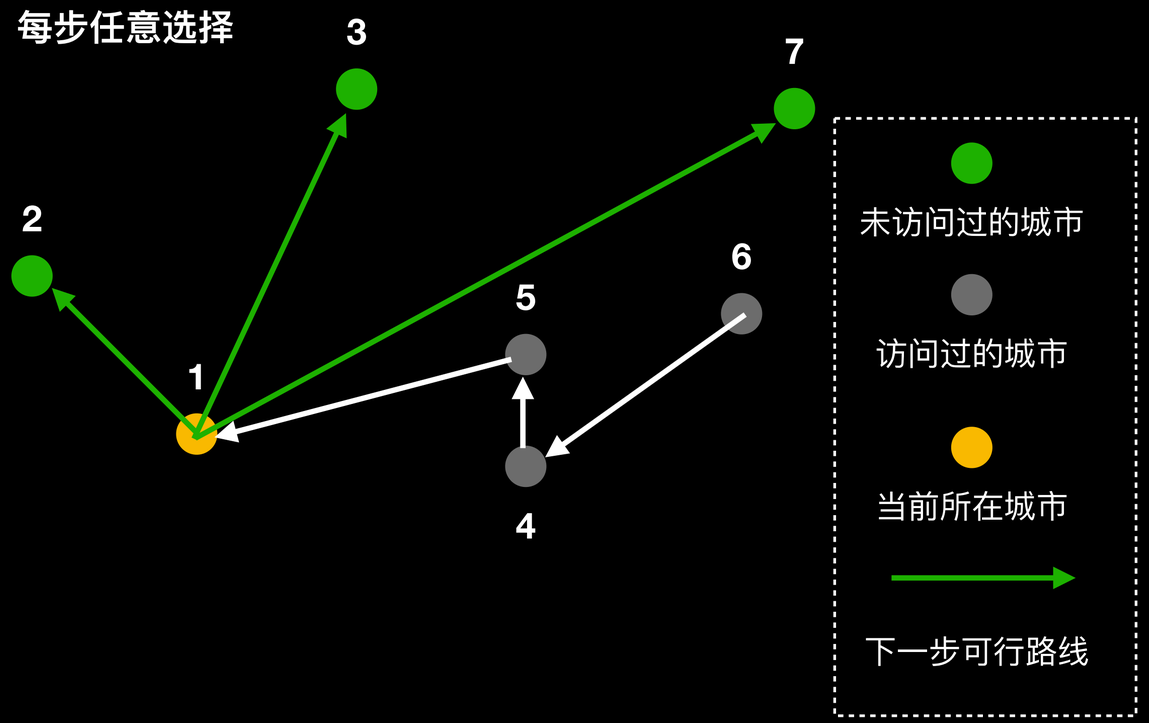
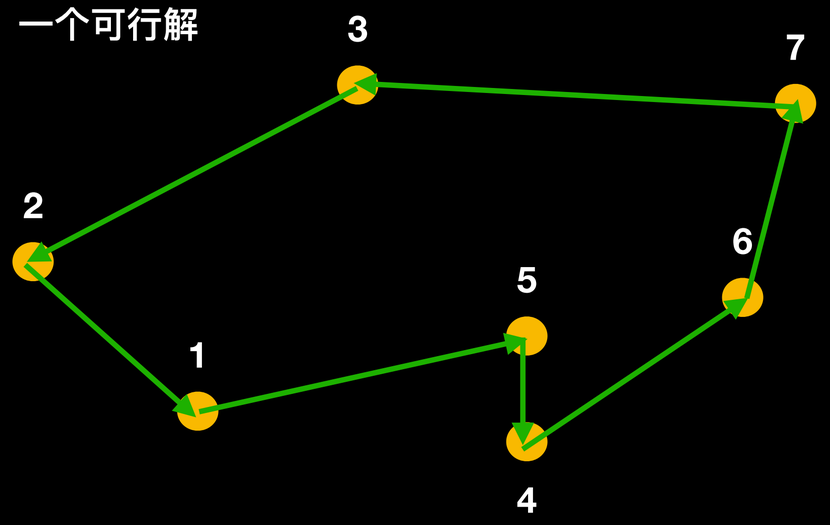
小练习：利用遗传算法(Genetic Algorithm)求解旅行商问题(Traveling Salesman Problem)

friedbee a.k.a. 极客鸭鸭

旅行商问题
Traveling Salesman Problem

- 选定任意一点作为起始点，每次可以选择任意点访问，访问所有的地点有且仅有一次，最后回到起始点
- 假设有n个城市，通过枚举法一共有 $n \cdot (n-1) \cdot (n-2) \ldots \cdot 1 = n!$中可行解 (huge searching space)
- 尝试设计一种遗传算法快速求解

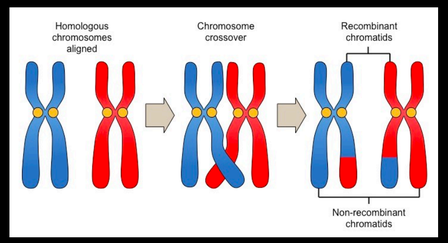


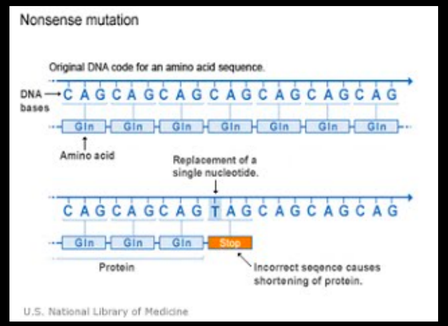


遗传算法思路

类比自然选择，“物竞天择”

- 将任意可行解编码成一段固定长度的数组(Chromosome)—染色体
- 三大步骤的类比：
  - 杂交(Crossover)—染色体片段交换
  - 突变(Mutation)—基因突变
  - 选择(Selection)—自然选择

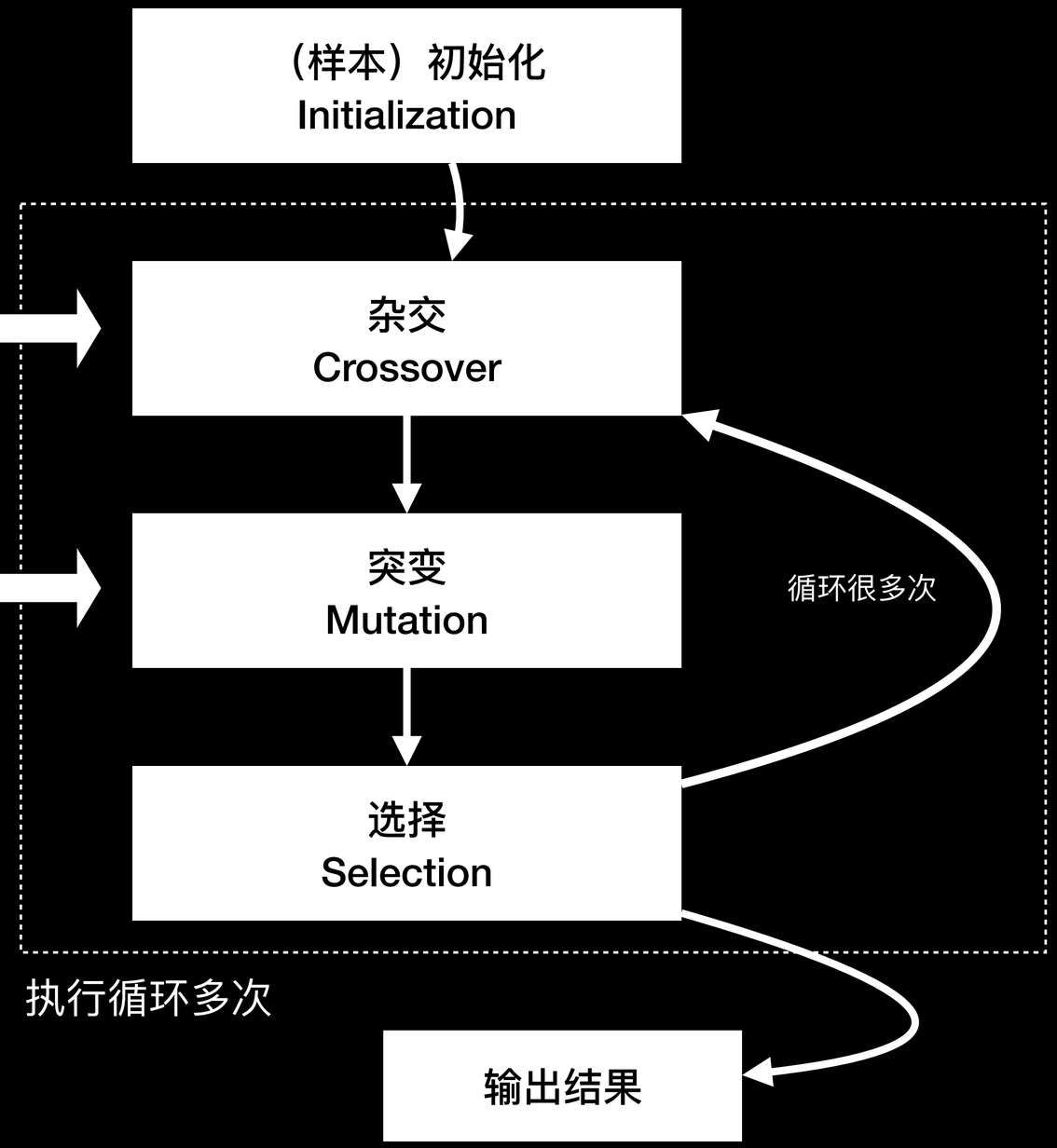




类比

类比

算法流程图



旅行商问题编码; 目标函数; 编码操作

Formulate TSP solution; Define Objective/Fitness Function; Actions

可行解编码为染色体：

- 用长度为n的数组表示，但其实是无头无尾的loop。所以，每个解有多种不同的表达
- 多种表达：[1, 4, 6, 7, 3, 2, 5] ==[3, 2, 5, 1, 4, 6, 7]

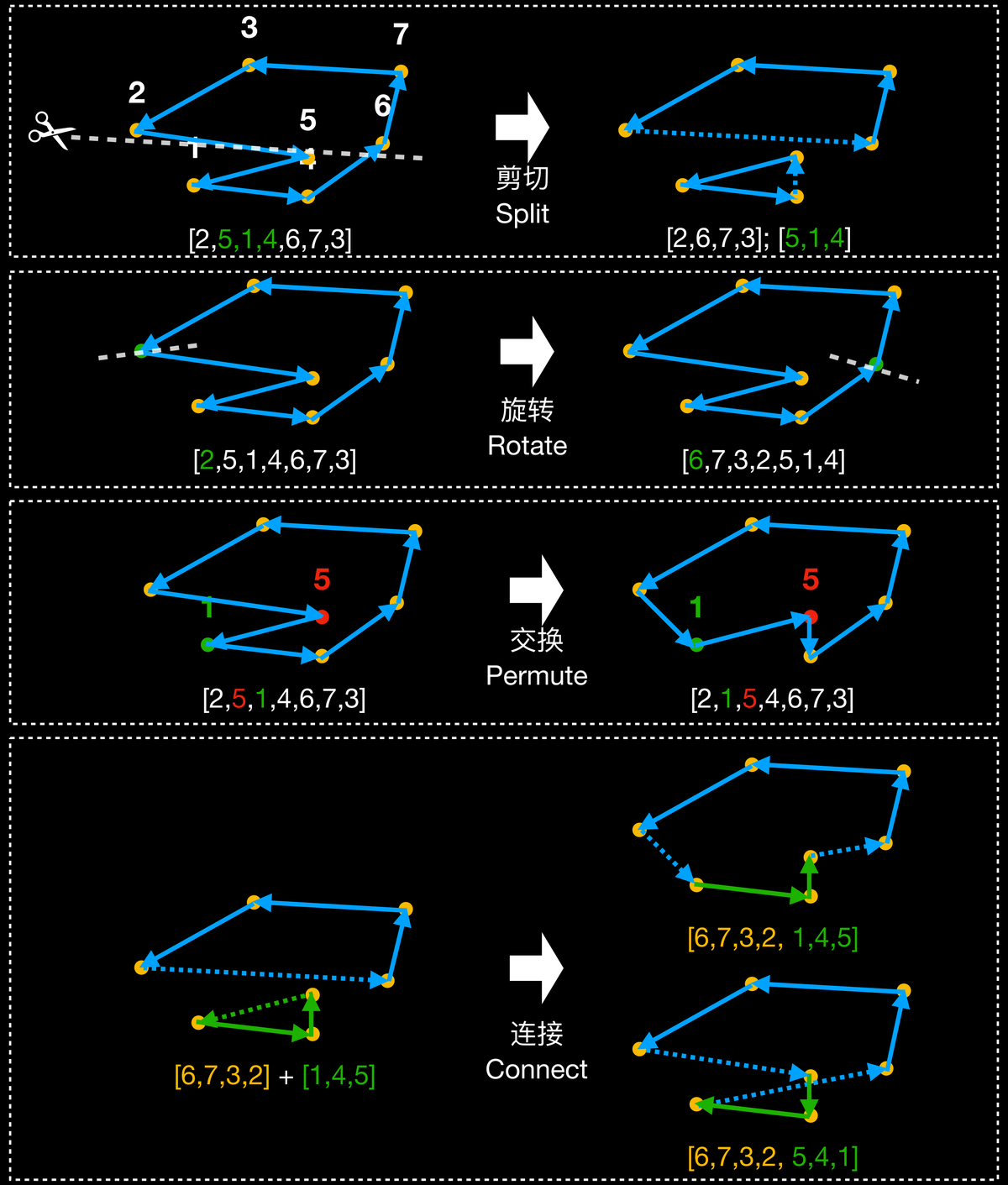
目标函数/适应性函数：总旅行距离（越短越好）

- $D = d_{[1,5]} + \ldots + d_{[3,2]} + d_{[2,1]} = \sum_{i=1}^{n} d_{i,i+1} + d_{n,1}$

定义编码动作（所有繁殖，突变用四种动作组合完成）

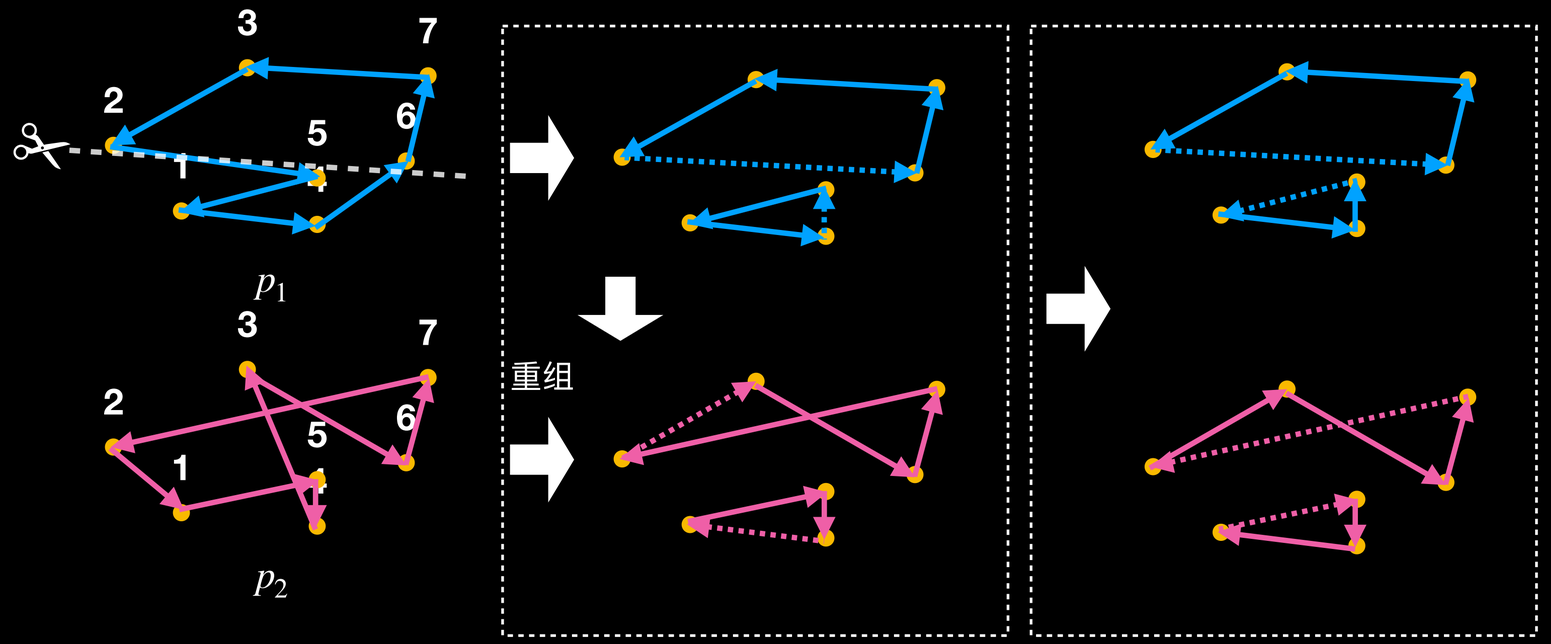
- 剪切（split）
- 旋转（rotate）
- 交换（permute）
- 连接（connect）两个方向/两种情况
- 重组（Re-construct）分解为两个集

  [1, 4, 6, 7, 3, 2, 5] ==> [6, 7, 2] + [1, 4, 3, 5]

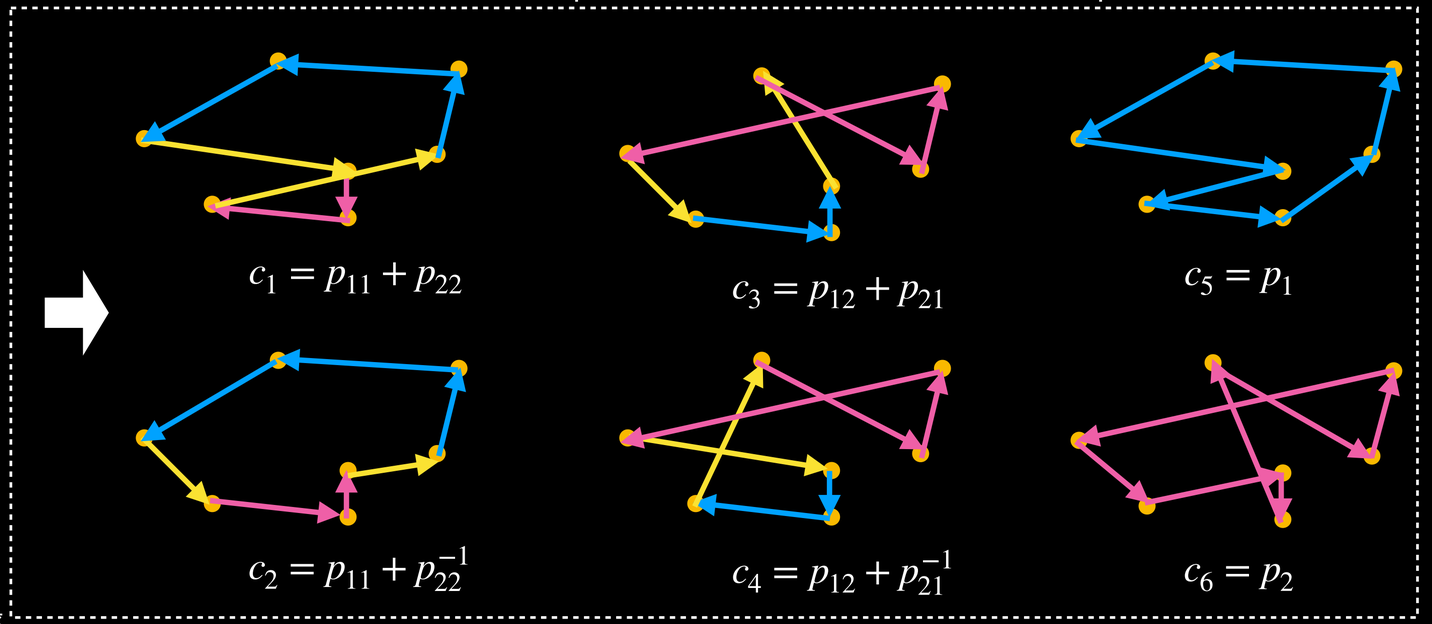


杂交
Crossover

- Step 1. 选定剪切点，每个染色体分成两小段染色体片段
  - p_1: [2,5,1,4,6,7,3]
    - p_{11}: [5,1,4]
    - p_{12}: [6,7,3,2]
  - p_2: [1,5,4,3,6,7,2]根据p_1剪切，得到两个部分分割：
    - p_{21}: [3,6,7,2] or p_{21}^{-1}: [2,7,6,3]
    - p_{22}: [1,5,4] or p_{22}^{-1}: [5,1,4]
- Step 2. 确定断点(break point)
  - 假定断点为最后一项指向第一项: $n \rightarrow 1$
  - 策略1: 随机选定断点
  - 策略2: 距离最长的点为断点
- Step 3. 四种可能交换方式（考虑染色体可以两个方向拼接）
  - $c_1 = p_{11} + p_{22}$; $c_2 = p_{11} + p_{22}^{-1}$
  - $c_2 = p_{21} + p_{22}^{-1}$; $c_1 = p_{11} + p_{22}^{-1}$
- 共四种交换方式+原来两种情况=6种解作为下一代
- 还有很多细节省略（例如：如何选定父母，如何选定切点，设置合理的突变率等等）



Step 1（剪切）　　Step 2 （旋转）



Step 3（拼接）

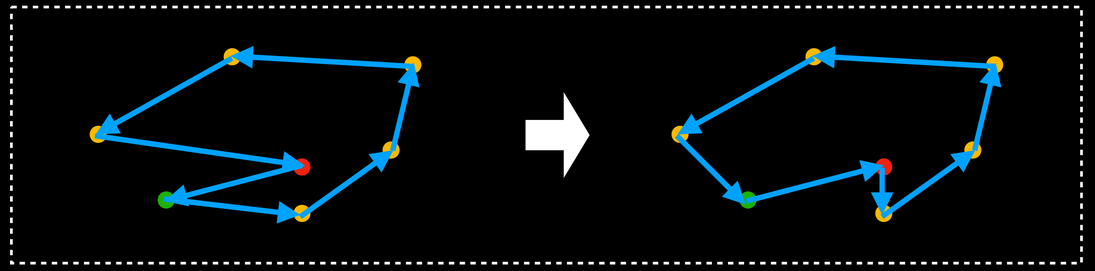
突变 与 选择
Mutation & Selection

突变(Mutation)：在数组中随机选择两个突变点位，交换两个点位

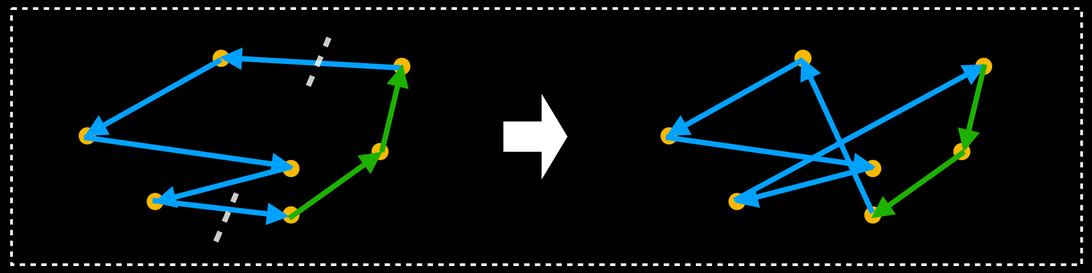
- 策略1：交换两个在数组上相邻的点位
- 策略2：交换两个距离上相邻的点位
- 策略3：自我杂交（改变拼接方向）

选择(Selection)：从这一代产生的所有样本，选择保留到下一代的样本

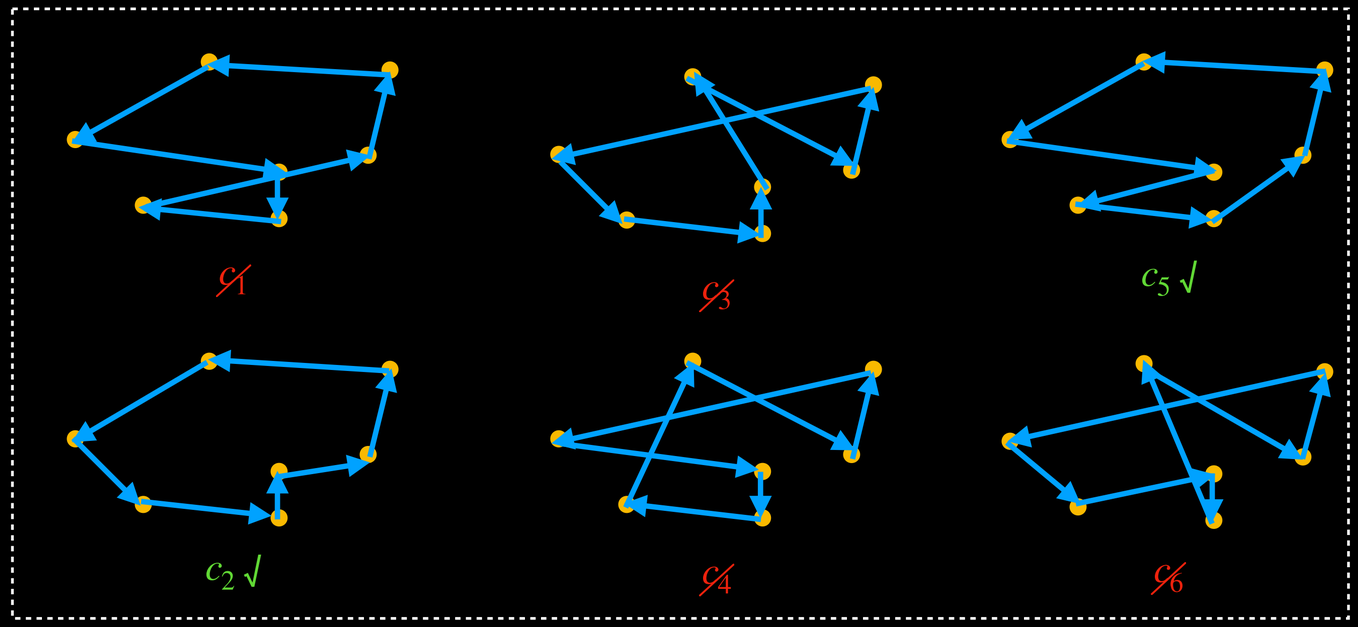
- 原则1：保留最优样本（得分高的一些样本）
- 原则2：保留多样性（足够多的不同样本）
- 限制条件：有限算力
- 综合抉择(Tradeoff)，选定选择策略



突变（策略1，2）



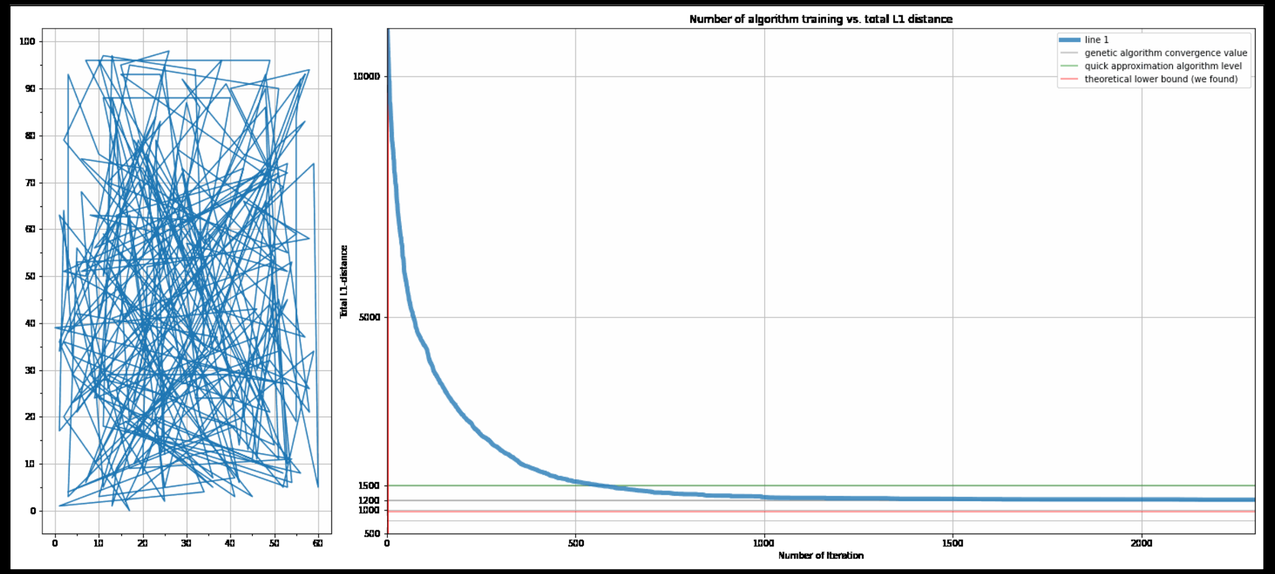
自我杂交（策略3）：剪切+交换方向



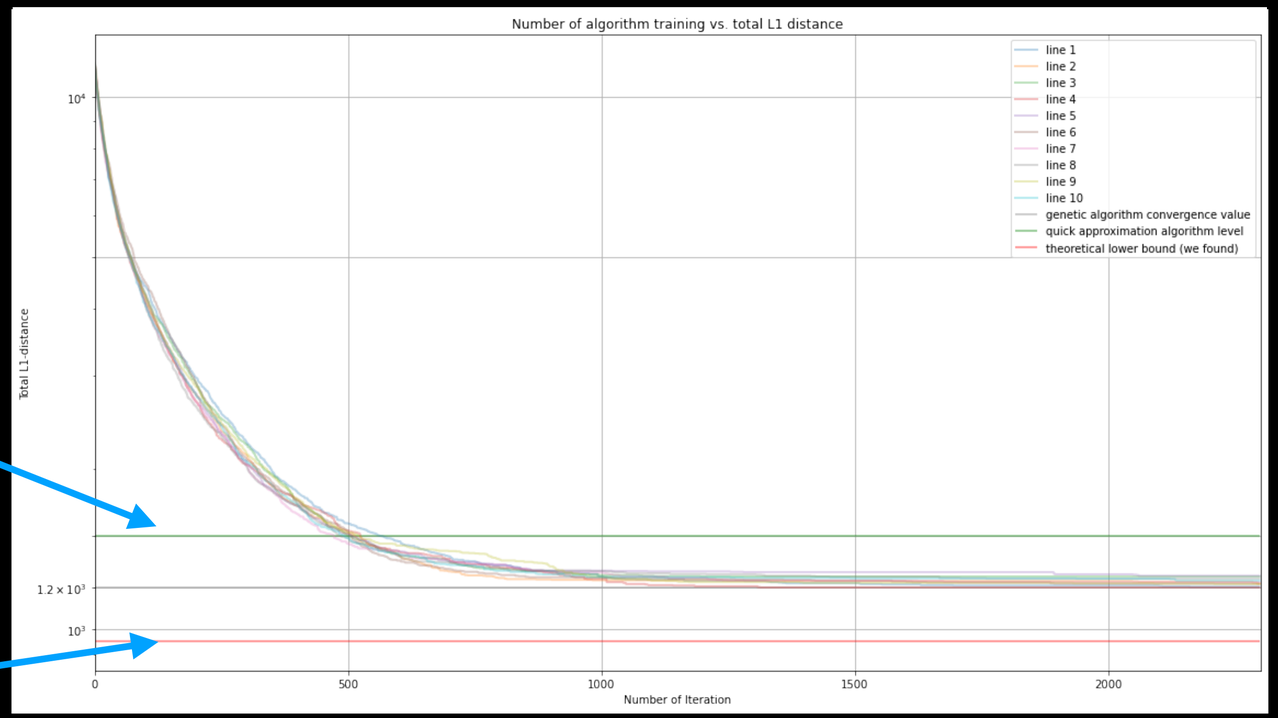
选择：淘汰旅行总距离太高的样本

实例展示
Case Study

- 260 个数据点在 60×100 的棋盘上
- 目标距离函数：（L1/ Manhattan Distance）
- 从随机数列开始



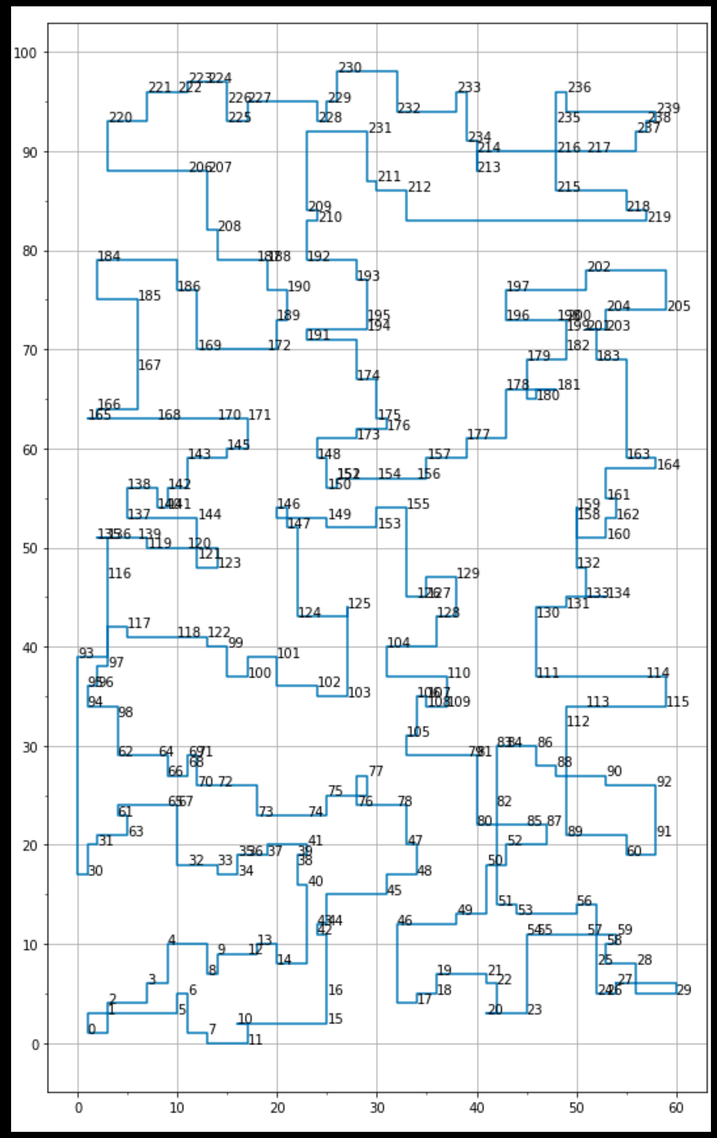
一次训练可视化结果



一个2倍以内近似解
(根据最小生成树
Minimum Spanning Tree得到)

找到的最优解的
最“高”理论下限

10次计算的得到不同的最终参数曲线

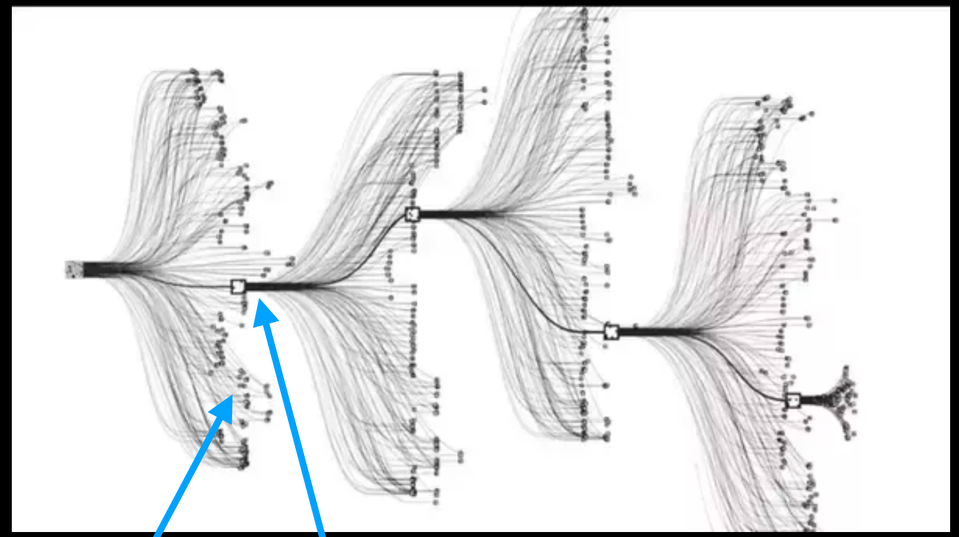


一个最终解旅行路径（总距离=1192）

实践总结
Summary

- 经验总结：
  - 一直保留群体中的最优解
  - 根据经验制定杂交与突变的方案，力求增大“有益进化”的概率
  - 通过解决小实例(toy sample)，可以帮助思考
  - 使用近似解作为算法起始解可以大大增快过程，增强结果
  - 写Code(implementation)简单，重要的是如何将可行性高的改进镶嵌如遗传算法中
  - 使用多线程（Multiprocessing）利用多个CPU可以同时得到不同的结果
  - Python中基因操作前使用copy.deepcopy防止在变量指向并改变parent的list数组内容
- 遗传算法的不足与优点：
  - 本质使用搜索算法，但是搜索方向较为随机，是靠选择过程筛选结果
  - 相比现在的ML/AI方法，没有让计算机掌握经验（改变问题还要从“零”开始）
  - 不好的设计容易卡在局部解(Converge to local optimal solution)
  - 好的算法很依赖编程者正确的理解问题，否则效率会很低
  - 可能主要的好处是通用性，几乎所有问题的解都可以编成固定格式的“染色体”编码

类比：棋类算法中的搜索树/决策树（*不严谨，选择空间不同）



选择 vs.
双方的实际走法

繁殖/突变 vs.
棋类的下一步可能走法

视频资料 & 参考资料
Video Material & Reference

视频中用到的数据与代码资源Github分享链接🔗

- https://github.com/thefriedbee/video-source-code
- 视频中用到的最小生成树近似解：https://www.geeksforgeeks.org/travelling-salesman-problem-set-2-approximate-using-mst/

其他一些可能的算法& 相关参考：

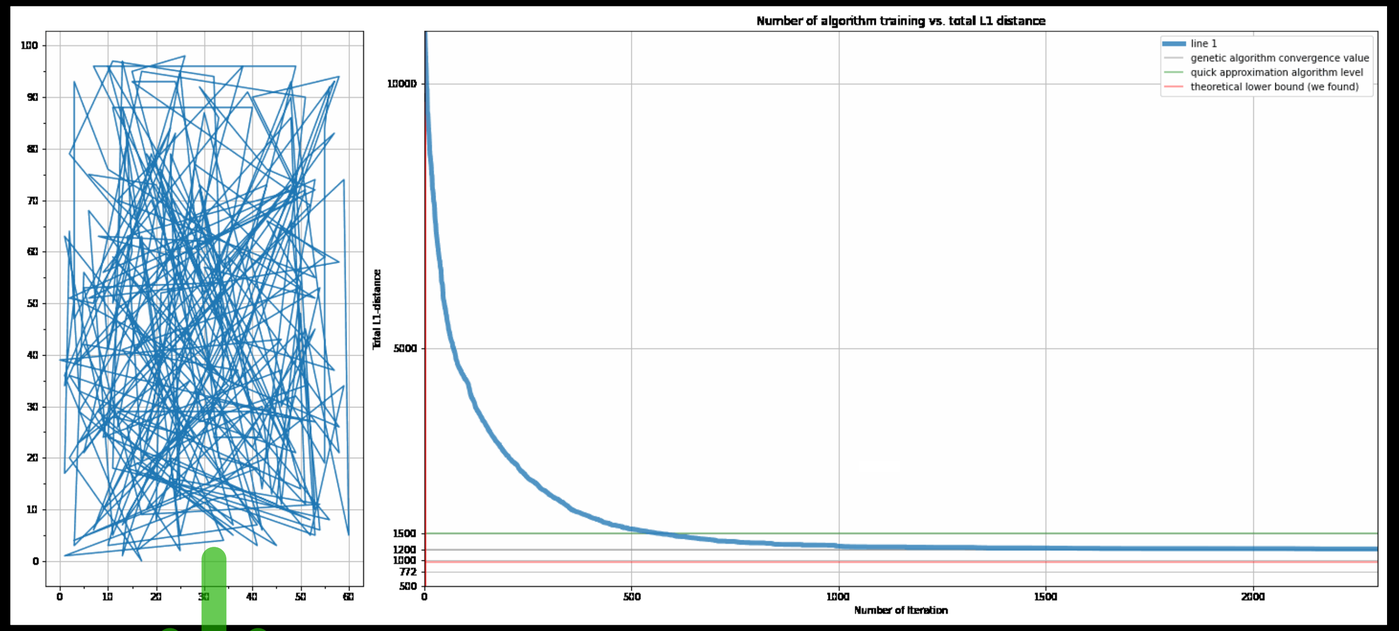
- 利用聚类（Clustering）例子：https://www.math.cmu.edu/~af1p/Teaching/OR2/Projects/P12/2010FinalProject.pdf
- 蚁群算法； 一个不错的Bili视频：https://www.bilibili.com/video/BV1vp4y1p78R
- 退火算法（Simulated Annealing）； 百科：https://en.wikipedia.org/wiki/Simulated_annealing

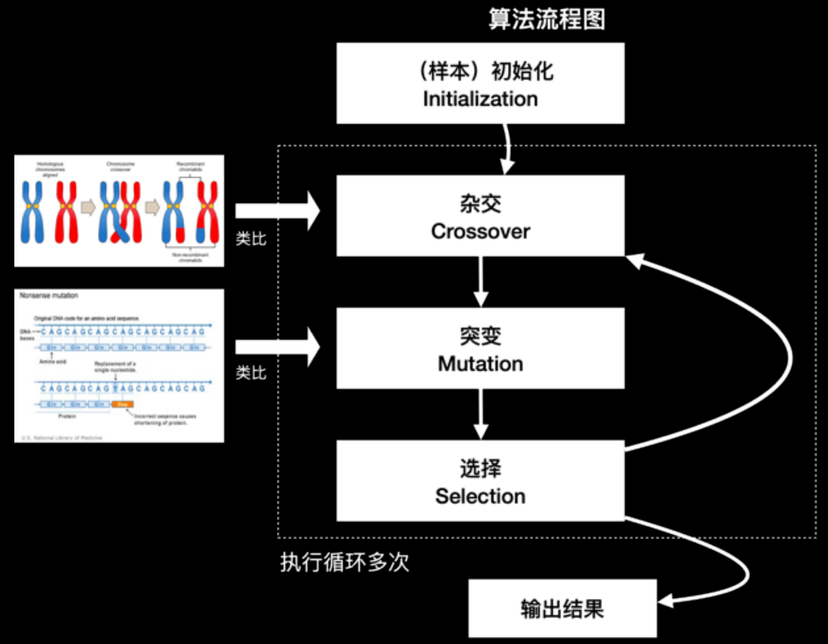
参考资料

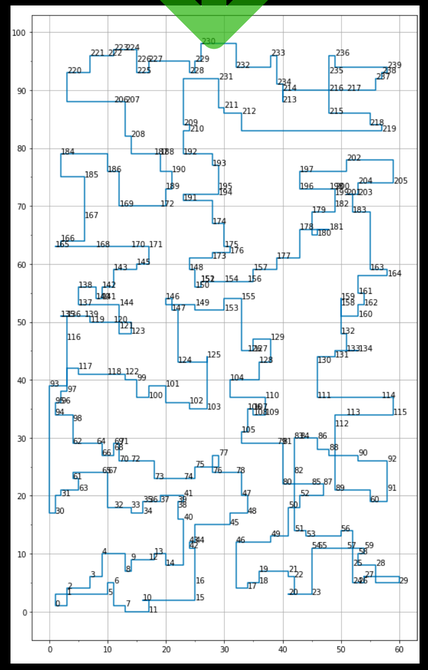
- https://en.wikipedia.org/wiki/Genetic_algorithm
- https://www.quora.com/What-does-it-mean-that-AlphaGo-relied-on-Monte-Carlo-tree-search
- https://en.wikipedia.org/wiki/Minimum_spanning_tree

插图资料

- https://ib.bioninja.com.au/standard-level/topic-3-genetics/33-meiosis/crossing-over.html
-







利用遗传算法(Genetic Algorithm)求解旅行商问题(Traveling Salesman Problem)