Применение раскраски графов для планирования работы беспроводных сетей

Кудряшова Анна
группа 371

Мотивация

На сегодняшний день беспроводные сети используются во многих сферах жизни общества: образовании, науке, медицине. Поэтому важно обеспечивать их бесперебойную и эффективную работу.

Вершинная раскраска графов применяется для распределения ограниченных ресурсов.

При работе беспроводных сетей проблема заключается в ограниченной пропускной способности узлов. По этой причине, при передаче информации в сети могут возникать помехи, что может привести к потере данных.

Для решения проблемы сосуществования, в зависимости от используемой модели распределения частоты, используют различные алгоритмы раскраски графов.

Алгоритм Луби

G -- исходный граф

$\forall$ u $\in$ V (G) $\exists$ L(u) -- список цветов. |L(u)| $\geq \Delta + 1$, где Δ -- максимальная степень в G.

На каждой итерации цикла любая вершина взаимодействует только со своими соседями.

Каждый цикл состоит из 4 шагов:

1. Любая еще не окрашенная вершина просыпается с вероятностью w и засыпает с вероятностью 1-w

Алгоритм Луби

2. Каждая бодрствующая вершина выбирает выбирает с равной вероятностью случайный цвет из своего списка L(u)

3. Каждая вершина u, чей сосед выбрал такой же цвет как и она, называется **неуспешной**; все остальные, еще не окрашенные и бодрствующие в настоящий момент вершины, называются **успешными**

4. Каждой успешной вершине v присваивается постоянный цвет t и этот цвет удаляется из L(w) для всех w смежных с v таких, что $t \in L(w)$. Неуспешные вершины обрабатываются на следующей итерации цикла.

Цикл продолжается до тех пор пока все вершины не будут окрашены.

Промежуточная модель присвоения частоты (Intermediate model)

- минимизация порога помех
- минимизация количества цветов, для которых возможно соблюдение данного порога

G -- абстрактный ненаправленный граф

$S = \{c_1 ... c_s\}$ -- спектр цветов

W -- матрица помех между каждой парой цветов (с неотрицательным значением)

$W_{ij} = W(c_i, c_j)$

Минимизации порога помех

Threshold Spectrum Coloring(TSC) -- порог спектрального окрашивания

Algorithm 1: TSC-DSATUR coloring algorithm

Input:
$G = (V, E)$: graph to be colored; $S = \{c_i\}$: spectrum of colors
W: matrix of interferences
$k \mid 2 \leq k \leq |S|$: maximum color number to be used from the spectrum
Output:
c: k-coloring of the graph G
$c(v) := \varnothing, \ \forall v \in V;$
1 **while** $\exists v \in V \mid c(v) = \varnothing$ **do**
2 $\quad v = \text{argmax}_{x \in V; c(x) = \varnothing} \ saturation\_degree(x);$
3 $\quad c(v) := \text{argmin}_{c_i \mid i \leq k} \sum_{u \in N(v); c(u) \neq \varnothing} W(c(u), c_i)$
end

Минимизация количества цветов

Алгоритм ищет цвет, не создающий в вершине помех превышающих произведение фиксированного t на долю уже окрашенных соседей. Таким образом гарантируется, что максимальный уровень помех в вершинах не превосходит t.

Chromatic Spectrum Coloring(CSC) -- хроматический спектр окраски

CSC-DSATUR

Algorithm 2: CSC-DSATUR coloring algorithm

Input:
$G = (V, E)$: graph to be colored; $S = \{c_i\}$: spectrum of colors
W: matrix of interferences
t: threshold on the maximum interference per vertex
Output:
c: coloring of the graph G, $c(v) := \varnothing,\ \forall v \in V$ if no valid coloring is found
$c(v) := \varnothing,\ \forall v \in V$;
while $\exists v \in V \mid c(v) = \varnothing$ **do**
  $v = \text{argmax}_{x \in V; c(x) = \varnothing}\ saturation\_degree(x)$
  $i = 1; I_{max} := \infty$
1 **while** $I_{max} > \frac{|\{v \in N(v) | c(v) \neq \varnothing\}|}{|N(v)|} t; i \leq |S|$ **do**
    $c(v) := c_i$
    $I_{max} = \sum_{u \in N(v); c(u) \neq \varnothing} W(c(u), c_i)$
    if $I_{max} \leq \frac{|\{v \in N(v) | c(v) \neq \varnothing\}|}{|N(v)|} t$ **then**
2 **foreach** $u \in N(v) \mid c(u) \neq \varnothing$ **do**
        $I_{max} = \sum_{w \in N(u); c(w) \neq \varnothing} W(c(w), c_i)$
        if $I_{max} > \frac{|\{v \in N(u) | c(v) \neq \varnothing\}|}{|N(u)|} t$ **then**
          break
        end
      end
    end
  end
  if $i > |S|$ **then**
    $c(v) := \varnothing,\ \forall v \in |V|$;
    break
  end
end

Помеховая модель присвоения частоты (Physical Interference Model)

- полученная информация декодирована успешно
- сеть была загружена максимально эффективно

Обычно данная проблема решается алгоритмически с помощью последовательности временных промежутков, в которых каждая связь встречается только один раз. Эту задачу можно рассматривать как раскраску графа, при которой каждая вершина раскрашивается в один цвет.

Далее рассмотрим алгоритм многоцветной раскраски, при котором связь может встречаться более одного раза (одинаковое количество для всех связей).

Обозначения

L -- множество связей

$\forall$ i $\in$ L $\exists$ {s_i -- отправляющий узел, r_i -- принимающий узел}

Величина определяющая способность r_i декодировать информацию называется коэффициент помех SINR(r_i,S).

$$\mathrm{SINR}(r_i, S) = \frac{P/d_{s_i r_i}^{\alpha}}{N + \sum_{j \in S \setminus \{i\}} P/d_{s_j r_i}^{\alpha}}$$

Обозначения

P -- сила исходящего сигнала

N -- уровень шума

$\mathbf{d_{ab}}$ -- евклидово расстояние между узлами a и b

α > 2 -- закон сокращения мощности сигнала с увеличением евклидового расстояния между узлами

S ⊂ L называется допустимым, если никакие две связи из S не имеют общего узла и SINR(i,S) ≥ β для ∀ i ∈ S, где β -- константа и β > 1

Обозначения

T -- количество допустимых подмножеств L

Связи между узлами будем считать вершинами графа и если некоторый набор связей допустим, то никакие две вершины из этого набора не являются смежными.

Последовательность допустимых наборов $\mathbf{S_1...S_T}$ задает раскраску графа в T цветов и все вершины набора S_k имеют цвет k.

Каждая связь может появиться более, чем в одном из T наборов, обозначим количество таких наборов **q**.

Каждая вершина получает q из T различных цветов, каждый из которых относится к определенному временному интервалу.

Раскраска одним цветом

R -- множество всех связей

Новые $\mathbf{S_k}$ наборы появляются до тех пор пока **R** не станет пустым.

Связь добавляется в набор, если ее добавление не нарушает условие допустимости.

Связи в **R** ранжируются согласно определенному критерию, который определяется дополнительно.

Алгоритм

1. k := 1, S_k := Ø, R := L. Порядок элементов в R согласно критерию ранжирования.
2. Если связь i ∈ R существует и S_k ∪ {i} допустимо, тогда перемещаем верхнюю связь i из R в S_k и переходим к шагу 3. Если нет, то k := k + 1, S_k:= ∅, и переходим к шагу 2.
3. Если R ≠ ∅, то переопределяем порядок в R, согласно критерию ранжирования и переходим к шагу 2.
4. T:= k и выходная последовательность S_1...S_k

MaxCRank -- пример критерия ранжирования

Основной принцип заключается в том, чтобы максимизировать количество связей в R, которые все еще имеют шанс присоединиться к текущему S_k.

Критерий ранжирования неубывающий и относится к числу связей $j \in R \setminus \{i\}$, с которыми i не может быть в одном временном интервале.

Раскраска несколькими цветами

Предлагается ограничить количество временных интервалов до T’ = T + 1 и использовать свободное время других промежутков.

Чтобы найти наибольшее q > 1, для которого T’< qT повторяем шаги 1-4 из алгоритма раскраски одним цветом и итерируем q. Причем S_k := Ø только первый раз.

В конце каждой итерации T’ обновляется и становится равно числу временных интервалов с начала работы.

Вычисляем T’/q и продолжаем итерации до тех пор, пока это отношение уменьшается.

Было найдено минимальное количество цветов для раскраски графа -- **T’**

Wireless Body Area Network (WBAN)

Не для всех типов беспроводных сетей можно использовать временные промежутки для планирования работы.

WBAN является одним из типов беспроводных сенсорных сетей. Он состоит из небольших беспроводных устройств, которые находятся вокруг человеческого тела или внутри него.

Применение WBAN:

- может предупредить по сети больницу прежде, чем у пациента случится сердечный приступ, путем слежения за показателями его состояния.
- позволяет автоматически вводить инсулин больным диабетом, как только уровень инсулина снижается

Алгоритм

Каждый WBAN это вершина в графе

k -- количество цветов, оно известно всем вершинам

Цвет представлен целым числом начиная с 1

Алгоритм состоит из двух этапов:

1. Начальный этап
2. Этап многоцветной раскраски

Начальный этап

1. Каждая вершина отправляет информацию о себе смежным с ней вершинам и получает информацию обо всех своих соседях, которые находятся на расстоянии менее 2 от нее.
2. Подсчет каждым узлом своего приоритета по таблицам 1 и 2

Table 1. Traffic Type Priority Level

Degree	Traffic Type Priority	Traffic Type
Low ↓ High	0	Background (BK)
	1	Best Effort (BE)
	2	Excellent Effort (EE)
	3	Control Load (CL)
	4	Video (VI)
	5	Voice (VO)
	6	Medical data/Network Control
	7	Emergency/Medical Event Report

Table 2. Traffic Volume Priority Level

Degree	Traffic Volume Priority	Traffic Volume/second
Low ↓ High	0	0-50 kb
	1	51-100 kb
	2	101-150 kb
	3	151-200 kb
	4	201-250 kb

Начальный этап

3. Подсчет итогового приоритета по формуле: $w_n = \frac{1}{m}\sum_{i=1}^{m}(p1_i + p2_i)$

n -- количество WBAN в сети

$p1_m$ -- приоритет типа информации (по таблице 1)

$p2_m$ -- приоритет объема информации (по таблице 2)

4. Выбор случайного цвета из палитры

5. Обмен информацией из пунктов 3 и 4 с соседями

Начальный этап

6. Если у двух соседей оказался одинаковый цвет, то узел с большим приоритетом оставляет цвет себе. Если приоритеты равные, то каждая вершина случайно выбирает целое число от 1 до 100 и вершина с большим числом оставляет цвет. Если числа равны, то промежуток удваивается и снова выбирается случайное число. Так продолжается до тех пор пока числа не станут различными.

Таким образом для каждой вершины выбирается начальный цвет.

Этап многоцветной окраски

Proposed Multi-Coloring Algorithm

k ⌙ Defined # of color; O_n ⌙ Overlapped degree

C_n ⌙ Color of node; c ⌙ Initial integer of color

value ⌙ For calculating value; n, α ⌙ Count value

Seq ⌙ Sequence of node

```
1:  C_n=c
2:  n=α=0
3:  i=0
4:  WHILE i<k DO
5:      value = c + (n*O_n)
6:      Add 1 to n
7:      IF(!(k*α < value ≤ k*(α+1)))THEN
8:        Add 1 to α
9:      ENDIF
10:     Seq[i] = value−(k*α)
11:     Add 1 to i
12: ENDWHILE
```

1. Для каждой вершины создается уникальная последовательность цветов по алгоритму представленному слева.
2. Поочередно цвета в последовательности перебираются и сообщаются соседям вместе с приоритетом. Если цвета у соседей совпали, то вершина с большим приоритетом забирает цвет, а другая вершина помечается как неокрашенная