УНИВЕРСИТЕТ ИТМО
Факультет прикладной информатики

Квантовые подходы к обработке информации и искусственному интеллекту

Лабораторная работа №1
Вариант 22

«Задача нескольких коммивояжёров: минимизация времени обхода»

Выполнил:
Проскуряков Роман Владимирович, 409413

Номер группы:
К3339

Преподаватель:
Чуруксаев Иван Валерьевич

Санкт-Петербург, 2026

ЦЕЛЬ РАБОТЫ

Целью работы является исследование и практическая реализация методов решения задачи множественной задача коммивояжёра с минимизацией максимальной длины маршрута. В работе рассматриваются три метаэвристических подхода: имитация отжига (Simulated Annealing), поиск в больших окрестностях (Large Neighborhood Search) и жадный алгоритм с локальными улучшениями. Практической целью является сравнение эффективности данных методов по критерию makespan, времени выполнения и качеству решения.

ХОД ВЫПОЛНЕНИЯ

1 Формулировка задачи моего варианта

Дано:

- Множество городов: $V = \{0, 1, ..., n-1\}$
- Координаты городов: (x_u, y_u) для каждого $u \in V$
- Число агентов (коммивояжёров): m
- Расстояния между городами: $d_{uv} = \|(x_u, y_u) - (x_v, y_v)\|_2$

Требуется найти:

Разбиение городов на m циклов (маршрутов) такое, что:

1. Каждый город посещается ровно один раз
2. Каждый агент посещает хотя бы 1 город
3. Минимизируется makespan: $M = \max_{k=0,...,m-1} L_k$, где L_k — длина маршрута агента k

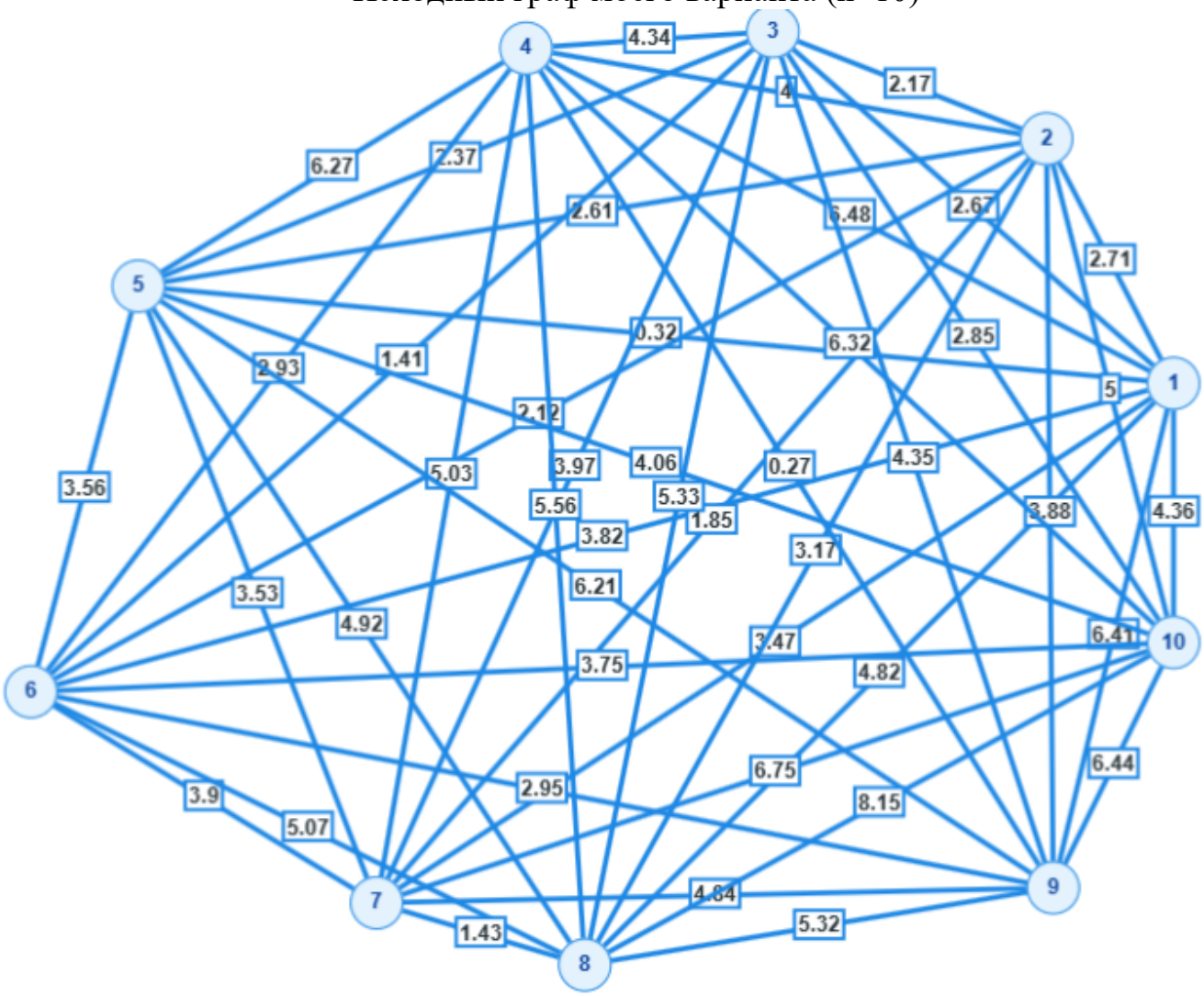
Целевая функция:

$$\min M = \min \max_{k=0,...,m-1} \left\{ \sum_{(u,v)\in \text{tour}_k} d_{uv} \right\}$$

Отличие от классической mTSP:

В классической mTSP минимизируется сумма длин всех маршрутов. В mTSP-makespan минимизируется максимальная длина среди маршрутов — это задача балансировки нагрузки между агентами.

Исходный граф моего варианта (n=10)



Число агентов (коммивояжёров): 4.

Стартовые города для агентов: 0, 2, 4, 6.

2 Математическая модель (QUBO-формулировка)

2.1. Бинарные переменные

$$x_{u,t,k} \in \{0, 1\}$$

где:

- u — номер города ($u = 0, ..., n-1$)
- t — позиция в маршруте ($t = 0, ..., L_{max}-1$)
- k — номер агента ($k = 0, ..., m-1$)

Интерпретация: $x_{u,t,k} = 1$ означает, что город u находится на позиции t в маршруте агента k.

Число переменных: $n \times L_{max} \times m$, где L_{max} — максимально возможная длина тура.

2.2. Ограничения

Ограничение 1: Каждый город посещается ровно один раз

$$\sum_{t=0}^{L_{max}-1} \sum_{k=0}^{m-1} x_{u,t,k} = 1, \quad \forall u \in V$$

Ограничение 2: В каждой позиции каждого маршрута не более одного города

$$\sum_{u=0}^{n-1} x_{u,t,k} \leq 1, \quad \forall t, k$$

Ограничение 3: Каждый агент посещает хотя бы один город

$$\sum_{u=0}^{n-1} \sum_{t=0}^{L_{max}-1} x_{u,t,k} \geq 1, \quad \forall k$$

2.3. Штрафы за нарушение ограничений

Штраф 1:

$$E_1 = A \sum_{u=0}^{n-1} \left(1 - \sum_{t,k} x_{u,t,k}\right)^2$$

Штраф 2:

$$E_2 = A \sum_{k=0}^{m-1} \sum_{t=0}^{L_{max}-1} \left(\sum_{u} x_{u,t,k}\right) \left(\sum_{u} x_{u,t,k} - 1\right)$$

Штраф 3:

$$E_3 = A \sum_{k=0}^{m-1} \max\left(0, 1 - \sum_{u,t} x_{u,t,k}\right)^2$$

2.4. Целевая функция

Длина маршрута агента k:

$$L_k = \sum_{t=0}^{L_{max}-2} \sum_{u,v} d_{uv} \cdot x_{u,t,k} \cdot x_{v,t+1,k} + \sum_{u,v} d_{uv} \cdot x_{u,L_{max}-1,k} \cdot x_{v,0,k}$$

Для минимизации времени используем аппроксимацию через штраф длинных маршрутов:

$$E_{obj} = B \sum_{k=0}^{m-1} L_k^2$$

или эквивалентно:

$$E_{obj} = B \sum_{k=0}^{m-1} \left(\sum_{t=0}^{L_{max}-1} \sum_{u,v} d_{uv} \cdot x_{u,t,k} \cdot x_{v,(t+1) \mod L_{max},k} \right)^2$$

2.5. Итоговая энергия

$$E_{total}(x) = E_1(x) + E_2(x) + E_3(x) + E_{obj}(x)$$

Задача: $\min_{x \in \{0,1\}^{n \times L_{max} \times m}} E_{total}(x)$

3 Описание реализованных алгоритмов

3.1. Имитация отжига (Simulated Annealing)

Идея: Метод глобальной стохастической оптимизации, основанный на аналогии с физическим процессом отжига металлов. Алгоритм начинает с высокой "температуры", при которой допускаются переходы в состояния с большей энергией (для исследования пространства решений и избежания локальных минимумов). Затем температура постепенно снижается, и алгоритм становится более "жадным".

Основные компоненты:

1. **Температурный график:** Управляет вероятностью принятия ухудшающих решений

$$T(t) = T_0 \cdot \alpha^t$$

   где T_0 — начальная температура, $\alpha \in (0, 1)$ — коэффициент охлаждения, t — номер итерации
2. **Критерий принятия решения:** Для перехода из состояния с энергией E в состояние с энергией E':
   - Если $\Delta E = E' - E < 0$ (улучшение) — принять всегда
   - Если $\Delta E \geq 0$ (ухудшение) — принять с вероятностью:

$$P(\text{accept}) = e^{-\Delta E / T}$$

   При высокой температуре T эта вероятность близка к 1 (принимаем почти любые ухудшения). При низкой температуре $T \to 0$ эта вероятность стремится к 0 (становимся жадными).

- **Начальная температура** T_0: Обычно подбирается так, чтобы в начале принималось около 80% ухудшающих переходов. Типичный диапазон: $T_0 \in [1.0, 3.0]$
- **Коэффициент охлаждения** $\alpha \in [0.93, 0.99]$: Чем ближе к 1, тем медленнее охлаждение. Типичные значения: 0.95-0.97.
- **Критерий остановки:** Обычно $T < T_{min}$ (например, $T_{min} = 0.001$) или фиксированное число итераций.

3.2. Поиск в больших окрестностях (Large Neighborhood Search, LNS)

Идея: Итеративный метод, на каждом шаге "разрушающий" часть текущего решения и восстанавливающий его оптимизацией. Позволяет исследовать далекие области пространства решений.

Основные компоненты:

- Фаза разрушения: случайный выбор блока городов и открепление их от текущих позиций
- Фаза восстановления: переоптимизация блока с помощью SA или локального поиска
- Критерий принятия: улучшение целевой функции

Параметры для подбора:

- Размер блока (обычно 20-30% от числа городов)
- Число итераций LNS
- Параметры внутренней оптимизации

3.3. Жадный алгоритм + локальные улучшения (Baseline)

Идея: Строит начальное решение жадным способом, затем применяет локальные улучшения (2-opt для каждого маршрута).

Этапы:

1. Сортировка пар городов по расстоянию
2. Жадное назначение городов агентам (выбирается агент с минимальной текущей длиной маршрута)
3. Применение 2-opt локального поиска к каждому маршруту отдельно
4. Попытки переноса городов между агентами для балансировки

4 Процесс подбора параметров

Параметры подбирались экспериментально на тестовых запусках:

Для SA:

T_0=400, α=0.96, Tmin=0.001, максимум итераций = 1500.

Для LNS:

Доля разрушаемых городов = 40%, итераций востановления= 30, итераций LNS = 15.

Для жадного алгоритма с локальными улучшениями: Число итераций балансировки = 70.

Параметры QUBO:

A=1000, *B*=1 (удовлетворяет условию A≫B).

5 Проведённые эксперименты

Для каждого метода выполнено по 10 запусков с фиксированными сидами (от 0 до 9). Для каждого запуска фиксировались:

- Метрика: makespan (целевая метрика)
- Метрика: общая длина
- Метрика: коэффициент баланса
- Общее время работы
- Время до первого допустимого решения
- Время до лучшего найденного решения
- Допустимость решения
- Конечное полученное решение

Далее данные всех запусков усреднялись и выводились для анализа.

РЕЗУЛЬТАТЫ

https://colab.research.google.com/drive/1vIiK45rQ0GwGlXrAgBZio4v0xUzGNoHN

1 Статистический анализ (программный вывод)

Число городов: 10

Число агентов: 4

Стартовые города: [0, 2, 4, 6]

==========

РЕЗУЛЬТАТЫ (средние значения по всем запускам)

==========

Метод: SIMULATED ANNEALING

Средний makespan: 10.6260

Минимальный makespan: 8.9700

Средняя общая длина: 33.0740

Средний баланс (стандартное отклонение): 2.0186

Среднее общее время: 0.3085 сек

Среднее время до первого допустимого: 0.0010 сек

Среднее время до лучшего: 0.2179 сек

Процент допустимых решений: 100.0%

Количество запусков: 10

Пример решения (seed=0):

Агент 0: [0, 1, 5] (длина: 8.65)

Агент 1: [2, 9] (длина: 5.70)

Агент 2: [4, 7] (длина: 9.84)

Агент 3: [6, 8, 3] (длина: 10.14)

Пример решения (seed=1):

Агент 0: [0, 1] (длина: 5.42)
Агент 1: [2, 5, 3] (длина: 8.68)
Агент 2: [4, 9] (длина: 8.12)
Агент 3: [6, 7, 8] (длина: 11.59)

Метод: LARGE NEIGHBORHOOD SEARCH

Средний makespan: 9.3000
Минимальный makespan: 8.9600
Средняя общая длина: 24.6160
Средний баланс (стандартное отклонение): 2.7586
Среднее общее время: 0.0052 сек
Среднее время до первого допустимого: 0.0000 сек
Среднее время до лучшего: 0.0007 сек
Процент допустимых решений: 100.0%
Количество запусков: 10

Пример решения (seed=0):
Агент 0: [0, 1] (длина: 5.42)
Агент 1: [2, 5, 3, 8] (длина: 8.96)
Агент 2: [4, 9] (длина: 8.12)
Агент 3: [6, 7] (длина: 2.86)

Пример решения (seed=1):
Агент 0: [0, 1] (длина: 5.42)
Агент 1: [2, 8, 3, 5] (длина: 8.96)
Агент 2: [4, 9] (длина: 8.12)
Агент 3: [6, 7] (длина: 2.86)

Метод: GREEDY WITH LOCAL SEARCH

Средний makespan: 10.8450

Минимальный makespan: 8.9600

Средняя общая длина: 30.9480

Средний баланс (стандартное отклонение): 2.7085

Среднее общее время: 0.0009 сек

Среднее время до первого допустимого: 0.0000 сек

Среднее время до лучшего: 0.0002 сек

Процент допустимых решений: 100.0%

Количество запусков: 10

Пример решения (seed=0):

Агент 0: [0, 9] (длина: 8.72)

Агент 1: [2, 7] (длина: 10.66)

Агент 2: [4, 5] (длина: 7.12)

Агент 3: [6, 1, 3, 8] (длина: 10.96)

Пример решения (seed=1):

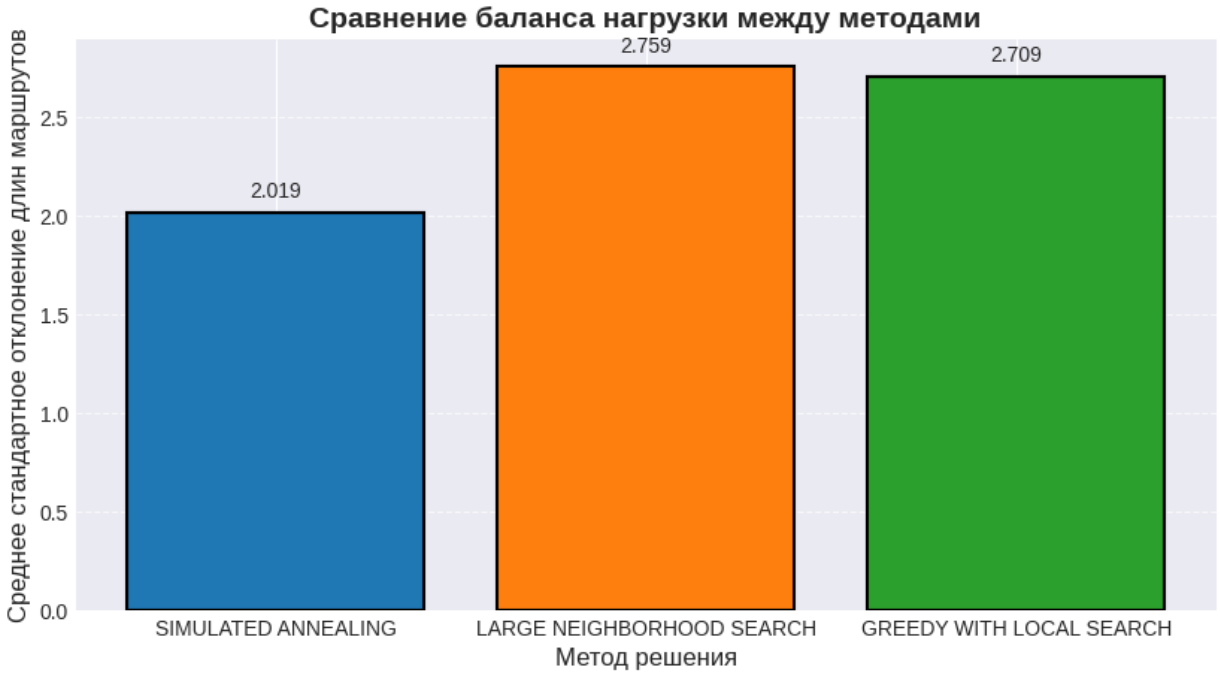
Агент 0: [0, 5, 1] (длина: 8.65)

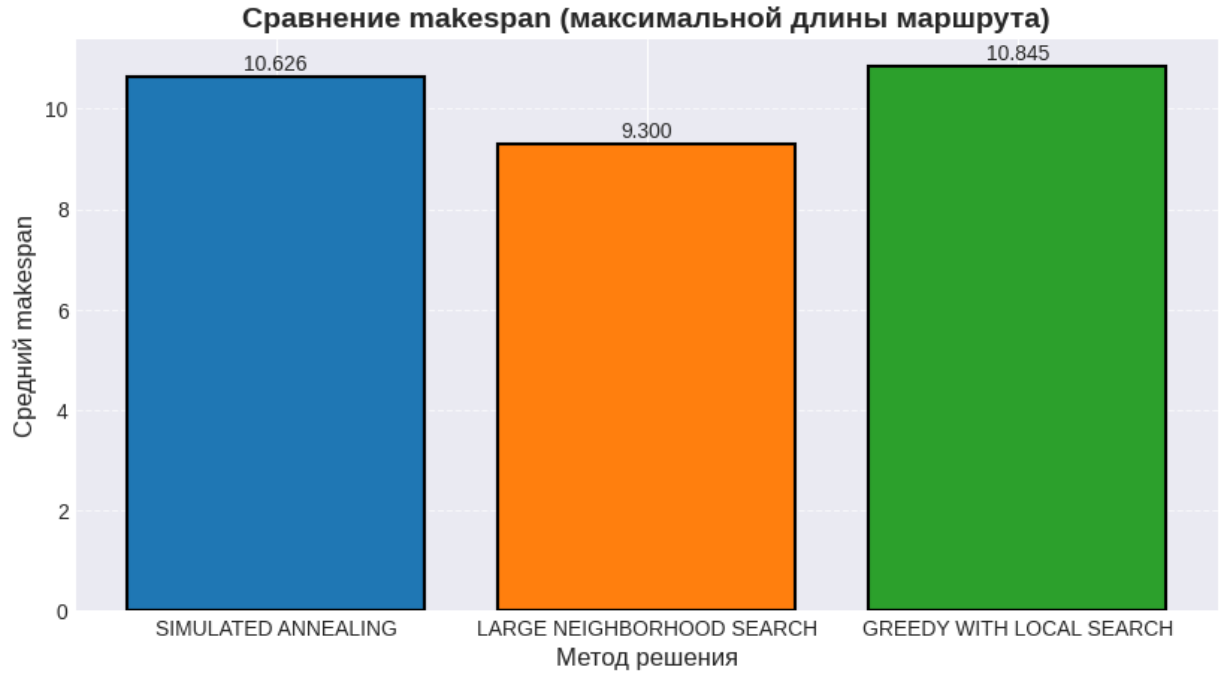
Агент 1: [2, 8, 3] (длина: 8.96)

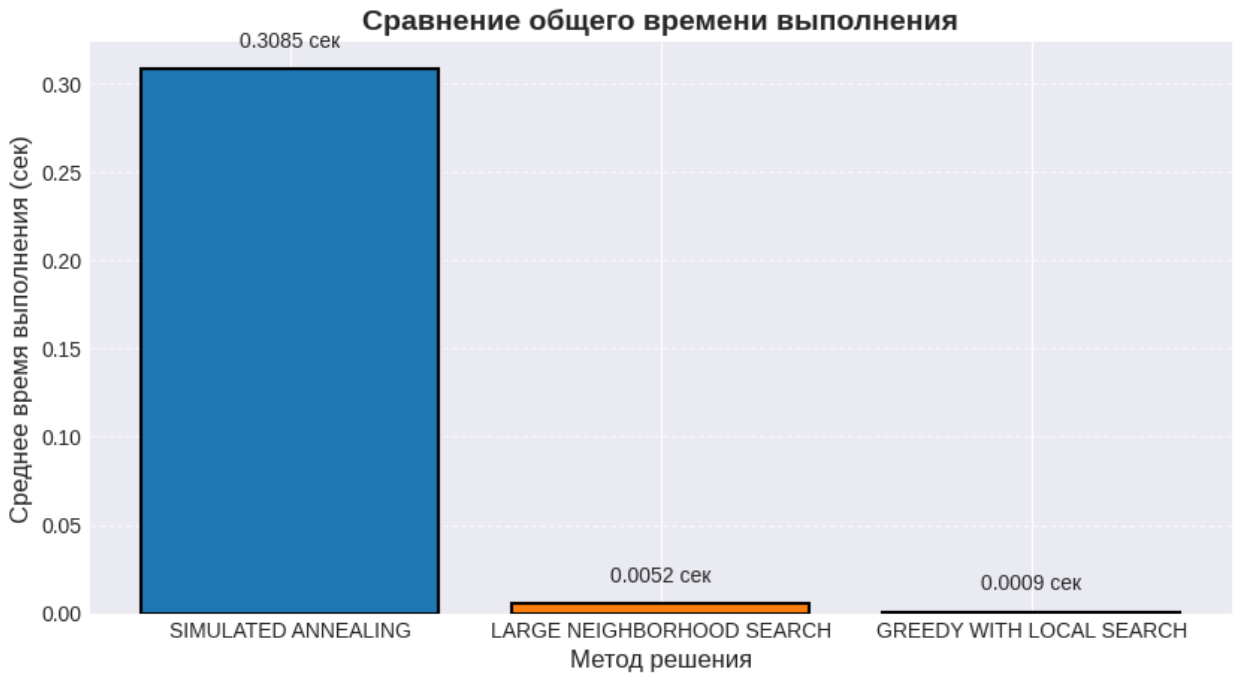
Агент 2: [4, 9] (длина: 8.12)

Агент 3: [6, 7] (длина: 2.86)

2 Сравнение эффективности методов решения







ВЫВОДЫ

Балансировка нагрузки:

Все методы обеспечили посещение всех городов без пустых маршрутов. Наилучший баланс (минимальное средние стандартное отклонение длин) достигнут методом SA и примерно равен 2. Для других методов он составляет примерно 2.7

Эффективность методов по makespan:

Все алгоритмы смогли достичь минимального **makespan** как минимум за 10 запусков. Он равен 8.96. Наилучшие средние значения makespan продемонстрировал метод LNS, что объясняется его способностью исследовать широкие окрестности и быстро избегать локальных минимумов. У него значение 9.3 против 10.6 у SA и 10.8 у жадного

Время выполнения:

Самый быстрый метод — жадный с локальными улучшениями, однако это единственный его плюс. LNS требует не сильно больше времени (примерно в 5 раз), но находит решения лучше. SA же требует гораздо больше

времени на работу. По моим данным ему нужно примерно в 1000 раз больше времени, чем жадному алгоритму.