基于灰狼优化算法的多旅行商问题的求解

一、灰狼优化算法

1.算法起源

灰狼优化算法是由Seyedali Mirjalili等人于2014提出的一种启发式智能优化算法。

灰狼被认为是顶级掠食者，意味着它们位于食物链的顶端部分。灰狼大多数喜欢群居，群体的平均数量一般为5到12只。它们有一个非常严格的社会等级制度，如下图所示：

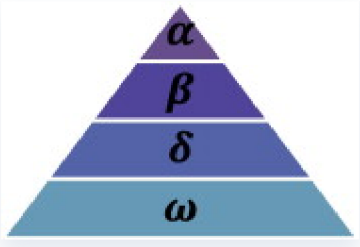

图 1.1.1 灰狼的社会等级制度

头狼是一只雄性和一只雌性，称为α狼，α狼主要负责狩猎、睡觉地点和起床时间等事项。所有狼都必须遵从α狼的决定。然而，在自然界中也观察到了一些"民主"行为，即α狼追随狼群中的其他狼。在聚集的时候，整个狼群会通过下垂尾巴的方式来表达对α狼的服从。α狼也被称为优势狼，因为狼群必须服从它的命令。只有α狼才被允许在狼群中交配。有趣的是，α狼不一定是狼群中最强壮的成员，但在管理狼群方面却是最优秀的。这表明在狼群中，与力量相比，它们更看重组织性和纪律性。

灰狼等级制度的第二层是β狼。β狼是从属狼，帮助α狼进行决策或参与狼群的其他活动。β狼可以是雄性也可以是雌性，如果α狼中的一只去世或变老，β狼很可能是成为α狼的最佳人选。β狼应该尊重α狼，同时指挥其他低级别的狼。它扮演着α狼的顾问和狼群纪律维护者的角色。β狼在整个狼群中强化α狼的命令，并向α狼提供反馈。

等级最低的灰狼是ω狼。ω狼扮演着替罪羊的角色。ω狼总是不得不服从所有其他占主导地位的狼。它们是最后被允许进食的狼。ω狼在狼群中看似并不重要，但据观察，如果失去ω狼，整个狼群都会面临内部争斗和问题。这是因为ω狼为所有狼宣泄了暴力和挫败感。这有助于满足整个狼群的需求，并保持其等级制度。在某些情况下，ω狼也是狼群中的保姆。

如果一只狼不是α狼、β狼或ω狼，那么他/她就被称为从属狼（在某些资料中也被称为δ狼）。δ狼必须服从α狼和β狼，但他们可以支配ω狼。侦察狼、哨兵狼、长老狼、猎狼和保育狼都属于这一类。侦察狼负责监视领地的边界，并在遇到危险时向狼群发出警告。哨兵狼负责保护和保障狼群的安全。长老狼是曾经担任过α狼或β狼的有经验的狼。猎狼在捕猎猎物和为狼群提供食物时协助α狼和β狼。最后，保育狼负责照顾狼群中弱小、生病和受伤的狼。

除了狼的社会等级制度外，群体狩猎也是灰狼另一种有趣的社会行为。根据研究人员的发现，灰狼狩猎的主要阶段如下：

- 追踪、追赶并接近猎物。
- 追捕、包围并骚扰猎物，直到其停止移动。
- 向猎物发起攻击。

图 1.1.2 灰狼狩猎行为：(A) 追赶、接近和追踪猎物 (B-D) 追捕、骚扰和包围 (E) 静止状态和攻击

对灰狼的狩猎技术和社会等级制度进行数学建模，从而设计出并优化灰狼优化算法（Grey Wolf Optimizer，GWO）。

2.数学模型和算法

社会等级制度

在设计GWO算法时，为了对数学建模狼的社会等级制度，将最优解视为α。因此，第二和第三优解分别被称为β和δ。其余候选解被假定为ω。在GWO算法中，狩猎（优化）过程由α、β和δ引导。ω狼跟随这三只狼。

包围阶段

灰狼在狩猎时会包围猎物。为了用数学方法模拟包围行为，提出以下方程：

$$\vec{D} = \left| \vec{C} \cdot \vec{X}_p(t) - \vec{X}(t) \right| \tag{1.2.1}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{1.2.2}$$

其中t表示当前迭代次数，$\vec{A}$和$\vec{C}$是系数向量，$\vec{X}_p$是猎物的位置向量，$\vec{X}$表示灰狼的位置。

$\vec{A}$和$\vec{C}$的计算公式如下：

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{1.2.3}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{1.2.4}$$

其中，$\vec{a}$的分量在迭代过程中从2线性递减到0，$\vec{r}_1$和$\vec{r}_2$是$[0, 1]$区间内的随机向量。

为了观察公式1.2.1和1.2.2的效果，图1.2.1(a)展示了一个二维位置向量和一些可能的邻近点。如图所示，位于（X，Y）的灰狼可以根据猎物（X^*，Y^*）的位置来更新自己的位置。通过调整$\vec{A}$和$\vec{C}$的值，可以从当前位置到达最佳个体周围的不同位置。

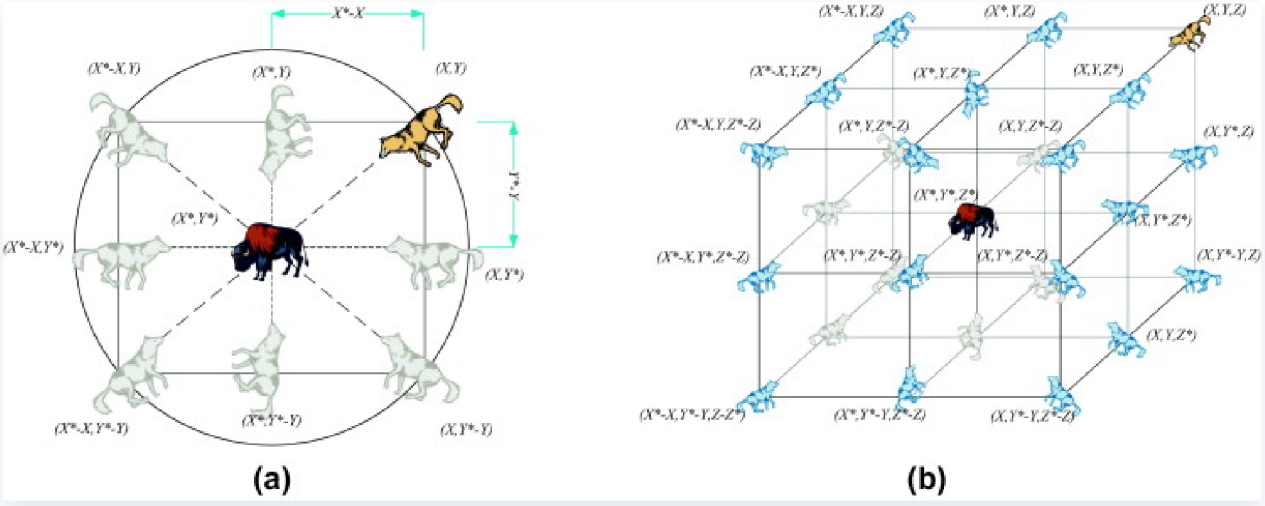

图 1.2.1 二维和三维位置向量及其可能的下一个位置

例如，通过设置$\vec{A} = (1, 0)$和$\vec{C} = (1, 1)$，可以达到（$X^* - X, Y^*$）的位置。灰狼在三维空间中可能的更新位置如图1.2.1(b)所示。请注意，随机向量r_1和r_2使灰狼能够到达图1.2.1(b)中所示各点之间的任何位置。因此，灰狼可以使用公式1.2.1和1.2.2在猎物周围的空间内的任何随机位置更新其位置。

同样的概念可以扩展到n维搜索空间，灰狼将在迄今为止获得的最佳解决方案周围的超立方体（或超球体）中移动。

狩猎

灰狼能够识别猎物的位置并将其包围。狩猎通常由α狼引导。有时，β狼和δ狼也可能参与狩猎。然而，在抽象的搜索空间中，我们不知道最优解（猎物）的位置。为了在数学上模拟灰狼的狩猎行为，我们假设α狼（最佳候选解）、β狼和δ狼对猎物可能的位置有更好的了解。因此，我们保存迄今为止获得的前三个最佳解，并强制其他搜索个体（包括最底层的ω狼）根据最佳搜索个体的位置来更新自己的位置。在这方面，我们提出了以下公式。

$$\vec{D}_\alpha = \left|\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}\right|, \vec{D}_\beta = \left|\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}\right|, \vec{D}_\delta = \left|\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}\right| \tag{1.2.5}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \left(\vec{D}_\alpha\right), \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \left(\vec{D}_\beta\right), \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \left(\vec{D}_\delta\right) \tag{1.2.6}$$

$$\vec{X}(t+1)=\frac{\vec{X}_1+\vec{X}_2+\vec{X}_3}{3} \tag{1.2.7}$$

图1.2.2展示了在二维搜索空间中，搜索个体如何根据α狼、β狼和δ狼的位置来更新自己的位置。可以观察到，最终位置将在由搜索空间中α狼、β狼和δ狼位置所定义的圆内的某个随机位置。换句话说，α狼、β狼和δ狼估计猎物的位置，而其他狼则在猎物周围随机更新自己的位置。

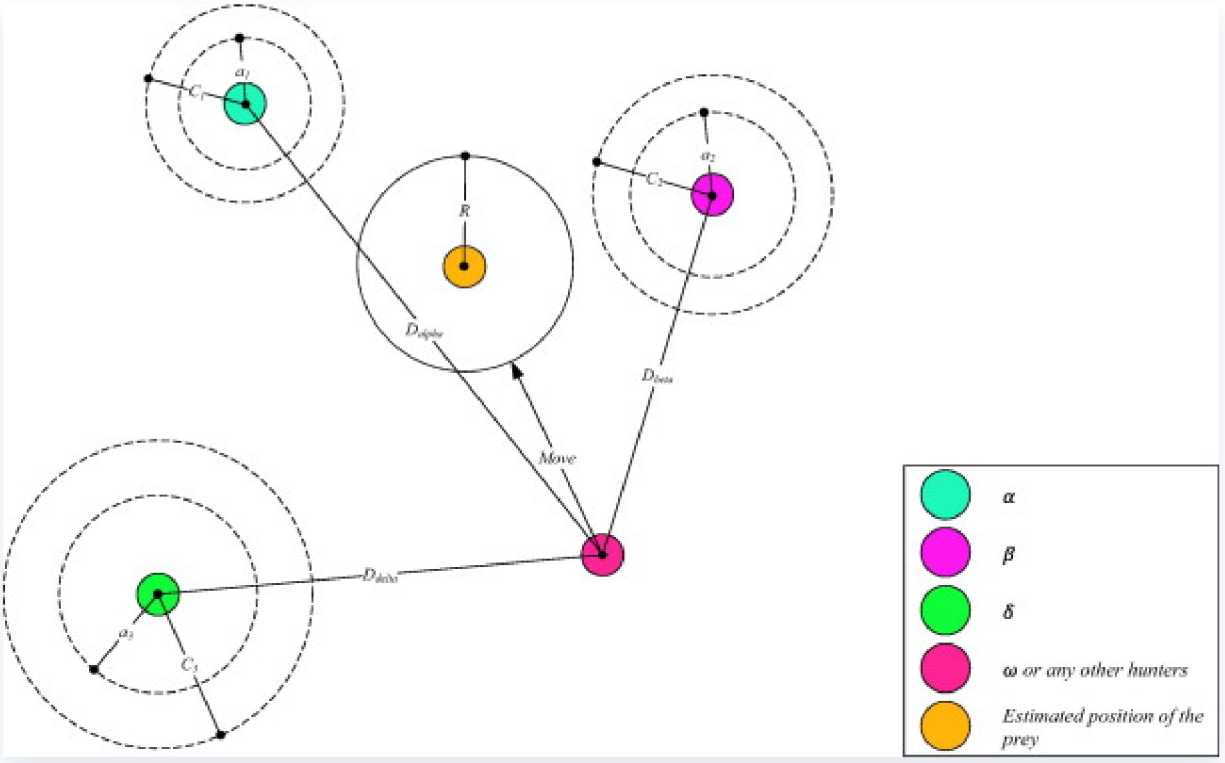

图 1.2.2 灰狼优化算法中的位置更新

攻击猎物（开发/利用）

如上所述，当猎物停止移动时，灰狼会通过攻击猎物来完成狩猎。为了在数学上模拟接近猎物的过程，我们会减小$\vec{a}$的值。请注意，$\vec{A}$的波动范围也会随着$\vec{a}$的减小而减小。换句话说，$\vec{A}$在区间$[-2a, 2a]$内是一个随机值，其中a在迭代过程中从2减小到0。当$\vec{A}$的随机值在$[-1, 1]$范围内时，搜索个体的下一个位置可以位于其当前位置和猎物位置之间的任何位置。图1.2.3(a)显示，$|A|<1$会迫使灰狼向猎物发起攻击。

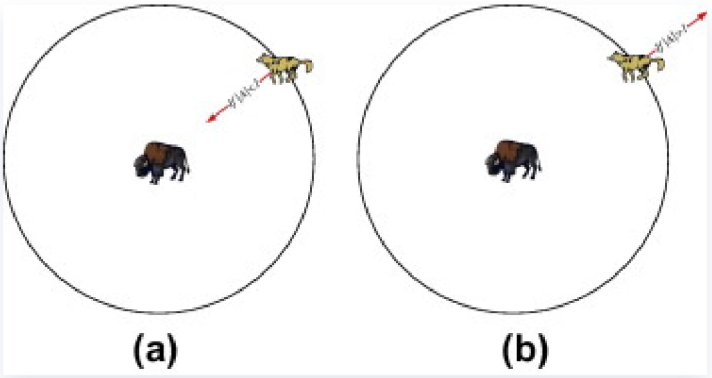

图 1.2.3 攻击猎物与搜寻猎物

到目前为止，所提出的操作符使灰狼优化算法允许其搜索个体根据α狼、β狼和δ狼的位置来更新自己的位置，并向猎物发起攻击。然而，使用这些操作符，灰狼优化算法容易陷入局部解的停滞状态。诚然，所提出的包围机制在一定程度上表现出了探索性，但灰狼优化算法需要更多的操作符来强调探索。

搜寻猎物（探索）

灰狼主要根据α狼、β狼和δ狼的位置进行搜索。它们会相互分散以搜寻猎物，并在发现猎物时聚拢以发起攻击。为了在数学上模拟这种分散行为，我们使用$\vec{A}$(随机值大于1或小于-1)的来强制搜索个体远离猎物。这强调了探索的重要性，并使灰狼优化算法能够进行全局搜索。图1.2.3(b)也显示，$|A| > 1$会迫使灰狼远离猎物，以期找到更合适的猎物。

灰狼优化算法中另一个有利于探索的组成部分是$\vec{C}$。如等式1.2.4所示，$\vec{C}$向量包含$[0, 2]$区间内的随机值。这个组成部分为猎物提供了随机权重，以便在等式1中定义的距离时随机地强调（$C > 1$）或淡化（$C < 1$）猎物的影响。这有助于灰狼优化算法在整个优化过程中表现出更随机的行为，从而有利于探索并避免陷入局部最优。值得一提的是，与A不同，C的值并不是线性减小的。我们故意要求C始终提供随机值，以便不仅在初始迭代期间强调探索，而且在最终迭代期间也强调探索。在陷入局部最优停滞的情况下，尤其是在最终迭代中，这个组成部分非常有用。

$\vec{C}$也可以被视为自然界中接近猎物时遇到的障碍的影响。一般来说，自然界中的障碍会出现在狼的狩猎路径上，实际上会阻止它们快速便捷地接近猎物。这正是$\vec{C}$的作用。根据狼的位置，$\vec{C}$可以随机地为猎物分配一个权重，使猎物对狼来说更难到达或距离更远，或者反之亦然。

综上所述，在灰狼优化算法中，搜索过程从创建一群随机的灰狼（候选解）开始。在迭代过程中，α狼、β狼和δ狼会估计猎物的可能位置。每个候选解都会更新其与猎物的距离。参数a从2减小到0，以分别强调探索和利用。当$|A| > 1$时，候选解倾向于远离猎物进行搜索（探索）；而当$|A| < 1$时，候选解则向猎物聚拢以发起攻击（利用）。最后，当满足终止条件时，灰狼优化算法结束。

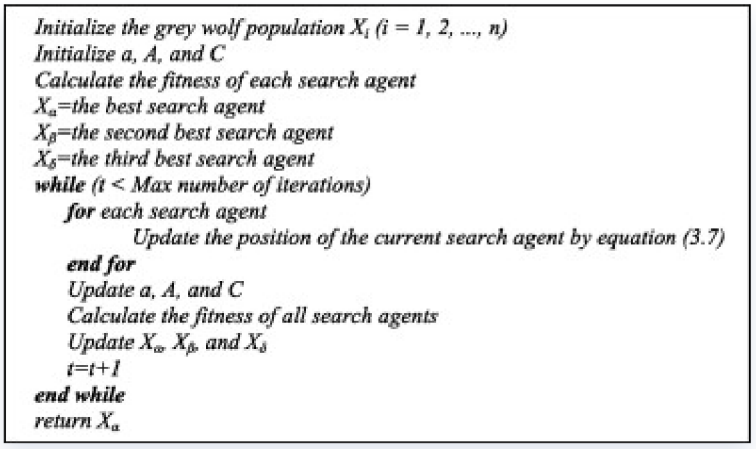
Initialize the grey wolf population X_i (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
X_α=the best search agent
X_β=the second best search agent
X_δ=the third best search agent
***while** (t < Max number of iterations)*
  ***for** each search agent*
    Update the position of the current search agent by equation (3.7)
  end for
  Update a, A, and C
  Calculate the fitness of all search agents
  Update X_α, X_β, and X_δ
  t=t+1
end while
return X_α

图 1.2.4 灰狼优化算法的伪代码

为了理解灰狼优化算法在理论上如何解决优化问题，我们可以注意以下几点：

- 所提出的社会等级制度有助于灰狼优化算法在迭代过程中保存迄今为止获得的最佳解决方案。

- 所提出的包围机制在解决方案周围定义了一个圆形邻域，该邻域可以扩展到更高维度以形成超球体。
- 随机参数A和C使候选解决方案具有不同随机半径的超球体。
- 所提出的狩猎方法允许候选解决方案定位猎物的可能位置。
- 参数a和A的自适应值确保了探索和开发的进行。
- 参数a和A的自适应值使灰狼优化算法能够平稳地在探索和开发之间过渡。
- 随着A的减小，一半的迭代用于探索（$|A| \geqslant 1$），另一半用于开发（$|A| < 1$）。
- 灰狼优化算法只有两个主要参数需要调整（a和C）。

虽然有可能整合变异和其他进化操作符来模拟灰狼的整个生命周期，但我们尽量简化了灰狼优化算法，使其需要调整的操作符尽可能少。

二、多旅行商问题

1.问题概述

多旅行商问题（MTSP）是众所周知的组合优化问题——旅行商问题（TSP）的变体，在TSP中规定只允许有一个旅行商从起始城市出发，遍历其余所有城市，最后返回起始城市。但在MTSP中，我们允许多个旅行商都从某一个起始城市出发，然后各自前往其余城市，但限定每个城市只能被一个旅行商所访问，且除起点外的所有城市都必须被旅行商所访问，最后所有旅行商再返回起始城市。

用更形式化的语言描述MTSP，假设城市数目为n，旅行商数目为m，则MTSP可简单地描述为：m个旅行商从同一城市S出发，各自沿着一条旅行路线行走，使每个城市有且仅有一个旅行商经过（除出发城市S），这m个旅行商最后返回起始城市。MTSP问题的求解目标是最小化m个旅行商行走距离的最大值。

可将MTSP定义在有向图$G=(V,A)$，其中V是n个顶点的集合，A是弧的集合。d_{ij}表示节点i和j之间的距离。x_{ij}（取值1或者0）表示弧(i,j)是否被选择。u_i表示一个旅行商从起点出发前往节点i时经过的节点数目，即节点i在当前旅行商路径中的位置序号。因此，MTSP的数学模型如下所示（节点1为起点城市，当然同时也是终点）：

$$\min\left[\max\left(z_1,z_2,\cdots,z_m\right)\right] \tag{2.1.1}$$

$$z_k=\sum_{i=1}^{n}\sum_{j=1}^{n}d_{ij}x_{ijk},k=1,2,\cdots,m \tag{2.1.2}$$

$$\sum_{j=2}^{n}x_{1j}=m \tag{2.1.3}$$

$$\sum_{j=2}^{n}x_{j1}=m \tag{2.1.4}$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 2, \cdots, n \tag{2.1.5}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 2, \ldots, n \tag{2.1.6}$$

$$u_i - u_j + (n-m)x_{ij} \leqslant n - m - 1 \tag{2.1.7}$$

$$2 \leqslant i \neq j \leqslant n \tag{2.1.8}$$

$$x_{ij} \in \{0, 1\}, \forall (i, j) \in A \tag{2.1.9}$$

式(2.1.1)表示最小化m个旅行商中行走距离最大的那个值；式(2.1.2)表示各个旅行商的行走距离；约束(2.1.3)和(2.1.4)保证恰好有m个旅行商从节点1出发，最后返回节点1；约束(2.1.5)和(2.1.6)保证每个节点只能被访问一次（除节点1外）；约束(2.1.7)防止形成任何不包含节点1的路线。

2.编码与解码

假设我们有9个城市，编号为$0 \sim 8$，旅行商数目为3，初始3个旅行商都从城市0出发，则GWO求解MTSP采用的编码如下图所示：

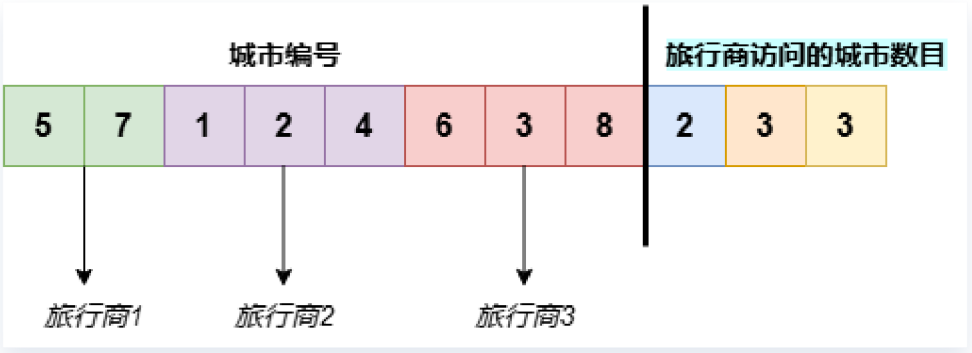

图 2.2.1 编码格式

灰狼个体被竖线分为两部分，左侧为8个城市，右侧为3个数字，表示每个旅行商访问的城市数目。在对灰狼个体进行编码后，需要对灰狼个体进行解码以得到旅行商的行走路线方案。上图中3个旅行商的行走路线如下：

- 旅行商1：$0 \to 5 \to 7 \to 0$
- 旅行商2：$0 \to 1 \to 2 \to 4 \to 0$
- 旅行商3：$0 \to 6 \to 3 \to 8 \to 0$

综上所述，若城市数目为n，旅行商数目为m，则灰狼个体长度为$n + m - 1$，其中前$n - 1$个数字表示城市编号（除起点城市以外），后m个数字表示m个旅行商各自访问城市的数目，且m必须大于等于1。

3.种群初始化

假设种群数目为$NIND$，城市数目为n，旅行商数目为m，那么初始种群中的任意一个灰狼个体都按照如下方式进行初始化：

- 将除起点城市以外的$n-1$个城市编号进行随机排序，将该排序结果作为灰狼个体的前$n-1$个数字。
- 随机生成m个数字，规定这m个数字之和等于$n-1$，且每个数字都必须大于等于1，然后将生成的m个数字作为灰狼个体的后m个数字。

4.灰狼位置更新

假设当前灰狼个体为$X(t)$，灰狼α为$X_\alpha(t)$，灰狼β为$X_\beta(t)$，灰狼σ为$X_\sigma(t)$，则$X(t)$的位置更新公式如下：

$$\mathrm{X(t+1)}=\begin{cases}\mathrm{Cross}\left[X(t),X_\alpha(t)\right], & \mathrm{rand}\leqslant\frac{1}{3}\\ \mathrm{Cross}\left[X(t),X_\beta(t)\right], & \frac{1}{3}<\mathrm{rand}<\frac{2}{3}\\ \mathrm{Cross}\left[X(t),X_\delta(t)\right], & \mathrm{rand}\geqslant\frac{2}{3}\end{cases} \tag{2.4.1}$$

其中，$Cross$为对两个灰狼个体进行交叉操作；$rand$为$0\sim1$的随机数。

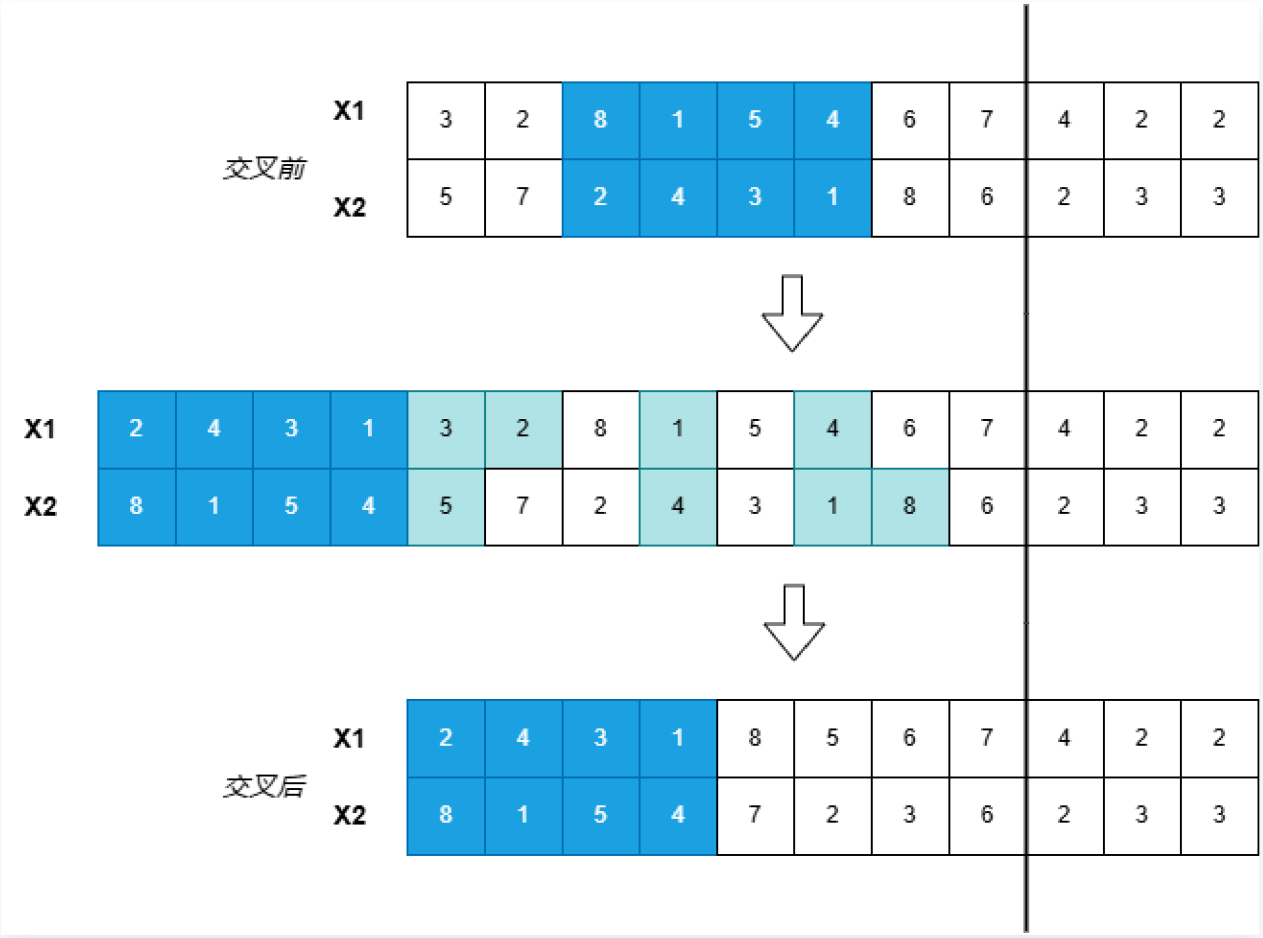

图 2.4.1 交叉操作示意图

5.局部搜索操作

假设灰狼更新位置后的种群为Population，那么将Population中的灰狼α，β，σ作为局部搜索操作的对象，即对这3个个体进行局部搜索操作以获得更优的灰狼个体，从而在整体上带动种群向更优的方向更新。

局部搜索操作采用了破坏和修复的思想，即首先使用破坏算子从当前解中移除若干个城市，然后使用修复算子将被破坏的城市重新插回到破坏的解中，目的是扩大搜索范围，提高得到最优解的概率。

破坏

假设城市数目为n，旅行商数目为m，则破坏算子从当前解的k（$2 \leq k \leq m$）条相邻路径中各移除l（$1 \leq l \leq L_{max}$）个城市。其中L_{max}的计算公式如下：

$$L_{\max} = \min\left(\left|\bar{t}_1\right|, |t_2|\right) \tag{2.5.1}$$

其中，$\left|\bar{t}_1\right|$为当前解中每条路径上城市数目的平均值，$|t_2|$为当前路径城市数目。

修复

首先介绍插入成本的概念。如果当前破坏后的解为S^*，那么在将R中的一个城市插回到S^*中的某个位置以后，此时修复后的解中将城市插回的路线距离减去S^*的各个行走距离中的最大值即为将该城市插入该位置的插入成本。

然后是遗憾值。如果R中的一个城市插回到S^*中的插回位置的总数目为lr，那么这lr个插入位置对应lr个“插入成本”。接下来将lr个“插入成本”按从小到大进行排序，若排序后的“插入成本”为up_delta，则将该城市插回到S^*中的“遗憾值”即为排序后排在第二位的“插入成本”减去排在第一位的“插入成本”，即$up_delta(2) - up_delta(1)$。

下面是修复算子的具体步骤：

- STEP1：初始化修复后的解S_{repair}，即$S_{repair} = S^*$。
- STEP2：如果R非空，转向STEP3，否则转向STEP6。
- STEP3：计算当前R中的城市数目nr，计算将R中各个城市插回到S_{repair}的遗憾值$regret$，即$regret$是nr行一列的矩阵。
- STEP4：找出$regret$中最大“遗憾值”对应的序号max_index，确定出即将被插回的城市$rc = R(max_index)$，将rc插回到S_{repair}中“插入成本”最小的位置。
- STEP5：更新$R(max_index) = []$，转回STEP2。
- STEP6：修复结束，输出修复后的解S_{repair}。

三、案例引入

1.问题简介

我们有51个城市，旅行商数目为2，从起始城市1出发（同时也是目标城市），要求最小化这两个旅行商行走距离的最大值。

城市的数据格式如下：

表 3.1.1 部分城市坐标

序号	x坐标	y坐标
1	37	52
2	49	49
3	52	64

同时，我们需要对GWO的参数进行设置：

表 3.1.2 实验参数设置

参数	取值
灰狼种群数目	50
最大迭代次数	1000
移除相邻路径的数目	2

2.实验结果

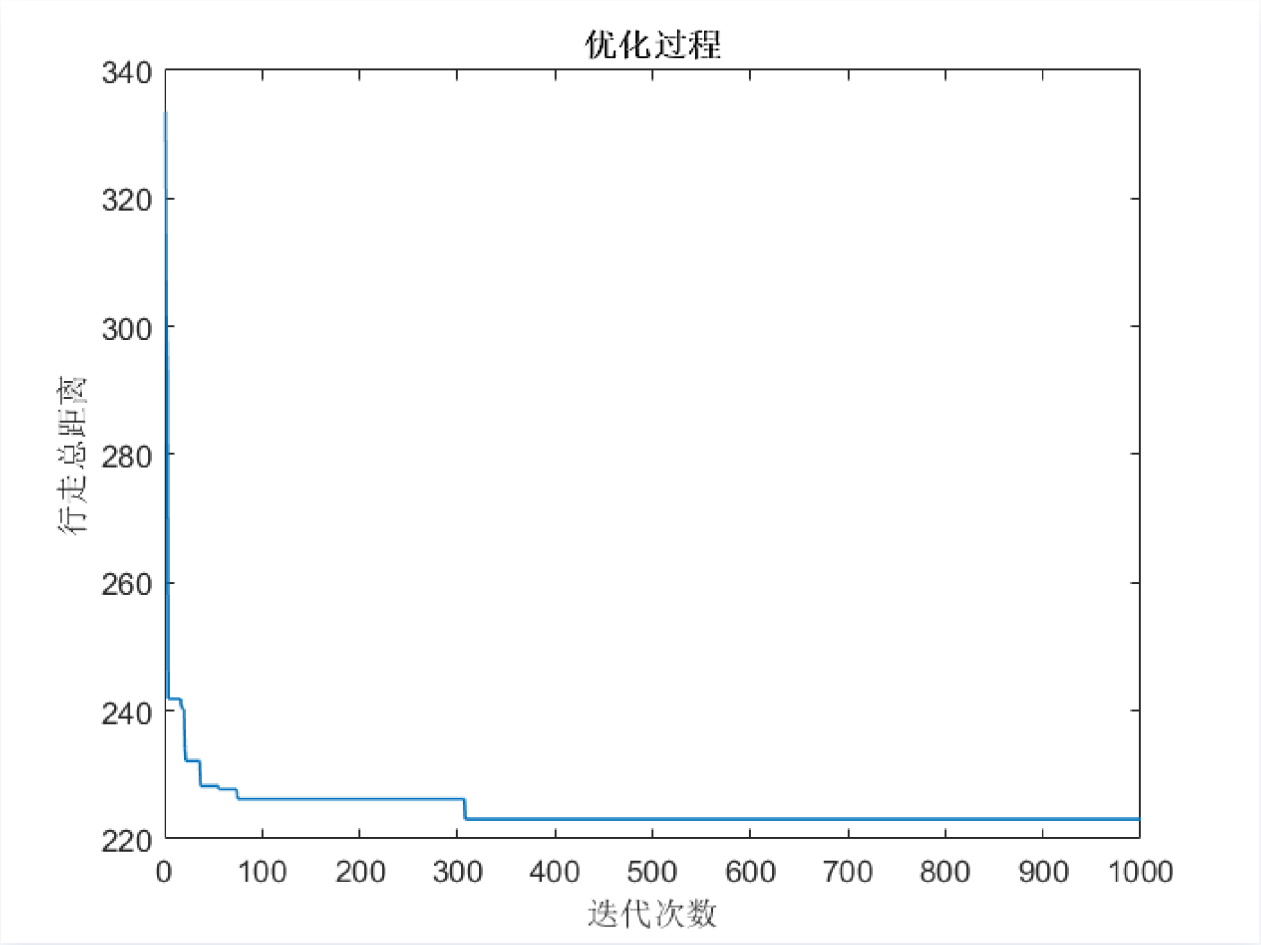

图 3.2.1 优化过程

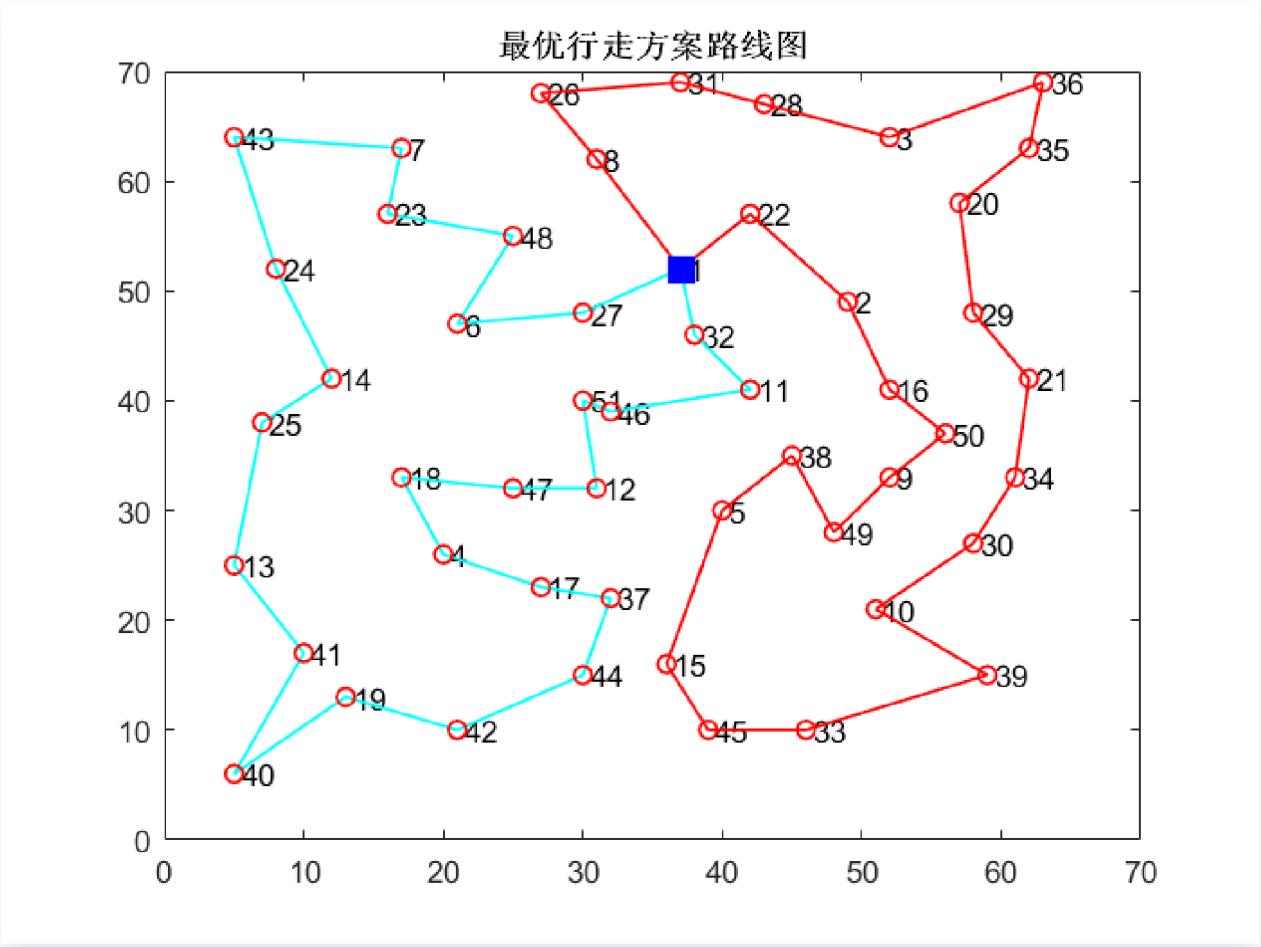

图 3.2.2 最优路线

四、讨论

1.讨论场景

2.讨论问题

- 灰狼优化算法有哪些优缺点?

优点:

1. **结构简单**：算法结构简单，容易理解和实现。
2. **参数少**：需要调节的参数较少，便于操作。
3. **自适应调整**：具有自适应调整的收敛因子和信息反馈机制，能够在局部寻优与全局搜索之间实现平衡。
4. **性能良好**：在求解精度和收敛速度方面都有良好的性能。

缺点:

1. **收敛速度慢**：在面对复杂的高维优化问题时，由于搜索空间巨大，可能需要较长时间才能找到较优解。

2. **全局搜索能力弱**：对于全局最优解的搜索能力相对较弱，容易陷入局部最优解。
3. **参数选择难度大**：存在需要手动调整的参数，如跟随系数和逃避系数等，参数设置不合理可能导致算法性能下降。

- 灰狼优化算法可以从哪些方面改进?

1. **调整控制参数**：通过调整算法中的控制参数，如非线性参数控制策略，来优化算法的性能。
2. **引入新的搜索策略**：例如基于Aquila探索方法的灰狼优化算法（AGWO），引入莱维飞行策略等新的搜索策略来提升算法的探索能力。
3. **与其他优化算法混合**：通过与其他优化算法如遗传算法、粒子群算法等的混合，提高全局搜索能力和收敛速度。

- 多旅行商问题的传统解法有哪些?

1. 精确算法：分支定界法、动态规划、线性规划、整数规划
2. 启发式算法：最近邻算法、模拟退火算法、遗传算法
3. 元启发式算法：禁忌搜索、蚁群优化

3.分工

五、其他

1.参考资料

- 论文：Grey Wolf Optimizer - ScienceDirect
- 课程：The Grey Wolf Optimizer | Udemy
- 书籍：MATLAB智能优化算法

2.工具

- Typora：用于撰写实验报告
- Mathpix：将手写公式转化为$latex$代码，提高效率
- draw.io：绘制部分图片
- MATLAB：代码编程