学号：__2016300030006__

密级：_____

武汉大学本科毕业论文

Wuhan University
旅行商问题粒子群算法设计与实现

院 (系) 名 称：　弘毅学堂

专 业 名 称：　计算机科学与技术

学 生 姓 名：　关焕康

指 导 教 师：　刘敏忠　教授

二〇二〇年五月

郑 重 声 明

本人呈交的学位论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本学位论文的知识产权归属于培养单位。

本人签名: _____ 日期: _____

摘　　要

旅行商问题有着广泛的应用，比如航班调度，印制电路板转孔等。然而，旅行商问题是经典的 NPC 问题。它的精确求解存在着计算爆炸的问题。此类问题目前更偏向于使用启发式算法进行求解。其中著名的启发式算法有蚁群算法和遗传算法等。在这篇文章中，我们将介绍自主改进的粒子群算法（下文称为“基于点交换的粒子群算法”，简称 ps-PSO）来求解旅行商问题。我们的基于点交换的粒子群算法以粒子群算法为雏形，重新定义了运算规则。使得粒子群算法更加适应求解旅行商问题。我们提出的基于点交换的粒子群算法-改进版能够快速收敛，在大规模旅行商问题上更具备优势。另一方面，为了评估基于点交换的粒子群算法，我们在四种算法上测试不同规模的 TSPLIB 数据集。四种方法包括随机，遗传算法，基于点交换的粒子群算法（ps-PSO-v1）和改进后的基于点交换的粒子群算法（ps-PSO-v2)。

关键词：粒子群算法；旅行商问题；遗传算法

ABSTRACT

Traveling salesman problem has a wide array of application, such as flight schedule, PCB(printed circuit board) etc.. And a lot of heuristic algorithm are employed to solve this problem, like ACO(Ant Conony Optimization), GA(Genetic Algorithm). In this work, we make two main contributions. First, to address traveling salesman problem, we introduce a modified PSO(Particle Swarm Optimization) algorithm named Point-Swap PSO and Point-Swap PSO-v2 (abbr. ps-PSO), in which we redefine some basic operations to adapt the classic PSO algorithm to focus on large-scale TSP. Second, we conduct experiments to evalute the proposed method, and contrast it with GA on multi-scale TSPLIB dataset.

Key words: Particle Swarm Optimization; Traveling Salesman Problem; Genetic Algorithm

目　　录

1 引言

旅行商问题有着广泛的应用，比如航班调度，印制电路板转孔等。在规模增大时，求解最优解将变得十分困难。我们希望在计算复杂度和解的优良程度上做出调整，希望可以在能忍受的时间限度内得到尽量优的解。启发式算法正好完成了这一任务。据我所了解到的，很多学者采用了各种变形的蚁群算法和遗传算法来解决旅行商问题 [1][2][3][4][5][6][7]。本文章中，我们提出以粒子群算法作为雏形，重新定义运算规则使得粒子群算法更加适应求解旅行商问题。我们提出了基于点交换的粒子群算法及其改进版，分别为 ps-PSO-v1 和 ps-PSO-v2。并使用遗传算法来做对比。通过分析不同算法在多规模的数据集上的结果，完成遗传算法和粒子群算法在旅行商问题上的探讨①。最终结果发现，ps-PSO-v2 的结果和遗传算法结果各有优劣。ps-PSO-v1 结果差于遗传算法和 ps-PSO-v2 但远优于随机生成的结果。相对于遗传算法，ps-PSO-v2 前期收敛很快，可以在较短时间内拿到一个较为满意的结果。这在数据规模较大时（(n>1000) 时，ps-PSO-v2 的优势会特别明显。但是，ps-PSO-v2 在进入平缓区域②后跳出局部解向全局最优解进发的动力不足③。

本文章分为八个部分。下一部分将介绍一些相关工作。第三部分详细介绍我们需要解决的旅行商问题，和使用到的数据集情况。第四部分讲述改进后的基于点交换的粒子群算法 ps-PSO-v2。第五部分是关于遗传算法解决旅行商问题的一种解法，该方法会成为我们对比的一个对象。第六部分是 ps-PSO-v1 和随机算法的设计思路。这两种方法都是作为参考对比。第七部分说明实验细节和四种不同算法在多规模的 TSPLIB 数据库上的结果。我们得出结论和讲明未来的方向在第八部分。

①所有代码和结果都会公布在 https://github.com/guanhuankang/TSP
②收敛曲线趋于平坦
③在第7部分我们将会详细说明

2 相关工作

旅行商问题最优化问题有两类，一类是连续变量问题，另一类是离散变量的问题。具有离散变量的优化问题又称为组合优化问题。旅行商问题是典型的 NP 难组合优化问题。需要从给出的解空间中寻求最有解或者较优解。原始的旅行商问题又称为静态哈密顿回路问题。在应用需要的背景下，人们又提出了新的旅行商问题，其中包括不对称旅行商问题（比如在航班调度上，在聚集度较高的城市和聚集度较低的城市之间飞行，成本和利润可能严重不对称），多人旅行商问题（即多个旅行商经过多个城市，每个城市仅仅被一个旅行商经过，求最短路径，在车辆调度路径安排问题具有重要意义）。目前，启发式算法是解决该问题最热门的手段。除了启发式算法，Hong Qu 等人结合 winner-take-all 的竞争机制设计了一个柱形竞争的神经网络求解多旅行商问题 [8]。

粒子群算法粒子群算法是群集智能算法的一种。从鸟群捕食中获取灵感，Dr.Eberhart 和 Dr.kennedy 首次提出 [9]。在该模型下，我们把鸟看作解空间的一个解（一个粒子），并赋予每个鸟有一定速度。每个鸟将根据社会信息和个体经验更新自己的速度，不断产生新的解。整个群体总体上不断向着全局最优解移动。该算法从提出起就不断被改进用于各种场景 [10][11]。

遗传算法遗传算法是模拟达尔文生物精华论的自然选择和遗传学机制的生物进化过程的计算模型。是一种模拟自然进化过程的迭代搜索最优解的方法。遗传算法定义了种群概念，种群的每个个体是解空间的一个解。引入交叉，变异等算子产生新一代种群。以目标函数计算每个个体的适应值，淘汰部分个体。遗传算法也有很多变体，比如免疫遗传算法，引入疫苗库保证种群的优良性质。

3 旅行商问题

3.1 概述

最早的旅行商问题提出是在 260 年前，第一次提出是 1759 年欧拉研究的骑士环游问题。1954 年，Geo eDanzig 等人用线性规划的方法（割平面法）取得了旅行商问题的历史性的突破——解决了美国 49 个城市的巡回问题。后来还提出了一种方法叫做分枝限界法。2010 年，英国伦敦大学皇家霍洛韦学院等机构研究人员报告说，小蜜蜂显示出了轻而易举破解这个问题的能力。这是首次发现能解决这个问题的动物，研究报告发表在《美国博物学家》杂志上[①]。

此外，旅行商问题也诞生了很多变种，如静态哈密顿回路问题，不对称 TSP，多目标 TSP。在该文章中，我们将解决静态哈密顿回路问题（最经典的模型）。简单来说，在 n 个点的完全图中寻找最短的哈密顿回路。下面给出其数学模型：

在二维平面上，n 个顶点 $v_1, v_2, v_3, ..., v_n$，寻找排列 P(排列 P 是 1,2,...,n 的一个排列), 使得

$$min \sum_{i=1}^{n} Dist(v_{p_{(i \bmod n)+1}}, v_{p_{(i+1 \bmod n)+1}}) \tag{3.1}$$

其中，$Dist(v_i, v_j) = \sqrt{(v_i.x - v_j.x)^2 + (v_i.y - v_j.y)^2}$ 和 $n >= 1$

3.2 计算复杂度

旅行商问题是 NP 难的组合优化问题。旅行商问题的描述非常简单，并且非常容易理解，但最优化求解非常困难，究其原因是因为求解最优解的复杂度很大。现有的计算机需要求解 TSP 的最优解时，运行时间和存储空间将随着问题规模剧烈增长。对于规模为 n 的 TSP 问题，表 3.1 将给出求解最优解的时空复杂度。

表 3.1 求解规模为 n 的 TSP 问题，最优化方法的时空复杂度比较

方法	时间复杂度	空间复杂度
穷举排列	约 O(n!)	约 O(n)
状压 DP [②]	约 O(2^n)	约 O(2^n)

①本段参考百度百科-旅行商问题 https://baike.baidu.com/item/旅行商问题/7737042?fr=aladdin
②状态压缩的动态规划

可以发现，复杂度随着 n 的增加，计算将变得十分困难。假设现有一个 PC 机器，每秒运算 10^{12} 次基本运算，用于求解 n=280 的 TSP 问题。穷举排列的方法将需要大于 10^{500} 年。状压 DP 需要超过 10^{60} 年。

所以，在问题规模增大时，求解旅行商问题使用这些最优化算法是不可行的。另一方面，启发式算法可以在较短的时间内提供一个较优解。仿生类的算法中，比如蚁群算法，遗传算法，粒子群算法，他们都采用迭代的方法不断寻找最优解。复杂度一般是多项式乘以迭代次数。在 n=280 时，我们采用的启发式算法都能在几十秒内收敛到一个比较满意的解。

3.3 数据集

我们采用 TSPLIB③数据集来测试我们的算法。TSPLIB 是一个公开的旅行商问题的数据集，由 Bob Bixby 和 Gerd Reinelt 提供。TSPLIB 数据集的数据大多与实际应用相关。比如 att48 数据集是美国 48 个州的首都城市。同时该数据集的规模覆盖从几十个城市到上千个城市。在小规模的数据集上，TSPLIB 还提供最优解答。

我们选取了 9 个不同规模的数据集，如表 3.2 所示（我们的实验代码和使用的数据集都会进行公开）：

表 3.2　选取的 TSPLIB 数据集

数据集名称	规模大小	提供最优解
att48	48	Yes
gr96	96	Yes
eil101	101	Yes
ch130	130	Yes
gr202	202	Yes
a280	280	Yes
d657	657	No
dsj1000	1000	No
fl1400	1400	No

③http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

4 基于点交换的粒子群算法-改进版 (ps-PSO-v2)

4.1 设计思路

基于点交换的粒子群算法以粒子群算法为雏形，重新定义运算规则，使得基于点交换的粒子群算法适应于解决旅行商问题。我们参考的粒子群算法雏形如下：

1. 设定参数：种群规模，社会学习因子，个体学子因子等
2. 随机化第一代种群
3. 粒子迭代更新：

   $v^{t+1} = w * v^t + c_1 * \alpha_1 * (pbest - x^t) + c_2 * \alpha_2 * (gbest - x^t)$

   $x^{t+1} = x^t + v^{t+1} * dt$
4. 若满足迭代停止条件退出，否则转 3

* w 是惯性系数，c_1, c_2 分别是个体学习因子和社会学习因子，α_1, α_2 是随机因子，pbest,gbest 分别是个体记忆中的最优解和种群中发现的最优解，v^T, x^T 分别代表第 T 时刻，个体的速度和位置。

可以发现，某个个体的迭代更新是依据个体经验和社会信息来完成的。个体学习因子和社会学习因子表征了个体对个体经验和社会信息的重视程度。随机因子的引入可以增加粒子的活跃能力。惯性系数越大，粒子的原有速度分量越大，扩大粒子的搜索空间①。

明显，上面的思路很难直接应用在旅行商问题上。因为旅行商问题是一个组合优化问题。我们不知道速度怎么定义，位置代表什么。我们需要做出一些修改。但是我们希望能够保留粒子群算法的思路-个体的更新依赖个体经验和社会信息。为了说明我们的方法，[4.2] 会给出粒子的定义，[4.3] 引入运算定义，[4.4] 给出我们的迭代公式。在 [4.5] 说明实验中的参数设置和一些实验细节。

①惯性越大，粒子更趋向于直线运动，能够走得更远，探索更大的解空间，能一定程度上帮助种群跳出局部最优解

4.2 粒子定义

旅行商问题的数学模型 (3.1) 中，解空间是所有的 1-n 的排列，解空间大小是 n!。根据一般规则，我们定义粒子为一个解，如公式 (4.1) 定义：

$$Particle_n = a_1 a_2 a_3 ... a_n \tag{4.1}$$

其中，$1 <= a_i <= n \, and \, a_i \in Z^+$ 且 $a_i == a_j \, iff. \, i == j$。

可见粒子的定义中，每个粒子都是一个长度为 n 的数字序列（(1,2,...,n 的排列)。给出粒子 $A_n = a_1 a_2 ... a_n$，唯一对应解空间的一个解，表示从城市 a_1 出发依次经过 $a_2, a_3, ..., a_{n-1}$ 到达 a_n，最后从 a_n 回到 a_1 形成一个哈密顿回路。

4.3 运算定义

章节 [4.1] 给出粒子群算法雏形，涉及数乘，加减法等。同样我们的迭代公式中也有对应的运算操作。但是参与运算的是粒子（4.1）无法直接应用我们熟悉的加减乘除等运算。为了便于说明我们提出的公式中的运算规则，我们给出运算的定义②。

4.3.1 定义运算元素集合

参与运算的元素集合 S，定义如 (4.2)

$$Set \, S = \{a_1 a_2 ... a_n \,|\, a_i \in Z^+, i \in set(1, 2, .., n), n >= 0\}. \tag{4.2}$$

为了后面叙述方便，如果 $A \in S, and A = a_1 a_2 ... a_n$，那么 $A_i = a_i$，A 的长度记为 A.length = n。

4.3.2 定义拼接

$$If \, A, B \in S, and A = a_1 a_2 ... a_n, B = b_1 b_2 ... b_m \, then,$$

$$< A, B >= a_1 a_2 ... a_n b_1 b_2 ... b_m = AB \tag{4.3}$$

这里定义了两个数字序列的拼接记为 AB 或者 <A,B>。

②我们使用的运算有减法，加法，除法，数乘，点乘，拼接

4.3.3 定义减法

$$If\ A,B,C \in S\ and\ A_i = a_i (A = a_1 a_2 ... a_n)\ then$$

$$C = A - B = A_{c_1} A_{c_2} ... A_{c_k} \tag{4.4}$$

其中，$A_{c_j} \notin B\ and\ c_j < c_{j+1}, i \in set(1,2,...,n), j \in set(1,2,...,k)$ 并且 k 应该尽量大。也就是说，数字序列减法 C=A-B，C 等于 A 的最长不包含 B 元素的有序子序列。

4.3.4 定义除法

$$If A,B \in S, Q = \frac{A}{B},\ then\ Q \in S\ and$$

$$Q = \frac{A}{B} =< A-(A-B), A-B, B-A > \tag{4.5}$$

直观来说，Q=A/B，Q 等于 A 和 B 的最长公共序列 (A 的顺序) 拼接上 A 的剩余部分，再拼接上 B 的剩余部分。

4.3.5 定义加法

$$If\ A,B \in S,\ X = A + B\ then$$

$$X = A + B =< A - B, B > \tag{4.6}$$

直观来说，X=A+B，结果 X 会保证 B 序列在结果的后面完整保留，前面是去除了属于 B 元素的 A 序列。

4.3.6 定义数乘

$$If\ k \in N\ and\ k <= A.length,\ A \in S\ and\ B = A * k\ then B \in S\ and$$

$$B = A * k =< A_{i+1}, A_{i+2}, ..., A_{i+k} > \tag{4.7}$$

其中，i 从 set(0,1,...,a-k) 随机抽取一个。简而言之，从 A 序列中随机抽出长度为 k 的字串。所以数乘要求 k 小于等于 A 的长度。

4.3.7 定义点乘

$$If\ k \in N\ and\ k <= A.length, A \in S\ and\ B = A \cdot \frac{k}{A.length}\ then B \in S\ and$$

$$B = A \cdot \frac{k}{A.length} =< A_1, A_2, ..., A_k > \tag{4.8}$$

简而言之，$B = A \cdot \frac{k}{A.length}$ =A 序列的长度为 k 的前缀。所以要求 k 小于等于 A 的长度。

4.4 基于点交换的粒子群算法迭代公式

章节 [4.2][4.3] 分别给出了粒子的定义和相关运算定义，接下来将给出迭代公式：

$$x^{t+1} = x^t + \frac{pbest * (c_1 * \alpha_1)}{gbest * (c_2 * \alpha_2)} \cdot (1 - \beta) \tag{4.9}$$

其中，x^{t+1}, x^t 分别是 t+1 代和 t 代的粒子 x。pbest 是粒子 x 历史最好解，gbest 是种群已发现最优的解。c_1 是对个体经验的依赖程度 (个体学习因子，c_1 是整数，范围是 [0,pbest.length③])。c_2 是对社会信息的信任程度 (社会学习因子，c_2 是整数，范围是 [0,gbest.length④])。α_1, α_2 是随机因子，是 [0,1] 之间的浮点数。β 是惯性因子，惯性因子越大，保留原序列长度越长。

公式 (4.9) 给出了种群中每个粒子的更新方法。x^t 是第 t 代的序列，x^{t+1} 是更新后的序列。可以看出，公式 (4.9) 更加看重增加的部分。增加的部分 $\frac{pbest*(c_1*\alpha_1)}{gbest*(c_2*\alpha_2)} \cdot (1-\beta)$ 后面使用点乘相当于截取 $\frac{pbest*(c_1*\alpha_1)}{gbest*(c_2*\alpha_2)}$ 的前 $(100 * (1 - \beta))$% 的序列。$\frac{pbest*(c_1*\alpha_1)}{gbest*(c_2*\alpha_2)}$ 这一部分是 pbest 和 gbest 的混合运算结果，pbest 和 gbest 的公共序列优先保留。这是因为 pbest 和 gbest 都有的序列较大概率是最优解的一个片段。同时公式 (4.9) 能够保证更新后的结果的合法性⑤。

另一方面，公式 (4.9) 中保留了粒子群算法的特性-更新依赖个体经验和社会信息。从而种群整体移向全局最优解。更新迭代公式 (4.9) 具有较强活性。即使在极端情况下，整个种群由一模一样的粒子组成时，由于随机因子 α_1, α_2 和数乘的随机性，种群可以创造出新的粒子。所以迭代公式 (4.9) 具有较强活性，增加种群多样性，提高收敛速度。

4.5 超参数设置

种群规模依赖于城市数量，个体/社会学习因子依赖于粒子长度 (= 城市数量 = 问题规模)。具体参考表 (7.1)。

③pbest 是 S 的一个元素，pbest.length 就是 pbest 的长度, 章节 [4.3.1] 有提及。
④gbest 是 S 的一个元素，gbest.length 就是 gbest 的长度, 章节 [4.3.1] 有提及。
⑤合法性指的是 x^t 是一个粒子，更新后 x^{t+1} 也是一个粒子。粒子对应解空间的一个解。

5 遗传算法

5.1 遗传算法

为了完成对比实验，我们设计了遗传算法来解决旅行商问题。使用的遗传算法流程：

1. 初始化参数
2. 随机产生初始化种群
3. 满足迭代停止条件退出，否则转 4
4. 交叉，变异，选择
5. 转 3

遗传算法的个体和公式 (4.1) 定义一样。超参数包含个体变异概率 ppossibility，单点/双点交叉概率 gpossibility, 种群规模 n。算子具体操作如下：

交叉：T 代种群 n 个个体随机组成 $\frac{n}{2}$ 对夫妻，根据单点/双点交叉概率 gpossibility 确定单点还是双点交叉得到两个子代。我们的实现与图 (5.1),(5.2) 存在区别。由于直接交换会导致个体 A，B 可能非法，所以，实际操作中单双点交叉会引入 [4.3] 中的定义。交叉片段分别记为 Seg_A, Seg_B，生成的新个体 A', B' 分别是 <$Seg_B, A - Seg_B$>, <$Seg_A, B - Seg_A$>。

变异：根据 ppossibility(个体变异概率) 确定是否执行变异，执行变异时，随机选取某一个点与另一个点进行交换。

选择：按照适应值升序排序，选取前 n 个进入下一代。

实际实验中，超参数设置见表 (7.1)。

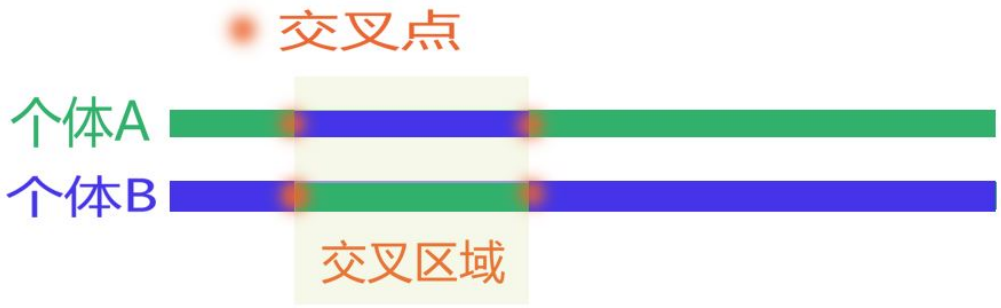

图 5.1　双点交叉

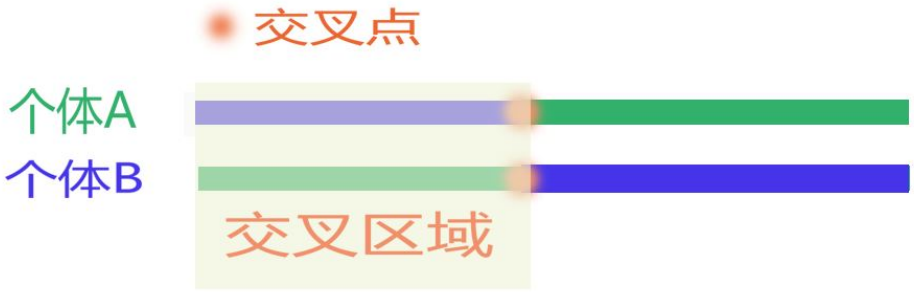

图 5.2　单点交叉

6 ps-PSO-v1 和随机生成

我们在改进基于点交换的粒子群算法前的版本 ps-PSO-v1 的效果并不好，通过总结分析我们提出了改进版 ps-PSO-v2。随机生成解的方法作为对照方案。下面我们将会简单介绍 ps-PSO-v1 和随机生成的方法。

6.1 ps-PSO-v1

个体仍然采用公式 (4.1) 定义。参考 [4.1] 的标准粒子群算法的思路。个体迭代更新应当参考个体经验和全局最有信息。粒子 x^t 更新到 x^{t+1}，过程如下:

$$x^{t'} = swap(x^t, pbest, ppossibility) \tag{6.1}$$

$$x^{t+1} = swap(x^{t'}, gbest, gpossibility) \tag{6.2}$$

其中，ppossibility①是对个体经验参考程度，gpossibility②是对社会信息参考程度。pbest，gbest 分别是个体最有记录和种群③最优记录。swap(input,target,p) 操作表示对 input 的序列的每一位根据概率 p 决定是否替换成目标 target 的对应元素。若 p=1 那么输出等于 target，p=0 输出为 input。伪代码如下：

```
Input: x^t (input)
Output: x^(t+1)

function update(input, target, p):
    for index in 1,2...,input.length:
        if random(0,1)<=p:
            targetIndex = indexOf(target[index])
            input.swapByIndex(index, targetIndex)
    return input
```

ps-PSO-v1 的迭代更新公式 (6.1),(6.2) 指出个体应当参考个体经验和社会信息。但是，个体的变化依赖个体经验参考程度和社会信息参考程度。并且当种群区域一致时，无法跳出局部解。所以，参考该方法的不足之处，我们提出改进版 ps-PSO-v2(参考第4部分)。

①ppossibility 设置见表 (7.1)
②gpossibility 设置见表 (7.1)
③种群规模设置见表 (7.1)

6.2 随机生成

随机生成④的方法作为对比对象。与 ps-PSO-v1, ps-PSO-v2 和遗传算法的不同之处在于迭代更新时，重新随机生成整个种群。所以，随机生成结果的最优解等价于随机生成 n*T 个解中的最优解。n 是种群规模，T 是迭代次数。

6.3 四种方法区别

我们的实验包括四种方法，分别为 ps-PSO-v1, ps-PSO-v2, 遗传算法 (下文使用 GA 表示) 和随机生成 (下文使用 Random 表示)。都可以认为时迭代算法，粒子定义均采用公式（(4.1)。种群规模均等于城市数量。在我们的实验中，迭代次数相同。在迭代更新的时候, 采用不同策略，对比如下：

表 6.1 ps-PSO-v1,ps-PSO-v2,GA and Random

方法	更新策略
ps-PSO-v1	$x^{t'} = swap(x^t, pbest, ppossibility)$ $x^{t+1} = swap(x^{t'}, gbest, gpossibility)$
ps-PSO-v2	$x^{t+1} = x^t + \frac{pbest*(c_1*\alpha_1)}{gbest*(c_2*\alpha_2)} \cdot (1-\beta)$
GA	交叉，变异，选择
Random	随机生成整个群体

④参数设置见表 (7.1)

7 实验

实验环境是使用个人 PC，处理器是 Intel(R) Core(TM)i5-5200U CPU @ 2.20GHz，内存 8GB，系统 Ubuntu 18。实验代码采用语言 C++, 利用多线程加速。同时跑 9 个数据集 (每个数据集一个线程)，总共耗时 12 小时 (主要耗时是规模较大的数据集 eg: dsj1000, fl1400)。

7.1 实验设置

表 7.1 实验设置

方法	更新策略	迭代次数	参数设置
ps-PSO-v1	$x^{t'} = swap(x^t, pbest, ppossibility)$ $x^{t+1} = swap(x^{t'}, gbest, gpossibility)$	2000	ppossibility=$\frac{1}{cityNumber}$ gpossibility=$\frac{1}{cityNumber}$ 种群规模 n= 城市数量 *2
ps-PSO-v2	$x^{t+1} = x^t + \frac{pbest*(c_1*\alpha_1)}{gbest*(c_2*\alpha_2)} \cdot (1-\beta)$	2000	$c_1 = 0.4, c_2 = 0.6$ $\beta = 0.1, n =$ 城市数量 *2
GA	交叉，变异，选择	2000	ppossibility = $\frac{1}{cityNumber}$ gpossibility=0.5, n= 城市数量 *2
Random	随机生成整个群体	2000	n= 城市数量 *2

7.2 实验评估标准

实验评估标准包含两个，一个是收敛速度，另一个是哈密顿回路路径长度（(下文称为代价 Cost)。由于收敛速度采用时间计算可能出现机器差异，IO 阻塞等系统因素和算法复杂度差异造成的评估标准不一致的情况。这里采用代价因迭代次数改变而发生的改变作为收敛速度评判标准。收敛速度采用定性评测，绘制代价-迭代次数曲线进行评估。代价采用定量评测，记录迭代完成后的最优解进行评估。

7.3 实验结果对比

代价表 (7.2) 展示了四种算法在 9 个数据集上测试的结果。可以看到在数据规模相对较小时 (n 小于等于 280),ps-PSO-v2 和 GA 结果接近 (GA 结果稍好于 ps-PSO-v2)。在规模较大时 (n>=300)，ps-PSO-v2 效果会比 GA 优，并在规模逐渐增大时，差距会比较明显。

表 7.2　代价对比

数据集	Random	ps-PSO-v1	ps-PSO-v2	GA	opt
att48	102271.0	62231.3	36252.0	**35979.6**	33523.70850743559
gr96	2661.51	1520.08	642.639	**595.97**	512.3093796093739
eil101	2771.4	1781.65	790.485	**732.833**	642.3095357906022
ch130	38371.6	24760.0	9775.76	**8180.42**	6110.86059813714
gr202	2717.87	1974.59	748.987	**702.368**	549.9980703917479
a280	29489.1	25028.1	6405.02	**6134.6**	2586.7696475631606
d657	786149.0	767662.0	**205420.0**	249147.0	-
dsj1000	514176000.0	507802000.0	**111640000.0**	149576000.0	-
fl1400	1559280.0	1527400.0	**193209.0**	327933.0	-

收敛速度① 图 (7.1),(7.2),(7.3) 展示的是代价-迭代次数曲线。我们选取了数据规模分别时 48，280，1400(分别是测试数据最小，中间，最大规模)。

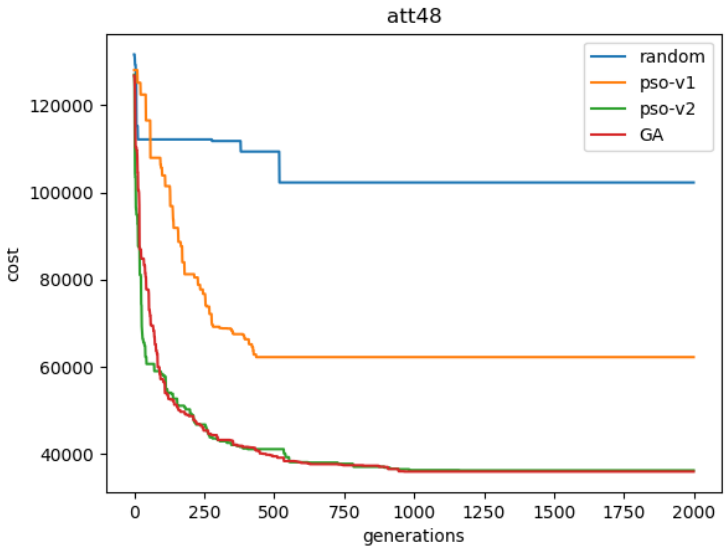

图 7.1　att48-48 capitals of the US (Padberg/Rinaldi)

7.4　结果分析

7.4.1　ps-PSO-v1

由表 (7.2), 图 (7.1),(7.2),(7.3)，可以看出，ps-PSO-v1 收敛速率远低于 ps-PSO-v2 和 GA，最终结果也不理想但比随机生成要好 (见图 (7.1))。该实验证明 PSO 算法的可行性。同时发现在图 (7.1) 的 500 代后，收敛曲线扁平。经过分析发现种群趋

①九个收敛曲线图见附录 A

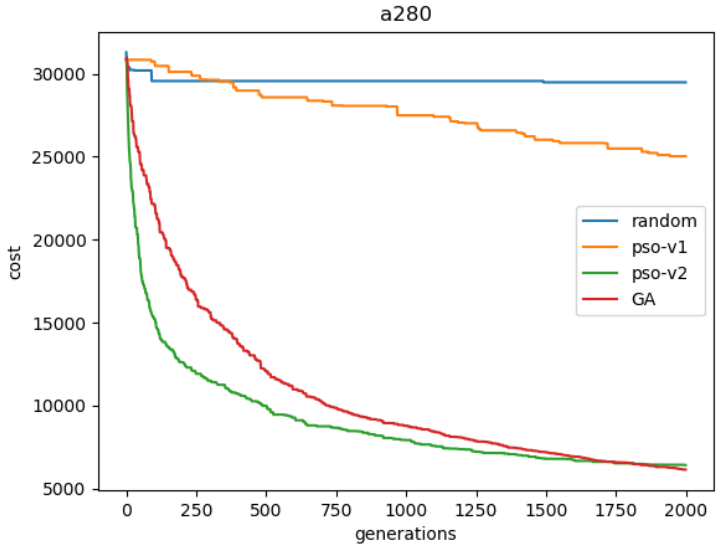

图 7.2　a280-drilling problem (Ludwig)

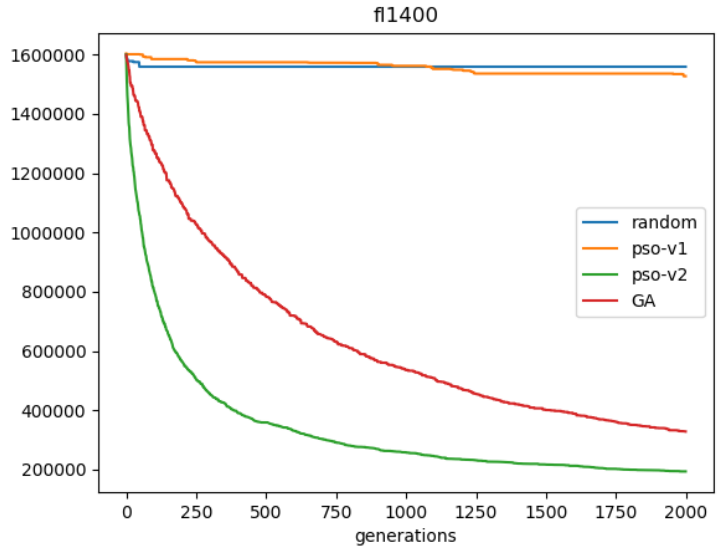

图 7.3　fl1400-Drilling problem (Reinelt)

于一致性，ps-PSO-v1 方法无法产生新的粒子。鉴于收敛速度慢和活性低 (种群趋于一致性后无法产生新粒子)，我们改进得到收敛速度快，活性强的 ps-PSO-v2。

7.4.2　ps-PSO-v2 和 GA

根据图 (7.2),(7.3)，发现 ps-PSO-v2 收敛速度快于 GA 算法，在较短时间内就能得到较为满意的结果。ps-PSO-v2 和 GA 的收敛曲线都不会完全平坦 (ps-PSO-v1 可能出现见图 (7.1))。GA 依赖变异算子保持种群活性。ps-PSO-v2 引入了随机因

子 α_1, α_2 和序列数乘点乘除法等新运算保证种群活力。这两种算法均展示出了很多优良特性，如快速收敛，保持种群优良基因等。

ps-PSO-v2 的收敛速度快，在大规模时优势会比 GA 明显。但是，可以发现后期 ps-PSO-v2 收敛过慢，会被 GA 超越 (图 (7.2))。这也给未来方向提出了新的问题。或许可以结合 GA 和 ps-PSO-v2 算法，得到更加优秀的解决旅行商问题的新算法。

7.4.3 逼近最优解

分析表 (7.2), 图 (7.2) 得出，我们实验的算法都出现了逼近最优解时阻力变大的情况。从而，我们在一定时间内得到的解与最优解存在一定的距离。为了明白我们算法进入后期后阻力变大的原因。我们对解进行了分析，如图 (7.4), (7.5), (7.6)(左上是城市点分布，右上最优解，左中 Random，右中 ps-PSO-v1，左下 ps-PSO-v2, 右下 GA)。我们认为，在后期，算法收敛到了一个比较好的解答时 (eg：图 (7.5)), 要跳出局部解需要算法刚好挑选中某些点进行交换。那么我们缺乏一个挑选点的规则，如果有一个指导的方向，那么就有希望迈一大步到达全局最优解附近。分析发现，最优解一般很少交叉线存在，也就是说我们或许可以根据减少交叉线的思路改进 ps-PSO-v2 算法得到一个前期快速收敛，后期不断突破局部最优解的新算法。

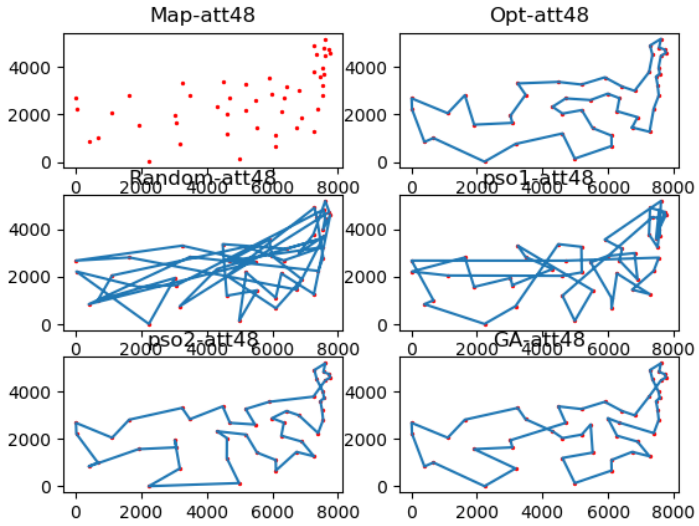

图 7.4 att48-48 capitals of the US (Padberg/Rinaldi)

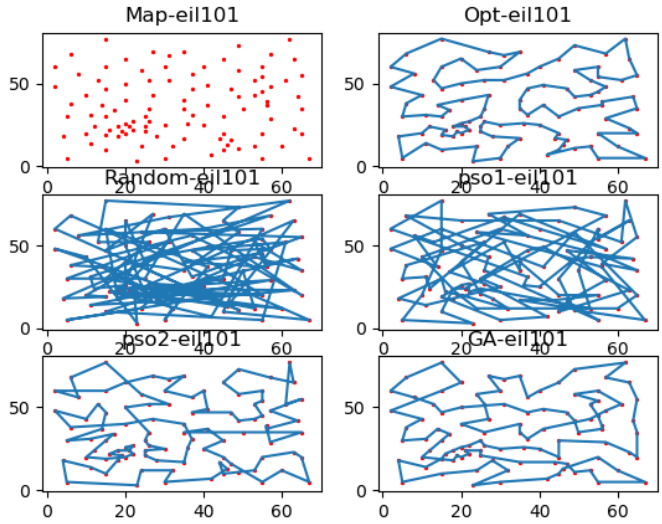

图 7.5　101-city problem (Christofides/Eilon)

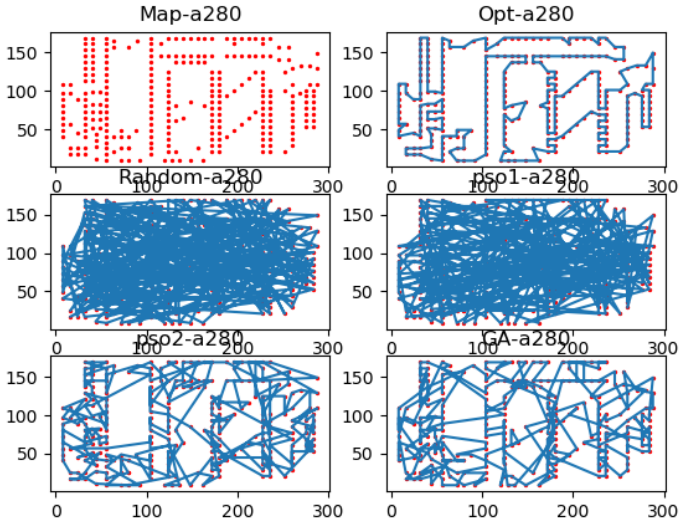

图 7.6　a280-drilling problem (Ludwig)

8 结论和未来方向

我们提出的基于点交换的粒子群算法可以快速收敛。ps-PSO-v2 能够在短时间内取得满意结果。但是，后期收敛缺乏动力。GA 算法前期收敛速度不如 ps-PSO-v2，但是能够持续降低代价。我们根据现有实验数据认为，GA 和 ps-PSO-v2 的收敛曲线会在某一点 t，在 t 点前 ps-PSO-v2 优于 GA，t 点后 GA 稍微优于 ps-PSO-v2。在规模变大时，该点会不断向后挪动。所以，大规模情况下，ps-PSO-v2 会更加实用。

针对 ps-PSO-v2 和 GA 的优劣，我们觉得未来可以综合两个算法的优点 (eg: 先 ps-PSO-v2 后 GA)。在后期引入消除交叉线的思路进一步逼近最优解。

参考文献

[1] 卓雪雪. 蚁群遗传混合算法在求解旅行商问题上的应用 [J]. 价值工程, , 39(02): 188 – 193.

[2] 费腾. 基于 T-ACO 算法的旅行商问题求解优化研究 [J]. 软件工程, 2020, 23(02): 82 – 52.

[3] 陶丽华. 基于 TSP 问题的动态蚁群遗传算法 [J]. 机械设计与制造, 2019: 147 – 149+154.

[4] 李汝佳. 基于蚁群算法的旅游路线规划问题研究 [J]. 电脑知识与技术, 2019, 15(08): 041 – 731.

[5] 尚宝平. 一种求解 TSP 问题的多策略改进蚁群算法 [J]. 数学的实践与认识, 2019, 49(02): 422 – 512.

[6] 王殿超. 一种改进的遗传算法在 TSP 问题中的应用 [J]. 辽宁工业大学学报 (自然科学版), 2019, 39(04): 932 – 532.

[7] 胡士娟. 求解工作量平衡多旅行商问题的改进遗传算法 [J]. 计算机工程与应用, 2019, 55(17): 132+551 – 051.

[8] Hong Qu, Zhang Yi, Huajin Tang. Improving local minima of columnar competitive model for TSPs[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2006, 53(6): 1353 – 1362.

[9] KENNEDY J, EBERHART R. Particle swarm optimization[A/OL]. Proceedings of International Conference on Neural Networks (ICNN'95), Perth, WA, Australia, November 27 - December 1, 1995[C], [S.l.]: IEEE, 1995: 1942 – 1948. https://doi.org/10.1109/ICNN.1995.488968.

[10] 王闯, 张勇, 李学贵, et al. 改进粒子群算法及其在聚类分析中应用 [J]. 系统仿真学报, 2020: 21 – 1.

[11] 王尔申, 孙彩苗, 黄煜峰, et al. 改进粒子群优化的卫星导航选星算法 [J]. 北京航空航天大学学报, 2020: 7 – 1.

致谢

谢谢提供帮助的各位老师，同学，还有 TSPLIB 数据集提供者和参考文献的作者们。

附录 A

A.1 收敛曲线和路线图

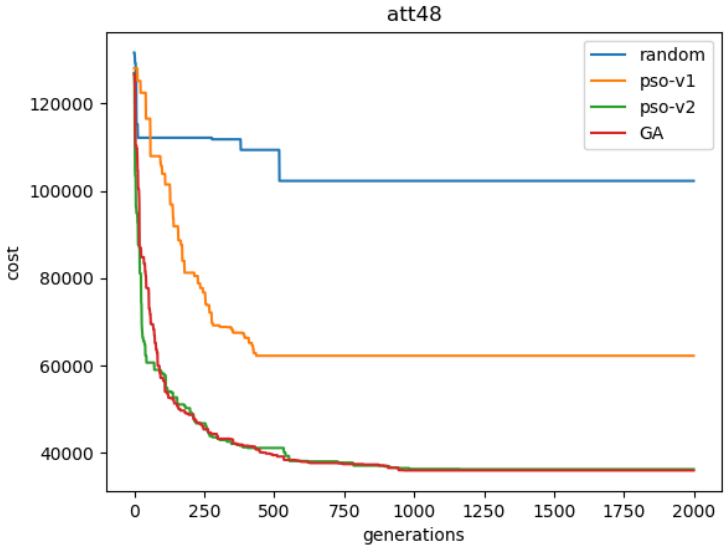

图 A.1 att48 - 48 capitals of the US (Padberg/Rinaldi)

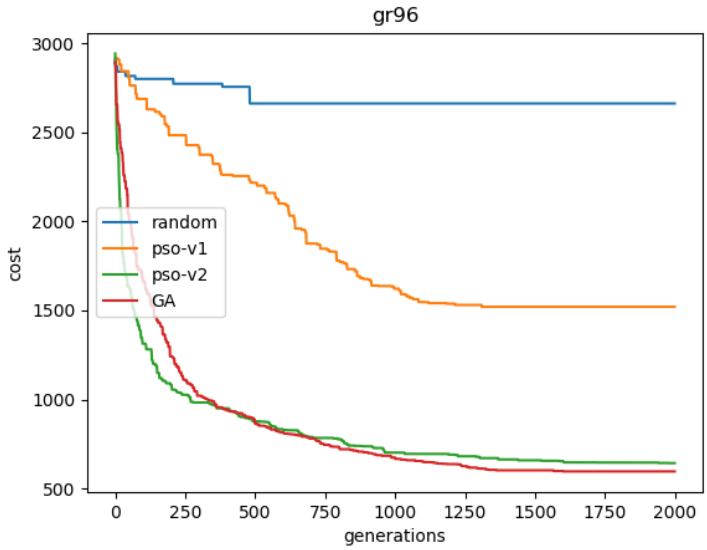

图 A.2 gr96 - Africa-Subproblem of 666-city TSP (Groetschel)

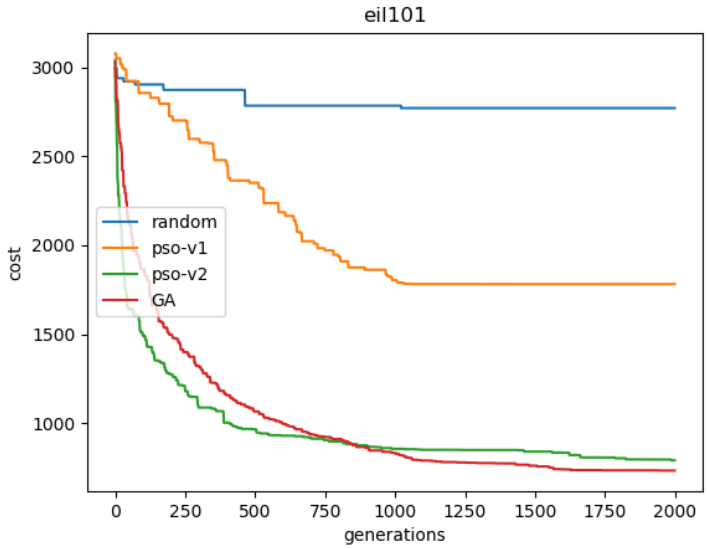

图 A.3　eil101 - 101-city problem (Christofides/Eilon)

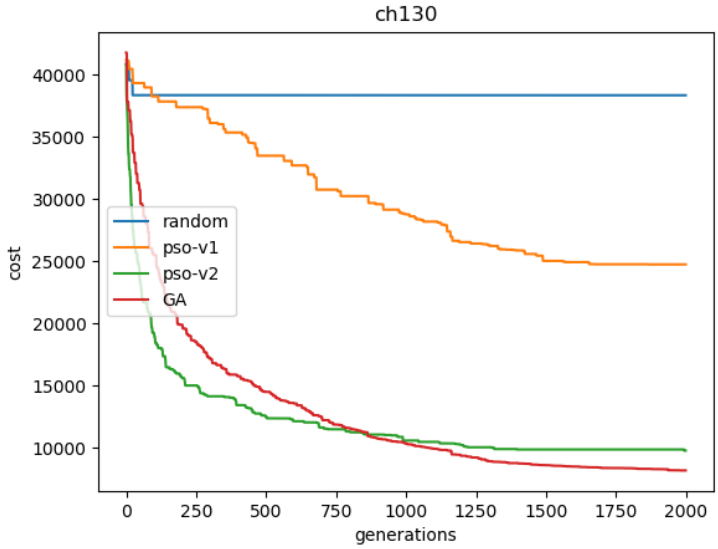

图 A.4　ch130 - 130 city problem (Churritz)

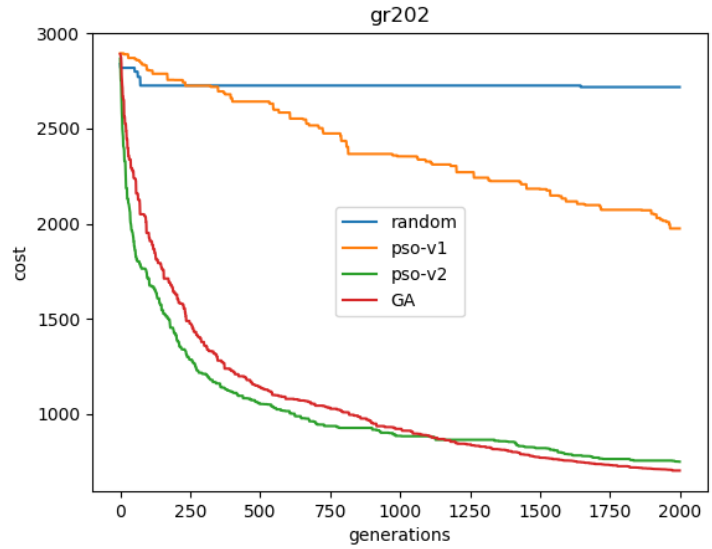

图 A.5　gr202 - Europe-Subproblem of 666-city TSP (Groetschel)

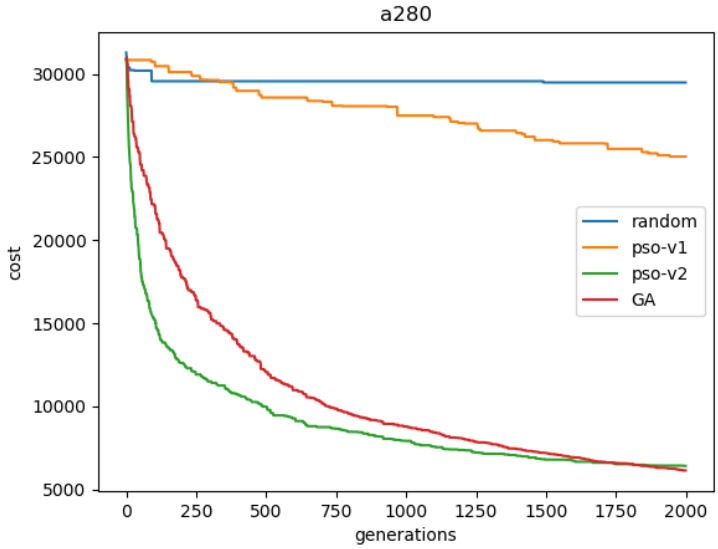

图 A.6　a280 - drilling problem (Ludwig)

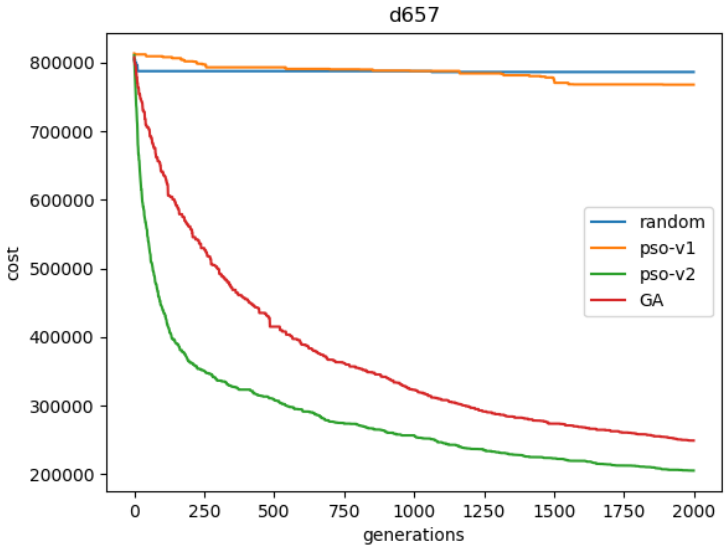

图 A.7　d657 - Drilling problem (Reinelt)

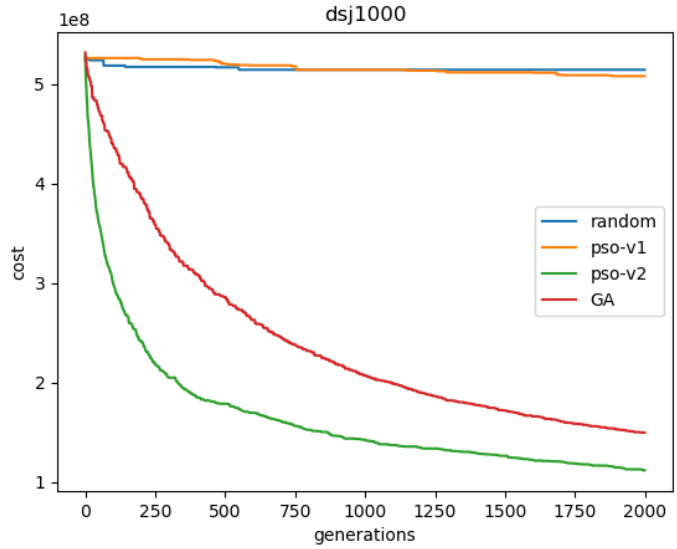

图 A.8　dsj1000 - Clustered random problem (Johnson)

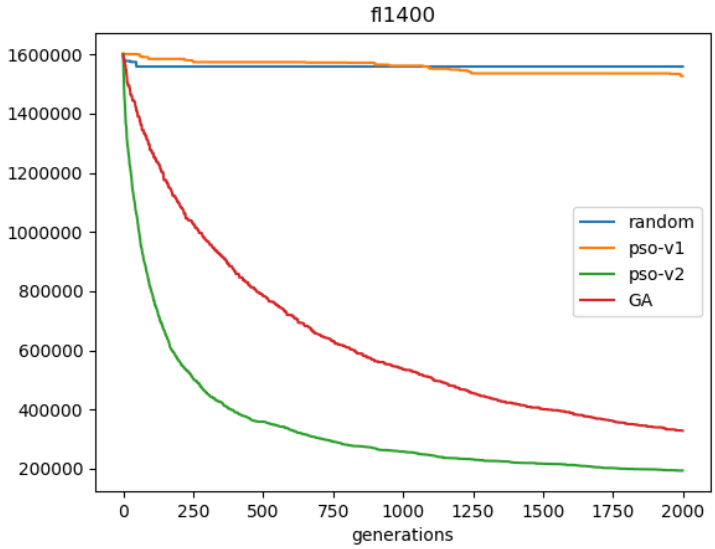

图 A.9　fl1400 - Drilling problem (Reinelt)

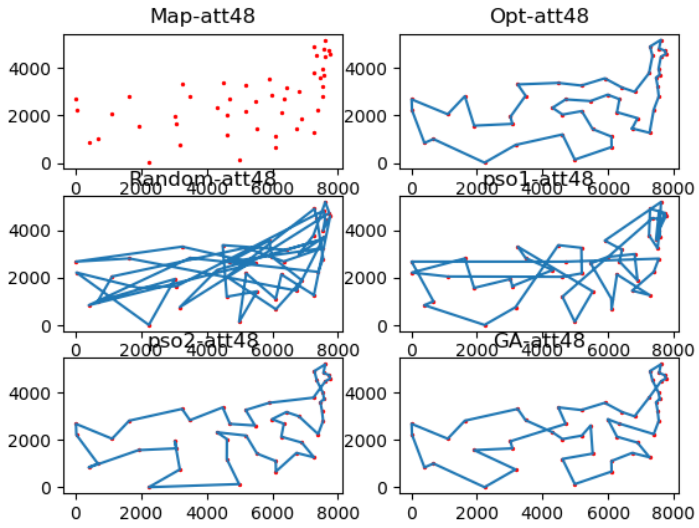

图 A.10　att48 - 48 capitals of the US (Padberg/Rinaldi)

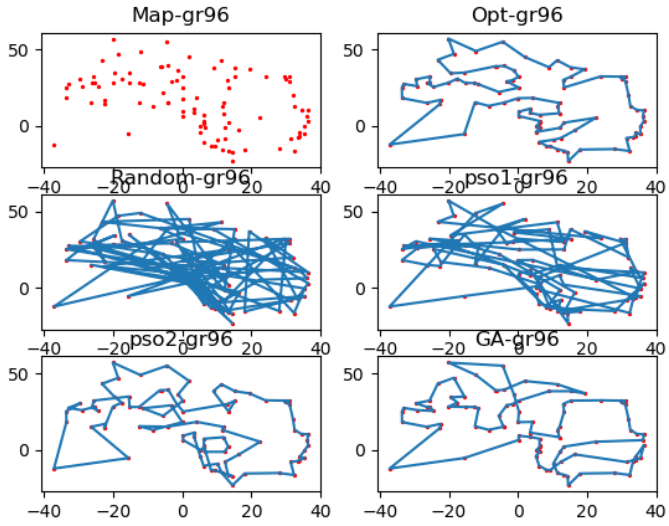

图 A.11　gr96 - Africa-Subproblem of 666-city TSP (Groetschel)

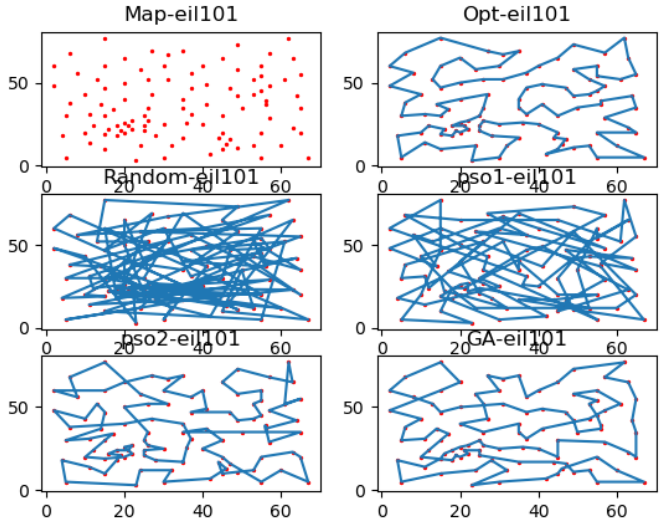

图 A.12　eil101 - 101-city problem (Christofides/Eilon)

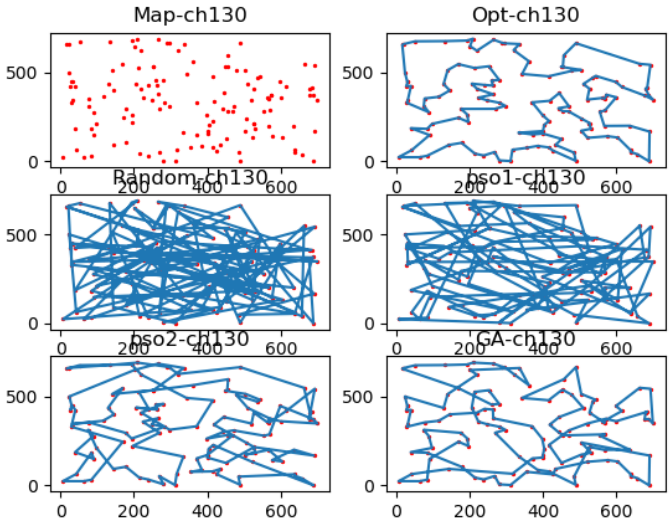

图 A.13　ch130 - 130 city problem (Churritz)

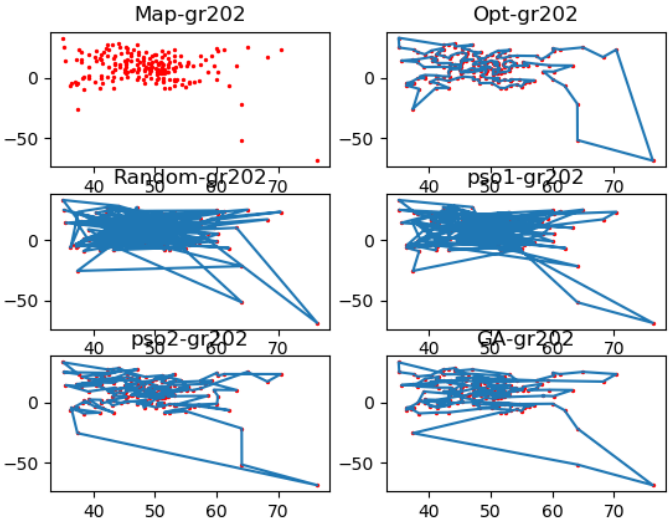

图 A.14　gr202 - Europe-Subproblem of 666-city TSP (Groetschel)

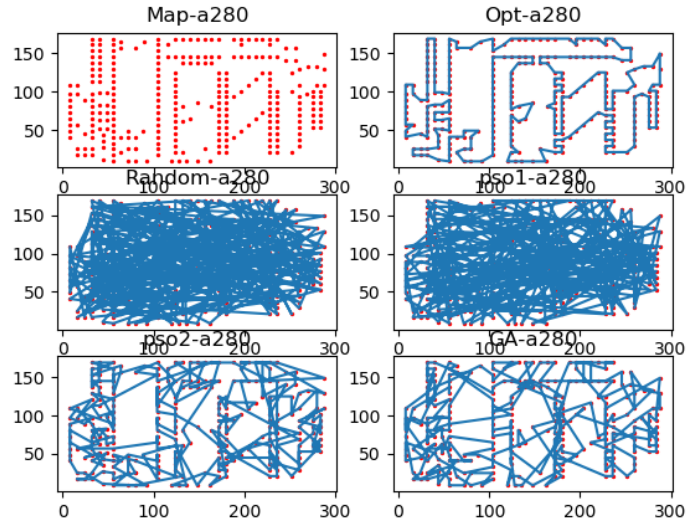

图 A.15　a280 - drilling problem (Ludwig)

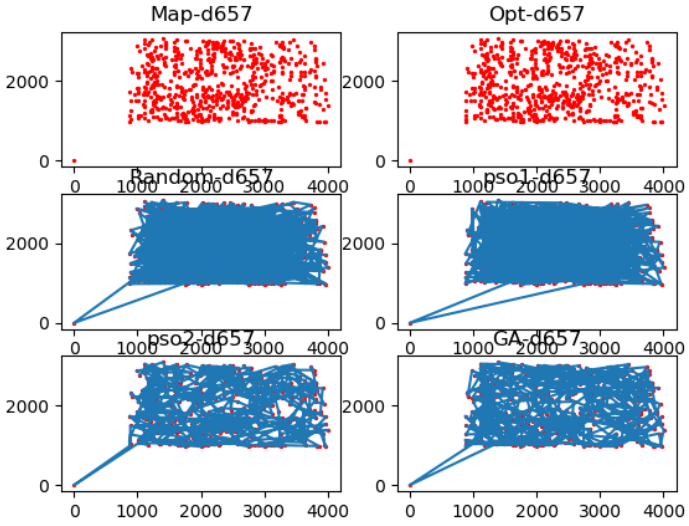

图 A.16　d657 - Drilling problem (Reinelt)

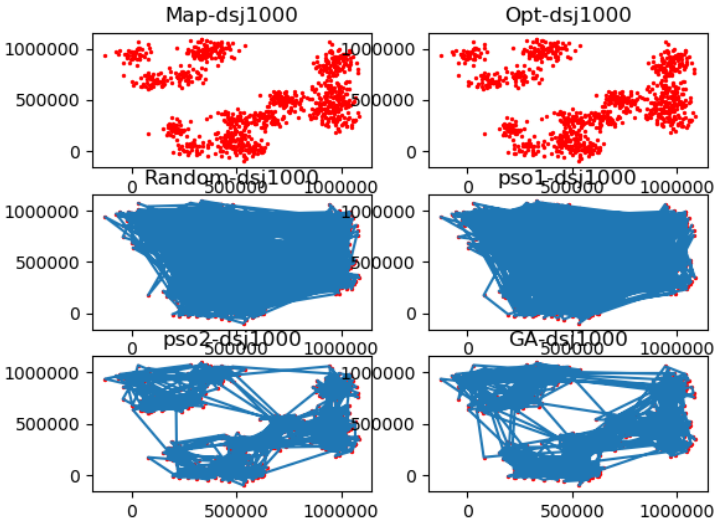

图 A.17　dsj1000 - Clustered random problem (Johnson)

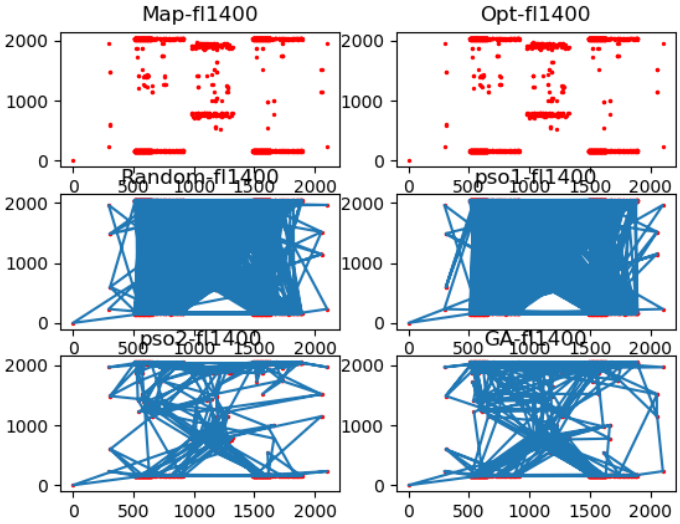

图 A.18　fl1400 - Drilling problem (Reinelt)