ЛАБОРАТОРНАЯ РАБОТА №3

Минимальное раскрашивание графа (Graph Coloring Problem)

Дисциплина: Квантовые подходы к обработке информации и искусственному интеллекту

1. ПОСТАНОВКА ЗАДАЧИ

Дано:

- Неориентированный граф $G = (V, E)$
- Множество вершин: $V = \{0, 1, ..., n-1\}$
- Множество рёбер: $E \subseteq V \times V$

Требуется найти:

Минимальное число цветов $\chi(G)$ (хроматическое число) и раскраску $c : V \to \{0, 1, ..., \chi(G)-1\}$ такую, что:

$$\forall (u, v) \in E : \quad c(u) \neq c(v)$$

Целевая функция:

$$\min \chi(G) = \min |\{c(v) : v \in V\}|$$

при ограничении отсутствия конфликтов между смежными вершинами.

Сложность: Задача является NP-полной. Даже проверка, можно ли раскрасить граф в 3 цвета, является NP-полной задачей.

Нижняя граница хроматического числа:

- $\chi(G) \geq \omega(G)$
- $\chi(G) \geq \lceil n / \alpha(G) \rceil$
- Для большинства графов: $\chi(G) \geq \Delta(G) + 1$ (максимальная степень + 1)

2. QUBO-ФОРМУЛИРОВКА

2.1. Итеративный подход к минимизации числа цветов

Вместо фиксированного числа цветов K, задачу решаем итеративно:

1. Начинаем с нижней границы $K = \omega(G)$ или $K = \Delta(G) + 1$
2. Пытаемся найти допустимую раскраску в K цветов
3. Если успешно — уменьшаем K, иначе увеличиваем
4. Возвращаем минимальное найденное K

Либо можно использовать формулы, которые мы выводили в классе.

2.2. Бинарные переменные для фиксированного K

$$x_{v,c} \in \{0, 1\}, \quad v \in V,\ c \in \{0, ..., K-1\}$$

Интерпретация: $x_{v,c} = 1$ означает, что вершина v имеет цвет c.

Число переменных: $n \cdot K$

2.3. Ограничения

Ограничение 1: Каждая вершина имеет ровно один цвет

$$\sum_{c=0}^{K-1} x_{v,c} = 1, \quad \forall v \in V$$

Ограничение 2: Смежные вершины имеют разные цвета

$$x_{u,c} \cdot x_{v,c} = 0, \quad \forall (u, v) \in E,\ \forall c$$

2.4. Штрафы

Штраф 1 (один цвет на вершину):

$$E_1 = A \sum_{v \in V} \left(1 - \sum_{c=0}^{K-1} x_{v,c} \right)^2$$

Раскрывая:

$$E_1 = A \sum_{v \in V} \left[1 - 2 \sum_{c=0}^{K-1} x_{v,c} + \sum_{c=0}^{K-1} \sum_{c'=0}^{K-1} x_{v,c} x_{v,c'} \right]$$

$$E_1 = A \sum_{v \in V} \left[1 - 2 \sum_{c=0}^{K-1} x_{v,c} + \sum_{c=0}^{K-1} x_{v,c} + 2 \sum_{c<c'} x_{v,c} x_{v,c'} \right]$$

Штраф 2 (конфликты между смежными вершинами):

$$E_2 = B \sum_{(u,v) \in E} \sum_{c=0}^{K-1} x_{u,c} \cdot x_{v,c}$$

2.5. Итоговая энергия

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

Задача для фиксированного K: $\min_{x \in \{0,1\}^{nK}} E_{total}(x)$

Критерий допустимости: $E_{total}(x^*) = 0$ означает, что граф можно раскрасить в K цветов.

3. МЕТОДЫ РЕШЕНИЯ

3.1 Degree Saturation

Идея: Жадный алгоритм, на каждом шаге красящий вершину с наибольшей "степенью насыщения" — числом разных цветов у её соседей.

Алгоритм:

1. Выбрать вершину с максимальной степенью, покрасить в цвет 0
2. Пока есть нераскрашенные вершины:
   - Для каждой вершины v вычислить $\text{saturation}(v)$ = число разных цветов у соседей
   - Выбрать вершину с максимальной степенью насыщения (при равенстве — с максимальной степенью)
   - Назначить ей минимальный допустимый цвет (не используемый соседями)
3. Вернуть раскраску и число использованных цветов

Свойства:

- Быстрый: $O(n^2)$
- Часто дает близкие к оптимальным результаты
- Детерминированный
- Используется как baseline и для получения верхней границы $\chi(G)$

3.2. 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).
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.

3.3. Backtracking с отсечениями

Идея: Рекурсивный полный перебор с ранним отсечением недопустимых ветвей.

Основные компоненты:

- Упорядочивание вершин по степени (убывание)
- Проверка допустимости на каждом шаге
- Отсечение ветвей при невозможности раскраски
- Эвристики для порядка выбора цветов

Применимость: Эффективен только для малых графов ($n \leq 20$) или графов со специальной структурой.

4. МЕТРИКИ СРАВНЕНИЯ

4.1. Основные метрики

Хроматическое число (целевая метрика):

$$\chi(G) = \min K \text{ такое, что существует допустимая раскраска}$$

Число конфликтов для фиксированного K:

$$\text{Conflicts} = \sum_{(u,v) \in E} 1[c(u) = c(v)]$$

Допустимость:

$$\text{Feasible} = 1[\text{Conflicts} = 0]$$

4.2. Отличие от baseline

Отличие от Degree Saturation:

$$\text{Gap}(\%) = \frac{\chi_{method} - \chi_{DegreeSaturation}}{\chi_{DegreeSaturation}} \times 100\%$$

(отрицательный gap означает улучшение)

Отличие от нижней границы:

$$\text{Gap}_{lower}(\%) = \frac{\chi_{method} - \omega(G)}{\omega(G)} \times 100\%$$

4.3. Качество раскраски

Баланс использования цветов:

$$\text{Balance} = \frac{\sigma(n_0, n_1, ..., n_{K-1})}{\bar{n}}$$

где n_c — число вершин цвета c, σ — стандартное отклонение.

5. ПОДБОР ПАРАМЕТРОВ

Добавление собственных модификаций

Возможные улучшения:

1. **Штраф за неиспользуемые цвета:**

$$E_{unused} = C \sum_{c=0}^{K-1} 1\left[\sum_{v} x_{v,c} = 0\right]$$

2. **Поощрение балансировки цветов:**

$$E_{balance} = D \cdot \text{Var}(n_0, ..., n_{K-1})$$

3. **Адаптивные штрафы:** Динамическое изменение A и B в процессе оптимизации

Требование: Обоснование и демонстрация улучшения.

6. ПЛАН ЭКСПЕРИМЕНТОВ

6.1. Требования

1. Запустить каждый метод **минимум 10 раз**
2. Зафиксировать random seed
3. Для каждого запуска записать:
    - Найденное хроматическое число χ
    - Число конфликтов (для проверки допустимости)
    - Время выполнения
    - Параметры A, B

6.2. Обязательные эксперименты

Эксперимент 1: Поиск χ(G)

- Применить все методы для поиска минимального числа цветов
- Сравнить найденные χ с нижней границей $\omega(G)$
- Метрики: χ, время, gap от нижней границы

Эксперимент 2: Влияние параметров

- Зафиксировать K (например, $K = \chi_{DegreeSaturation}$)
- Варьировать A и B
- Построить тепловую карту: допустимость раскраски vs (A, B)

Эксперимент 3: Сравнение с Degree Saturation

- Degree Saturation как baseline
- SA и Tabu для улучшения результата Degree Saturation
- Анализ: удалось ли уменьшить число цветов?

6.3. Анализ результатов

Обязательные элементы:

1. **Таблица результатов:**
    - Метод | χ (min/avg/max) | Время | Gap от $\omega(G)$
2. **Статистический анализ:**
    - Какой метод нашёл минимальное χ?

- Насколько близко к нижней границе?
- Баланс использования цветов

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Почему задача раскрашивания графа является NP-полной? Объясните на примере.
2. Что такое хроматическое число графа $\chi(G)$? Приведите примеры графов с известным $\chi(G)$.
3. Объясните, как квадратичный штраф E_1 гарантирует, что каждая вершина получит ровно один цвет.
4. Почему штраф за конфликты E_2 пропорционален числу рёбер? Что произойдёт при малом B?
5. Что произойдёт, если K меньше хроматического числа графа? Может ли QUBO-оптимизация это обнаружить?
6. В чём преимущество Degree Saturation перед простой жадной раскраской по степеням вершин?
7. Как Tabu Search избегает зацикливания? Объясните механизм списка запретов и критерия аспирации.
8. Опишите процесс подбора параметров A и B в вашей реализации. Какие значения вы выбрали и почему?
9. Сколько цветов потребовалось для раскраски вашего графа? Насколько это близко к нижней границе $\omega(G)$?
10. Предложите способы вычисления или оценки нижней границы $\omega(G)$ (размера максимальной клики).
11. Как можно модифицировать QUBO-формулировку для случая, когда некоторые вершины предраскрашены?

УДАЧИ В РАБОТЕ!