算法进行优化，首先将选择策略改为锦标赛选择，更加适合离散多解的情况；调整了算法的局部搜索策略，将 2-opt 改为 3-opt 以提高算法的搜索能力；同时，设计了动态调整变异率与交叉率的算子，这样不容易破坏整个群体的优化，能够进行局部寻优，也有利于搜索出最优点。

总体而言，本文的主要贡献总结如下：

A.调整选择性局部搜索策略

局部搜索有助于快速收敛，但副作用是，它通常会通过穷尽地探索任意给定解的邻域来带来巨大的适应度评估消耗。在 NMA 算法中，为了避免在一些低劣和冗余的解上进行低效的局部搜索，设计了一种选择性局部搜索策略，该策略偏向于有前途的和不同的个体，以及不关键的边，以最大化搜索效率。原算法采用 2-opt 作为局部搜索算法，耗时短有成效，但是在后续研究中认为[6]，3-opt 策略比 2-opt 效果更好。3-opt 算法是一种针对 TSP 问题的局部搜索算法，通过选取路径中不相邻的三个节点之间的连接删除，然后尝试其他 7 种不同连接方式，选取路径长度最短的连接方式作为新的连接方式，拥有更好的效果。但 3-opt 较高的时间复杂度使得面对大规模问题速度较慢。因此，本文采用 3-opt 的子集算法 or-opt，只需要进行一部分的 3-opt 交换，既拥有速度，又有较好的性能。

B. 选择、交叉、变异策略调整

原算法采用的种群迭代基于基本的遗传算法，选择采用轮盘赌策略，交叉率为 0.9，变异率为 0.01。群体任何的个体都是以同等的概率被交叉变异，这样不能保证群体的优化程度. 在进化的过程中，根据适应度值大小来调整交叉变异算子大小[7]. 对于适应度比

较低的个体，交叉概率比较大，从而可以迅速对整个群体进行优化. 而在进化的后期，个体的适应度都接近平均水平，这时交叉概率较小，从而保留最优个体给下一代。

C.丰富的验证性实验

原算法在[8]中开发了一个标准的基准测试套件。 测试套件由 25 个具有不同特征的 MSTSP 实例组成，这些 实例用于测试 NMA 和本文中的其他方法的性能。实验结果表明，与竞争对手相比，NMA+可以在相对较短的时间内获得更高质量和更多样化的解，同时受问题规模的影响较小，与原算法生成的解相似，而多样性更加明显。

Ⅱ.背景

A. 多解旅行商问题

在许多与 TSP 相关的实际应用中，需要多个近似好的解。这些应用背景激发了[]对 MSTSP 领域的研究。MSTSP 与 TSP 具有完全相同的数学形式，给定一个城市列表，一个推销员在 每个城市旅行一次且只旅行一次，构造一条汉密尔顿路径。MSTSP 求解器的目标是找到所有最短的汉密尔顿路径。数学上，考虑一个图 *G=(V, E)*，其中 *V={1,2,3，…，N}*是一组城市(用城市指数表示)，*E={(i, j)|i, j∈ V, i= j}*是一组边，表示城市 i 和 j 之间的道路/边。每条边(i, j)都有一个权值，记为 *dij*，它测量了两个城市 i 和 j 之间的距离。汉密尔顿路径可以表示为 *π*。因此，MSTSP 就是在所有可能的排列中找到一个最短的长度。更准确地说，考虑

$$\min f(\pi) = \sum_{k=1}^{N} d_{\pi(k)\pi(k+1)} + d_{\pi(N)\pi(1)} \quad (1)$$

其中 N 是城市的数量，*π(k)*是排列 π 的第 k 个元素。MSTSP 的目标是找到一组具有完全相同最小行程长度的解。 最近，我们发布了 25 个不同特征的 MSTSP 实例的综合基准测试[18]。根据设计方法，将 MSTSPs 分为简单 MSTSPs、几何 MSTSPs 和复合 MSTSPs 三类。

1)简单 MSTSPs 是随机生成的，最优解的数量不可控，但可以通过算法得到最优解集。

2)几何 MSTSPs 采用不同的对称几何形状设计，如矩形、正六边形。可以通过控制特定几何图形的组合、几何图形的位置以及几何图形的旋转来进行调整。

3)复合 MSTSPs 更复杂，它是由几个小规模的 MSTSPs 组成一个更大的实例。 每个小规模 MSTSP 都是一个城市群，而城市群分布在不同的地理位置。特征来源于两种关系:集群内关系和集群间关系。通过组合不同类型的小型 MSTSPs，控制了最优的数量。

总体而言，在 25 个 MSTSP 实例中，城市数量在 9 ~ 66 个之间，最优规模在 2 ~ 196 个之间。基于当前的计算条件，一些精确的算法可以处理小规模的 MSTSPs，当处理更大的 MSTSPs 时， 例如复合 MSTSPs (N≥28)，精确算法的时间开销将变得无法忍受。需要注意的是，由于需要找到多条最优路由， MSTSPs 比传统的 tsp 要困难得多，所以 MSTSPs 和 tsp 对“大规模”的感知是不同的。本文将 N≥28 的 MSTSPs 视 为大规模案例，对于此类案例，精确算法无法在可接受的时间内停止执行。

B. MSTSP 的多解优化算法

TSP 是一个 np 困难问题，MSTSP 比 TSP 更难获得多个最优解。MSTSP 优化器既要对满意解具有较高的搜索效率，又要对不同解保 持良好的种群多样性。这两个目标存在一定的矛盾，难以同时实现。有几种多解优化算法可用于解决 MSTSP 问题，包括 MCC-GA[9]、基于小生境的蚁群系统(NACS)[10]和基于邻域的 GA(NGA)[8]。

与以前的算法不同， NACS 使用多个信息素矩阵来定位不同的解。当蚂蚁找到一个优越且独特的解决方案时，创建一个新的信息素矩阵。在不同信息素矩阵上的蚂蚁往往会找到不同的最优解。NACS 遇到的问题是关于生态位参数的问题，如果没

有先验知识，很难适当设置参数。NGA 将相似的邻居聚为一个群体，并在群体内执行 GA 的基本遗传操作。遗传漂移和选择压力被限制在相应的群体中，从而定位多样化的最优。然而，NGA 的收敛速度慢，且难以设置合适的邻域大小。综上所述，离散 MMOP 算法的发展主要考虑了多样性保持、收敛速度和适当的参数设置。

Ⅲ.算法改进分析

本节介绍改进的算法 NMA+。改进后的算法与原算法框架基本相同，其中改变了局部选择策略与选择交叉变异的方式，首先概述该算法，随后进行各个部分的细节描述。

A. 框架

NMA+算法框架与原算法框架相同，以 MA 算法为基准，并引入邻域小生境策略。将整个种群划分为几个邻域群体，每个群体倾向于不同定位的最优。在给定的搜索空间内，应用 MA 进行有效的搜索。在此框架下，改变了几种增强策略来提高所提出算法的性能。

染色体采用上述定义的基于排列的表示，NMA 的过程描述如下：首先，使用部分贪婪策略初始化 NP 染色体。对于一条有 N 个基因的染色体(N为城市数量)，前面 $N/2$ 基因随机选择不同的城市构建，后面 $N/2$ 基因使用贪婪策略逐个选择城市——离染色体中最后确定的城市最近的未选择城市。然后，使用小生境策略重新排列染色体的顺序，将接近的染色体分组组成一个交配池。在测量两个解距离时，采用的是解的边缘集，而不是绝对节点位置集。

同时，采用了多样性增强的方法，避免陷入局部最优。首先利用邻域领导来构造该问题的临界边集(CES)，然后采用 CES 来实现复制和局部搜索策略。根据 GA 的例程，我们以 Pc 的概率进行交叉以及变异操作，概率为 Pm，而 Pm 会根据个体与种群的适应度进行动态调整，以保持优化性的同时增强多样性。然后，采用改进后选择性局部搜索来增强挖掘能力。获得后代后，使用替换操作确定下一代，持续迭代，直到满足终端条件。当算法结束时，得到一组最终解。采用后处理程序从最终解集选择精英。**算法 1 概述了 NMA 的总体过程。**下面将逐个对有关策略进行详细描述。

Algorithm 1 Niching Memetic Algorithm

Input: An MSTSP instance T, the population size NP, the crossover rate P_c, the mutation rate P_m, a termination criteria, and the minimum (maximum) neighborhood size $M_{\min}$ ($M_{\max}$)
Output: A representative solution set $\mathbb{S}$

```
Parent ← Initialize(NP)
Evaluate(Parent)
CES ← {}
while the termination criterion not satisfied do
    MatingPool ← NeighborhoodStrategy(Parent, M_min, M_max) /* Algorithm 2 */
    eMatingPool ← DiversityEnhancement(MatingPool, CES) /* Algorithm 3 */
    UpdateCriticalEdgeSet(eMatingPool, CES) /* Algorithm 4 */
    cOffspring ← Crossover(eMatingPool, CES, P_c)
    mOffspring ← Mutation(cOffspring, CES, P_m)
    lsOffspring ← LocalSearch(mOffspring, CES)
    Evaluate(lsOffspring)
    Parent ← Replacement(lsOffspring, MatingPool, CES)
end while
S ← Preserve (Parent) /* Algorithm 5 */
```

B. 自适应邻域策略

为了保持种群多样性，使用邻域生态位策略，其本质是限制子空间内群体的繁殖或选择压力。通过这种方法，同一生态位成员可以产生空间上接近的后代。为了解决邻域大小问题，根据小生境状态自适应地确定邻域大小。这里，邻域大小为 m，M_{min} 和 M_{max} 分别表示允许的最小和最大邻域大小。

Algorithm 2 $MatingPool \leftarrow$ NeighborhoodStrategy($Parent$, $M_{\min}$, $M_{\max}$)

```
tmpParent ← Parent
MatingPool ← {}
L_min ← MinLength(Parent)
L_β ← betaLength(Parent, NP/M_min)
while tmpParent ≠ ∅ do
    leader ← FindBest(tmpParent) /*Find out the best individual in tmpParent.*/
    if |tmpParent| ≥ M_max then
        γ ← min((leader.Len − L_min)/(L_β − L_min), 1.0)
        m ← floor(γ × (M_max − M_min) + M_min)
        if mod(m, 2) ≠ 0 then
            m = m + 1
        end if
    else
        m ← |tmpParent|
    end if
    CalculateShareDist(tmpParent, leader) /*Calculate the sharing distance between each individual in tmpParent and leader.*/
    sortParent ← Sort(tmpParent.shareDist, "desc") /*Sort the individuals by the sharing distance in an descending order.*/
    NeighborGroup ← sortParent[1, ..., m] /*Assign the first m closest sorted individuals to NeighborGroup.*/
    Shuffle(NeighborGroup) /*Shuffle the order of NeighborGroup.*/
    MatingPool ← MatingPool + NeighborGroup
    tmpParent ← tmpParent − NeighborGroup
end while
```

在算法 2 中，首先将长度最短的未处理染色体识别为群体 leader(领导者)。如果未加工染色体的数量大于等于 M_{max}，则根据 leader 的长度自适应确定邻域大小 m。然后，根据(2)中定义的共享距离对未处理的成员降序排序，前 m 个成员形成一个新的邻域组。新形成的群体被洗牌，然后加入交配池。与此同时，新加入的成员被标记，并从当前的种群中剔除。重复上述过程，直到所有成员都被处理完毕。

在上述邻域划分过程中，我们需要计算当前 leader 与每个未处理成员之间的两两相似性。在本文中，两个排列 πi 和 πj 的相似度是通过共享距离来度量的，如下所示:

$$S(\pi_i,\pi_j)=\frac{|\Phi(\pi_i)\cap\Phi(\pi_j)|}{N} \quad (2)$$

其中$\Phi(\pi_i)$和$\Phi(\pi_j)$分别表示 π_i和 π_i的边集;∩表示相交集，|·|表示相交集中边的个数。共享距离度量操作在路线的边缘集上，而不是绝对节点位置上。因此具有不同节点，但访问边集相同的解被识别为相同的解。

自适应调整邻域大小 m，提高算法的泛化性能。在本文提出的算法中，一个小生境被一个邻域组覆盖,邻域成员由邻域 leader 支配。首先，我们根据邻居 *leader* 的适应度值(即长度)计算一个生态位状态测度 $\gamma\in[0,1]$：

$$\gamma=\min\left(\frac{L_{leader}-L_{min}}{L_\beta-L_{min}},1.0\right) \quad (3)$$

式中 L_{leader} 为 leader 长度； L_{min} 和 L_β 为种群的最短长度和第 2 短长度；而 $\beta=NP/M_{min}$ 表示可能的邻里领袖的最大数量。适应度值 Lβ 可以看作是识别邻居 leader 是“优”还是“劣”的截断点。

然后，基于 γ 使用来调整邻域大小 m。

$$m=\lfloor\gamma\times(M_{max}-M_{min})+M_{min}\rfloor \quad (4)$$

1) 如果 L_{leader} 小于 Lβ，且等于 L_{min}，说明该 leader 较优。计算出的 γ 值为 0，因此，邻域大小 m 为 M_{min}。上级领导被分配最少的搜索资源，因为这个解决方案比其他解决方案所需要的调整更小。

2) 如果 L_{leader} 大于 L_β，则该 leader 是劣等的。根据(3)和(4)，γ = 1，m=M_{max}。算法将最多的搜索努力分配给劣等邻域群，帮助它们快速识别好的解。

3) 如果 L_{leader} 比 Lβ 短但比 L_{min} 大，则 leader 既不是最好的也不是最差的。接下来，邻域大小介于 M_{min} 和 M_{max} 之间，取决于 γ。

C. 多样性增强方法

在趋同的群体中进一步搜索，既不能带来多样性，也不能有更好的解决方案。因此原算法设计了一种多样性增强方法，以帮助融合群体跳出吸引力盆地，向更好的位置移动。

具体而言，为每个交配池，我们计算分享第一个成员之间的距离和组中的每个成员来检查他们是否共享相同排列(即排列等于 1)。如果成员都是相同的，认为这一群体收敛，随后突变除了 leader 以外的所有成员。突变率会随着种群的适应值变化，以增强多样性。

$$\begin{cases}P_m(k)=P_m-0.05*\left(\frac{f(k)-f_{avg}}{f_{max}-f_{avg}}\right), & f(k)\le f_{avg}\\ P_c(k)=P_m, & f(k)>f_{avg}\end{cases} \quad (5)$$

式中 $P_m(k)$ 为第 k 个个体的变异概率，在变异操作中，对于适应度值低于平均水平的个体，希望它的变异概率较大,对于适应度值较大的个体则以较小固定的概率进行变异，这样不容易破坏整个群体的优化，能够进行局部寻优，也有利于搜索出最优点。

扰动后，收敛的成员被重新分配到邻近区域。**该过程在算法 3 中描述。**

Algorithm 3 *eMatingPool* ← DiversityEnhancement (*MatingPool*, *CES*)

```
eMatingPool = {}
for each NeighborGroup ∈ MatingPool do
    leader ← NeighborGroup.Leader
    flag ← true
    for each p ∈ NeighborGroup do
        if S(p, leader) ≠ 1 then
            flag ← false
            break
        end if
    end for
    if flag == true then
        newNeighborGroup ← Mutation(NeighborGroup − leader, CES, 1.0)
        eMatingPool = eMatingPool + newNeighborGroup
    else
        eMatingPool = eMatingPool + NeighborGroup
    end if
end for
```

D. 关键边缘集的更新

MSTSP 具有离散(有限)问题空间和需要找到多个问题的最优的解决方案，因此，MSTSP 的不同最优解可能具有一些共同的元素。在本算法中认为，如果能够识别出关键边缘，将有助于增强 NMA 算子，从而进一步提高搜索效率，特别是对于大规模的 mstsp。因此，制定了 NMA 中的 CEA 演化策略，CEA 在演进过程中进行增量更新。NMA+与原算法流程一致。

算法 4 总结了 CES 更新的过程。然后，采用 CES 改进 NMA 的复制和局部搜索操作。

Algorithm 4 UpdateCriticalEdgeSet(*eMatingPool*, *CES*)

```
1: pLeader = {}
2: for each NeighborGroup ∈ eMatingPool do
3:    leader ← NeighborGroup.Leader
4:    if leader.sg ≥ N then
5:       pLeader = pLeader + leader
6:    end if
7: end for
8: candidates = ⋂_{leader∈pLeader} Φ(leader) /*Calculate the
   intersection of solutions' edges in pLeader.*/
9: for each edge ∈ candidates do
10:   CES = CES + edge
11:   conflictedFIFO(CES, edge) /*Once conflicted, use FIFO.*/
12: end for
13: deleteOldest(CES) /*Delete the edges inserted before N earlier
   generations.*/
```

E. 临街边缘感知的再现

以 GA 为基准，所提算法的再现由交叉和突变操作组成。交叉操作将亲本染色体配对繁殖，培养出新的竞争力个体。在整个种群中进行的交叉操作会导致遗传漂移，使所有的候选解决方案趋于相同。为了避免基于相似的亲本会产生相似的后代最终趋同这一观点，在适应性邻域策略中，我们将相似的染色体聚集在同一邻域群体中，同时构建群体内的交配池。此后，交叉操作可以方便地将两个亲本按照生成的交配池的顺序配对。在进化的过程中，根据适应度值大小来调整交叉算子大小[9]. 对于适应度比较低的个体， 交叉概率比较大，从而可以迅速对整个群体进行优化. 而在进化的后期， 个体的适应度都接近平均水平， 这时交叉概率较小， 从而保留最优个体给下一代. 改进算法的交叉算子设计如下：

$$\begin{cases} P_c(k) = 0.8 - 0.3 * \left(1 - \frac{f(k)}{f_{avg}}\right)^{\frac{1}{2}}, f(k) \geq f_{avg} \\ P_c(k) = 0.8, \qquad f(k) < f_{avg} \end{cases} \quad (6)$$

P_C(k)为第 k 个个体的交叉概率，f_{avg}当代种群的平均适应度， f(k)为个体 k 的适应度. 进化过程中，个体的适应度值低于平均水平的个体， 希望有较大的交叉概率。随着种群的进化，种群的平均适应度也逐渐增大， 个体的适应度值逐渐趋向与平均水平。这样的个体以较小的固定的概率进行交叉，从而保持群体的优化性。

除此之外，NMA 通过结合 CEA 方法改进了部分映射交叉(PMX)[61]。 由此产生的交叉被命名为 CEA-PMX。首先，从染色体中随机选择两个交换点。将亲本一方的两个交换点之间的片段性基因挑出来，直接传递给孩子。其次，父节点和 CES 相交的边也由子节点继承。第三，这个孩子的剩余部分是从另一个父母那里继承的。在遗传过程中，如果候选基因已经包含在孩子身上，那么对应的定位基因(另一方亲本的等位基因)就会被复制。另一个孩子也会以类似的方式产生。

突变操作扰乱染色体，带来多样性。我们修正了交换突变(exchange mutation, EM)如图 4 所示。对于一条候选染色体，EM 随机交换两个基因。当选择的基因相邻时，交换的边数为 2;否则为 4 条。为了避免全相邻这种低效的情况，强制选择突变不相邻的基因。此外，突变保留了染色体上的临界边缘。

F. 选择性局部搜索

局部搜索期望在给定解决方案的小范围内进行利用，当对同一邻域组执行局部搜索时，它被认为是围绕该组的覆盖子空间进行搜索。原算法采用 2-opt 作为局部搜索方法，因 为 2-opt 在局部搜索操作中消耗的时间最少。但是后续的研究[6]认为，3-opt 的搜索效果将比 2-opt 的效果更好。由于 3-opt 的时间复杂度较高，我将选择算法替换为 or-opt2，在选择效果与时间上达成更好的平衡。Or-opt 搜索算法是由 Or（1976）提出的一种路径内改进

方法。[6]该方法是将一条路径中的 m 个连续的顾客节点在该路径中重新定位。由于置换的特殊性，无需在每次移动后重新评估遍历的长度(一次重新评估消耗一次适应度评估)。从运算复杂度来看，每一步搜索减去两条边，再加上两条边，也就是说操作次数是 4 次。一次完整的评估需要 N 次操作(N 表示排列中元素的个数)，因此， 当 or-opt-2 交换操作达到 N/4 次时，我们计算一次适应度评估。为了节省不必要的搜索步骤，所提出的选择性策略 将局部搜索偏向于有前途的和多样化的候选解的非关键边缘。

我们将邻域群大致分为以下两种情况：

1)大小恰好为 M_{min} 的邻居群倾向于收敛。对所有成员进行局部搜索是多余的。因此，选取最好的成员之一执行局部搜索。

2)对于其他群体，它们相对多样化，需要探索性搜索。选取行程最短的前一半成员，采用局部搜索。在上述过程中，忽略了 CES 中的临界边，这些临界边不经历 or-opt。

G. 替换操作

经过一系列的操作，我们得到了最终的后代。替换操作带来了选择压力，以决定哪些后代能够存活到下一代。选择压力促进了进化过程，选择过程为了适应原算法中的“离散多解”现象，我们采用更适合离散选择情况的锦标赛策略。为了有效地维持多个解，我们将选择压力限制在各自的邻域群体内，而不是限制在全局空间上，从而 同时在不同的潜在子空间中进行搜索。具体操作是，每个子节点都与最近的对应父节点邻域群的成员/成员竞争。同时，如果不止一个成员与孩子有相同的最大共享距离，则选择行程最长的成员作为竞争对手。比赛结束后，胜出者存活下来，进入下一代。

H. 精英选择方法

当满足终端条件时，算法停止并提供最终解。考虑到提高决策者的体验，原算法缩小了最终的解决方案。剔除冗余、不充分的解决方案，最终提供具有代表性的解决方案。邻域内群体应该覆盖一个子空间，群体中的最佳成员支配其他成员。在这种情况下，应该提供邻域群中的最优解。此外，我们还考虑了在同一邻群中包含多个最优的情况，当这些最优位置很近时。确定提供哪些解的具体操作如下所述：

找出解集的最短长度，并设置为 L_{min}。将 L_{min} 放大得到阈值长度 $L_{threshold}$。本文将阈值比率设置为 0.01。之后，通过分组收集有代表性的解决方案。对于每一个组，按行程长度对成员进行降序排序。在逐一检查它们之前，我们首先检查它们是否存在于档案中。如果候选解已经存在，我们就跳过它。否则，我们按如下三种情况检查候选解。

1)如果候选者的长度符合 $L_{\min}$。将此候选人直接添加到存档中。

2)如果候选解是其关联组中的最佳成员，且其长度大于 $L_{\min}$ 但小于等于 $L_{\text{threshold}}$，则依次计算候选解与存档中成员之间的共享距离。最大共享距离记录为 Sh_{max}。当 $Sh_{\max}$ 小于 0.8×N 时，候选方案与存档中的其他成员区分开来。当候选方案被区分或存档为空时，允许候选方案进入存档。

3)如果候选人在前两次不合格，则表明候选解决方案不充分且未被区分，因此该解决方案被忽略。**整个过程在算法 5 中总结**。

Algorithm 5 $\mathbb{S} \leftarrow$ Preserve(*Parent*)

```
L_min ← MinLength(Parent)
L_threshold ← L_min × (1 + ε)
S = {}
for each NeighborGroup ∈ Parent do
    sort(NeighborGroup.Len, "acs") /*Sort the members in NeighborGroup by length, in a descending order.*/
    for i = 1 to |NeighborGroup| do
        c ← NeighborGroup[i]
        if existed(S, c) then
            continue
        end if
        /* Occasion 1 */
        if c.Len == L_min then
            S ← S + c /*Preserve the chromosome best so far.*/
            /* Occasion 2 */
        else if i == 1 and c.Len ≤ L_threshold then
            Sh_max ← 0
            for s ∈ S do
                if S(s, c) > Sh_max then
                    Sh_max ← S(s, c)
                end if
            end for
            if Sh_max < 0.8 then
                S ← S + c /*Preserve the superior and diverse solution.*/
                break
            end if
            /*Occasion 3 */
        else
            break
        end if
    end for
end for
```

I. 复杂性分析

在 TSP 或 MSTSP 中，对一个解的求值就是对该解所表示的路径中边的长度求和。因此，对于一次评价，适应度评价的复杂度为 O(N)。相似度计算测量其复杂度也是 O(N)。为简单起见，将邻域大小表示为固定值 m，算法 2 中自适应邻域策略的计算开销即 $O(N\times NP^2/m)$。算法 3 中的多样性增强策略检测小生境是否收敛的代价为 $(NP/m)\times O(N\times m) = O(NP\times N)$。算法 4 中的 CES 更新策略成本最高的一步是识别不同持久领先者之间的共同边缘，时间复杂度为 $O(N\times NP/m)$。由于交叉和突变操作，NMA 的繁殖消耗 O(NP×N)。在最坏的情况下，选择性局部搜索也消耗 $O(NP\times N^2)$。每代种群的适应度评估平均花费 O (NP×N)。综上所述，每代算法的计算复杂度主要由局部搜索的复杂度所决定，即 $O(NP\times N^2)$，这与经典的以 or-opt-2 为局部搜索的 MA 相同。

Ⅳ.实验结果

A. 实验装置

在实验中，将所提出的 NMA 算法与四种离散 MMOP 算法进行了比较。首先，NACS[10]和 NGA[8]是之前发表的两种 MMOP 算法，此外，比较还实现了两种基本的 MMOP 算法，即 CGA[11]和 ShGA[11]。此外， 他们两人使用$(1-S(\pi_i,\ \pi_j))$测量两个排列之间的距离π_i和π_j。它们的交叉操作和突变操作分别为 PMX 和 EM。

ga 相关参数为:CGA 的交叉率 Pc=0.9，突变率 Pm=0.01，拥挤因子 CF=NP,CGA 和 ShGA 的生态位半径设为 0.2。在 NMA 自适应邻域策略中，邻域大小 m 被限制在 $M_{\min}$=4，$M_{\max}$=12 的范围内。对于所有算法，将种群大小 NP 统一设置为 150，最大适应度评估(MaxFEs)列于表 1，最终条件为 MaxFEs 耗尽。这五种算法在 25 个 MSTSTP 测试实例上进行了测试[8]。

B. 性能测量

1) f_β测度: f_β是精度值 P 和召回值 R 的综合指标，用来评价得到的解的质量。P 为所得解的分数即最优解:

$$P=\frac{TP}{TP+FP} \tag{7}$$

其中，TP 为返回解中最优解的个数，FP 为最终解集中非最优解的个数。R 是成功定位的真值解的分数:

$$R=\frac{TP}{TP+FN} \tag{8}$$

其中 FN 是算法遗漏的最优解的个数。实际上，TP 和 FN 的总和是基准中期望解决方案的总数。基于 P 和 R 的值，计算出f_β为

$$F_\beta=\frac{(1+\beta^2)\times P\times R}{\beta^2\times P+R} \tag{9}$$

对于具有大量最优解的测试实例，获得最具代表性的最优解比穷尽地将它们全部定位更重要，便将 β^2的值设置为 0.3，以放大精度的影响来评价所获得的解。此外，三个指标(P、R 和F_β)在[0,1]中为实值。特别是有两种极端情况:当算法在理想条件下提供了完全期望的解时，F_β得分为 1;当算法无法找到任何令人满意的解时，F_β得分为 0。

2) 多样性指标(Diversity Indicator,DI):多样性指标是评价算法性能的另一个重要指标。当算法找不到任何想要的解时，它们的F_β值都为零。在这些情况下，DI 有助于进一步区分不同算法的性能。具体来说，DI 是根据获得的解与真解之间的平均最大相似度来定义的，其计算式为

$$DI(\mathbb{P},\mathbb{S})=\frac{\sum_{i=1}^{|\mathbb{P}|}\max S(P_i,P_j)\big|_{j=1,\ldots\ldots|\mathbb{S}|}}{N\times\mathbb{P}} \tag{10}$$

其中P_i是 P 的第 i 个排列，$S(P_i,P_j)$是 S 的 i,j 个排列。$\sum_{i=1}^{|\mathbb{P}|}\max S(P_i,P_j)\big|_{j=1,\ldots\ldots|\mathbb{S}|}$使用(2)测量排列$P_i$与$S$中每个排列之间的最大共享距离。

C. 与离散多解优化器的比较

1) f_β分数: f_β分数衡量解的质量。分数高表示解集质量好。进行模拟 MA，数值结果和显

著性结果见表二，其中对应的 MSTSP 放在左边，对应的f_β得分放在右边。如表所示，NMA 在大多数 MSTSPs 上取得了显着更好的结果， 特别是在简单 MSTSPs 和几何 MSTSPs 上。除了 ShGA 的 MSTSP12 外，两种改进的基本 MMOP 算法 CGA 和 ShGA 在大多数 MSTSPs 上表现不佳。随着城市规模的增大，5 种算法的f_β值迅速下降，但 NMA+算法的表现略好，甚至好过原算法。

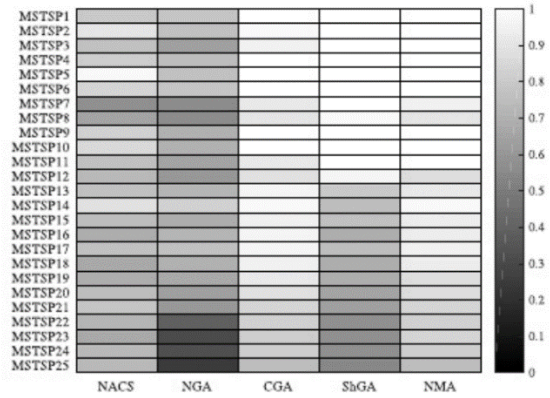
实验结果验证了 NMA+算法在f_β得分方面优于其他比较算法，但它们在处理大型城市规模的 MSTSPs 时都遇到困难。此外，NMA+可以在各种最优尺寸的实例中获得高精度值，从而进一步获得良好的f_β值。

表一 在 25 次 MSTSPS 上的F_β得分

TSP	NACS	NGA	CGA	SHGA	NMA	NMA+
1	0.684	0.973	0.024	0.026	1.000	**1.000**
2	0.804	0.959	0.030	0.034	1.000	**1.000**
3	0.497	0.936	0.078	0.110	1.000	**1.000**
4	0.724	0.932	0.034	0.034	1.000	**1.000**
5	0.989	0.846	0.017	0.017	1.000	**1.000**
6	0.643	0.877	0.034	0.034	1.000	**1.000**
7	0.125	0.769	0.261	0.435	0.901	**1.000**
8	0.137	0.578	0.337	0.700	0.772	**0.778**
9	0.768	0.974	0.034	0.034	1.000	**1.000**
10	0.813	0.969	0.034	0.034	1.000	**1.000**
11	0.459	0.949	0.072	0.118	1.000	**1.000**
12	0.090	0.331	0.275	0.919	0.535	**0.564**
13	0.025	0.096	0.255	0.003	**0.611**	0.562
14	0.087	0.172	0.090	0.000	0.883	**0.905**
15	0.004	0.416	0.219	0.003	**0.732**	0.697
16	0.000	0.054	0.211	0.000	0.532	**0.553**
17	0.004	0.416	0.219	0.003	0.557	**0.650**
18	0.000	0.031	0.062	0.000	0.571	**0.650**
19	0.000	0.007	0.040	0.000	0.168	**0.188**
20	0.000	0.000	0.015	0.000	0.118	**0.128**
21	0.012	0.000	0.001	0.000	0.000	**0.118**
22	0.000	0.000	0.000	0.000	0.000	**0.000**
23	0.000	0.000	0.000	0.000	0.000	**0.000**
24	0.000	0.000	0.009	0.000	**0.101**	0.009
25	0.000	0.000	0.000	0.000	0.000	**0.000**

2） DI:DI 评估解的多样性。DI 越高，多样性越好。将 NMA+与 NMA 和其他算法相比较。

图一 在 25 次 MSTSPS 上的F_β得分



表二 在 25 次 MSTSPS 上的DI得分

DI	NMA	NMA+
MSTSP1	1.000	**1.000**
MSTSP2	1.000	**1.000**
MSTSP3	1.000	**1.000**
MSTSP4	1.000	**1.000**
MSTSP5	1.000	**1.000**
MSTSP6	1.000	**1.000**
MSTSP7	0.933	**0.951**
MSTSP8	0.875	**0.885**
MSTSP9	1.000	**1.000**
MSTSP10	1.000	**1.000**
MSTSP11	1.000	**1.000**
MSTSP12	0.874	**0.875**
MSTSP13	**0.926**	0.919
MSTSP14	0.985	**0.981**
MSTSP15	0.930	**0.931**
MSTSP16	0.913	**0.924**
MSTSP17	0.920	**0.965**
MSTSP18	0.954	**0.958**
MSTSP19	**0.982**	0.875
MSTSP20	0.857	**0.884**
MSTSP21	0.796	**0.802**
MSTSP22	0.775	**0.781**
MSTSP23	0.822	**0.824**
MSTSP24	**0.850**	0.737
MSTSP25	0.730	**0.749**

由图 1 可知，相对而言，CGA 和 NMA 在总体 MSTSP 实例上具有较好的多样性;ShGA 在简单和几何 MSTSPs 上表现良好，但在复合 MSTSPs 上表现不佳;NACS 和 NGA 在总体问题实例上的结果并不令人满意。进行显著性检验，结果见表 2。从表中可以看出，NM+显著优于 NMA，分别为 25 分中的 22 分。这些结果表明，CGA 和 ShGA 可以获得高 DI，因为它们提供了一个大的存档，而 NMA 也可以通过提供代表性的最优来实现类似的性能，而 NMA+通过改变局部搜索策略和动态调整变异率实现了多样性的进一步增强，比原算法有稳定的提升。

V.结论

本文改进了一种求解 mstsp 的 NMA 算法。NMA+利用邻域生态位策略和 MA 有效地实现了 MSTSP 多解定位的目标。邻域小生境策略在搜索过程中保持了候选对象的多样性，MA 保证了搜索能力。为了进一步提高算法性能与多样性，设计了随种群适应度而动态变化的变异率与交叉率，使得种群在保持最优化的同时增强了多样性。本文将四种增强方法中的局部选择策略修改，将 2-opt 搜索方法改为 or-opt 算法，在保证时间复杂度可接受的情况下进一步提高局部搜索的性能，以最大化适应度评估的利用率。为了验证所提算法的效率和有效性，将所提 NMA 算法与现有的两种离散 MMOP 算法和两种改进的小生境 MMOP 算法进行了比较。NMA 在两个评估指标方面优于四种比较算法。本文仅仅是针对原本的小生境算法进行修改，未来仍有值得挖掘的地方，比如调整预生成种群的方法，在 PMX 交叉过程中引入顺序交叉等方式，以进一步优化算法的性能。

References

[1] G. Gutin and A. P. Punnen, The Traveling Salesman Problem and Its Variations. NewYork, NY, USA: Springer, 2007, pp. 1–15.

[2] K. Ilavarasi and K. S. Joseph, “Variants of travelling salesman problem: A survey,”in Proc. Int. Conf. Inf. Commun. Embedded Syst., Feb. 2014, pp. 1–7.

[3] Z.Beheshti and S.M.H.Shamsuddin, “A review of population-based metaheuristic algorithms,” Int. J. Adv. Soft Comput.Appl., vol.5, no.1, pp.1–35, 2013.

[4] Y. Wang, H.-X. Li, G. G. Yen, and W. Song, “MOMMOP: Multi objective optimization for locating multiple optimal solutions of multimodal optimization problems,” IEEE Trans. Cybern., vol. 45, no. 4, pp. 830–843, Apr. 2015.

[5] T. Huang, Y. Gong, S. Kwong, H. Wang and J. Zhang, “A Niching Memetic Algorithm for Multi-Solution Traveling Salesman Problem,” IEEE Transactions on Evolutionary Computation. DOI: 10.1109/TEVC.2019.2936440.

[6] Zhong, X. (2021). On the Approximation Ratio of the 3-Opt Algorithm

[7] 张友鹏,熊伟清.一种改进的实数编码遗传算法[J].铁道学报,2004,26(6):67-70

[8] T. Huang, Y.-J. Gong, and J. Zhang, “Seeking multiple solutions of combinatorial optimization problems: A proof of principle study,” in Proc. IEEE Symp. Comput.Intell., 2018, pp. 1212–1218. for the (1, 2)-TSP. Oper. Res. Lett., 49, 515-521.

[9] S. Ronald, “Finding multiple solutions with an evolutionary algorithm,” in Proc.IEEE Int. Conf. Evol. Comput., vol. 2, Nov. 1995, pp. 641–646.

[10] X.-C. Han, H.-W. Ke, Y.-J. Gong, Y. Lin, W.-L. Liu, and J. Zhang, “Multimodal optimization of traveling salesman problem: A niching ant colony system, ” in Proc. Genet. Evol. Comput. Conf. Companion, 2018, pp. 87–88.

[11] R. Thomsen, “Multimodal optimization using crowding-based differ-ential evolution,” in Proc. IEEE Congr.Evol. Comput., vol. 2, 2004, pp. 1382–1389

[12] Y.-X. Liu etal. , “Solving NP-hard problems with physarum-based ant colony system,” IEEE/ACM Trans. Comput. Biol. Bioinf., vol. 14, no. 1, pp. 108–120, Jan./Feb. 2017

作者 黄鸿展，华南理工大学计算机科学与工程学院 2022 级计算机科学与技术本科生。

Author Hong-Zhan Huang is the 2022 undergraduate student of Computer Science and Technology, School of Computer Science and Engineering, South China University of Technology, Guangzhou, China