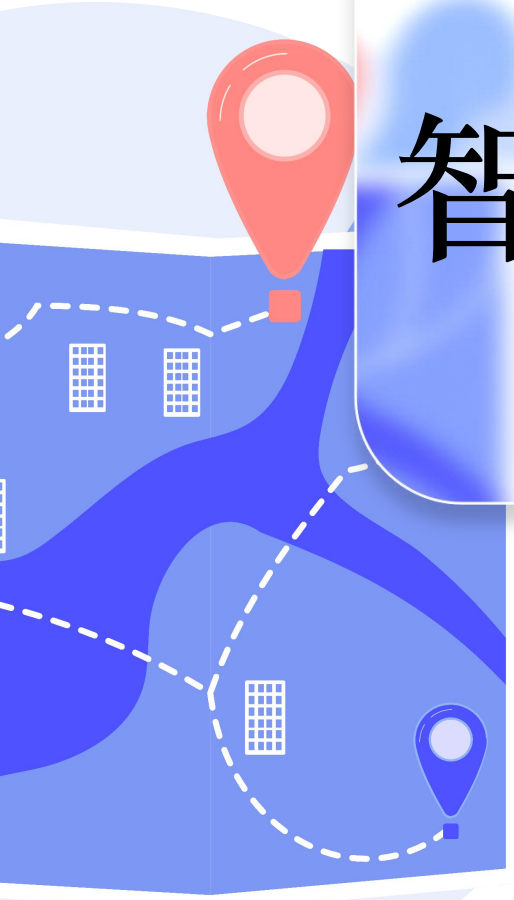
浙江大学
ZHEJIANG UNIVERSITY



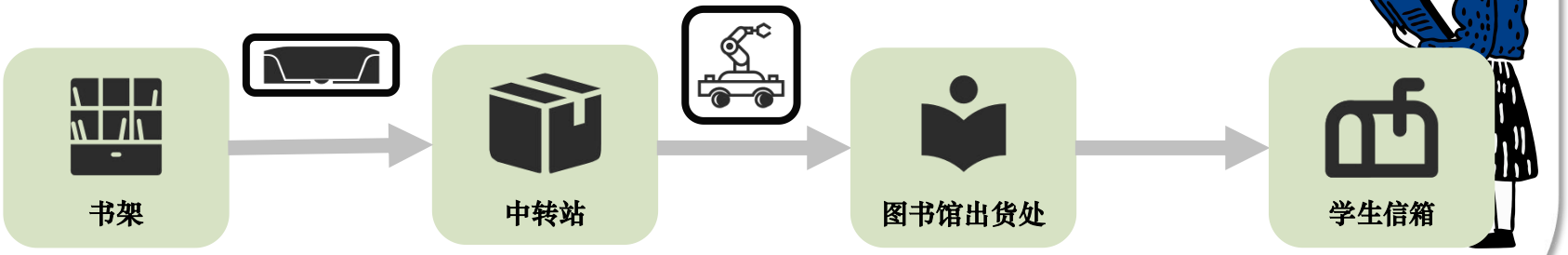
智能图书馆派送路径优化

2025年 6 月 3 日

背景导入

国民人均阅读量↑　　找不到书、忘记还书☹

便捷、高效的借阅服务需求↑



网上预定 — 自动分拣 — 送书上门

基本假设

假设1: 对于所有问题，不考虑机器人的实际体积可能造成的碰撞问题

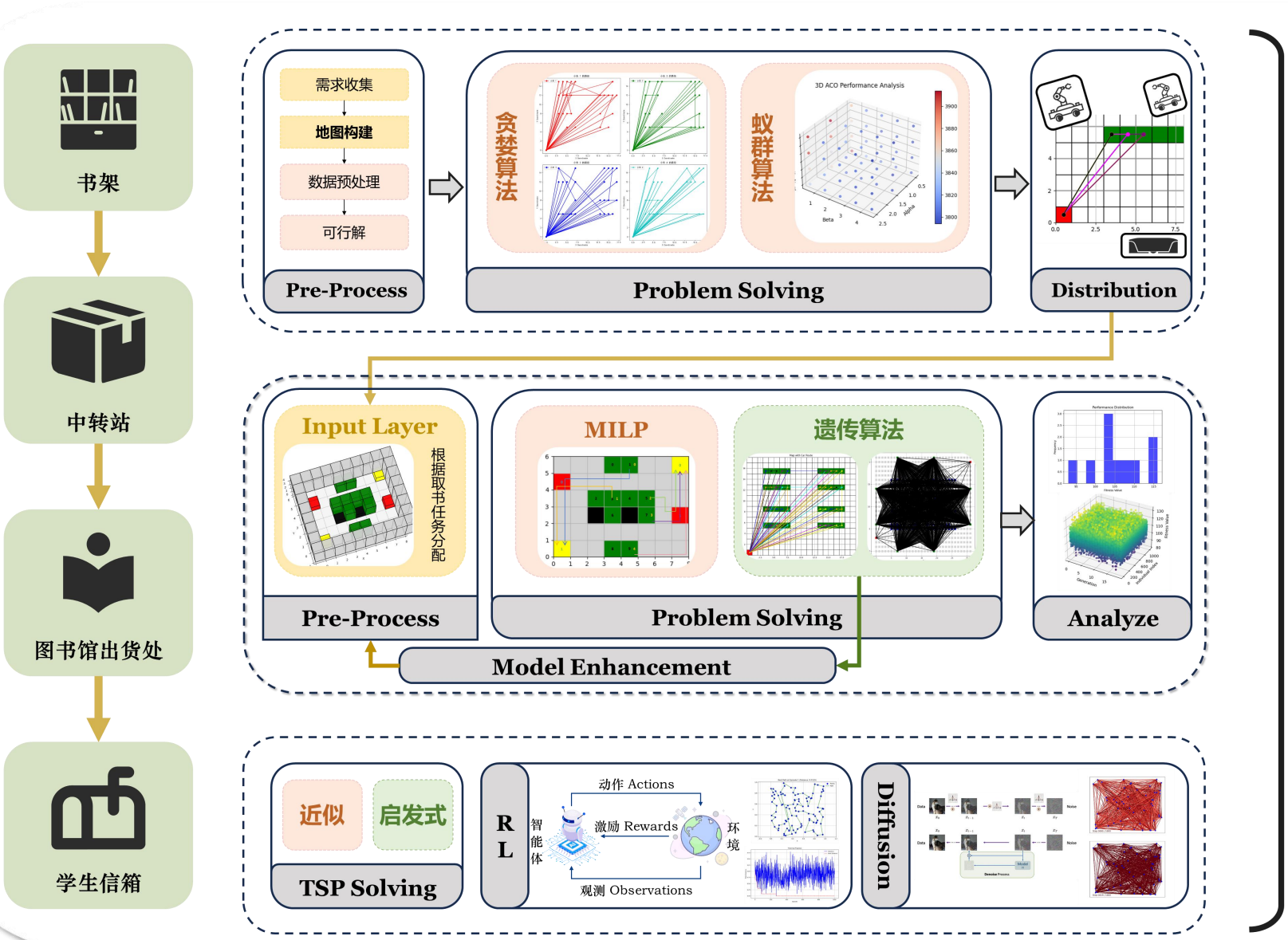
假设2: 为了提高解决效率，假设图书馆每天指定时间集中处理订单

假设3: 我们假设同一个地点只有一个用户进行订阅

假设3：对于问题二，我们假设一个货架只能储存一个支架来存放书籍

假设4: 对于问题二，我们只考虑完成取书——挑拣——回收/放回这一任务的最短距离，不考虑AGV 小车的调度问题，即不考虑多辆小车的调度问题

工作框架


书架
中转站
图书馆出货处
学生信箱
需求收集
地图构建
数据预处理
可行解
Pre-Process
贪婪算法
蚁群算法
3D ACO Performance Analysis
Problem Solving
Distribution
Input Layer
根据取书任务分配
Pre-Process
MILP
遗传算法
Problem Solving
Analyze
Model Enhancement
近似
启发式
TSP Solving
RL
动作 Actions
智能体
激励 Rewards
环境
观测 Observations
Diffusion
1
2
3
4
System Analyze

书架

中转站

图书馆出货处

学生信箱

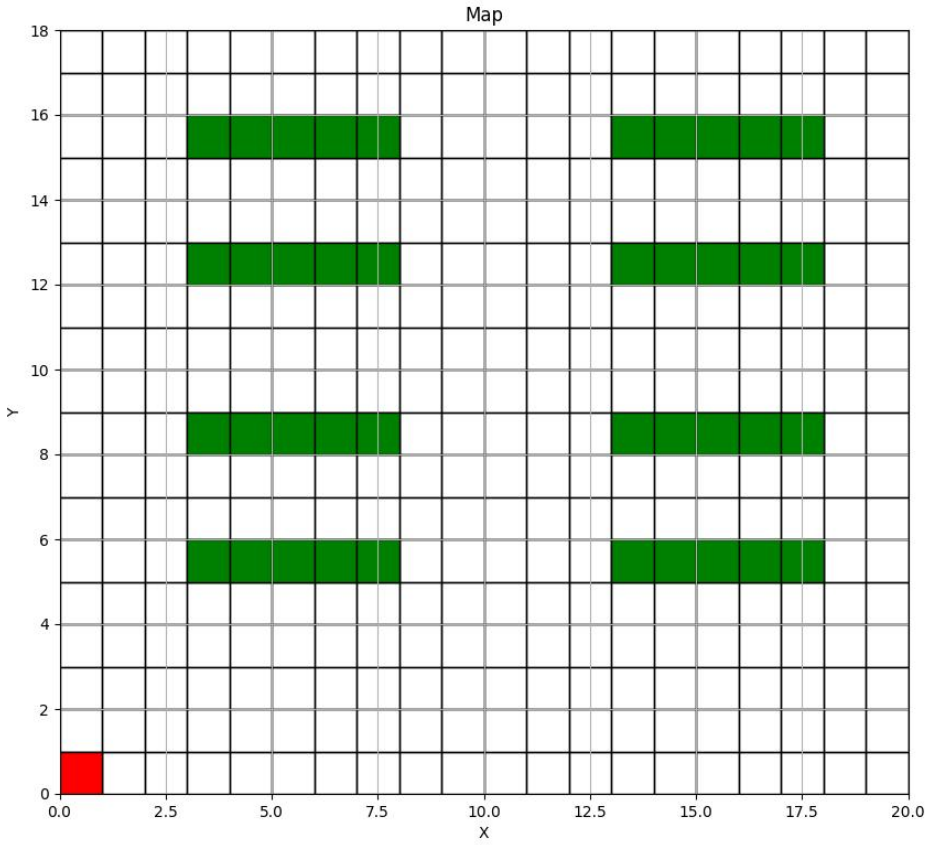
问题一定义

N 辆机器人同时取书

从起始位置出发往返书架取书

每个机器人一次最多M本书

规划机器人路径使总路径最短




书架


中转站

目标函数

$$Z = \min \sum_{i=0}^{N} \sum_{j=0, j\neq i}^{N} \sum_{k=1}^{M} x_{ijk} \cdot d_{ij}$$

配送量不能超过书架的需求量

$$0 \leq y_{ik} \leq q_i, i \in N, k \in C$$

至少有一辆运输小车为书架进行配送

$$\sum_{i=0}^{N} \sum_{k=1}^{M} x_{ijk} \geq 1, j \in N$$

到达某点的车辆数应该等于离开该点的车辆数

$$\sum_{i=0}^{N} x_{ipk} - \sum_{j=0}^{N} x_{pjk} = 0, p \in N, k \in C$$

所有书架的取书任务必须全部被完成

$$\sum_{k=1}^{M} y_{ik} = q_i, i \in N$$

运输小车不超载

$$\sum_{i=1}^{N} y_{ik} \leq Q, k \in C, 0 < q_i < Q, i \in N$$

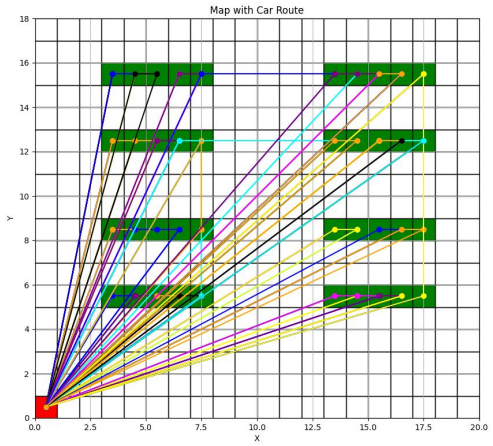
限制没有回路

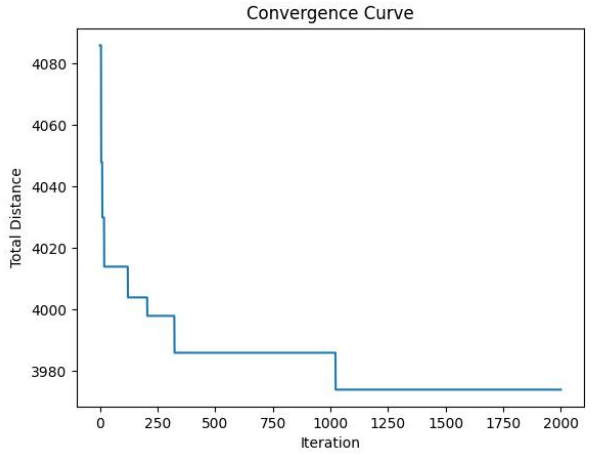
$$\sum_{i=0}^{N} \sum_{j=0, j\neq i}^{N} x_{ijk} \leq |s| - 1, 2 \leq |s| \leq n - 1$$

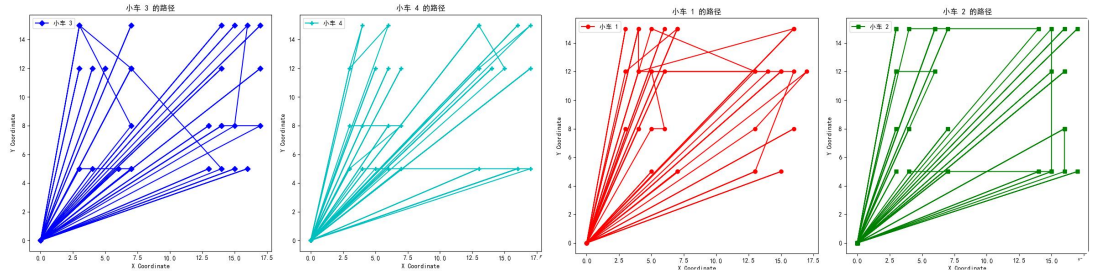
算法实现 —— 贪婪算法

```
function generate_initial_solution
    初始化解决方案列表和剩余需求
    while 剩余需求大于 0 do
        初始化路线，从原点开始
        初始化剩余容量为车辆容量
        while 剩余容量大于 0 且剩余需求大于 0 do
            找到可访问的书架并随机访问一个书架
            if 剩余容量大于等于书架需求 then
                将书架添加到路线中
                更新剩余容量和需求
            else
                将书架添加到路线中
                更新书架需求和剩余容量为 0
            end if
        end while
        将路线添加到解决方案中
    end while
```

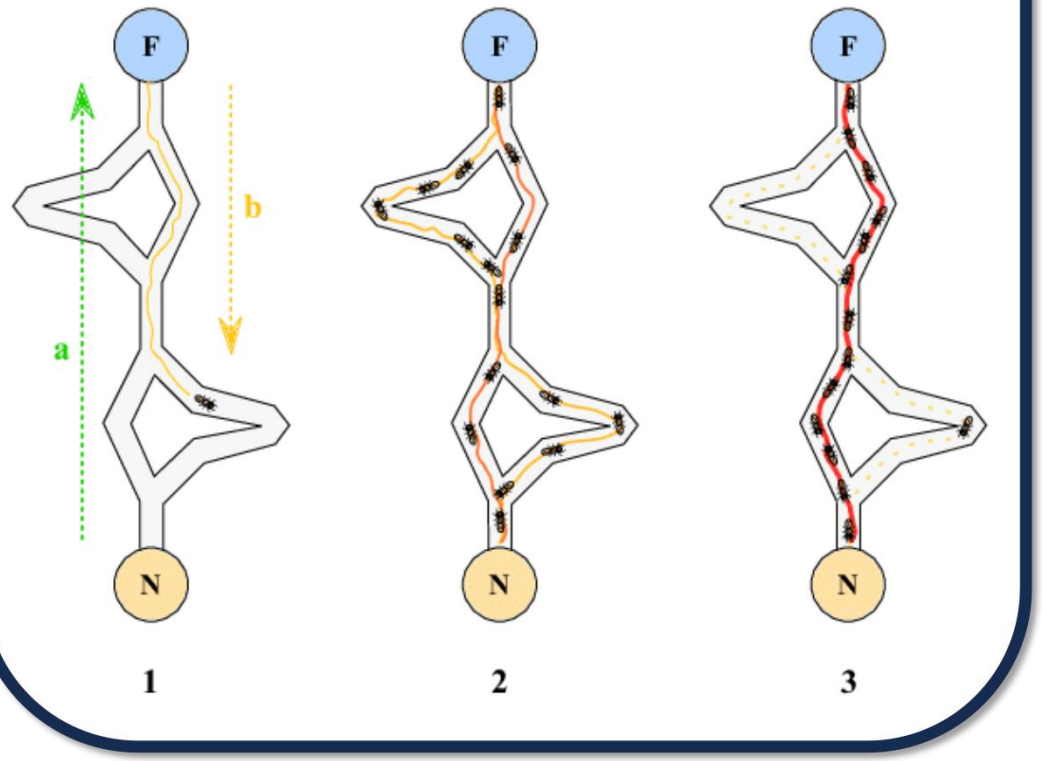
结果







蚁群算法



科学家发现，蚁群在可以在有障碍物的环境下，找到一条最短到达食物的路径。

蚂蚁在路径上释放**信息素**。
后到的蚂蚁沿**信息素浓度高**的路径行走。

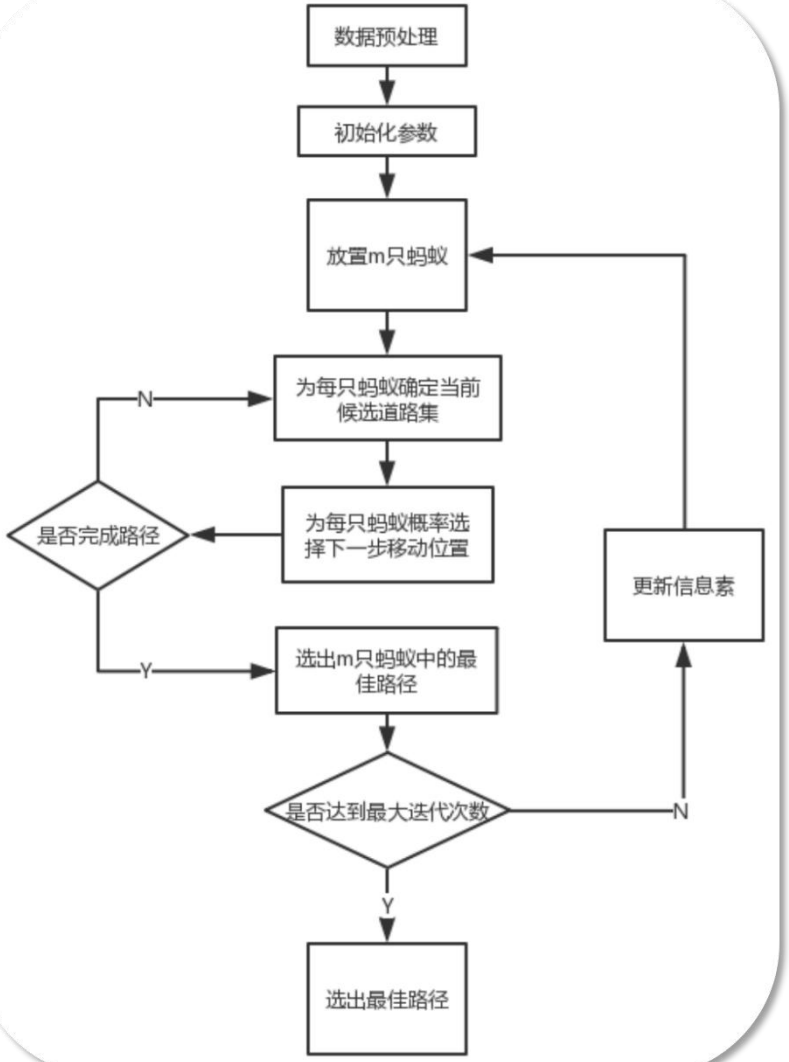
在所有可能的路径中，由于**往返最短路径**花的时间少，通过频率高，所以信息素浓度高。

这又会吸引更多蚂蚁走这条路径，
形成**正反馈**。

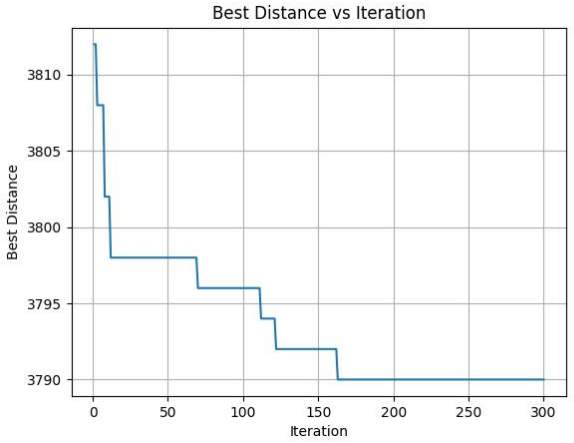
每只蚂蚁只关注局部信息 —— 群体的智能涌现

算法实现

蚁群觅食	蚁群优化算法
蚁群	搜索域内的一组有效解
信息素	信息素浓度变量
蚁巢到食物的一条路径	一个有效解
找到的最短路径	问题最优解



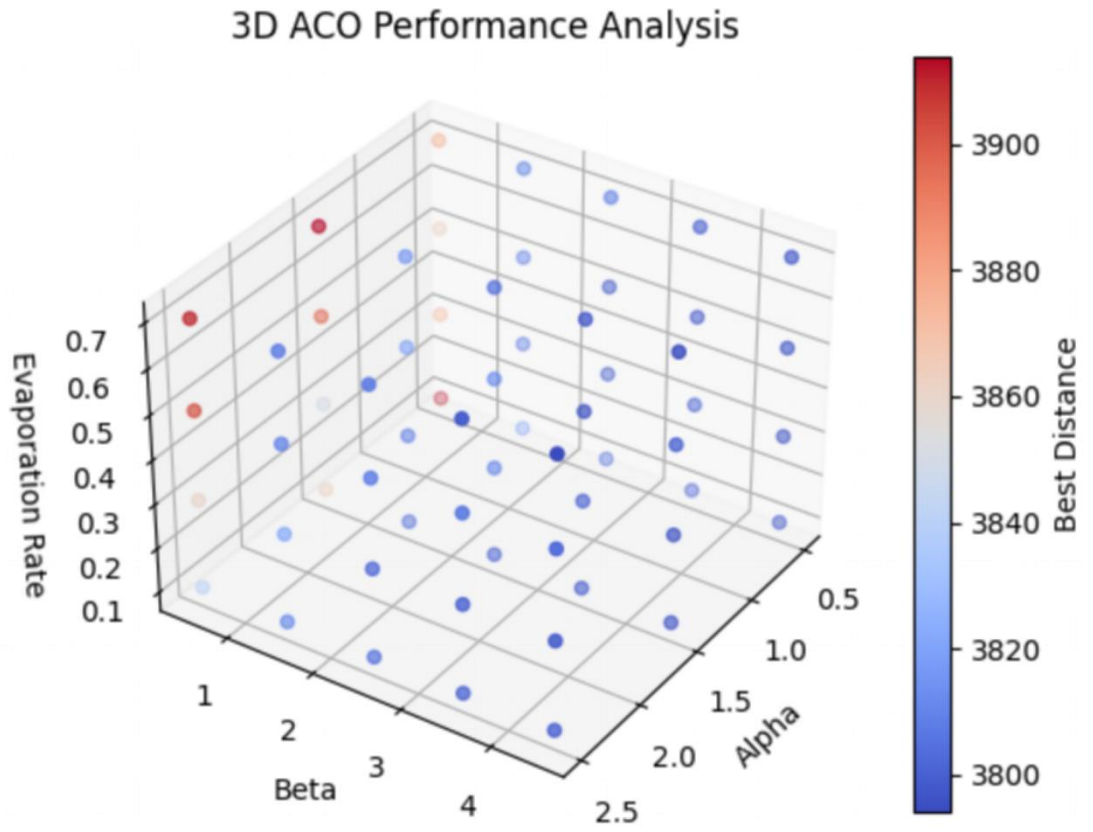
结果



相对于贪婪算法

其收敛速度更快，

需要的迭代次数更少



参数敏感性分析

Alpha、Beta、信息素浓度

- Alpha 参数控制信息素的重要性
- Beta 参数控制启发式信息的影响力
- 信息素浓度在算法中用于表示一条路径的优劣，蚂蚁在选择路径时会倾向于选择信息素浓度较高的路径

书架

中转站

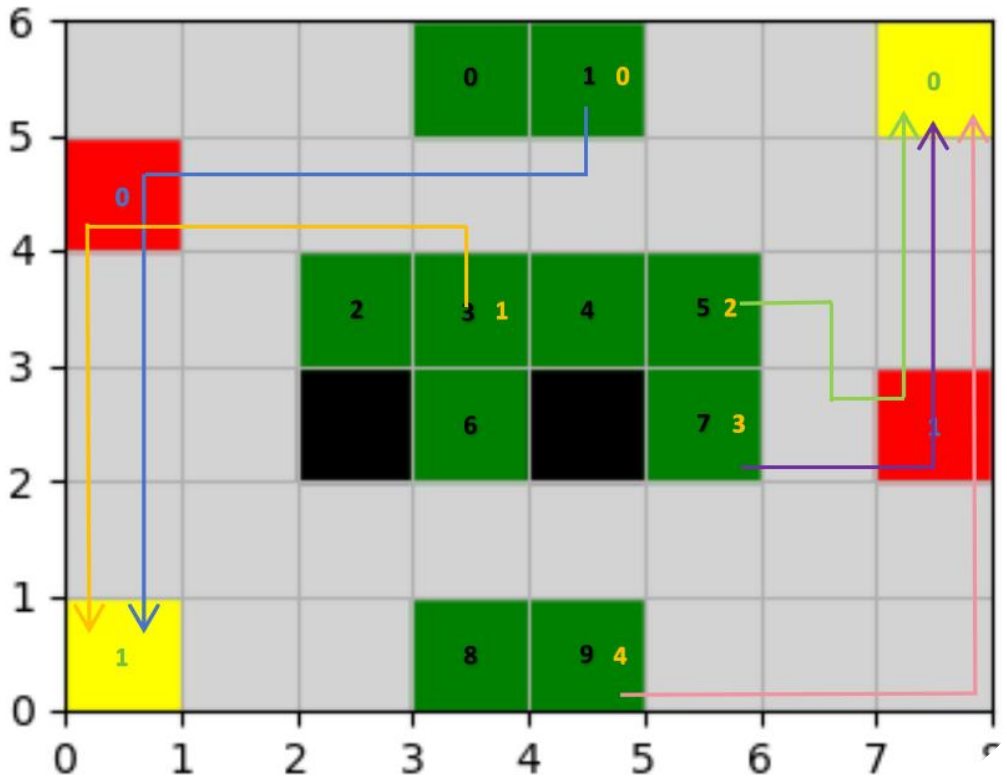
图书馆出货处

学生信箱

问题二定义

从起始位置出发，先前往储位取下装有预定订单中书籍的托盘，然后送往挑选处将订单中需要的书籍挑选出去，挑选完成之后，如果托盘中没有书籍，则前往托盘回收处，如果仍然有书籍，则将托盘送回货架上

托盘运行总路径长度最短



模型构建——4类约束

托盘内书本数量与预约订单匹配约束

托盘与书籍挑选节点的匹配

如果托盘还有书籍，则需要放到空储位节点

若托盘无书籍，则考虑托盘与回收节点的匹配

1. 托盘内书本数量与预约订单匹配约束

考虑托盘 i 在拣选工位应被挑选走的第 h 种书的数量小于第 i 得数量

考虑若托盘 i 被挑

应大于 0

考虑若托盘 i 没有

等于 0

剩从所有托盘中

盘中的书籍全部

果没有被挑选完

2. 托盘与书籍挑选节点

考虑托盘 i 至多前往一

托盘 i 前往拣选

托盘 i 从储位 q

路线长度 d_{iqm} 应

$d_{iqm} + M(3 -$

托盘 i 从储位 q

工作台 m 的行走路线长度 d

3. 如果托盘还有书籍，则需要放到空储位节点

考虑若托盘 i 没有被挑选出前往拣选工位，则托盘 i 中必有书籍存在

盘 i 在被拣选

果托盘 i 中的

一个储位

托盘 i 没有被

意一个托盘被

4. 若托盘无书籍，则考虑托盘与回收节点的匹配

考虑托盘 i 中的书籍全部被挑选走，则从托盘 i 中挑选出来的书籍数量 于托盘 i 中拥有的书籍数量总和

$$\sum_{h\in H} k_{ih} - \sum_{h\in H} O_{ih} + M(1-y_i) \geqslant 0, \forall i \in I \quad (5)$$

托盘 i 在被拣选完商品后，至多前往一个空盘回收位

$$\sum_{r\in R} f_{ir} \leq 1, \forall i \in I \quad (6)$$

考虑如果托盘 i 被挑选后没有书籍，则分配一个回收处，如果挑选后 则不分配回收处

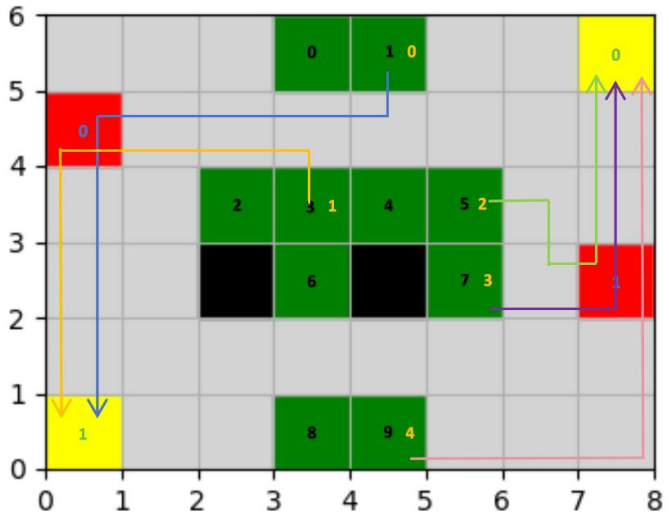
$$\sum_{r\in R} f_{ir} = y_i, \forall i \in I \quad (7)$$

托盘 i 没有被挑选出前往拣选工作台，则不需要为其分配空盘回收位

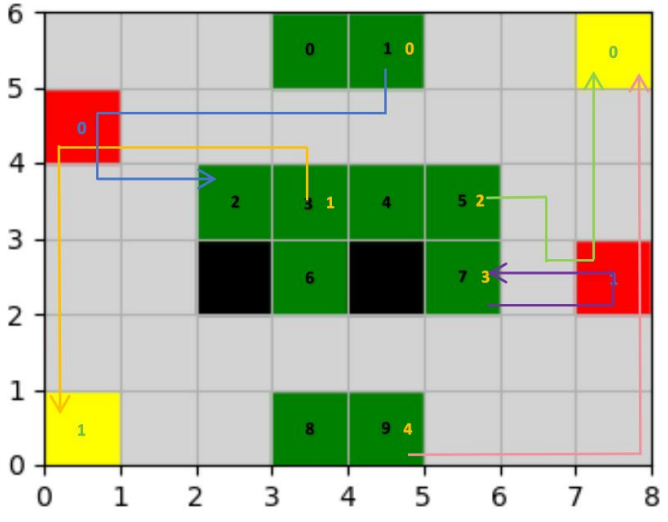
$$x_i - \sum_{r\in R} f_{ir} \geqslant 0, \forall i \in I \quad (58)$$

PULP 库结果

```
d_iqm solution:
d_iqm[0][1][0] = 5.0
d_iqm[1][3][0] = 4.0
d_iqm[2][5][1] = 3.0
d_iqm[3][7][1] = 2.0
d_iqm[4][9][1] = 5.0

d_imq solution:

d_imr solution:
d_imr[0][0][1] = 4.0
d_imr[1][0][1] = 4.0
d_imr[2][1][0] = 3.0
d_imr[3][1][0] = 3.0
d_imr[4][1][0] = 3.0
```



Example 1



```
d_iqm solution:
d_iqm[0][1][0] = 5.0
d_iqm[1][3][0] = 4.0
d_iqm[2][5][1] = 3.0
d_iqm[3][7][1] = 2.0
d_iqm[4][9][1] = 5.0

d_imq solution:
d_imq[0][0][2] = 3.0
d_imq[2][1][7] = 2.0

d_imr solution:
d_imr[1][0][1] = 4.0
d_imr[3][1][0] = 3.0
d_imr[4][1][0] = 3.0
```

Example 2

PULP 库结果

```
Objective value:                370.00000
Enumerated nodes:               15041
Total iterations:               196911
Time (CPU seconds):             21.52
Time (Wallclock seconds):       21.52

Option for printingOptions changed from n
Total time (CPU seconds):       22.07   (
```

```
x solution:
x[0] = 1.0
x[1] = 1.0
x[2] = 1.0
x[3] = 1.0
x[4] = 1.0
x[5] = 1.0
x[6] = 1.0
x[7] = 1.0
x[8] = 1.0
x[9] = 1.0

y solution:
y[0] = 1.0
y[1] = 1.0
y[2] = 1.0
y[3] = 1.0
y[4] = 1.0
y[5] = 1.0
y[6] = 1.0
y[7] = 1.0
y[8] = 1.0
y[9] = 1.0

d_iqm solution:
d_iqm[0][37][6] = 7.0
d_iqm[1][100][6] = 17.0
d_iqm[2][131][3] = 10.0
d_iqm[3][97][6] = 12.0
d_iqm[4][58][6] = 11.0
d_iqm[5][101][6] = 18.0
d_iqm[6][63][6] = 16.0
d_iqm[7][141][5] = 12.0
d_iqm[8][4][6] = 12.0
d_iqm[9][8][6] = 16.0

d_imq solution:

d_imr solution:
d_imr[0][6][1] = 3.0
d_imr[1][6][1] = 3.0
d_imr[2][3][0] = 6.0
d_imr[3][6][1] = 3.0
d_imr[4][6][1] = 3.0
d_imr[5][6][1] = 3.0
d_imr[6][6][1] = 3.0
d_imr[7][5][0] = 13.0
d_imr[8][6][1] = 3.0
```

Example 3: 所有托盘被运到挑拣处并到达回收处的总路程为 370

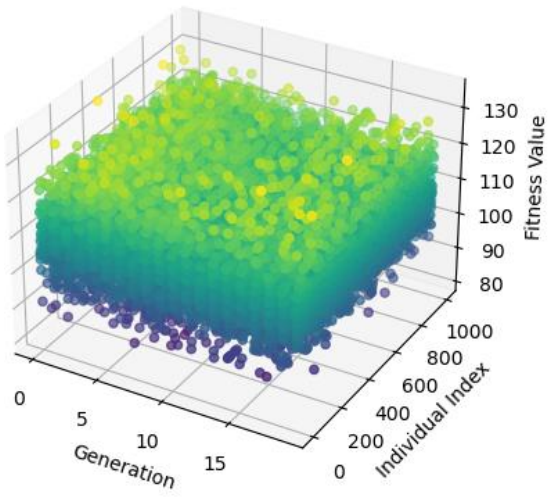
```
Objective value:                181.0000
Enumerated nodes:               214
Total iterations:               2549
Time (CPU seconds):             1.41
Time (Wallclock seconds):       1.41

Option for printingOptions changed from
Total time (CPU seconds):       1.72    (
```

```
x[0] = 0.0
x[1] = 1.0
x[2] = 1.0
x[3] = 1.0
x[4] = 1.0
x[5] = 1.0
x[6] = 1.0
x[7] = 1.0
x[8] = 1.0
x[9] = 1.0
x[10] = 1.0
x[11] = 0.0

d_iqm solution:
d_iqm[1][153][3] = 7.0
d_iqm[2][65][6] = 18.0
d_iqm[3][122][6] = 20.0
d_iqm[4][51][3] = 18.0
d_iqm[5][125][3] = 18.0
d_iqm[6][74][3] = 12.0
d_iqm[7][4][6] = 12.0
d_iqm[8][38][6] = 8.0
d_iqm[9][118][6] = 15.0
d_iqm[10][77][6] = 11.0

y solution
y[0] = 0.0
y[1] = 1.0
y[2] = 1.0
y[3] = 1.0
y[4] = 1.0
y[5] = 1.0
y[6] = 1.0
y[7] = 1.0
y[8] = 1.0
y[9] = 1.0
y[10] = 1.0
y[11] = 0.0

d_imq solution:

d_imr solution:
d_imr[1][3][0] = 6.0
d_imr[2][6][1] = 3.0
d_imr[3][6][1] = 3.0
d_imr[4][3][0] = 6.0
d_imr[5][3][0] = 6.0
d_imr[6][3][0] = 6.0
d_imr[7][6][1] = 3.0
d_imr[8][6][1] = 3.0
d_imr[9][6][1] = 3.0
d_imr[10][6][1] = 3.0
```

Example 4: 三步的距离如上图所示

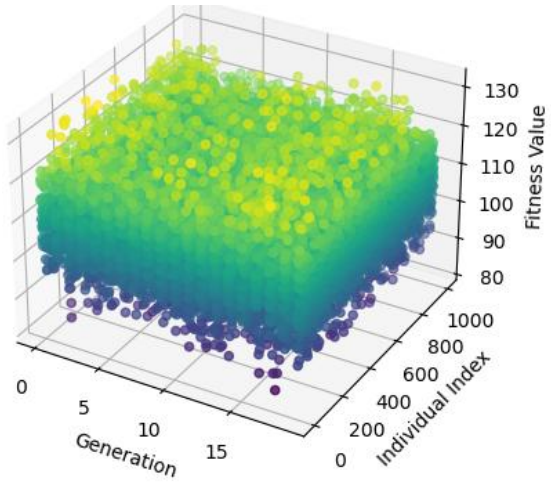
遗传算法 + Deap库

Example 1



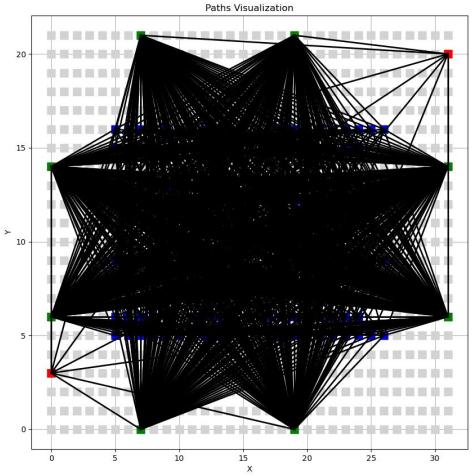
小地图
预订 = 处理

Example 2



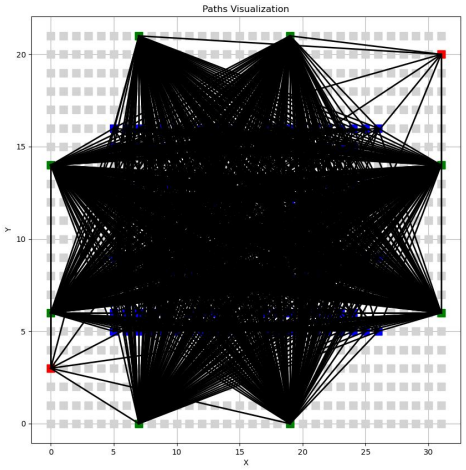
小地图
预订 > 处理

Example 3



大地图
预订 = 处理

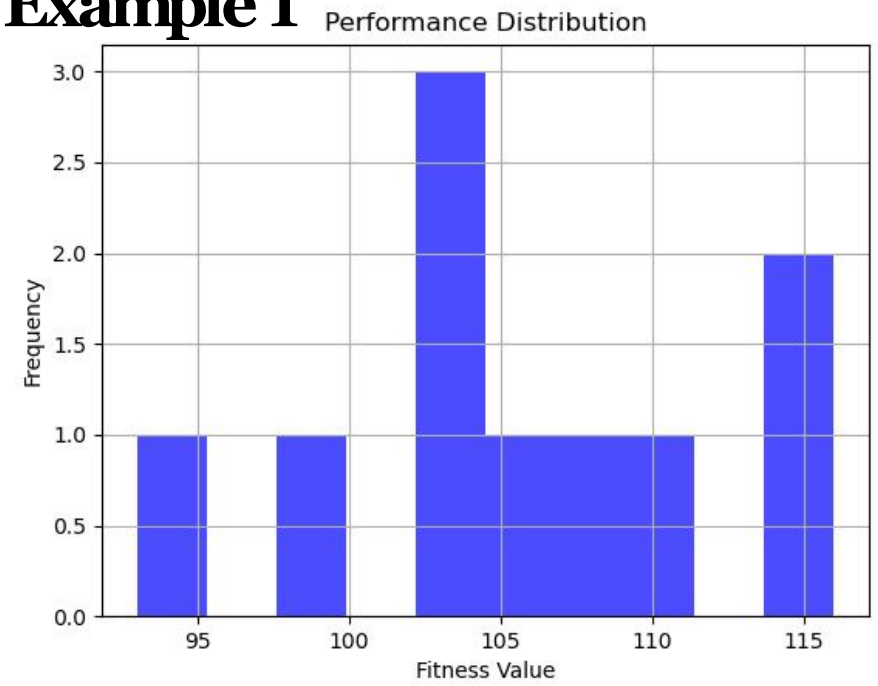
Example 4



大地图
预订> 处理

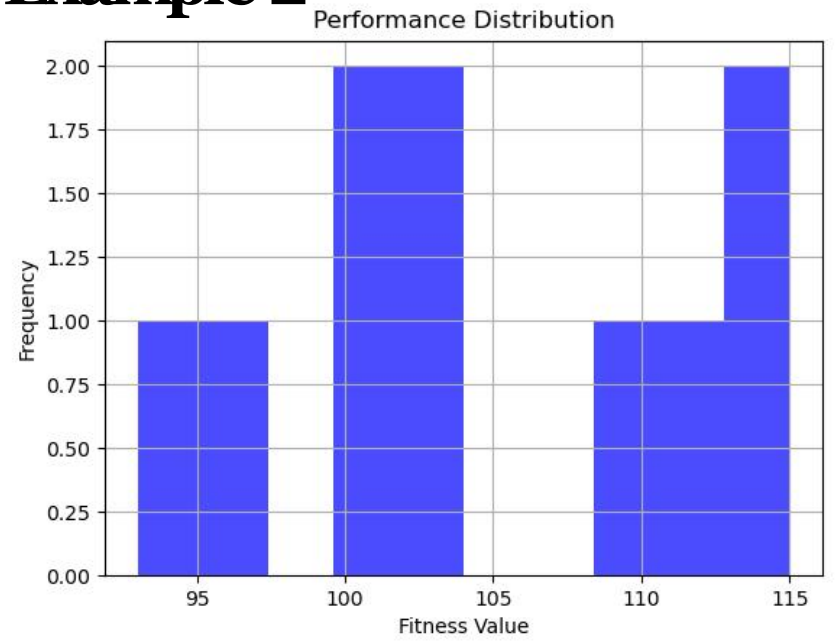
误差分析

Example 1




```
Mean: 105.8
Standard Deviation: 6.896375859826668
Best Result: 93
Worst Result: 116
```

Example 2




```
Mean: 104.7
Standard Deviation: 7.430343195303969
Best Result: 93
Worst Result: 115
```

问题三定义

小车将已经打包好的书籍

送往每一位同学的信箱中

TSP 旅行商问题

模型构建

目标函数

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$$

每个城市必须仅被访问一次

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i \in n \quad \sum_{i=1}^{n} x_{ij} = 1, \forall j \in n$$

从起点城市出发，回到起点城市

$$\sum_{i=1}^{n} x_{i1} = 1, \sum_{j=1}^{n} x_{1j} = 1$$

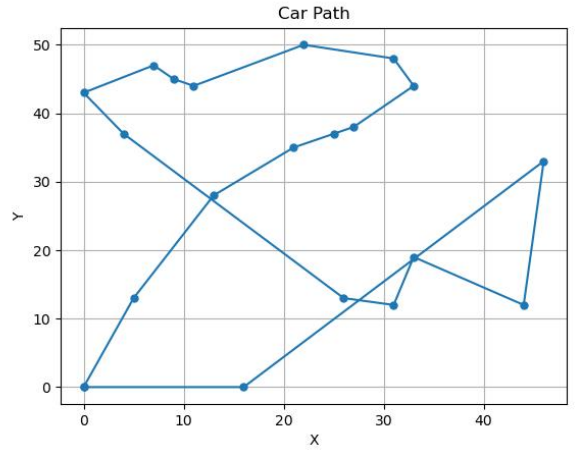
避免子回路

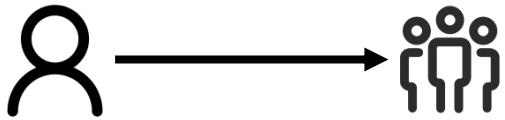
$$\sum_{i=1, i \neq j}^{n} \sum_{k=1, k \neq i, k \neq j}^{n} x_{ik} x_{kj} \leq n - 1, \forall j \in n$$

算法实现

```
最佳路径:
(0, 0) (5, 13) (13, 28) (21, 35) (25, 37) (27, 38) (33, 44) (31, 48) (22, 50)
 (11, 44) (9, 45) (7, 47) (0, 43) (4, 37) (26, 13) (31, 12) (33, 19) (44, 12)
 (46, 33) (16, 0) (0, 0)
路径长度:  242.97992405372588
```

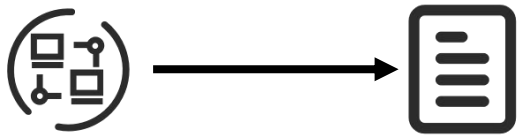
可视化结果





用户需求激增时（即点集数量很大时）

使用传统方法求解的效率和精度的balance很难平衡



为此，我们尝试使用机器学习方法对问题进行求解

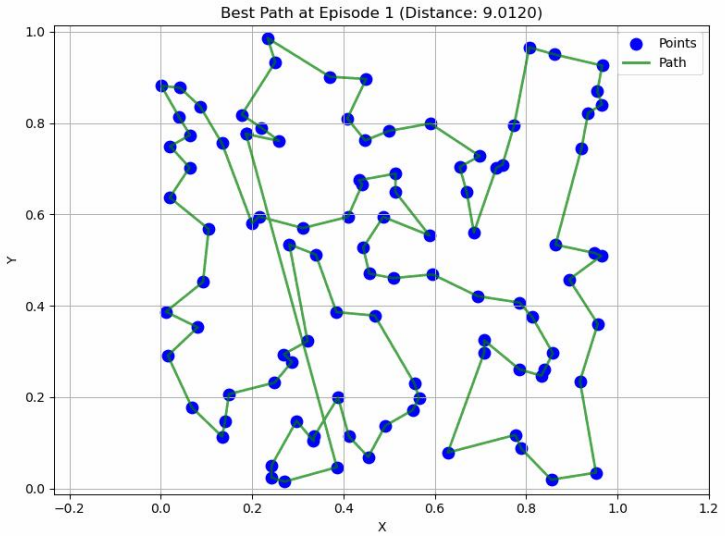
我们阅读了几篇前沿的工作，并在本问题上进行了实现。

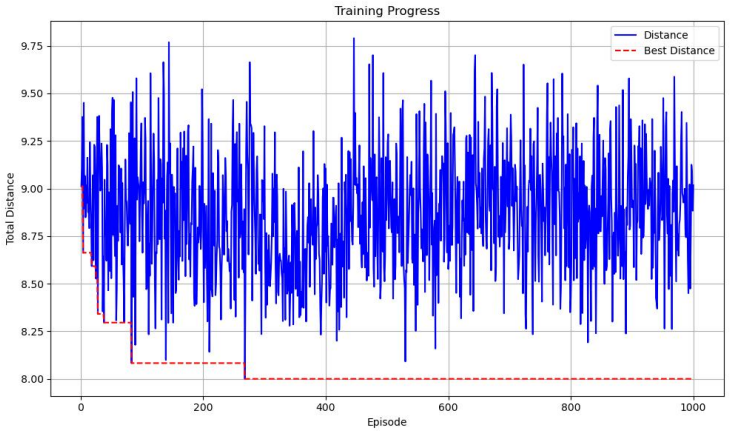
数据集的构建：
使用concorde求解，构建50、100两个规模下的TSP数据集，用于模型训练；**且为了方便训练，对坐标做了归一化**

强化学习

1. Pointer Network 架构
   - Encoder：双向 LSTM 编码所有点坐标
   - Decoder：LSTM 依次输出访问顺序
   - Pointer机制：注意力机制指向下一个访问点
2. ε-Greedy 探索策略
   - ε 从 0.8 衰减至 0.1，控制探索与利用的平衡
3. 2-Opt 路径优化
   - 使用 2-opt 算法优化解，减少交叉与锯齿，提高路径质量

训练结果

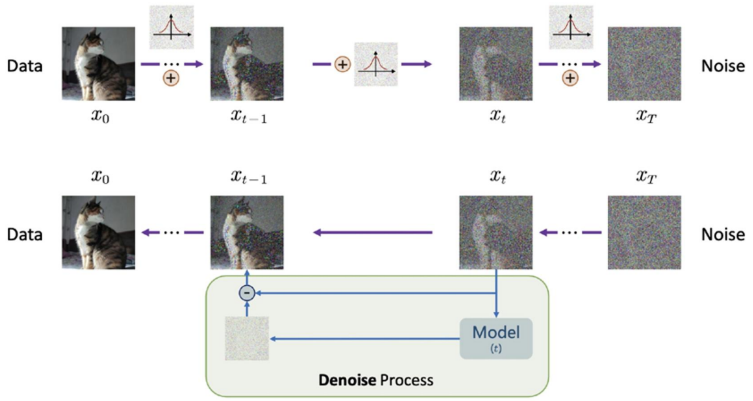




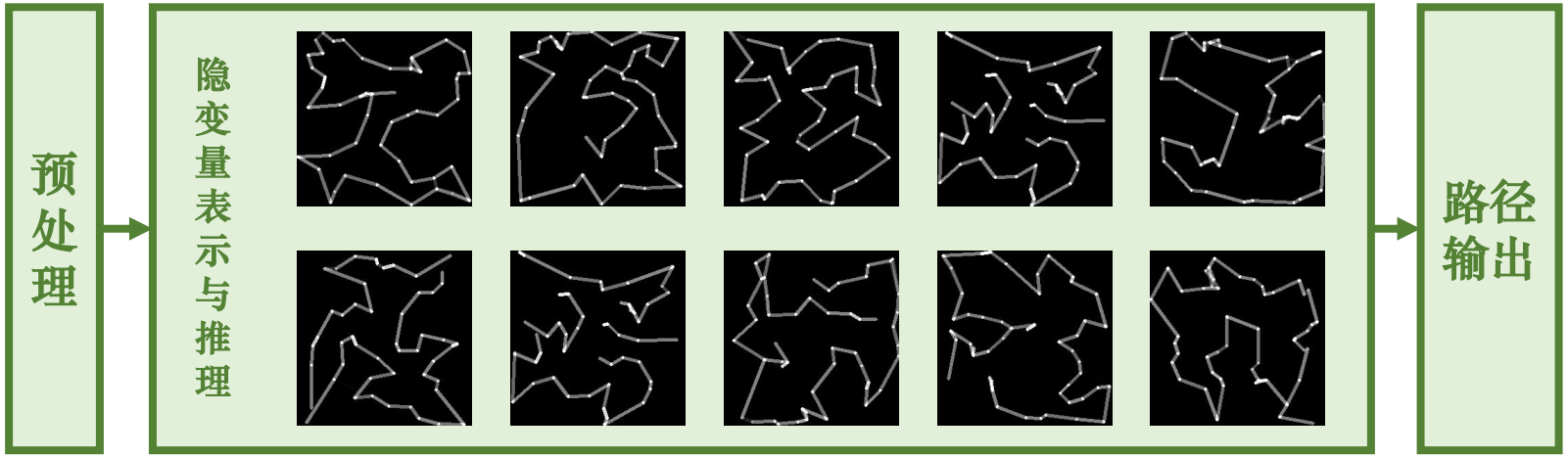
Diffusion Model

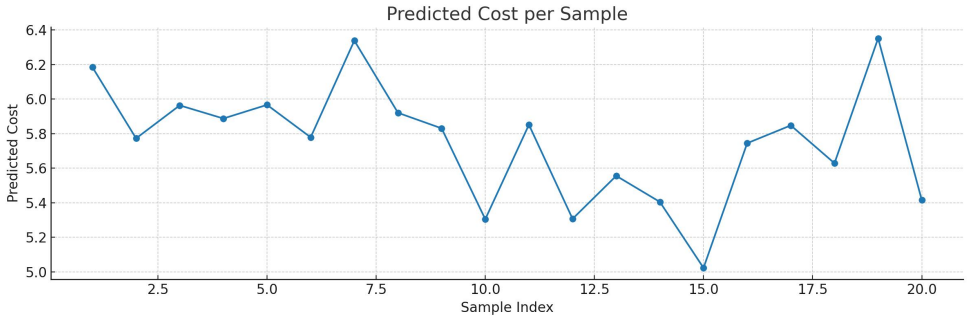
最开始由DDPM这篇论文提出，目前引用量已超2w

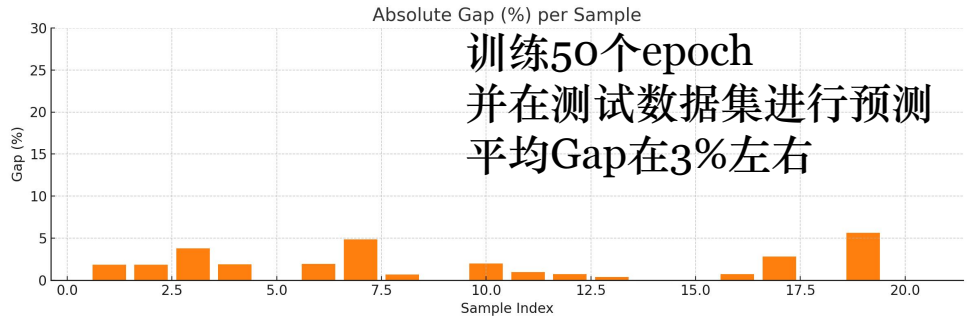
广泛用于图像生成 text2img领域，也逐渐用在LLM，机器人轨迹规划当中。



Denoising Diffusion Probabilistic Models



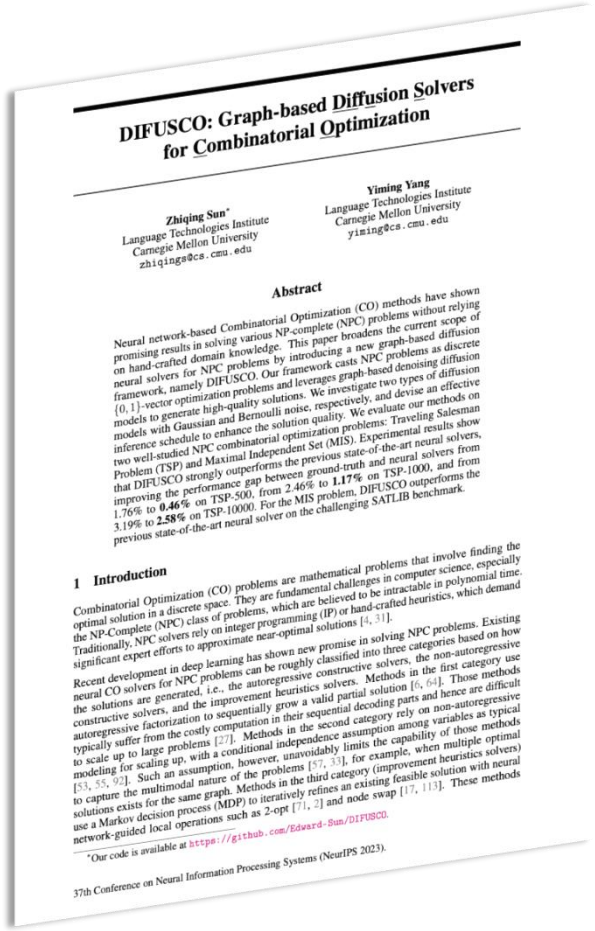




训练50个epoch
并在测试数据集进行预测
平均Gap在3%左右

并且我们发现使用TSP-50训练的模型，在TSP-100，TSP-500数据集上进行测试，仍然可以达到比较不错的效果。

经验可重复利用
机器学习的泛化性能



DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization

Zhiqing Sun*
Language Technologies Institute
Carnegie Mellon University
zhiqings@cs.cmu.edu

Yiming Yang
Language Technologies Institute
Carnegie Mellon University
yiming@cs.cmu.edu

Abstract

Neural network-based Combinatorial Optimization (CO) methods have shown promising results in solving various NP-complete (NPC) problems without relying on hand-crafted domain knowledge. This paper broadens the current scope of neural solvers for NPC problems by introducing a new graph-based diffusion framework, namely DIFUSCO. Our framework casts NPC problems as discrete $\{0,1\}$-vector optimization problems and leverages graph-based denoising diffusion models to generate high-quality solutions. We investigate two types of diffusion models with Gaussian and Bernoulli noise, respectively, and devise an effective inference schedule to enhance the solution quality. We evaluate our methods on two well-studied NPC combinatorial optimization problems: Traveling Salesman Problem (TSP) and Maximal Independent Set (MIS). Experimental results show that DIFUSCO strongly outperforms the previous state-of-the-art neural solvers, improving the performance gap between ground-truth and neural solvers from 1.76% to **0.46%** on TSP-500, from 2.46% to **1.17%** on TSP-1000, and from 3.19% to **2.58%** on TSP-10000. For the MIS problem, DIFUSCO outperforms the previous state-of-the-art neural solver on the challenging SATLIB benchmark.

1 Introduction

Combinatorial Optimization (CO) problems are mathematical problems that involve finding the optimal solution in a discrete space. They are fundamental challenges in computer science, especially the NP-Complete (NPC) class of problems, which are believed to be intractable in polynomial time. Traditionally, NPC solvers rely on integer programming (IP) or hand-crafted heuristics, which demand significant expert efforts to approximate near-optimal solutions [4, 31].

Recent development in deep learning has shown new promise in solving NPC problems. Existing neural CO solvers for NPC problems can be roughly classified into three categories based on how the solutions are generated, i.e., the autoregressive constructive solvers, the non-autoregressive constructive solvers, and the improvement heuristics solvers. Methods in the first category use autoregressive factorization to sequentially grow a valid partial solution [6, 64]. Those methods typically suffer from the costly computation in their sequential decoding parts and hence are difficult to scale up to large problems [27]. Methods in the second category rely on non-autoregressive modeling for scaling up, with a conditional independence assumption among variables as typical [53, 55, 92]. Such an assumption, however, unavoidably limits the capability of those methods to capture the multimodal nature of the problems [57, 33], for example, when multiple optimal solutions exists for the same graph. Methods in the third category (improvement heuristics solvers) use a Markov decision process (MDP) to iteratively refines an existing feasible solution with neural network-guided local operations such as 2-opt [71, 2] and node swap [17, 113]. These methods

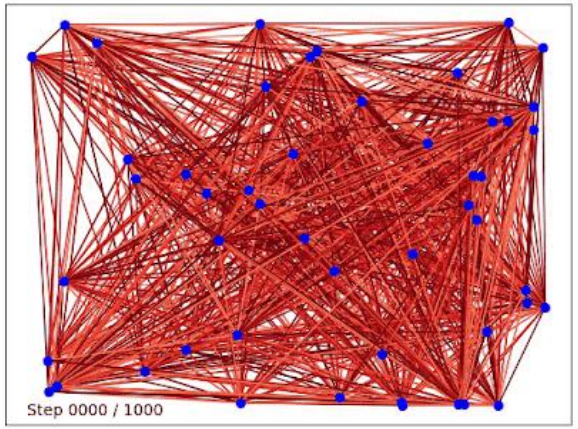
*Our code is available at https://github.com/Edward-Sun/DIFUSCO.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

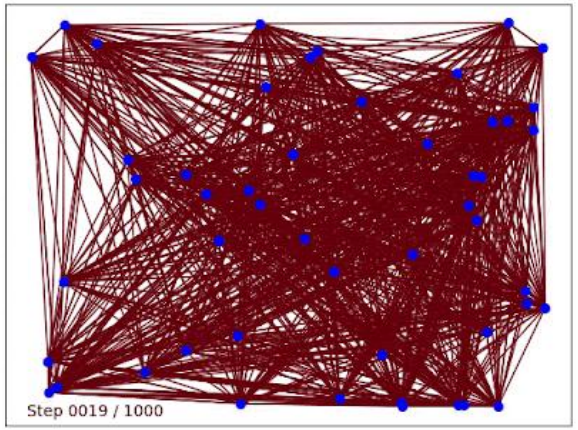
我们找到了一篇NeurIPS 2023的Best Paper
核心思想是引入图结构的扩散模型，将组合优化问题表述为 {0,1} 向量优化

我们复现这篇paper中的算法，相较于我们之前采用的ACO和遗传算法，Gap更小。

Algorithm	Type	TSP-50 Length↓	TSP-50 Gap(%)↓	TSP-100 Length↓	TSP-100 Gap(%)↓
Concorde*	Exact	5.69	0.00	7.76	0.00
2-opt	Heuristics	5.86	2.95	8.03	3.54
AM	Greedy	5.80	1.76	8.12	4.53
GCN	Greedy	5.87	3.10	8.41	8.38
Transformer	Greedy	5.71	0.31	7.88	1.42
POMO	Greedy	5.73	0.64	7.84	1.07
Sym-NCO	Greedy	-	-	7.84	0.94
DPDP	$1k$-Improvements	5.70	0.14	7.89	1.62
Image Diffusion	Greedy†	5.76	1.23	7.92	2.11
Ours	Greedy†	**5.70**	**0.10**	**7.78**	**0.24**
AM	$1k\times$Sampling	5.73	0.52	7.94	2.26
GCN	$2k\times$Sampling	5.70	0.01	7.87	1.39
Transformer	$2k\times$Sampling	5.69	0.00	7.76	0.39
POMO	$8\times$Augment	5.69	0.03	7.77	0.14
Sym-NCO	$100\times$Sampling	-	-	7.79	0.39
MDAM	$50\times$Sampling	5.70	0.03	7.79	0.38
DPDP	$100k$-Improvements	5.70	0.00	7.77	0.00
Ours	$16\times$Sampling	**5.69**	**-0.01**	**7.76**	**-0.01**

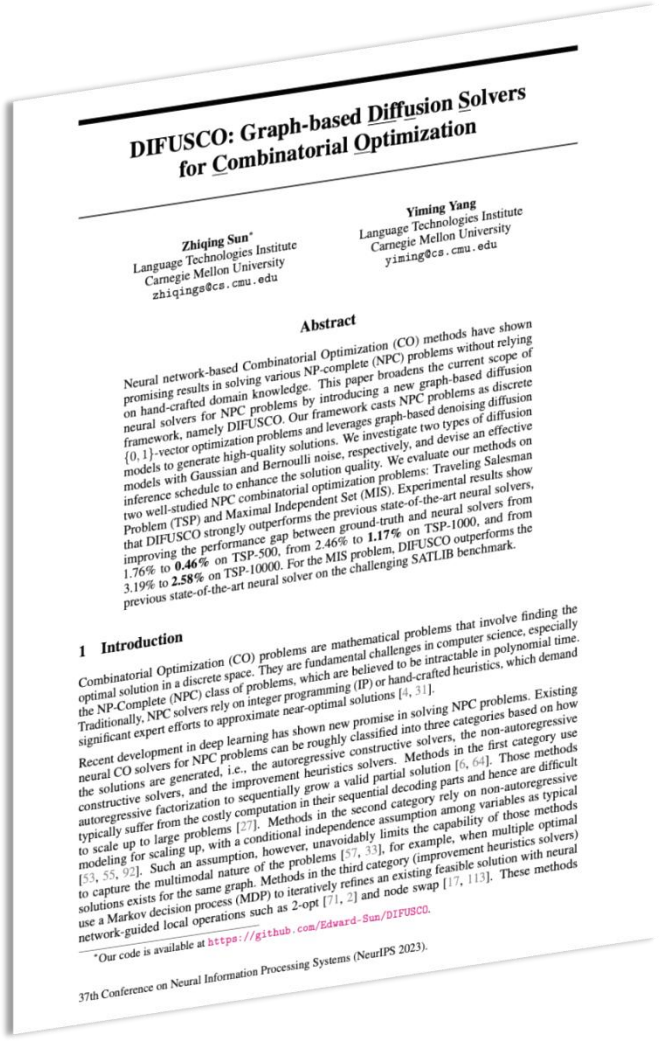


Gaussian Noise



Bernouli Noise

DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization



DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization

Zhiqing Sun*
Language Technologies Institute
Carnegie Mellon University
zhiqings@cs.cmu.edu

Yiming Yang
Language Technologies Institute
Carnegie Mellon University
yiming@cs.cmu.edu

Abstract

Neural network-based Combinatorial Optimization (CO) methods have shown promising results in solving various NP-complete (NPC) problems without relying on hand-crafted domain knowledge. This paper broadens the current scope of neural solvers for NPC problems by introducing a new graph-based diffusion framework, namely DIFUSCO. Our framework casts NPC problems as discrete $\{0,1\}$-vector optimization problems and leverages graph-based denoising diffusion models to generate high-quality solutions. We investigate two types of diffusion models with Gaussian and Bernoulli noise, respectively, and devise an effective inference schedule to enhance the solution quality. We evaluate our methods on two well-studied NPC combinatorial optimization problems: Traveling Salesman Problem (TSP) and Maximal Independent Set (MIS). Experimental results show that DIFUSCO strongly outperforms the previous state-of-the-art neural solvers, improving the performance gap between ground-truth and neural solvers from 1.76% to **0.46%** on TSP-500, from 2.46% to **1.17%** on TSP-1000, and from 3.19% to **2.58%** on TSP-10000. For the MIS problem, DIFUSCO outperforms the previous state-of-the-art neural solver on the challenging SATLIB benchmark.

1 Introduction

Combinatorial Optimization (CO) problems are mathematical problems that involve finding the optimal solution in a discrete space. They are fundamental challenges in computer science, especially the NP-Complete (NPC) class of problems, which are believed to be intractable in polynomial time. Traditionally, NPC solvers rely on integer programming (IP) or hand-crafted heuristics, which demand significant expert efforts to approximate near-optimal solutions [4, 31].

Recent development in deep learning has shown new promise in solving NPC problems. Existing neural CO solvers for NPC problems can be roughly classified into three categories based on how the solutions are generated, i.e., the autoregressive constructive solvers, the non-autoregressive constructive solvers, and the improvement heuristics solvers. Methods in the first category use autoregressive factorization to sequentially grow a valid partial solution [6, 64]. Those methods typically suffer from the costly computation in their sequential decoding parts and hence are difficult to scale up to large problems [27]. Methods in the second category rely on non-autoregressive modeling for scaling up, with a conditional independence assumption among variables as typical [53, 55, 92]. Such an assumption, however, unavoidably limits the capability of those methods to capture the multimodal nature of the problems [57, 33], for example, when multiple optimal solutions exists for the same graph. Methods in the third category (improvement heuristics solvers) use a Markov decision process (MDP) to iteratively refines an existing feasible solution with neural network-guided local operations such as 2-opt [71, 2] and node swap [17, 113]. These methods

*Our code is available at https://github.com/Edward-Sun/DIFUSCO.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

值得一提的是，论文也与多种前沿模型进行了对比，在TSP-500，TSP-1000，TSP-10000数据集上取得了SOTA的效果。这展现出Diffusion等概率模型在解决组合优化问题上的潜力

Algorithm	Type	TSP-500 Length ↓	TSP-500 Gap ↓	TSP-500 Time ↓	TSP-1000 Length ↓	TSP-1000 Gap ↓	TSP-1000 Time ↓	TSP-10000 Length ↓	TSP-10000 Gap ↓	TSP-10000 Time ↓
Concorde	Exact	16.55*	—	37.66m	23.12*	—	6.65h	N/A	N/A	N/A
Gurobi	Exact	16.55	0.00%	45.63h	N/A	N/A	N/A	N/A	N/A	N/A
LKH-3 (default)	Heuristics	16.55	0.00%	46.28m	23.12	0.00%	2.57h	71.77*	—	8.8h
LKH-3 (less trails)	Heuristics	16.55	0.00%	3.03m	23.12	0.00%	7.73m	71.79	—	51.27m
Farthest Insertion	Heuristics	18.30	10.57%	0s	25.72	11.25%	0s	80.59	12.29%	6s
AM	RL+G	20.02	20.99%	1.51m	31.15	34.75%	3.18m	141.68	97.39%	5.99m
GCN	SL+G	29.72	79.61%	6.67m	48.62	110.29%	28.52m	N/A	N/A	N/A
POMO+EAS-Emb	RL+AS+G	19.24	16.25%	12.80h	N/A	N/A	N/A	N/A	N/A	N/A
POMO+EAS-Tab	RL+AS+G	24.54	48.22%	11.61h	49.56	114.36%	63.45h	N/A	N/A	N/A
DIMES	RL+G	18.93	14.38%	0.97m	26.58	14.97%	2.08m	86.44	20.44%	4.65m
DIMES										
Ours (DIFUSCO)	SL+G†	18.35	10.85%	3.61m	26.14	13.06%	11.86m	98.15	36.75%	28.51m
Ours (DIFUSCO)	SL+G†+2-opt	**16.80**	**1.49%**	**3.65**m	**23.56**	**1.90%**	**12.06**m	**73.99**	**3.10%**	**35.38**m
EAN	RL+S+2-opt	23.75	43.57%	57.76m	47.73	106.46%	5.39h	N/A	N/A	N/A
AM	RL+BS	19.53	18.03%	21.99m	29.90	29.23%	1.64h	129.40	80.28%	1.81h
GCN	SL+BS	30.37	83.55%	38.02m	51.26	121.73%	51.67m	N/A	N/A	N/A
DIMES	RL+S	18.84	13.84%	1.06m	26.36	14.01%	2.38m	85.75	19.48%	4.80m
DIMES										
Ours (DIFUSCO)	SL+S	17.23	4.08%	11.02m	25.19	8.95%	46.08m	95.52	33.09%	6.59h
Ours (DIFUSCO)	SL+S+2-opt	**16.65**	**0.57%**	**11.46**m	**23.45**	**1.43%**	**48.09**m	**73.89**	**2.95%**	**6.72**h
Att-GCN	SL+MCTS	16.97	2.54%	2.20m	23.86	3.22%	4.10m	74.93	4.39%	21.49m
DIMES	RL+MCTS	16.87	1.93%	2.92m	23.73	2.64%	6.87m	74.63	3.98%	29.83m
DIMES	RL+AS+MCTS	16.84	1.76%	2.15h	23.69	2.46%	4.62h	74.06	3.19%	3.57h
Ours (DIFUSCO)	SL+MCTS	**16.63**	**0.46%**	**10.13**m	**23.39**	**1.17%**	**24.47**m	**73.62**	**2.58%**	**47.36**m

DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization

相似应用领域

问题1、3可以用来解决


自动化仓库中搬运机器人


自动焊接设备的焊枪移动路径


校园环卫车收垃圾路径

问题2可以用来解决


生鲜配送—— 一鸣


冷链配送——盒马


港泉宿舍搬迁