

ЛАБОРАТОРНАЯ РАБОТА №1

Задача нескольких коммивояжёров: минимизация времени обхода

Дисциплина: Квантовые подходы к обработке информации и искусственному интеллекту

1. ПОСТАНОВКА ЗАДАЧИ

Дано:

- Множество городов: $V = \{0, 1, \dots, n - 1\}$
- 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: $M = \max_{k=0, \dots, m-1} L_k$, где L_k — длина маршрута агента k

Целевая функция:

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

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.

2. QUBO-ФОРМУЛИРОВКА

2.1. Бинарные переменные

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

где:

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

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:

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

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. МЕТОДЫ РЕШЕНИЯ

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 (принимая почти любые ухудшения).

При низкой температуре $T \rightarrow 0$ эта вероятность стремится к 0 (становимся жадными).

3. **Генерация соседних состояний:** Зависит от задачи (перестановки, перевороты битов, изменения цветов и т.д.)

Параметры для подбора:

- **Начальная температура T_0 :** Обычно подбирается так, чтобы в начале принималось около 80% ухудшающих переходов. Типичный диапазон зависит от масштаба энергии задачи.
- **Коэффициент охлаждения $\alpha \in [0.93, 0.99]$:** Чем ближе к 1, тем медленнее охлаждение и тем дольше работает алгоритм. 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. МЕТРИКИ СРАВНЕНИЯ

4.1. Основные метрики

Makespan (целевая метрика):

$$M = \max_{k=0, \dots, m-1} L_k$$

Общая длина:

$$L_{total} = \sum_{k=0}^{m-1} L_k$$

Коэффициент баланса:

$$\text{Balance} = \frac{\sigma(L_0, \dots, L_{m-1})}{\bar{L}}$$

где σ — стандартное отклонение, \bar{L} — средняя длина. Чем меньше, тем лучше балансировка.

Gap от baseline:

$$\text{Gap}(\%) = \frac{M_{method} - M_{baseline}}{M_{baseline}} \times 100\%$$

4.2. Сравнение времени выполнения

Для каждого метода измерять:

- Время до первого допустимого решения
- Время до лучшего найденного решения
- Общее время работы

4.3. Допустимость решения

Проверка выполнения всех ограничений:

- Все города посещены ровно один раз
- Нет пустых маршрутов
- Маршруты образуют корректные циклы

5. ПЛАН ЭКСПЕРИМЕНТОВ

5.1. Требования

1. Запустить каждый метод **минимум 10 раз** на вашем варианте
2. Зафиксировать random seed для воспроизводимости

3. Записать для каждого запуска:

- Makespan
- Общую длину
- Баланс (стандартное отклонение)
- Время выполнения
- Допустимость решения

5.2. Анализ результатов

Статистический анализ:

- Достигнут ли баланс нагрузки?
- Какой метод лучше по makespan?
- Какой метод быстрее?

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чём отличие минимизации makespan от минимизации суммарной длины маршрутов? Почему makespan важен в реальных приложениях?
2. Почему параметр A должен быть намного больше B ? Что произойдет, если A слишком мал?
3. Объясните, как квадратичный штраф E_1 гарантирует, что каждый город посещён ровно один раз.
4. Сколько бинарных переменных в вашей задаче? Как выбрать L_{max} ?
5. Почему аппроксимация makespan через $\sum L_k^2$ работает? Какие у неё недостатки?
6. В чём преимущество имитации отжига перед жадным поиском? Почему SA может находить лучшие решения?
7. Как поиск в больших окрестностях помогает избежать локальных минимумов? Чем он отличается от SA?
8. Опишите процесс подбора параметров A и B в вашей реализации. Какие значения вы выбрали и почему?
9. Достигается ли в вашем лучшем решении хороший баланс нагрузки между агентами? Как это измерить количественно?
10. Предложите модификацию QUBO-формулировки для случая с центральным депо (все агенты стартуют и финишируют в городе 0).

УДАЧИ В РАБОТЕ!