$$\nabla f(x) + \sum_i \lambda_i \nabla g_i(x) + \sum_j \mu_j \nabla h_j(x)$$

其中$\lambda_i$是不等式约束$g_i(x)$的拉格朗日乘数，$\mu_j$是等式约束$h_j(x)$的拉格朗日乘数。

2. 原始可行性条件

$$g_i(x) \leq 0 \text{ 对所有的} i \text{和} h_j(x) = 0 \text{ 对所有} j$$

3. 对偶可行性条件

$$\lambda_i \geq 0 \text{ 对所有} i$$

4. 互补松弛性

$$\lambda_i g_i(x) = 0$$

上述条件在凸优化问题中，KKT 条件是充分必要条件，这意味着任何满足上述条件的解均是全局最优解。在非凸优化问题中，满足 KKT 条件的解可能仅是局部最优解或鞍点。

假设当前优化问题是：

$$\text{最小化} f(x) = x^2\text{，约束条件：} x \geq 1\text{，即} g(x) = 1 - x \leq 0$$

KKT 条件：

a. 稳定性 $2x - \lambda = 0$

*b.* 原始可行性条件 $1 - x = 0$

*c.* 对偶可行性条件 $\lambda \geq 0$

d. 互补松弛性条件 $\lambda(1 - x) = 0$

最终根据上述四个等式与不等式，可求得 $x = 1$，$\lambda = 2$。

## 四、多目标优化求解

- 无约束的单目标优化问题

  利用启发式优化算法求解无约束单目标优化问题。

- 无约束多目标优化问题

- 带约束多目标优化问题（Multi Operation Objective)问题

$$min_x F(x) = [f_1(x), f_2(x), \ldots\ldots f_K(x)]^{\mathrm{T}}$$
$$s.t.\ \ g_i(x) \geq 0, i \in [1, M]$$
$$h_j(x) = 0, i \in [1, L]$$

令$D$为上述多目标优化问题的可行域，即

$$D = \{x | g_i(x) \geq 0, , i \in [1, M], h_j(x) = 0, i \in [1, L]\}$$

求解多目标带约束的优化问题的常见方法：

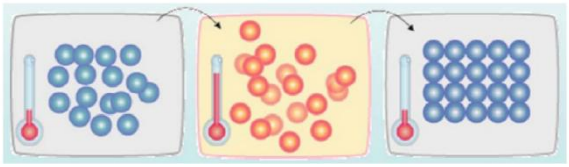
## 五、启发式算法(Heuristic Algorithms)

启发式算法定义：一个基于直观或经验构造的算法，在可接受的从成本（计算时间和空间）下给出待解决组合优化问题每一个实例的一个可行解，该可行解与最优解的偏离程度一般不能被预计。启发式优化算法相对于最优化算法提出，最优化算法为一个问题找到最优解，启发式算法是找到一个问题的可行解。

**代码链接：https://github.com/1998-Chen/Operations-Research-Optimization-Algorithm**

### ➢ 模拟退火算法（Simulated Annealing，SA）

概念与定义：模拟固体退火原理，将固体加热至充分高，再慢慢冷却。加热时，固体的粒子随着温度上升是无序的，加温时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。目标函数视为内能$E(x)$函数，寻找使得目标函数达到最小的解。固体在某温度$T$的内能就是当前解实现的目标函数取值。



图：模拟退火算法

Metropolis 准则

$$P(\Delta E)=\begin{cases}1, & if\ \Delta E<0\\ e^{-\frac{\Delta E}{T}}, & if\ \Delta E\geq 0\end{cases}$$

其中，$\Delta E=E(x_{t+1})-E(x_t)$。Metropolis 准则促进跳出局部最优解的关键设计。当出现更优解时，百分之百接受作为新解；当不是更优解时，算法允许接受一些坏解，并且这种接受坏解的概率$e^{-\frac{\Delta E}{T}}\in[0,1]$随着温度的下降不断减少，可以理解为以动态的概率跳出局部最优解，随着温度下降不断稳定。SA 区别与 PSO、GA 等算法，首先 SA 不是群优化算法（不具有初始化种群操作）

最模拟退火算法伪代码参考如下：

| 模拟退火算法伪代码 |
|---|
| 1. 初始化温度$T_t$，随机产生初始解$x_t$，温度下降系数$k\in[0,1]$，终止温度系数$\overline{T}$，系数 $L$，计算$E(x_t)$，局部最优解存储器 local_x，全局最优解存储器 global_x |
| 2. while $T_t\geq\overline{T}$: |
| 3. for $l$ in $L$: |
| 4. $x_{t+1}=x_t+\varepsilon$ 对当前可行解$x_t$进行扰动得到新解并计算$E(x_{t+1})$ |
| 5. 计算$\Delta E=E(x_{t+1})-E(x_t)$ |
| 6. 根据 Metropolis 准则，判断是否接受$x_{t+1}$作为当前可行解 |
| 7. 存储每个可行解于 local_x 中 |
| 8. 基于 argmin 函数找出当前循环 $l$ 最优解 |
| 9. $T_t$=$T_{t-1}*k$ |
| 10. 寻找 local_x 中的全局最优解 |

## ➢ 粒子群算法(Particle Swarm Optimization, PSO)

粒子群算法是基于社会信息共享进行最优化搜索算法。每个粒子代表着问题的解。假设在 D 维搜索空间，存在 N 个粒子，每个粒子（代表一个向量）代表问题的解。算法主要参数：

(1) 第$i$个粒子的位置：$X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$

(2) 第$i$个粒子的速度为：$V_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,D})$

(3) 第$i$个粒子搜索到的最优位置（个体最优解）：$P_{i,pbest} = (p_{i,1}, p_{i,2}, \ldots, p_{i,D})$

(4) 群体搜索到最优位置（群体最优解）：$P_{d,gbest} = (p_{1,gbest}, p_{i,gbest}, \ldots, p_{D,gbest})$

(5) 第$i$个粒子搜索到的最优位置的适应值（优化函数的目标值）：$f_p$

(6) 群体搜索到的最优位置的适应值：$f_g$

| PSO 算法参数说明 |
| --- |
| $N$是粒子群规模 |
| $k$是迭代次数 |
| $i$是粒子序号，$i = 1, \ldots\ldots, N$ |
| $D$是粒子维度（解的维度） |
| $d$是粒子维度序号 |
| $k$是迭代次数 |
| $w$是权重 |
| $c_1$属于个体学习因子 |
| $c_2$属于群体学习因子 |
| $r_1$、$r_2$是属于区间[0,1]内的随机数，增加模型的探索能力 |
| $v_{id}^k$是粒子在第$k$次迭代中第$d$维的速度向量 |
| $x_{id}^k$是粒子在第$k$次迭代中第$d$维的位置向量 |
| $p_{id,pbest}^k$是粒子（个体）在$k$次迭代中第$d$维的历史最佳位置，表示在$k$次迭代后，粒子个体最优解 |
| $p_{d,dbest}^k$是群体第$k$次迭代中第$d$维的历史最优位置，表示在$k$次迭代后，粒子群中的最优解 |

速度更新公式：

$$v_{id}^{k+1} = wv_{id}^k + c_1 r_1 (p_{id,pbest}^k - x_{id}^k) + c_2 r_2 (p_{d,dbest}^k - x_{id}^k)$$

其中，$wv_{id}^k$属于惯性部分（由惯性权重和粒子自身速度构成），表明对本身运动状态的信任；$c_1 r_1 (p_{id,pbest}^k - x_{id}^k)$是认知部分，属于粒子与自身历史最优位置之间的距离和方向；$c_2 r_2 (p_{d,dbest}^k - x_{id}^k)$属于社会部分，表示粒子间信息共享和合作，该粒子当前位置与群体历史最有位置的距离和方向。$c_1$属于个体学习因子，$c_2$属于群体学习因子。

- 改善性权重：$w = w_{max} - (w_{max} - w_{min}\frac{k}{k_{max}})$

速度方向：第$k+1$次迭代的方向由：惯性方向+个体方向+群体最优方向

位置更新公式：

$$x_{id}^{k+1} = x_{id}^k + v_{id}^{k+1}$$

粒子群算法伪代码参考如下

粒子群算法伪代码

1. 初始化粒子群个数 $N$，最大迭代次数$M$，惯性权重 $w$，个体学习因子$c_1$，群体学习因子$c_2$，维度$D$，随机因子
2. 初始化粒子群，位置矩阵$X$，速度矩阵$V$

   for $\boldsymbol{i}$ in range($N$):

   for $\boldsymbol{j}$ in range($D$)：

   随机初始化位置$x_{ij}$

   随机初始化速度$v_{ij}$

   设置个体最优位置$\boldsymbol{p_i}$=$\boldsymbol{x_i}$

   初始全局最优位置$\boldsymbol{p_{gbest}}$
3. 进行迭代

   for $\boldsymbol{k}$ in range($M$):

   对于每个粒子 $\boldsymbol{k}$，计算适应值$f(\boldsymbol{x_k})$

   如果$f(\boldsymbol{x_k})$优于$f(\boldsymbol{p_k})$:

   更新个体最优位置$\boldsymbol{x_k} = \boldsymbol{p}_k$

   如果$f(\boldsymbol{p_k})$优于$f(\boldsymbol{p_{gbest}})$:

   更新全局最优位置$\boldsymbol{p_{gbest}}$=$\boldsymbol{x_k}$

   对于每个粒子 $\boldsymbol{k}$

   更新速度$\boldsymbol{v_k}$， $\boldsymbol{v_k} = w \times \boldsymbol{v_k} + c_1 \times r_1 \times (\boldsymbol{p}_k - \boldsymbol{x_k}) + c_2 \times r_2 (\boldsymbol{p_{gbest}} - \boldsymbol{x_k})$

   更新位置$\boldsymbol{x_k}$， $\boldsymbol{x_k} = \boldsymbol{x_k} + \boldsymbol{v_k}$

输出全局最优位置$\boldsymbol{p_{gbest}}$及其适应值$f(\boldsymbol{p_{gbest}})$

## ➢ 遗传算法（Genetic Algorithm, GA)

遗传算法是一类进化算法，它通过模拟自然界的进化过程（如遗传、变异、选择和交叉）来寻找问题的最优解，旨在通过模拟自然选择过程来解决复杂的优化和搜索问题。

**基本原理**

遗传算法的基本原理基于自然选择和遗传学，包括以下几个步骤：

a. 编码（Encoding）：将问题的解表示为染色体（通常为二进制字符串）。

b. 初始种群（Initial Population）：随机生成一组染色体作为初始种群。

c. 适应度函数（Fitness Function）：评估每个染色体的适应度，适应度函数根据问题的具体要求定义，用于衡量染色体的优劣。

d. 选择（Selection）：根据适应度值选择染色体进行繁殖。常用的选择方法有轮盘赌选择、锦标赛选择等。

e. 交叉（Crossover）：选择两个染色体并交换部分基因片段，生成新的后代。交叉操作模拟自然界的基因重组过程。

f. 变异（Mutation）：随机改变染色体中的某些基因值，增加种群的多样性，防止早熟收敛。

g. 替换（Replacement）：用新生成的后代替换种群中的一些个体，形成新的种群。

h. 终止条件（Termination Condition）：当达到预定的迭代次数或找到满意的解时，

背景知识

一条染色体中存在多个基因。例如 x = [0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0]。其中 0 和 1 就是染色体 x 的一个基因，将 x 表示的二进制‘01100111001010101110’转为十进制值是 422086，则 422086 是问题的一个解（单变量求解问题）。

二进制如何转化为十进制：

- 从右到左，给每个二进制位赋予权重，权重是 2 的指数，从 0 开始。
- 乘以对应的权重
- 将所有乘积相加
- 第 0 位: $0 * 2^0 = 0$
- 第 1 位: $1 * 2^1 = 2$
- 第 2 位: $1 * 2^2 = 4$

将所有的值相加：0+2+4+……

## ➢ 蚁群算法(Ant Colony Optimization, ACO)

利用自然界觅食时最短路径原理。从起点到终点存在多个路径，蚁群初始阶段随机选择路径，甚至每条路径都能走一遍。每个走过的路径均会存在信息素。在单位时间内，所有可行的路径中，最短路径留下的信息素最多（浓度更高）。为此，蚁群再次面临选择时，优先考虑信息素浓度高的路径走。

路径选择（转移）公式

$$P_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}(t)]^{\beta}}{\sum_{s\in S_k}[\tau_{is}(t)]^{\alpha}\cdot[\eta_{is}(t)]^{\beta}} & s\in S_k\\ 0 & s\notin S_k\end{cases}$$

其中，$P_{ij}^{k}(t)$表示$t$时刻，第$k$只蚂蚁，从$i$点到$j$点的概率，$S_k$是蚂蚁$k$可以访问的节点集合。$\tau_{ij}(t)$是时刻$t$时$i$点到$j$点的信息浓度，$\eta_{ij}(t)$是时刻$t$时$i$点到$j$点路径长度的倒数，$\alpha$与$\beta$分别是超参数。

信息素浓度的更新公式

$$\begin{cases}\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}\\ \Delta\tau_{ij}=\sum_{k=1}^{n}\delta\,\tau_{ij}^{k}\end{cases}$$

其中$\rho$是信息挥发因子，$1-\rho$表示信息素残留因子，$\Delta\tau_{ij}$表示信息素的更新量，$\delta\tau_{ij}^{k}$是第$k$只蚂蚁在节点$i$到节点$j$路径上信息素的更新量。

单个蚂蚁释放信息素的增量公式

$$\Delta\tau_{ij}=\begin{cases}\dfrac{Q}{L_k}, \text{第}k\text{只蚂蚁从城市}i\text{到城市}j\\ 0, \text{其他}\end{cases}$$

其中，$L_k$是蚂蚁自身完成从起点到终点走过的距离，Q 是蚂蚁$k$身上总的信息素。

总结

| 蚁群算法伪代码 |
| --- |
| 1. 初始化蚂蚁的位置 |
| 2. 蚂蚁行走，根据随机策略进行路径的选择直到走到终点。待到全部走完后，全局中是否存在蚂蚁行走的路径是否满足最优条件 |
| 3. 更新每个路径的信息素 |
| 4. 蚂蚁重新选择起点，开始新一轮探索 |

基于蚁群算法进行物流节点城市路径规划