AcoFlow: Ant Colony Optimization for Maximum Flow

摘要

寻找网络中的最大流问题在 Ford-Fulkerson Method 中得到了很好的解决，然而问题在于如何设计更好的启发式信息来帮助我们寻找每个迭代下的增广路径。然而启发式搜索往往面临近似得到某个结果而并非精确解的弊病，且启发式系统的性能在许多高度离散问题上表现得并不稳定，但是这种搜索范式能否在最大流问题上凑效依然是值得探究的问题。本文试图通过以蚁群算法为代表的进化算法来探究启发式搜索在寻找网络最大流问题上的可行性。我们的代码和实验结果都已上传到:https://github.com/Selen-Suyue/ML-Homework/tree/main/Max_ low。

1 蚁群算法解决网络最大流问题的框架

蚁群算法（ACO）解决网络最大流问题的框架包括以下两个主要过程：

1. **将流量图的特征迁移到适应蚁群算法的特征：路径长度、信息素等。**
2. **更新余量图，使其适配蚁群算法的搜索需求。**

1. 特征迁移到蚁群算法

在蚁群算法中，蚂蚁需要在图中搜索路径，而每条路径的“长度”需要被定义。为了适应最大流问题，我们将路径长度表示为流量的倒数：

$$L_{ij} = \frac{1}{Q_{ij}}$$

其中，L_{ij} 表示从节点 i 到节点 j 的路径长度，Q_{ij} 是该路径的流量。这样，流量越大的路径，蚁群搜索的可能性就越高，从而促使蚂蚁选择流量更大的路径。

对于信息素更新，我们假设蚂蚁在搜索路径后，会将路径上的信息素值更新为该路径上的最大流量 $f_{ij}^{\max}$，即：

$$\tau_{ij}^{\text{new}} = \tau_{ij}^{\text{old}} + \alpha \cdot f_{ij}^{\max}$$

其中，τ_{ij}^{old} 是路径上原有的信息素值，τ_{ij}^{new} 是更新后的信息素值，α 是信息素的蒸发系数，$f_{ij}^{\max}$ 是路径 (i, j) 上的最大流量。

2. 余量图的更新和迭代过程

在每次搜索到增广路径后，蚁群算法需要更新余量图。假设当前图的流量为 f_{ij}，则经过一轮增广后，路径 (i, j) 的流量更新为：

$$f_{ij}^{\text{new}} = f_{ij}^{\text{old}} - f_{ij}^{\max}$$

这意味着每次增广路径的流量会从原图中减去，生成新的余量图。下一轮迭代则在余量图上进行，计算新的路径流量，并根据余量图重新初始化信息素。这个过程将持续进行，直到蚂蚁无法找到任何增广路径为止。

3. 剪枝操作

在某些情况下，图中的某些边流量很大，但其后续没有连接到有余量的边，这些边会成为“死路”，导致蚂蚁陷入局部最优解。为避免这种情况，我们采用剪枝操作，即当蚂蚁走到这些“死路”时，立即将所有经过该边终点的边的信息素值永久置为零：

$$\tau_{ij} = 0, \quad \forall j \in \mathcal{N}_i \text{ where } f_{ij} = 0$$

其中，$\mathcal{N}_i$ 表示与节点 i 相邻的所有节点集合，$f_{ij} = 0$ 表示该边没有余量流量。

通过这种剪枝操作，我们可以有效避免搜索陷入局部陷阱，促使蚁群继续向其他可能的路径搜索。

在前面的基础上，我们设计出了 AcoFlow 算法以实现最大流问题如表1所示。

Algorithm 1: AcoFlow: Ant Colony Optimization for Maximum Flow

Input: Graph $G = (V, E)$, ants N, parameters α, β, ρ, iterations T

Output: Maximum flow sum_flow

Initialize pheromone $\tau_{ij} \leftarrow 1$ for all $i, j \in V$;

sum_flow $\leftarrow 0$

for $t = 1$ ***to*** T **do**

  Reset $\tau_{ij} \leftarrow 1$ for all $i, j \in V$;

  max_flow $\leftarrow 0$

  for $k = 1$ ***to*** N **do**

    path $\leftarrow []$, flow $\leftarrow \infty$, current $\leftarrow 0$

    while *current* $\neq$ *end* **do**

      path $\leftarrow$ path $\cup$ {current} Compute probabilities:

$$P_{ij} = \frac{\tau_{ij}^{\alpha} \cdot c_{ij}^{\beta}}{\sum_{j \in \mathcal{N}(i)} \tau_{ij}^{\alpha} \cdot c_{ij}^{\beta}}$$

      next $\leftarrow$ SelectNextNode(P_{ij}) **if** *next* $= \varnothing$ **then**

        $\tau_{ij} \leftarrow 0$ for all $(i, j) \in$ path;

        return $\varnothing, \varnothing$

      flow $\leftarrow \min(\text{flow}, c_{\text{current, next}})$ current $\leftarrow$ next

    path $\leftarrow$ path $\cup$ {end}

    if *flow* $>$ *max_flow* **then**

      max_flow $\leftarrow$ flow

    Update pheromone: $\tau_{ij} \leftarrow \tau_{ij} +$ flow for $(i, j) \in$ path

  if *max_flow* $= 0$ **then**

    return sum_flow

  Reduce flow in G along best_path: $f_{ij} \leftarrow f_{ij} -$ max_flow

  sum_flow $\leftarrow$ sum_flow + max_flow

return sum_flow

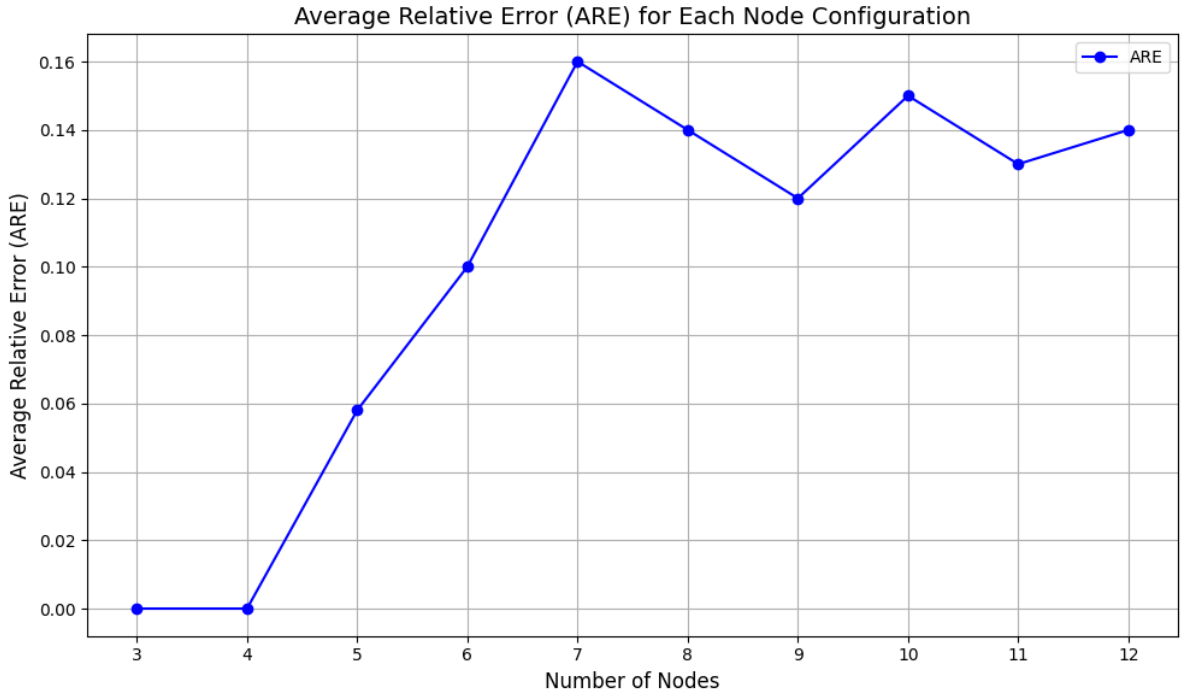

图 1: AcoFlow 优化误差

2 实验

在实验部分，我们以 Ford-Fulkerson 算法的结果为 Ground Truth，节点数依次设置为 3-12，并在每个节点树上随机生成 10 张有向图和对应流量以进行更充分的测试，我们的每个测试点结果均发布在摘要提供的链接中，本文暂且不附。我们以相对误差的平均值作为评价指标：

$$\text{ARE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{|y_i|}$$

这其中 y_i 为真实流量而 $\hat{y}_i$ 为 AcoFlow 搜索出的最大流量。结果如图1所示。可以发现在节点数较低的时候，AcoFlow 可以做到较高的成功率，而在此之后误差逐渐增高，最后在 14% 上下波动（由于设备原因单次实验耗时较长，因此不便进行后续实验，但是我们测试在节点为 20 的时候，ARE 为 0.1198，依然在此区间内）。

我们认为造成 AcoFlow 不够精确和可以改进的地方有以下几点：

1. 首先，AcoFlow 对于路径的搜索完全依赖于蚁群算法的信息素激励（我们没有和 FF 算法一样借助 BFS 搜索），它不太稳定，尤其是在面临节点数和路径增多的条件下，迭代到后期流量都相对较少的情况下，往往不能够搜索完全部的增广路径。在后续的改进版本中，我们考虑是否借助 BFS 直接化简余量图为最简形式而不依赖于蚁群减枝，减小搜索范围再交给 AcoFlow 处理。另一种思路是提供一种长期的信息素变化函数，而不是每次余量图更新后直接初始化而放弃先验信息。

2. 其次，我们应考虑是否需要动态调整规模：如迭代次数靠前的周期，是否需要更多的蚁群来搜索出更值得信赖的增广路径以避免因为过度贪心而错误地删除了有效边。而在迭代次数靠后的周期，我们目前的做法是一旦依次迭代完全没有搜出增广路径就宣告结束，是否需要更合理的机制，如尝试更多次的搜索，或在这个节点再调用 BFS 之类的辅助工具帮助蚁群避免落入局部最优。

3 总结

AcoFlow 的结果与预期的优化方法处理该问题所可能出现的缺陷有较多一致之处，启发式搜索在优化后期的乏力是 AcoFlow 后续的核心问题，我们将继续优化该项目以期达到更好的解决方案。