Практика

1	Сегментація послідовності. Для бінарного зображення тексту, написаного моноширинним шрифтом, зашумленого за допомогою застосування операції XOR до кожного пікселя з випадковою величиною з розподілом Бернуллі, запропонувати ефективний алгоритм пошуку найбільш ймовірних місць пробілів. Положення пробілів мають бути рішенням Баєсової задачі розпізнавання на ланцюзі Маркова, що відповідає штрафу 1, якщо результат не співпадає з дійсним розміщенням пробілів, і 0 якщо співпадає. Апріорні ймовірності біграм та окремих символів вважати відомими.
2	Постановка та алгоритм розв'язку головоломки судоку в термінах OR-AND задачі розмітки.
3	Локально-кон'юнктивні функції: означення та приклади.
4	Застосування розв'язку MIN-MAX задачі для пошуку розв'язку MIN-SUM задачі.
5	Постановка та розв'язок задачі сегментації зображення на два класи (небо і земля) з плавною границею між сегментами.
6	Постановка та розв'язок задачі сегментації зображення на три класи (небо, земля і вода) з обмеженнями на взаємне розташвання, накладеними за допомогою введеня спеціальних міток для границь між ними.
7	Означення двовимірної контекстно-вільної граматики. Навести алгоритм перевірки належності зображення до мови, яку породжуэ граматика, в різних напівкільцях.

8	Постановка та розв'язок задачі корекції рисунку, на якому зображено бінарне сумування двох чисел, на яке накладено шум Бернуллі. Еталонні зображення цифр відомі. Для розв'язку застосувати двовимірні контекстно-вільні граматики.

OR-AND, MAX-MIN та MIN-MAX задачі розмітки

9	Зв'язок між OR-AND, MIN-MAX та MAX-MIN задачами розмітки. Як ефективно розв'язувати будь-яку з цих задач, якщо є ефективний алгоритм розв'язку лише однієї з них?
10	Поняття інваріантності та поліморфізму (для бінарного та тернарного оператора) для OR-AND, MAX-MIN та MIN-MAX задач розмітки. Показати, що інваріантність $\langle \{0,1\}, \max, \min \rangle$ задачі відносно оператора p означає інваріантність $\langle \{0,1\}, \vee, \& \rangle$ задачі відносно того ж самого оператора p. Надати визначення оператора напіврешітки, мажоритарного оператора та оператора Мальцева. Показати відшукання кращої розмітки за умови наявності відомого оператора напіврешітки, відносно якого задача інваріантна.
11	Дано задачу, що інваріантна відносно оператора **напіврешітки**. Маємо оракул, який вірно відповідає на запитання "Яка вага найкращої розмітки?". Навести ефективний алгоритм пошуку кращої розмітки та показати, що під час роботи цього алгоритму задача залишається інваріантною відносно оператора напіврешітки.
12	Дано задачу, що інваріантна відносно **мажоритарного** оператора. Маємо оракул, який вірно відповідає на запитання "Яка вага найкращої розмітки?". Навести ефективний алгоритм пошуку кращої розмітки та показати, що під час роботи цього алгоритму задача залишається інваріантною відносно мажоритарного оператора.
13	Навести алгоритм викреслювання **другого** порядку. Показати, що алгоритм не надасть результат, що гірший за справжній. Показати, що застосування алгоритму не позбавляє задачу інваріантності відносно оператора **напіврешітки**.
14	Показати, що взяття проєкції не позбавляє задачу інваріантності відносно її поліморфізму.

15	Показати, що задачу, яка інваріантна відносно **мажоритарного** оператора, можна представити у вигляді графу, а ваги дуг є проєкціями ваг цієї задачі на ці дуги.
16	Вивести алгоритм перетворення зірки на симплекс.
17	Навести алгоритм викреслювання **третього** порядку. Показати, що застосування алгоритму дає той самий результат, що й перетворення зірки на симплекс.

Сегментація послідовності. Для чорно-білого зображення монохромної строки, зашумленого за допомогою XOR до кожного піксела з випадковою величиною з розподілом Бернуллі, запропонувати ефективний алгоритм пошуку найбільш ймовірних місць пробілів. Положення пробілів мають бути рішенням Байєсової задачі розпізнавання на ланцюзі Маркова, що відповідає штрафу 1, якщо результат не співпадає з дійсним розміщенням пробілів, і 0 якщо співпадає. Апріорні ймов-ті біграм та окремих символів вважати відомими.

$x \in X = [0,1]^{n\times m}$ – мн-на вхідних зображень.

$K = \{T \in A^* \mid r(T) \in \{0,1\}^{n\times m}\}$ –

– мн-на слів, що $r(T)$ – є зображенням цього слова.

$r : A^* \to \bigcup\limits_{a \in N_0} \{0,1\}^{n\times a}$ – переводить текст у зображення

A – алфавіт з $\sqcup$-пробілом.

A^* – мн-на всіх можливих текстів, що містять тільки символи з A

Ризик: $\sum\limits_{x\in X} \sum\limits_{k\in K} p(x,k)\cdot w(k, q(x)) \longrightarrow \min\limits_{q: X\to K}$.

$w(k,\sqcup) = \#(k \neq \sqcup)$ – штрафуємо, якщо k це не пробіл.

N – шум Бернуллі, $x = r(k) \oplus N$.

$(r(k)\oplus N)\oplus r(k) = r(k)\oplus r(k)\oplus N = N$

$\Rightarrow$ виділимо шум $\Rightarrow x \oplus r(k) = (r(k)\oplus N)\oplus r(k)$

$\Rightarrow x\oplus r(k) = N$.

$$p(x\oplus r(k)) = \prod_{i=\overline{1,n}} \prod_{j=\overline{1,m}} p^{x_{ij}\oplus r(k)_{ij}} \cdot (1-p)^{1\oplus x_{ij}\oplus r(k)_{ij}}$$

$K \in \{i_1, \dots, i_l : l = \overline{0,n}\}$; T – стовпці, $|T|$ – довжина вхідного зображення $|T| = n$.

Цільова ф-ція

$$\max_{l: T\to\{0,1\}} \sum_{k\in K^T} \prod_{t\in T} p_t(k_1)\cdot \prod_{t=1}^{n} p_{t,t+1}(k_t, k_{t+1})\cdot p_{t+1}(k_{t+1}) =$$

$$= \max_{l:T\to\{0,1\}} \sum_{k\in K^T} p_1(k_1)\cdot \prod_{t=1} p_{t,t+1}(k_t,k_{t+1})\cdot p_{t+1}(k)\cdot p_{t i_1}(k_{i_1})\cdot$$

$$\cdot \prod_{t=i+1}^{n} p_{t,t+1}(k_t,k_{t+1})\cdot p_{t+1}(k_{t+1}) = \left\{\begin{matrix} l_1 - \text{позиція 1-го } ``\sqcup" \\ l_i - \text{позиція } i\text{-го } ``\sqcup" \end{matrix}\right\} =$$

Scanned with CamScanner

$$= \max_{l:T\to\{0,1\}} \sum_{k_1 \ldots k_{l_1-1}\in K} p_1(k_1)\cdot \prod_{t=1}^{l_1-1}\left[p_{tt+1}(k_t,k_{t+1})\cdot p_{t+1}(k_{t+1})\right]\cdot$$

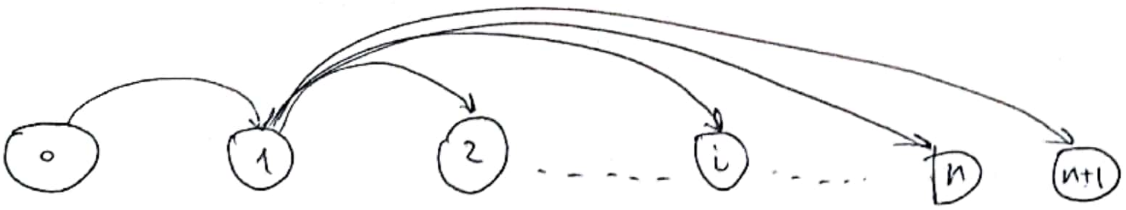
$$\ldots\ldots \underbrace{\sum_{k_{l_i+1}\ldots k_{l_{i+1}-1}\in K} \prod_{t=l_i}^{l_{i+1}-1} p_{tt}(k_t,k_{t+1})\cdot p_{t+1}(k_{t+1})}_{F_{l_i l_{i+1}}} \ldots\ldots$$

$$\ldots \sum_{k_{l_m+1},\ldots k_n\in K} \prod_{t=l_m+1}^{n-1} p_{tt}(k_t,k_{t+1})\cdot p_{t+1}(k_{t+1}) =$$

$$= \max_{\{l_1,\ldots,l_n\}} F_{0l_1}\cdot \prod_{i=1}^{m-1} F_{l_i l_{i+1}}\cdot F_{l_m l_n}.$$

$$F_{ij} = \sum_{(k_i,\ldots,k_j)\in K^{j-i}} \prod_{t=i}^{j-1}\left[p_{tt'}(k_t,k_{t+1})\cdot p_{t+1}(k+1)\right]\ ;\ i,j=1\ldots n.$$

$$G_{i+1} = \max_{j=\overline{0,i}}\left(G_j\cdot F_{j\,i+1}\right)$$



$$F_{1i}\cdot p_i(\sqcup)\, F_{in}\cdot p_n(\sqcup)\cdot F_{n\,n+1}.$$

Скл-ть n^2, якщо всі F_{ij} відомі

Скл-ть всього $k^2\cdot n^2$ - чому?

бо $k^2 n$ - F_{ij},

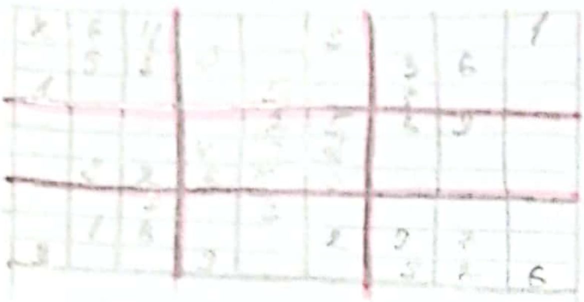
$\times$

n - $\forall i$.

Scanned with CamScanner

Постановка та алгоритм розв'язку головоломки судоку в термінах OR-AND задачі розмітки

Гра Судоку (9×9)



Ігрове поле складається з квадрата розміром 9×9, розділеного на менші квадрати 3×3 клітинки. Таким чином все поле налічує 81 клітинку. У деяких з них вже на початку гри розташовані числа (від 1 до 9). Залежно від того, скільки клітинок вже заповнено, конкретні судоку можна віднести до легких або складних.

Мета головоломки – необхідно заповнити порожні клітинки цифрами від (1 до 9) так, щоб у будь-якому рядку, стовпчику і блоці розміром 3×3 клітинки не було однакових цифр (класична постановка).

Математична постановка

81 клітинка

$K = \{1 \dots 9\}$ - множина міток.

T - множина об'єктів (клітинок), $T = \{1 \dots 9\}^{m \times n}$, $t_{ij} \in T$. клітинка

$T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9$ - розбиття мн-ни T.

Кожний T_l ; $l = \overline{1,9}$ розміром 3×3:

$\bigcup_{l=1}^{9} T_l = T$; Структура сусідства: $\tau = \{t_i t_j : \begin{cases} \exists l! : t_i^l \in T_l, t_j^l \in T_l \\ \exists l : \begin{cases} t_i^l \in T^l, t_j^l \notin T^l, t_j \in t_i \\ t_j^l \in T^l, t_i^l \notin T^l, t_i \in t_j \end{cases} \end{cases}$

всі ел-ти i-го рядка

$N_t = \{t' \in T | tt' \in \tau\}$.

Задача представляється з вигляді графу розміром m×n×, де потрібно заповнити унарні штрафи та бінарні.

В нашому випадку

$g_t(k)$ відповідає за допустимість вершини (мітки) k.

$g_{tt'}(k,k')$ відповідає за допустимість сусідства елементів t і t' з мітками k і k' відповідно

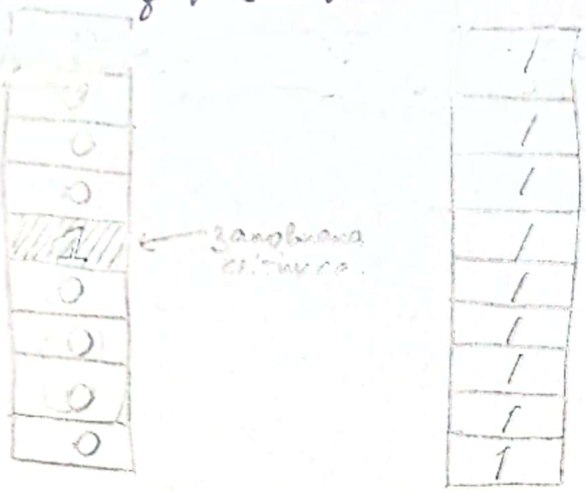
сусіди для цього елемента це всі елементи, що знаходяться в його квадраті 3×3 і на його рядку і стовпцях

Scanned with CamScanner

$q: T \times K \to \{0,1\}$; $g: T^2 \times K^2 \to \{0,1\}$; Нехай $t_{ij}=0$ де клітинка пуста.

Заповнення q

масив q розміром $m \times n \times |K|$;



$\forall t \in T, \forall k \in K.$

Якщо клітинка заповнена ($t_{ij} \neq 0$)

Якщо клітинка не заповнена $t_{ij}=0$.

Заповнення g

$\forall t \in T$ $\forall t' \in N_t$ ставимо: $g=0$

value(t) - значення клітинки в судоку.

якщо value(t) != 0

    якщо value(t') != 0 і value(t) != value(t')

        $g_{tt'}(value(t), value(t')) = 1$.

    інакше:

        $g_{tt'}(value(t), value(t)) = 0$, а решта $g_{tt'}(value(t), :) = 1$

інакше:

    якщо value(t') != 0

        $g_{tt'}(value(t'), value(t')) = 0$, а решта $g_{tt'}(:, value(t')) = 1$.

    інакше

        $g_{tt'}(:,:) = 1$

Алгоритм викреслення 2-го порядку

ініціалізація: $q^0 = q$, $g^0 = g$, $i=0$

$\forall t \in T$:

$\forall k \in K$:

$$q_t^{i+1}(k) = q_t^i(k) \,\&\, \underset{t' \in N_t}{\&} \; \bigvee_{k' \in K} g_{tt'}^i(k,k')$$ $\Big\}\; O(|T| \cdot |K| \cdot |K|)$

$\forall t \in T$:

$\forall t' \in N_t$:

$\forall k \in K$:

$\forall k' \in K$:

$$g_{tt'}^{i+1}(k,k') = g_{tt'}^i(k,k') \,\&\, q_t^{i+1}(k) \,\&\, q_{t'}^{i+1}(k')$$ $\Big\}\; O(|T| \cdot |K| \cdot |K|)$

$i \leftarrow i+1$

$q^i \neq q^{i-1} \vee g^i \neq g^{i-1}$.

інакше висновки q^i, g^i

Scanned with CamScanner

Умови самоконтролю

Беремо $t_1 \in T'$; нехай $\gamma(K)$ - вираз, що каже True або False.
таке, що value $(t_1)!=0$:

```
for k in |K|:
    if γ(K) == True:
        value'(t') = 1
    else:
        "система обмежень не допустима"
```

Якщо система обмежень допустима продовжуємо

```
for t ∈ T'\{t_1} таких, що value(t)!=0.
    for k in |K|.
        if γ(K) == True.
            value(t) = 1
        if k = |K|
            "відмова від розпізнавання"
```

не забувати перераховувати g і q.

Складність загальна: $O(|T'||\tau|\cdot|K|^3)$.

Локально-кон'юнктивні функції: означення та приклади.

До локально-кон'юнктивних функцій відносять функції такого виду:

$f(\{0,1\}^n) \to \{0,1\}$.

$f(X) = \underset{X_i}{\&} f_i(X_i)$.

$\max_i (|X_i|) < |X|$ – порядок структури

$X_1, \ldots X_k$ – структура (розбиття множини X).

$\bigcup_i X_i = X$

Scanned with CamScanner

...на кон'юнктивність наступних понять.

- монотонно зростаюча посл-ть – це послідовність елементи з збільшенням номера збільшуються.

$\{x_n\}$ – зростаюча $\Leftrightarrow \forall n \in \mathbb{N}: x_n < x_{n+1}$.

$$g_{tt'}(x(t), x(t')) = \mathbb{1}\{x(t) < x(t')\}.$$

$$g(\bar{x}) = \underset{tt' \in S}{\&} g_{tt'}(x(t), x(t')) = \underset{tt' \in S}{\&} \mathbb{1}\{x(t) < x(t')\}, \; t' = t+1.$$

^структура.

- одноекстремумна посл-ть.

$g_s(x(s)) = \{s = \{t, t', t''\}$ – будь які трійки (сусідні) $\} = \mathbb{1}\{x(t) > x(t') < x(t'')\}$

якщо посл-ть увігнута.

$g_s(x(s)) = \mathbb{1}\{x(t) < x(t') < x(t'')\}$, якщо посл-ть випукла

$$g(\bar{x}) = \underset{s \in S}{\&} g_s(x(s))$$

- лінійно змінююча (зростаюча або спадаюча) послідовність $\bar{x}$, така, що $x(i) = ai + b$; a, b – цілі числа.

$g_s(x(s)) = g_{t_1 t_2}(x(t_1), x(t_2)) : \mathbb{1}\{x(t_2) - x(t_1) = a\}$?

$$g(\bar{x}) = \underset{s \in S}{\&} g_s(x(s))$$

- квадратична послідовність $\bar{x}$, така, що $x(i) = a i i + b \cdot i + c$; a, b, c – цілі числа.

Потрібно перевірити, що посл-ть має лише один екстремум.

$(a > 0)$ $(a < 0)$

← будь-які трійки.

$f_s(x(s)) = f_{tt't''}(x(t), x(t'), x(t''))_s$

$= \mathbb{1}\{a > 0\} \cdot \mathbb{1}\{x(t) < x(t') > x(t'')\} +$

$+ \mathbb{1}\{a < 0\} \cdot \mathbb{1}\{x(t) > x(t') < x(t'')\}$

$t = -2 \Rightarrow 4 \cdot a - 2b + c$
$t = -1 \Rightarrow a - b + c$
$t = 0 \Rightarrow c$
$t = 1 \Rightarrow a + b + c$

$\} - \Rightarrow -3a + b$; $\} - \Rightarrow -a + b$; $\} - \Rightarrow a + b$

$\} - \Rightarrow -a + b + 3a - b = 2a$.

$\} - \Rightarrow a + b + a - b = 2a$

$$[x(t+2) - x(t+1)] - [x(t+1) - x(t)] = 2a.$$

$$g_{s'}(x(s')) = \{s' = \{t, t+1, t+2\}\} = \mathbb{1}\{[x(t+2) - x(t+1)] - [x(t+1) - x(t)] = 2a\}$$

$$g(\bar{x}) = \underset{s \in S}{\&} f_s(x(s)) \; \& \underset{s' \in S}{\&} g_{s'}(x(s'))$$

$x(t+2) - 2x(t-1) + x(t)$.

– поліноміальна послідовність.

$x(t) = at^3 + bt^2 + c\cdot t + d.$

$x(-2) = -8a + 4b - 2c + d.$

$x(-1) = -a + b - c + d.$

$x(0) = d$

$x(1) = a + b + c + d$

$x(-1) - x(-2) = 7a - 3b + c \quad (1)$

$x(0) - x(-1) = a - b + c \quad (2)$

$x(1) - x(0) = a + b + c \quad (3)$

$(2) - (1) = -6a + 2b \quad (I)$

$(3) - (2) = 2b \quad (II)$

$(II) - (I) = 6a$

$x(t) = at^4 + bt^3 + ct^2 + dt + e \Rightarrow 24a$

$x(t) = at^5 + bt^4 + ct^3 + dt^2 + e\cdot t + f \Rightarrow 120a.$

$n=1 \to a$

$n=2 \to 2a$

$n=3 \to 6a.$

$n=4 \to 24a$

$n=5 \to 120a.$

$\ldots$

$\underline{n=3} \Rightarrow g_s(x(s)) = \{ s = \{t, t+1, t+2, t+3\} \} = 1\!\!1\{x(t+3) - 3x(t+2) + 3x(t+1) - x(t) = 6a\}.$

$\underline{n=4} \Rightarrow g_s(x(s)) = \{ s = \{t, t+1, t+2, t+3, t+4\} \} = 1\!\!1\{x(t+4) - 4x(t+3) + 6x(t+2) - 4x(t+1) + x(t) = 24a\}.$

↑ трикутник Паскаля.

$$g(\bar{x}) = \mathop{\&}_{s \in S'} g_s(x(s))$$

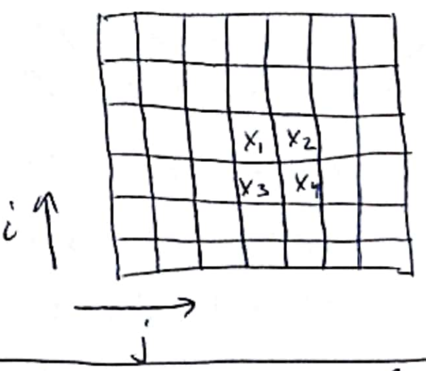
– двовимірний масив X, такий, що $x(i,j) = ai + bj + c$; a, b, c – цілі числа.

Потрібно розглянути всі можливі трійки по вертикалі та горизонталі

$g_s(x(s)) = 1\!\!1\{x_2 - x_1 = x_4 - x_3\}$ $1\!\!1\{x_3 - x_1 = x_4 - x_2\}$ – тільки на границі.

$x^{в}(i,j) = x(i+1,j) - x(i,j)$

$x^{г}(i,j) = x(i,j+1) - x(i,j)$



Для наступних прикладів функцій покажемо, що вони не є лок. кон'юнктивними на структурі будь-якого порядку. У всіх прикладах мова йде про посл-ті фіксованої довжини, що побудовані із 0 та 1.

– функція приймає значення 0 на деяких посл-ті і 1 на решті посл-ті.

$X = \{0,1\}$, $\bar{x} = T \to X.$

$\varphi(\bar{x}) = 1\!\!1\{ y_1 = 1\!\!1\{x_1 \neq x_1^*\}\} \,\&\, \mathop{\&}_{i=\overline{2,n}} 1\!\!1\{y_i = y_{i-1} + 1\!\!1\{x_i \neq x_i^*\}\} \,\&$

$\& \; 1\!\!1\{y_n \neq 0\}.$

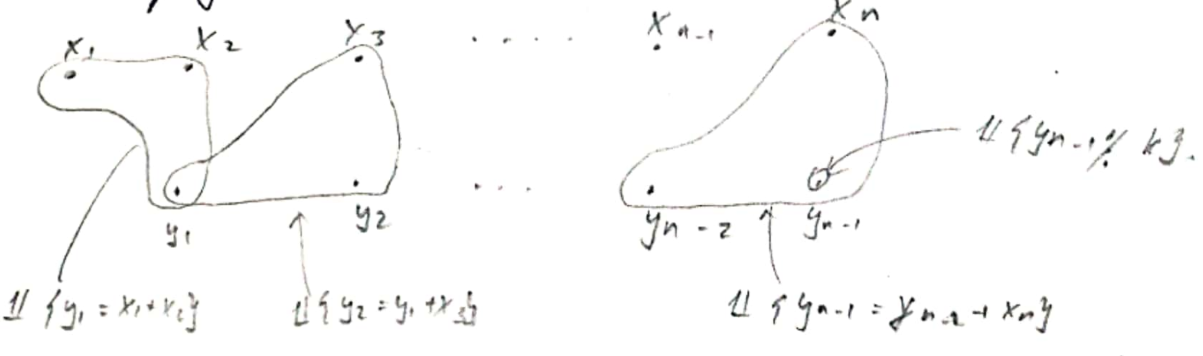
Scanned with CamScanner

...имо недопустиме заповнення $\overline{Y}$, а саме таке, що =
допустимої посл-ті $\varphi(\overline{X}^*) = \underset{s \in S}{\&} \varphi_s(x^*(s)) = 0$

Має існувати таке віконце s^*, де $\varphi_{s^*}(x^*(s^*)) = 0$
Ту комірку, що не входить до s^* позначимо за t''
Розглянемо другу розмітку $\overline{X}'$, таку, що $x'(t) = x^*(t) \ \forall t \neq t^*$,
а $x'(t'') = \overline{x^*(t^*)}$ - інвертоване значення старої розмітки
Розмітка $\overline{X}'$ - допустима допустима, так як відрізняється від
єдиної недопустимої розмітки лише в одній комірці, але
на цій розмітці число $\varphi_{s^*}(x(s^*))$ залишається $= 0$, оскільки
значення в комірці не змінилось

- функція приймає значення 0 на всіх послідовностях в яких кількість одиниць кратно заданому числу, і приймає значення 1 на всіх інших посл-тях.
Одну із таких ф-цій розділяють всі посл-ті з непарною кіл-тю одиниць і забороняють всю решту.

Розглядаємо недопустиме заповнення, а саме таке де кіл-ть одиниць = заданому числу. $\varphi(\overline{X}^*) = \underset{s \in S}{\&} \varphi_s(x^*(s)) = 0$.
Має існувати таке віконце s^*, де $\varphi_{s^*}(x^*(s^*)) = 0$
Ту комірку, що не входить в s^*, ~~об~~ позначимо t^*
Розглянемо другу розмітку $\overline{X}'$, таку, що $x'(t) = x^*(t)$, для
всіх $t \neq t^*$, а $x'(t^*) = \overline{x^*(t^*)}$ - інв. значення старої розмітки.
Від цього розмітка із недопустимої стала допустимою, але
на цій розмітці число $\varphi_{s^*}(x'(s^*)) = 0$



$\varphi(\overline{x}) = 1\!\!1\{y_1 = x_1 + x_2\} \ \& \underset{i=2,n-1}{\&} 1\!\!1\{y_i = y_{i-1} + x_{i+1}\} \ \& \ 1\!\!1\{y_{n-1} \% k\}$
або
$1\!\!1\{y_{n-1} \% 2\}$

Scanned with CamScanner

– Функція забороняє всі посл-ті, в яких к-ть од-иць = заданому числу, і ~~забороняє~~ дозволяє всю решту посл-тей. Дана ф-я забороняє посл-ті, які ~~забороняють~~ містять одну і тільки одну одиницю

Аналогічно до попереднього випадку, але

$$\varphi(\bar{x}) = \mathbb{1}\{y_1 = x_1 + x_2\} \,\&\, \underset{i=2,n-1}{\&} \mathbb{1}\{y_i = y_{i-1} + x_{i+1}\} \,\&\, \underline{\mathbb{1}\{y_{n-1} \neq k\}}$$

або $\mathbb{1}\{y_{n-1} \neq 1\}$

Scanned with CamScanner

Застосування розв'язку MIN-MAX задачі для пошуку розв'язку MIN-SUM задачі.

Розмітка для MIN-Sum і MAX-SUM задач.

Позначимо g^* - найкращі дуги: $g^*_{tt'} = \begin{cases} \max\limits_{x, x' \in K} g_{tt'}(x, x'), & \text{якщо } \langle \max, + \rangle \\ \min\limits_{x, x' \in K} g_{tt'}(x, x'), & \text{якщо } \langle \min, + \rangle \end{cases}$

Labeling relaxation:

$$k^* = \arg \bigvee_{k: T \to K} \& \left[\!\left[g_{tt'}(k_t, k_{t'}) = g^*_{tt'} \right]\!\right]$$

рішень може і не бути

ε - хороша розмітка

$$k^*(\varepsilon) = \arg \bigvee_{k: T \to K} \&_{tt' \in \tau} \left[\!\left[|g_{tt'}(k_t, k_{t'}) - g^*_{tt'}| \le \varepsilon \right]\!\right], \ \varepsilon \in \mathbb{R}^+$$

визначено нами.

Якщо для задачі z існує ε - хороша розмітка, то задача наз-ся ε - узгодженою

Пошук кращої ε - хорошої розмітки

1) Знайти кращі дуги $g^*_{tt'}$, $\forall tt' \in \tau$.
2) Знайти всі можливі епсилон: $E = \{ |g_{tt'}(x, x') - g^*_{tt'}| : tt' \in \tau, x \in K, x' \in K \}$.
3) Сортуємо всі $\varepsilon \in E$ в порядку зростання:
$$0 < \varepsilon_0 < \varepsilon_1 < \dots < \varepsilon_n = \max_{\substack{tt' \in \tau \\ x, x' \in K}} |g_{tt'}(x, x') - g^*_{tt'}|$$
4) Двійковим пошуком знаходимо таке ε_i, що розмітка $k^*(\varepsilon_i)$ допустима, але при цьому $k^*(\varepsilon_{i-1})$ недопустима,
5) $k^*(\varepsilon_i)$ - відповідь.

Вирішуються задачі:

$$k^* \in \arg\min_{k: T \to K} \max_{tt' \in \tau} |g_{tt'}(k_t, k_{t'}) - g^*_{tt'}|, \quad \varepsilon = \min_{k: T \to K} \max_{tt' \in \tau} |g_{tt'}(k_t, k_{t'}) - g^*_{tt'}|$$

Scanned with CamScanner

Постановка та розв'язок задачі сегментації зображення на два класи (небо і земля) з плавною границею між сегментами.

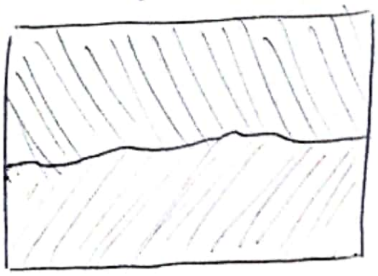
$T = \{0, 255\}^{m \times n}$ - зображення.

$K = \{0, 1, 2\}$ - небо, границя, земля.

Вибір функції q:

1. Відстань до кольору (зелений / блакитний).
2. EM для пошука параметрів розподілів „трави" і „неба"
   a. Суміш Гаусових розподілів для кожного з класів.
   b. Попередні параметри + донавчання на контексті зображення

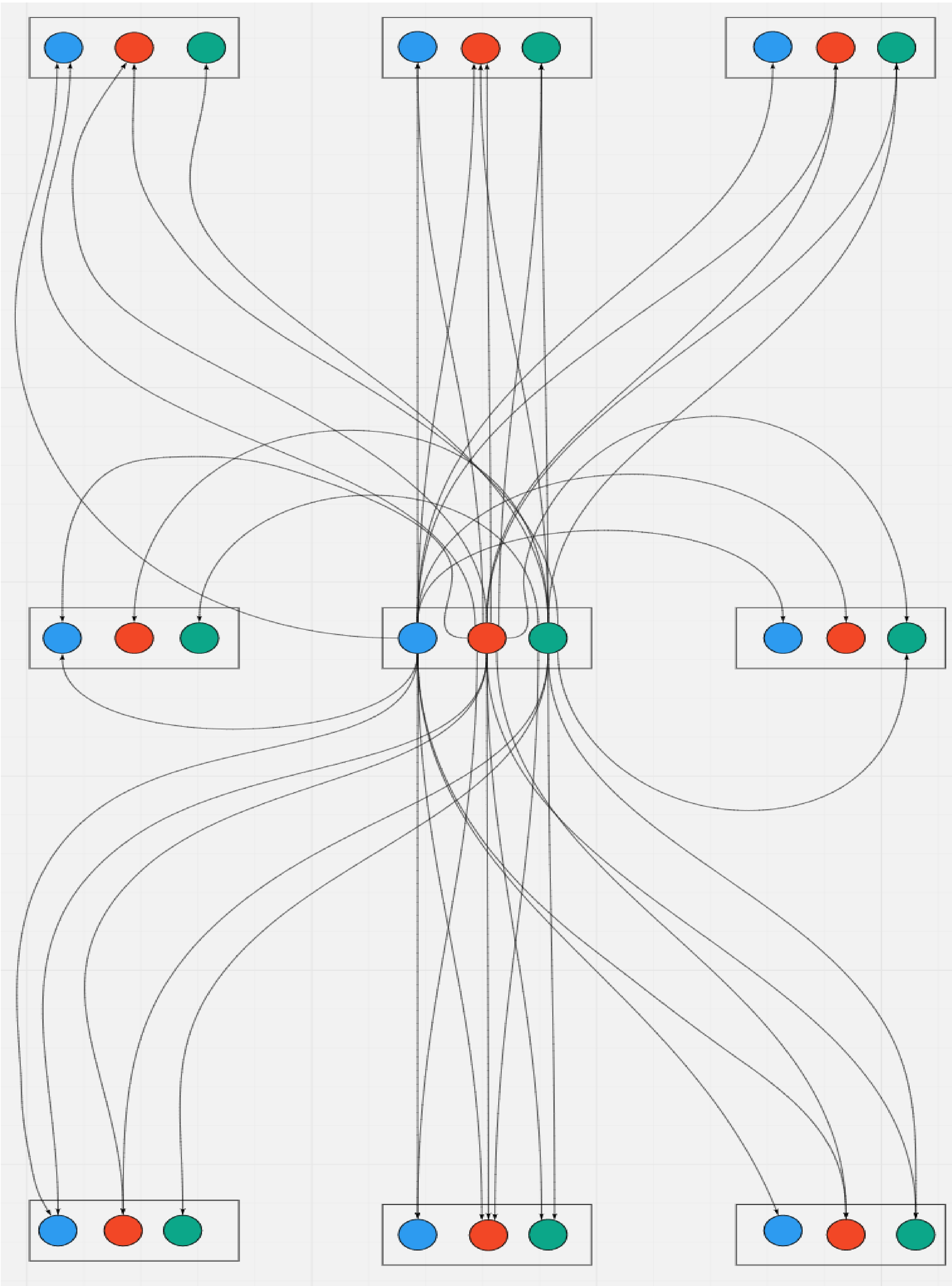
Головна задача: сегментувати зображення на два класи.



Задання дуг (на наступній сторінці).

Алгоритм:

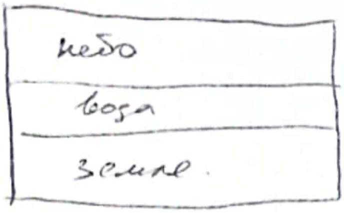
1. Отримати g
2. Отримати q з EM (розподілів.)
3. Провести n_iter ітерацій дифузії.
4. Отримати репараметризовані φ з дифузії.
5. Отримати q-rep та g-rep на основі φ.
6. Використанням знайти необхідний epsilon використавши двійковий (бінарний) пошук.
   Тут необхідно знаходити кожен раз q-binary та g-binary і дивитись чи консистентна задача при певному epsilon.
7. Результуючі q-binary та g-binary даємо на вхід самоконтролю
8. Отримаємо сегментацію.



Rotate(green - up, red - middle, blue - bottom)

Постановка та розв'язок задачі сегментації зображення на три класи (небо, земля, вода) з обмеженнями на взаємне розташування, накладеними за допомогою введення спеціальних міток для границь між ними.

$\Pi = \{0, \ldots, 255\}^{m \times n}$ - зображення. , K = {земля, вода, небо, границя між небом, між водою}



Головна задача: сегментувати зображення разом з границями.

Вибір ф-ії q:

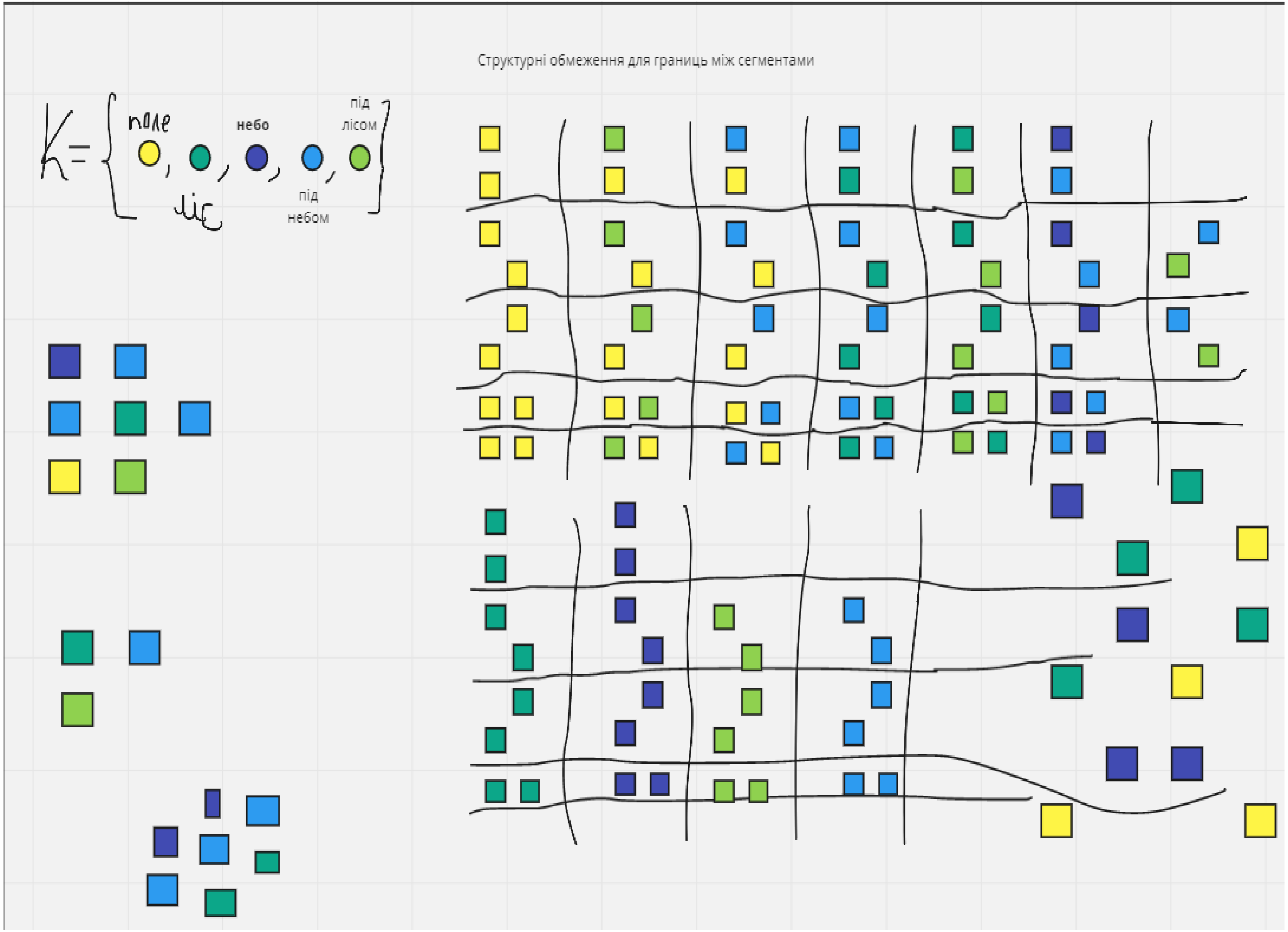
1. EM для пошуку параметрів розподілів. суми Гаусових розподілів для кожного з класів.

Задання g (на наступній сторінці).

<u>Алгоритм:</u>

1. Отримати g та q
2. Провести n-iter ітерацій дифузії.
3. Отримати репараметризовані φ з дифузії.
4. Отримати q-rep та g-rep на основі φ.
5. Викреслюванням знайти необхідний epsilon використавши двійковий (бінарний) пошук. Тут необхідно знаходити кожен раз q-вінагу та g-вінагу і дивитись чи консистентна розмітка при певному epsilon
6. Результуючі g-вінагу та q-вінагу даємо на вхід самоконтролю.
7. Отримуємо сегментацію.

Scanned with CamScanner



Структурні обмеження для границь між сегментами
K = {
поле
ліс
небо
під небом
під лісом
}

Означення двовимірної контекстно-вільної граматики. Навести алгоритм перевірки належності зображення до мови, яку породжує граматика, в різних напівкільцях.

Поле зору: $T(m,n) = \{(i,j) \mid 1 \le i \le m,\ 1 \le j \le n\}$ – складається із m рядків і n стовпців.

Прямокутник (віконце): $\pi \subset T$

Множина прямокутників: $\Pi \ni \pi$

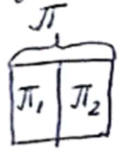
X – множина сигналів – ті значки, які можуть бути записані в поле зору.

Зображення: $\bar{x} = \langle m, n, T(m,n) \to X \rangle$

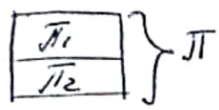
$x(i,j)$ – сигнал, що спостерігається в пікселі (i,j)

$x(\pi)$ – фрагмент (зображення), що спостерігається у віконці π.

$\pi = \pi_1 \square \pi_2$ – операція конкатенації прямокутників по горизонталі



$\pi = \dfrac{\pi_1}{\pi_2}$ – операція конкатенації прямокутників по вертикалі



K – імена

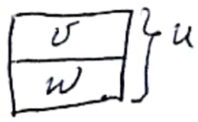
$P_Г: K \times K \times K \to \{0,1\}$ – правило горизонтальної конкатенації зображень

$P_Г(u, v, w) = 1 \Leftrightarrow u = v \square w$

Щоб намалювати зображення з іменем u, поле зору потрібно розділити на дві частини. У лівій частині намалювати зображення з іменем v, в правій w

$P_В: K \times K \times K \to \{0,1\}$ – правило вертикальної конкатенації

$P_В(u, v, w) = 1 \Leftrightarrow u = \dfrac{v}{w}$



Щоб намалювати зображення з іменем u, поле зору потрібно розділити на дві частини. У верхній частині намалювати зображення з іменем v, в нижній w

$P_Т: K \times X \to \{0,1\}$ – термінальне правило

$P_Т(u, \square) = 1 \Leftrightarrow$ білому пікселю потрібно дати ім'я u.

Scanned with CamScanner

Двовимірно контекстно-вільна граматика.

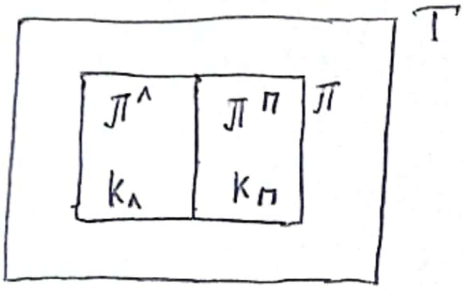
$G = \langle K, X, P_Г, P_В, P_Т \rangle$ в $(\vee, \&)$ варіанті.

$f(\pi, k) \in \{0,1\}$ – чи можна прямокутнику π дати ім'я k

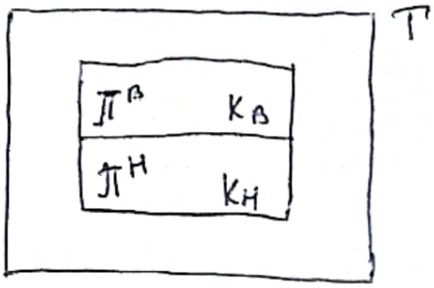
(чи породжує прямокутник ім'я k).

$f((i,j), k) = P_T(k, x(i,j))$

Алгоритм КЯК



$$f(\pi, k) = \bigvee_{\substack{\pi^Л, \pi^П: \\ \pi^Л \, [] \, \pi^П = \pi}} \; \bigvee_{k_Л, k_П} f(\pi^Л, k_Л) \,\&\, f(\pi^П, k_П) \,\&\, P_Г(k, k_Л, k_П) \;\vee$$



$$\vee \bigvee_{\substack{\pi^В, \pi^Н \\ \frac{\pi^В}{\pi^Н} = \pi}} \; \bigvee_{k_В, k_Н} f(\pi^В, k_В) \,\&\, f(\pi^Н, k_Н) \,\&\, P_В(k, k_В, k_Н)$$

алгоритм, який для прямокутн π та імені k каже, чи можна цьому π дати це ім'я.

Чи можна зображенню $\bar{x}^*$ дати ім'я k^*?

Рішення: $f(T, k^*)$

Всього чисел $f(\pi, k)$ $\underbrace{m^2 \cdot n^2}_{\text{кіл-ть прямокутників}} \cdot \underbrace{|K|}_{\text{кіл-ть всіх імен.}}$

Одне число обраховується за: $n \cdot |K|^2 + m \cdot |K|^2$

Складність рішення задачі: $m^2 n^2 |K| \cdot (n \cdot |K|^2 + m \cdot |K|^2) =$

$= m^2 \cdot n^2 \cdot |K|^3 \cdot (m+n)$

Для min-max

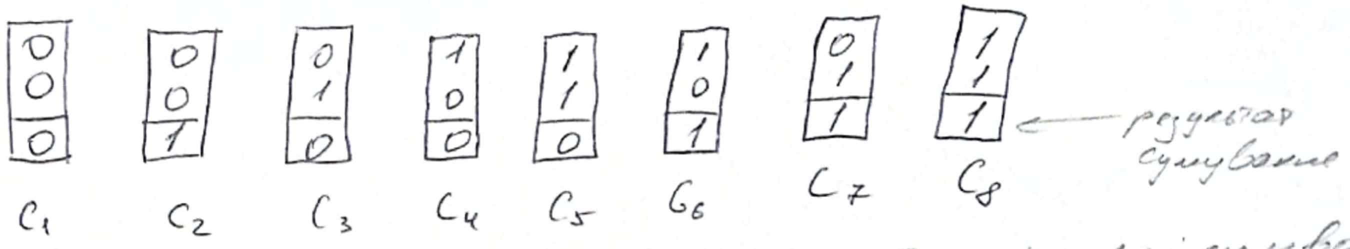
$f(\pi, k) = \min\left(\min\min(\sqcup + \sqcup + \sqcup), \min\min(\sqcup + \sqcup + \sqcup)\right).$

і P – буде грати роль штрафів за конкатенацію.

Scanned with CamScanner

Постановка та розв'язок задачі корекції рисунку, на якому зображено бінарне сумування її двох чисел, на які накладено шум Бернуллі. Еталонні зображення цифр відомі. Для розв'язку застосувати двовимірні контекстно-вільні граматики.

Є еталони:



Тут наведені всі можливі стовпчики. - це термінальні символи

0 1 0 0 1
0 1 1 1 0 — Приклад
1 1 1 1 1

Введемо такі поняття:

US - uncompleted sample

US' - uncompleted sample with carry-over (переносом) (зак-ся переносом)

CS - completed sample

Правила побудови.

Наприклад US' це C_5.
US це C_6

(*) US ← ·|US (C_1, C_6, C_7)

US ← ·|US' (C_2)

конкатенація

(**) US' ← ·|US (C_5)

US' ← ·|US' (C_3, C_4, C_8)

CS ← US.

Нехай у нас є зображення X (зашумлене)
Нам потрібно знайти найкращий шлях
n_cols – кількість вхідних стовпчиків.

Створюємо масив f розміром n_cols × 8 × 2 (к-сть еталонів; US або US') і заповнюємо −∞ всі значеннями. Вирішуємо задачу (max, +). Спочатку ми заповнюємо f для останнього стовпця і потім рухаємося на початок.

Нехай $C^* = [C_1, C_2, \dots C_8]$ стовпці еталони

Заповнюємо для C_1, C_6, C_7 для US:

```
for c in [0,5,6]:
    f[-1,c,0] = xor-sum(C*[c], X[-1])
for c in [4]:
    f[-1,c,1] = xor-sum(C*[c], X[-1])
```

X[-1] — останній стовпець (елемент) зашумленого зображення

Аналогічно для C_5 для US'

Scanned with CamScanner

Далі робимо для решти стовпців; Нехай $C^* = [C_1, \ldots, C_8]$

```
for colum in range(n_cols -2,-1,-1):
   for c in([0,5,5,  1])
      f[column,c,0] = xor(C*[c],X[column]) + max(f[column+1,:,0])
   for c in([2,3,7, 4])
      f[colum,c,1] = xor(C*[c],X[column]) + max(f[column+1,:
```

```
for column in range(n_cols -2,-1,-1).
   for c in [0,5,6]
      f[column,c,0] = xor_sum(C*[c],X[column]) + max(f[column+1,:,0])
   for c in [1]:
      f[colum,c,0] = xor-sum(C*[c],X[column]) + max(f[column+1,:,1])
   for c in [4]:
      f[column,c,1] = xor-sum(C*[c],X[column]) + max(f[column+1,:,0])
   for c in [2,3,7]:
      f[column,c,1] = xor-sum(C*[c],X[column]) + max(f[column+1,:,1])
```

Відновлення:

```
result - масив n_cols x1.
result[0] = argmax(f[0,:,0])
for i in range(1, n_cols):
      if result[i-1] in [0,4,5,6]:
            result[i] = argmax(f[i,:,0])
      else:
         result[i] = argmax(f[i,:,1])
```

Scanned with CamScanner

Звʼязок між OR-AND, MIN-MAX та MAX-MIN задачами розмітки. Як ефективно розвʼязувати будь-яку з цих задач, якщо є ефективний алгоритм розвʼязку лише однієї з них?

Зведення MAX-MIN до OR-AND.

Нехай ми вміємо розвʼязувати будь-яку VΛ задачу $Z^{\varepsilon}_{\wedge\vee} = \langle T, \tau \subseteq T^2, K, g^{\varepsilon}: \tau \times K^2 \to \{0,1\} \rangle$, що полягає у знаходженні значення:

$$G^{\varepsilon}_{\vee\wedge} = \bigvee_{k: T \to K} \bigwedge_{tt' \in \tau} g^{\varepsilon}_{tt'}(k_t, k_{t'})$$, ефективно.

На вході маємо max-min задачу: $Z_{maxmin} = \langle T, \tau \subseteq T^2, K, g: \tau \times K^2 \to \mathbb{R} \rangle$, що полягає у знаходженні $G_{maxmin} = \max\limits_{k: T \to K} \min\limits_{tt' \in \tau} g_{tt'}(k_t, k_{t'})$.

Нехай $g^{\varepsilon}_{tt'}(k, k') = \mathbb{1}(g_{tt'}(k, k') \leq \varepsilon)$, $tt' \in \tau, k \in K, k' \in K$, тоді рівносильними є умови:

$$1 = G^{\varepsilon}_{\wedge\vee} = \bigvee_{k: T \to K} \bigwedge_{tt' \in \tau} g^{\varepsilon}_{tt'}(k_t, k_{t'}),$$

$$\exists k: T \to K, \forall tt' \in \tau: g_{tt'}(k_t, k_{t'}) < \varepsilon \Rightarrow \exists k: T \to K, \max_{tt' \in \tau} g_{tt'}(k_t, k_{t'}) \leq \varepsilon$$

$$\Rightarrow \min_{k: T \to K} \max_{tt' \in \tau} g_{tt'}(k_t, k_{t'}) \leq \varepsilon.$$

Іншим набором рівносильних умов є:

$$0 = G^{\varepsilon}_{\vee\wedge} = \bigvee_{k: T \to K} \bigwedge_{tt' \in \tau} g^{\varepsilon}_{tt'}(k_t, k_{t'}) \Rightarrow \forall k: T \to K, \exists tt' \in \tau: g_{tt'}(k_t, k_{t'}) > \varepsilon$$

$$\Rightarrow \forall k: T \to K \; \max_{tt' \in \tau} g_{tt'}(k_t, k_{t'}) > \varepsilon \Rightarrow \min_{k: T \to K} \max_{tt' \in \tau} g_{tt'}(k_t, k_{t'}) > \varepsilon$$

Оскільки $\max A \in A$ і $\min A \in A$ => $\forall$ скінченної мн-ни A:

$$G_{maxmin} \in \mathcal{E} = \{g_{tt'}(k, k'): tt' \in \tau, k \in K, k' \in K\}.$$

Впорядкуємо елементи множини $\mathcal{E}$

$$\min \mathcal{E} = \varepsilon_1 < \varepsilon_2 \ldots < \varepsilon_{|\mathcal{E}|} = \max \mathcal{E}, \; \varepsilon_i \in \mathcal{E}, \; i = \overline{1, |\mathcal{E}|}$$

Scanned with CamScanner

Тоді

$$G_{maxmin} = \begin{cases} \varepsilon_1, & 1 = G^{\varepsilon_1}_{V\Lambda} \Leftrightarrow G_{maxmin} \le \varepsilon_1 = \min\limits_{k \in T \to K} \min\limits_{tt' \in \Gamma} g_{tt'}(k_t, k_{t'}). \\ \varepsilon_i, & \exists i > 1 : 0 = G^{\varepsilon_{i-1}}_{V\Lambda} \ne G^{i}_{V\Lambda} = 1 \Leftrightarrow \varepsilon_{i-1} < \min\limits_{k: T \to K} \max\limits_{tt' \in \Gamma} g_{tt'}(k_t, k_{t'}) \le \varepsilon_i \end{cases}$$

Знайти розмітку можемо, розв'язавши OR-AND задачу розмітки $Z^{\varepsilon_i}_{V\Lambda}$

Зведення OR-AND до MAX-MIN, MIN-MAX

В алгоритмі викреслювання.

$$g^{i+1}_{t}(k) = \min\left(g^{i}_{k}(k), \min_{t' \in N_t} \max_{k' \in K} g_{tt'}(k, k')\right)$$

$$g^{i+1}_{tt'}(k, k') = \min\{g^{i}_{tt'}(k, k'), g^{i}_{t}(k), g^{i}_{t'}(k')\}$$

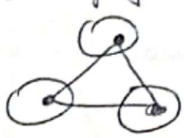
А взагалі розв'язується як звичайна MAX-MIN; MIN-MAX.

Scanned with CamScanner

Поняття інваріантності та поліморфізму (для бінарного та тернарного оператора) для OR-AND, MAX-MIN, MIN-MAX задач розмітки. Показати, що інваріантність ({0,1}, max, min) задачі відносно оператора p означає інваріантність ({0,1}, ∨, &) задачі відносно того ж самого оператора p. Назвати визначення оператора напівґратки, мажоритарного оператора та оператора Мальцева. Показати відшукання кращої розмітки за умови наявності відомого оператора напівґратки, відносно якого задача інваріантна.

Хочемо, щоб розмітку було легко шукати

Для цього нам потрібно, щоб в розмітці не було не допустимих вершин і дуг, що з'єднують хоча б одну не допустиму вершину. Тобто залишаємо тільки таке!



. Назвемо таку задачу <u>узгодженою</u> (arc-consistent)

Тобто, це така задача.! $g_t(k) \Rightarrow \mathop{\&}\limits_{t'\in N_t} \bigvee\limits_{k'\in K} g_{tt'}(k,k')$; $g_{tt'}(k,k') \Rightarrow g_t(k) \,\&\, g_{t'}(k')$

$\forall tt'\in T, k\in K, k'\in K.$

$\forall t\in T, k\in K$

Хочемо, щоб існував оператор: $\exists\, \theta: K^* \to K: K^*_t = \bigodot\limits_{k:\, g_t(k)=1} k$

зірка Кліні.

Тобто він дає мітку в кожному об'єкту → (яку $g_t(k)=1$).

Цей оператор <u>бінарний</u>.

$g_t(k) \,\&\, g_t(k') \to g_t(k \circ k')$

$g_{tt'}(k,k') \,\&\, g_{tt'}(\varkappa,\varkappa') \to g_{tt'}(k\circ\varkappa, k'\circ\varkappa')$

импликация «из истинного утверждения следует истинное, из ложное → любое»!

«Если 2x2=5, то я папа Римский» Крыгин. В.М.

$\circ$ — поліморфізм задачі $Z \Leftrightarrow$ задача Z є інваріантною відносно $\circ$

Поліморфізм — це оператор.

Властивості оператора $\circ$!

I. вл: комутативність: $a\circ b = b\circ a$.

II вл. идемпотентность: $a\circ a = a \Rightarrow$ «Якщо лише одна вершина допустима ми маємо отримати її»

III вл. асоціативність: $(a\circ b)\circ c = a\circ(b\circ c)$

IV. вл. Замкнутість: $a\circ b \in K$.

Це все вл-ті <u>напівґратки</u> (полурешётки) (semilattice).

Мажоритарний оператор

$p(k,\dots,k,k') = p(k,\dots,k,k',k) = \dots p(k,k',k,k,\dots,k) = k.$

$p(k,k,k') = p(k,k',k) = p(k',k,k) = k.$

Оператор Мальцева

$p(k,k,k') = p(k',k,k) = k'.$

для $p(k,k',k)$ - невідомо

Scanned with CamScanner

Задача t – інваріантна відносно оператора $\circ$,

$g_t(k) \,\&\, g_t(k') \to g_t(k \circ k')$, $\forall t \in T, \forall k \in K, \forall k' \in K$.

$g_{tt'}(k, k') \,\&\, g_{tt'}(\varkappa, \varkappa') \to g_{tt'}(k \circ \varkappa, k' \circ \varkappa')$, $\forall tt' \in T, \forall k \in K, \forall k' \in K, \forall \varkappa \in K, \forall \varkappa' \in K$

Важливо $\boxed{\circ \text{ – це оператор поліморфізму} \Leftrightarrow \text{задача є інваріантною відн. оп. } \circ}$

Інваріантність в Max-Min $\min(f(k), f(\varkappa)) \leq f(k \circ \varkappa)$

Інваріантність в OR-AND $f(k) \,\&\, f(\varkappa) \to f(k \circ \varkappa)$

або g, або g.

Ці інваріантності однакові для оператора $\circ$.

бо $\underset{0}{0 \,\&\, 1} \to$ будь-яке $\Rightarrow \min(0,1) = 0 \leq$ всього

з істинного в істинне

Поліморфізм для MaxMin випливатиме з інваріантності

Для тернарних операторів: $\min(f(k), f(k'), f(k'')) \leq f(k \circ k' \circ k'')$

$f(k) \,\&\, f(k') \,\&\, f(k'') \to f(k \circ k' \circ k'')$.

Якщо $\circ$ відомий і є поліморфізмом і оператором напівґратки $\Rightarrow$ можем знайти розмітку якщо навіть не існує допустимої розмітки

$\Rightarrow \forall t: k_t^* = \underset{\varkappa \in K: g_t(\varkappa) = 1}{\circ\, \varkappa} \Rightarrow \underset{t \in T}{\&} g_t(k_t^*) \,\&\, \underset{tt' \in T}{\&} g_{tt'}(k_t^*, k_{t'}^*) = G$

Дано задачу, що інваріантна відносно оператора напівреш ітки. Маємо оракул, який вірно відповідає на запитання „Яка вага найкращої розмітки?" Навести ефективний алгоритм пошуку кращої розмітки та показати, що під час роботи цього алгоритму задача залишається інваріантною відносно оператора напівреш ітки

Алгоритм пошуку розмітки із самоконтролем

Вхід: будь-яка задача, що задовольняє обмеження Z.

Вихід: $K^T \cup$ {„відмова від розпізнавання", „нема рішення"} (якщо на першому кроці ми все викреслили то одразу нема рішення).

1. крок: $\bigvee_{k^T \in K^T} f(k) = 0 \Rightarrow$ нема рішення

$i = \overline{1,T}$ — Проходимо по всім об'єктам.

$\varphi_i(k) = \bigvee_{k_{i+1} \in K} \cdots \bigvee f(k_1^*, \dots, k_{i-1}^*, k, k_{i+1}, \dots, k_{|T|})$, $\forall k \in K$.

if $\exists k \in K : f_i(k) = 1 \Rightarrow k_i^* = k$.

if $\forall k \in K : f_i(k) = 0 \Rightarrow$ {„відмова від розпізнавання", НЗК.

Значить, задача не інв. відносно окремого сімейства операторів

Якщо $\forall i = \overline{1,|T|}\ \exists k_i^* \in K : f_i(k_i^*) = 1 \Rightarrow$ відповідь $K^* = \langle k_1^*, k_2^*, \dots, k_{|T|}^* \rangle$

Якщо для кожного об'єкта знайшли мітку, то в сукупності ці мітки створюють розмітку.

На першому кроці нам потрібно перевірити, чи взагалі в задачі можливе рішення, тому і перевіряємо (запускаємо викреслювання).

- - -

Алгоритм самоконтролю не порушує інваріантність задачі

■ Припустимо, задача інваріантна відносно оператора P

Виконуємо крок алгоритму пошуку розмітки із самоконтролем.

Знаходимо перше k_{t^*}, означає інші k ми занулюємо.

Якщо $f_t(k_{t^*}) = 0 \Rightarrow$ „йдемо до наступного k_{t^*}"

Якщо $f_t(k_{t^*}) = 1 \Rightarrow$ необхідно $f_t\big(P(k_t^*, \dots, k_t^*)\big) = 1$ (щоб виконувалась інваріантність)

Що має виконуватись для цього оператора? (Він може бути n-місний).

↳ це має виконуватись, якщо оператор ідемпотентний.

Тобто, $\forall$ ідемпотентного оператора (якщо задача інваріантна відносно нього) алгоритм пошуку розмітки з самоконтролем не порушує інваріантність.

Scanned with CamScanner

Ідемпотентність оператора ρ - це достатня умова інваріантності

Оператори напіврешітки, Мальцева, Мажоритарний є ідемпотентними.

Ідемпотентність означає, якщо ми цьому оператору подамо один і той самий аргумент декілька разів, то він його і видасть: $\rho(k, k, k) = k$

~~Оператор напіврешітки~~.

Scanned with CamScanner

Дано задачу, що інв. відносно мажоритарного оператора. Маємо оракул, який вірно відповідає на запитання „Яка вага найкращої розмітки?". Навести ефективний алгоритм пошуку кращої розмітки та показати, що під час його роботи задача залишається інв. відносно маж. оператора.

Алгоритм пошуку розмітки із самоконтролем.

Вхід: будь-яка задача, що задовольняє обмеження Z. OR-AND

Вихід: $K^T \cup$ {„відмова від розпізнавання", „нема рішення"}.

1. Крок: Запускаємо зірку в симплекс: якщо $\bigvee_{K^T \to K} f(K) = 0$ => „нема рішення".

Якщо ∈ => $i = \overline{2,T}$ – Проходимо по всім об'єктам.

$$\varphi_i(K) = \bigvee_{K_{i+1} \in K} \dots \bigvee_{K_n \in K} f(K_1^*, \dots, K_{i-1}^*, K, K_{i+1}, \dots, K_{|T|}), \forall K \in K$$

Якщо $\exists K \in K: f_i(K) = 1 \Rightarrow K_i^* := K$

Якщо $\forall K \in K: f_i(K) = 0 \Rightarrow$ „відмова від розпізнавання";

Якщо $\forall i = \overline{1,T}\ \exists K_i^* \in K: f_i(K_i^*) = 1 \Rightarrow$ відповідь $K^* = \langle K_1^*, K_2^*, \dots, K_{|T|}^* \rangle$.

В MAX-MIN.

$$\exists K_i \in K: \max_{K_{i+1} \in K} \max_{K_{i+2} \in K} \dots \max_{K_{|T|} \in K} f(K_1^*, K_2^*, \dots, K_i, K_{i+1}, \dots, K_{|T|}) = G^{AC}.$$

Алгоритм самоконтролю не порушує інваріантність задачі

■ Виконуємо крок алгоритму пошуку розмітки із самоконтролем

Припустимо, що задача була інв. відносно оператора p.

Знаходимо перше K_t^*,

Якщо $f_t(K_t^*) = 0 \Rightarrow$ йдемо до наступного K_t^*.

Якщо $f_t(K_t^*) = 1 \Rightarrow$ необхідно $f_t(p(K_1^*, \dots, K_t^*)) = 1$.

Для цього оператора має виконуватись ідемпотентність.

а мажоритарний є ідемпотентним

Ідемпо-ть оператора це достатня умова інваріантності

$\forall$ задачі, що інв відн ідемпотентного оператора алгоритм пошуку розмітки з самоконтролем не порушує інваріантність. ■

Scanned with CamScanner

Навести алгоритм викреслювання другого порядку. Показати, що алгоритм не надасть результат, що гірший за справжній. Показати, що застосування алгоритму не позбавляє задачу інваріантності відносно оператора напівґратки.

$Z = \langle T, \tau \subseteq T^2, K, q: T \times K \to \{0,1\}, g: \tau \times K^2 \to \{0,1\} \rangle$ — задача.

T — об'єкти, $\tau \subseteq T^2$ — структура, K — мітки, q — ваги міток, g — ваги дуг.

Викреслювання 2-го порядку

Вхід: Z (задача).

Ініціалізація: $q^0 = q$, $g^0 = g$, $i = 0$.

$\forall t \in T$

$\forall k \in K$

$$q_t^{i+1}(k) = q_t^i(k) \ \& \bigwedge_{t' \in N_t} \bigvee_{k' \in K} g_{tt'}^i(k, k')$$

$O(|\tau| \cdot |K| \cdot |K|)$

$\forall t \in T$:

$\forall t' \in N_t$:

$\forall k \in K$:

$\forall k' \in K$:

$$g_{tt'}^{i+1}(k, k') = g_{tt'}^i(k, k') \ \& \ q_t^{i+1}(k) \ \& \ q_{t'}^{i+1}(k')$$

$O(|\tau| \cdot |K| \cdot |K|)$

$i \leftarrow i+1$.

$q^i \neq q^{i-1} \vee g^i \neq g^{i-1}$.

інакше і відповідь q^i, g^i.

Для max-min

$$q_t^{i+1}(k_t) = \min\{q_t^i(k_t), \min_{t' \in N_t} \max_{k' \in K} g_{tt'}^i(k_t, k_{t'})\}$$

$$g_{tt'}^{i+1}(k_t, k_{t'}) = \min\{g_{tt'}^i(k_t, k_{t'}), q_t^i(k_t), q_{t'}^i(k_{t'})\}.$$

Алгоритм викреслювання 2-го порядку не надасть рез-т, що гірший за справжній

Розмітка.

$\exists k^* \in K^T: f(k^*) = G(Z) = \bigvee_{k: T \to K} \{ \bigwedge_{t \in T} q_t(k_t) \ \& \bigwedge_{tt' \in \tau} g_{tt'}(k_t, k_{t'}) \} \to$

$\to G^{AC}(Z)$. Arc-consistency. тобто $G(Z) \to G^{AC}(Z)$.

$\Rightarrow (\forall t \in T: q_t(k_t^*) = 1, \ \forall t' \in N_t: g_{tt'}(k_t^*, k_{t'}^*) = 1$.

результат викреслювання.

Scanned with CamScanner

$G^{*c}(z)$ – результат задачі z після застосування алгоритму викреслювання.

Якщо у нас існує допустима розмітка, то алгоритм викреслювання скаже, що розмітка існує (а в Min Max результат викреслювання буде не вірне тим справжнє рішення).

Тобто якщо розмітка існує ⇒ викреслювання щось залишить.

Б Min-Max.

$$\exists k^* \in K^T : f(k^*) \overset{\text{реальне}}{=} G(z) \le \overset{\text{результат на останній ітерації}}{G^{*c}(z)} = \max_{t \in T} \max_{k \in K} q_t^{n}(k)$$

$$\Rightarrow \begin{cases} \forall t \in T : q_t(k_t^*) \ge f(k^*),\ \forall t' \in N_t : g_{tt'}(k_t^*, k_{t'}^*) \ge f(k^*) \\ \exists t \in T : q_t(k_t^*) = f(k^*) \lor \exists tt' \in T : g_{tt'}(k_t^*, k_{t'}^*) = f(k^*) \end{cases}$$

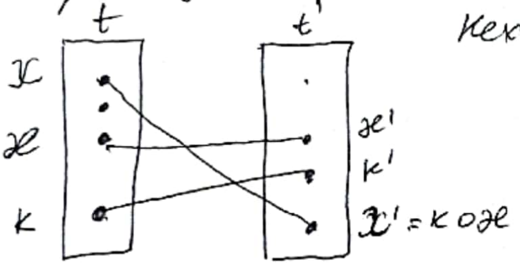
Викреслювання I порядка не порушує інв. відносно оператора напіврешітки, що значить порушує ⇒ значить порушується якась вл-ть напіврешітки.

Нехай задача інв. відносно оператора напіврешітки. D

і $q_t^i(k) = q_t^i(\varkappa) = q_t^i(k \circ \varkappa) = 1$, але $q_t^{i+1}(k \circ \varkappa) = 0$

I випадок
Із $\varkappa$ і k виходять дуги

Розглянемо приклад.



Нехай $k \circ \varkappa = x$;

Застосуємо вл-ть поліморфізму:

із того, що вже є дві доп. дуги $\varkappa\varkappa'$, kk' слідує, що обов'язково існує ще одна дужка, що рахується по формулі

$$g_{tt'}(k,k') \,\&\, g_{tt'}(\varkappa,\varkappa') \to g_{tt'}(k\circ\varkappa, k'\circ\varkappa') = g_{tt'}(x,x')$$

⇒ Тобто не можлива така ситуація, що $q_t^{i+1}(k\circ\varkappa) = 0$, а з неї не виходить жодної дуги, якщо були $q_t^i(k) = q_t^i(\varkappa) = q_t^i(k\circ\varkappa) = 1 = q_t^i(x)$

Тобто дуга з x в x' має бути ⇒ вершина $\varkappa$ не має бути викресленою ⇒ $q(k\circ\varkappa) = 1$

Розглянемо випадок, коли із $\varkappa$ і k дуги не виходять

⇒ не може бути такого, що із кожного сусіда t' виходить хоча б одна дуга із k і із кожного сусіда t' виходить одна дуга із $\varkappa$ одночасно.

$$\underset{t' \in N_t}{\&} \bigvee_{k' \in K} g_{tt'}(k,k') \,\&\, \underset{t' \in N_t}{\&} \bigvee_{\varkappa \in K} g_{tt'}(\varkappa,\varkappa') = 1.$$

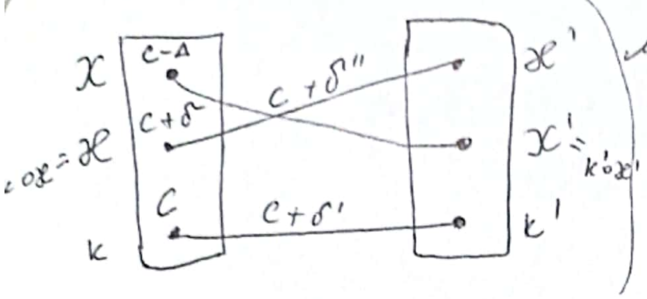
⇒ $\varkappa$ або k буде викреслено ⇒ $0 = q_t^{i+1}(k) \,\&\, q_t^{i+1}(\varkappa) \to q_t^{i+1}(x) = 0$.

Scanned with CamScanner

Max-Min

Хай $\min\{g_t^i(k), g_t^i(\varkappa)\} \le g_t^i(\underset{X}{k \circ \varkappa})$; $= C$.

але $g_t^{i+1}(\underset{X}{k\circ\varkappa}) < C$



Ми знаємо, що задача інв. відн. оп. напівґратки

$\Rightarrow \min\{g_{tt'}(k,k'), g_{tt'}(\varkappa,\varkappa')\} \le g_{tt'}(X,X')$

тобто це виконувалось.

Не може бути такої ситуації:

з X в X' дужка була не менше ніж з $\varkappa$ в $\varkappa'$ та k в k'

тобто не гірше ніж C

Опуститись, вона могла і тільки якщо була якась „погана дуга“,

тобто існувала б дуга з ваги меншою за $C-\Delta$ (з X в X')

$$\Rightarrow \min\{\min_{t'\in N_t}\max_{k'\in K} g_{tt'}(k,k'), \min_{t'\in N_t}\max_{\varkappa'\in K} g_{tt'}(\varkappa,\varkappa')\} < C.$$

$$\Rightarrow g_t^{i+1}(\varkappa) < C \text{ або } g_t^{i+1}(k) < C \Rightarrow C > g_t^{i+1}(X) \ge \min\{g_t^{i+1}(k), g_t^{i+1}(\varkappa)\}$$

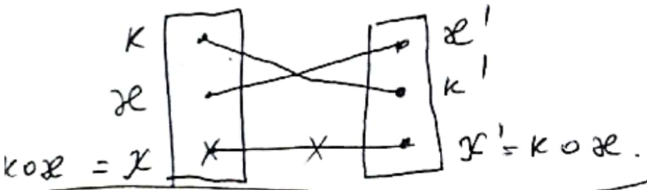
Такого бути не може

У нас це є дуги; якщо викреслилась дуга і порушилась інв-ть

то потрібно зрозуміти, що робити в загальному випадку.

Для OR-AND

Нехай $g^i_{tt'}(k,k') = g^i_{tt'}(\varkappa,\varkappa') = g^i_{tt'}(\underset{k\circ\varkappa}{X},\underset{k'\circ\varkappa'}{X'}) = 1$, але $g^{i+1}_{tt'}(X,X') = 0$



Нехай викреслилась $X X'$; вона могла викреслитись, якщо вершина або X або X' викреслилась.

Нехай це була X.

Але ми знаємо, якщо задача була інв. і дуги kk', $\varkappa\varkappa'$ існували б.

$\Rightarrow g^{i+1}_{tt'}(X,X')=0 \Rightarrow g^i_t(X) \,\&\, g^i_{t'}(X')=0$

Але, якщо нема дуги XX' $\Rightarrow$ або k або $\varkappa$ (або k', або $\varkappa'$) мали викреслитись.

$g_t(k) \,\&\, g_t(\varkappa) \,\&\, g_{t'}(k') \,\&\, g_{t'}(\varkappa') = 0 \Rightarrow$ а це значить викреслилась якась дуга $\Rightarrow g^{i+1}_{tt'}(k,k') \,\&\, g^{i+1}_{tt'}(\varkappa,\varkappa') = 0 \Rightarrow$ інв. не порушена

Scanned with CamScanner

Показати, що взяття проекції не позбавляє задачу інваріантності відносно поліморфізму

f - функція : $f: K^T \to \{0,1\}$.

T - множина об'єктів ; T_1, T_2 - розбиття множини T.

$T = T_1 \cup T_2$; $T_1 \cap T_2 = \varnothing$

$f^{p_2}_{T_1} : K^{T_1} \to \{0,1\}$ - проекція діє на множині T_1

Визначимо її:

$$f^{p_2}(K(T_1)) = \bigvee_{K(T_2) \in K^{T_2}} f(K(T_1), K(T_2)) : K(T_1) \in K^{T_1}.$$

$f \in inv(p)$ - означає, що ф-я f є інваріантною відносно оператора p.

Властивість: $\begin{cases} f(a) \& f(b) \to f(p(a,b)) \\ f(a,a') \& f(b,b') \to f(p(a,b), p(a',b')) \end{cases}$

Доведемо, що,
якщо $f(K(T_1)) \& f(K(T_2)) \to f(p(K(T_1), K(T_2)))$

$\Rightarrow f^{p_2}(K(T_1)) \& f^{p_2}(K(T_2)) \to f^{p_2}(p(K(T_1), K(T_2)))$

для того ж оператора p.

Імплікація: "з істинного в істинне" : $1 \to 1$

Перевіримо для $f^{p_2}(K(T_1)) = 1$ і $f^{p_2}(K(T_2)) = 1$.

$f^{p_2}(K(T_1)) = 1 \Rightarrow \bigvee_{K(T_2) \in K^{T_2}} f(K(T_1), K(T_2)) = 1 \Rightarrow \exists a \in K^{T_2} : f(K(T_1), a) = 1$

$f^{p_2}(K(T_2)) = 1 \Rightarrow \bigvee_{K(T_1) \in K^{T_1}} f(K(T_2), K(T_1)) = 1 \Rightarrow \exists b \in K^{T_1} : f(K(T_2), b) = 1$.

Для фіксованих : $K(T_1), K(T_2)$.

$\Rightarrow$ маємо $\underbrace{f(K(T_1), a) \& f(K(T_2), b)}_{1} \to f(p(K(T_1), K(T_2)), p(a,b))$

$\bigvee_{K(T_2) \in K^{T_2}} f(p(K(T_1), K(T_2)), a, b) = 1$

$\Rightarrow$ візьмемо диз'юнкцію:

$\Rightarrow f^{p_2}(K(T_1)) \& f^{p_2}(K(T_2)) \to f^{p_2}(p(K(T_1), K(T_2)))$

■

Scanned with CamScanner

Показати, що задачу, яка інваріантна відносно мажоритарного оператора, можна представити у вигляді графу, а ваги дуг є проекціями ваг цієї задачі на ці дуги

Мажоритарний оператор: $p(k,k,k') = p(k,k',k) = p(k',k,k) = k$

Є дві задачі: $Z = \langle T, \tau \subseteq T^2, K, g: \tau \times K^2 \to \{0,1\} \rangle$ → тут $g = 1$

$Z' = \langle T, \tau \subseteq T^2, K, g': \tau \times K^2 \to \{0,1\} \rangle$. Тобто $\langle g, g' \rangle \to \langle g' \rangle$

g і g' інв. відносно якогось оператора p ⇒ $g \,\&\, g'$ теж інв. відносно того ж p. = "Кон'юнкція не знищує інв-ть"

$(g \& g')_{tt'}(k,k') \,\&\, (g \& g')_{tt'}(\varkappa,\varkappa') = [g_{tt'}(k,k') \,\&\, g'_{tt'}(k,k')] \,\&\, [g_{tt'}(\varkappa,\varkappa') \,\&\, g'_{tt'}(\varkappa,\varkappa')] =$ {переставимо за комутативністю} $=$

$= [g_{tt'}(k,k') \,\&\, g_{tt'}(\varkappa,\varkappa')] \,\&\, [g'_{tt'}(k,k') \,\&\, g'_{tt'}(\varkappa,\varkappa')] \to$

$\to g_{tt'}(p(k,\varkappa), p(k',\varkappa')) \,\&\, g'_{tt'}(p(k,\varkappa), p(k',\varkappa')) =$

$= (g \& g')_{tt'}(p(k,\varkappa), p(k',\varkappa'))$

У нас є розмітка (функція) $f: K^T \to \{0,1\}$ і мажоритарного оператора $p^n: (K^n)^3 \to K^n$ $\forall n \in 2^T$

Задача f інваріантна відносно мажоритарного оператора $f(k) \,\&\, f(\varkappa) \,\&\, f(x) \to f(p^T(k,\varkappa,x))$ (*)
$\forall k, \varkappa, x \in K^T$

$p^T(k,\varkappa,x)_t = p^t(k_t, \varkappa_t, x_t)$
↑ оператор p застосували до розмітки

Розіб'ємо мн-ну T на 3 під-множини. $\{T_1, T_2, T_3\}$ – розбиття T.
T – мн-на об'єктів ;
$T_1 \cup T_2 \cup T_3 = T$, $T_1 \cap T_2 = T_2 \cap T_3 = T_1 \cap T_3 = \varnothing$.

Визначимо проекції
Проекція f на $T_i \cup T_j$:

$$f_{T_i \cup T_j}(k_{T_i \cup T_j}) = \bigvee_{k_{T_a}: T_a \to K} f(k_{T_a}, k_{T_i \cup T_j}), \quad \forall k_{T_i \cup T_j}: T_i \cup T_j \to K;$$
$$\forall \{i,j,a\} = \{1,2,3\}$$

Ми хочемо показати, що якщо задача інв. відносно маж. оператора то вона може бути представлена у вигляді кон'юнкції допустимості дуг. і більше того самі g будуть проекціями цієї задачі на ці дуги.

(I) Якщо розмітка допустима => проекції теж допустимі
$\exists k \in K^T: f(k) = 1 \Rightarrow f_{T_i \cup T_j}(k_{T_i \cup T_j}) = 1$, $\forall i \neq j$; $i,j \in \{1,2,3\}$
$\forall i \neq j: i,j \in \{1,2,3\}: (\exists k_{T_i \cup T_j}: T_i \cup T_j \to K): f_{T_i \cup T_j}(k_{T_i \cup T_j}) = 1 \Rightarrow \exists k \in K^T: f(k) = 1$

Scanned with CamScanner

II :

Якщо у нас проекції γ на підмножини $T_i \cup T_j = 1$ в сукупності означає, що вся розмітка K тоді допустима!

$$\Rightarrow \gamma_{T_1 \cup T_2}(K_{T_1 \cup T_2}) \,\&\, \gamma_{T_1 \cup T_3}(K_{T_1 \cup T_3}) \,\&\, \gamma_{T_2 \cup T_3}(K_{T_2 \cup T_3}) \rightarrow \gamma(k)\ \forall k \in K^T$$

~~Ми знаємо, якщо розмітка допустима $\Rightarrow \forall$ T проекція теж буде допустимою~~

~~$\exists k \in K^T: \gamma(k) = 1 \Rightarrow \gamma_{T_i \cup T_j}(K_{T_i \cup T_j}) = 1,\ \forall i \neq j,\ i, j \in \{1, 2, 3\}$~~

$$\gamma_{T_1 \cup T_2}(K_{T_1 \cup T_2}) \,\&\, \gamma_{T_1 \cup T_3}(K_{T_1 \cup T_3}) \,\&\, \gamma_{T_2 \cup T_3}(K_{T_2 \cup T_3}) = 1,\ \forall k \in K^T.$$

$$\Rightarrow \exists k'_{T_a} \in K^{T_a},\ a \in \{1,2,3\}:\ \gamma(K_{T_1}, K_{T_2}, K'_{T_3}) \,\&\, \gamma(K_{T_1}, K'_{T_2}, K_{T_3}) \,\&\, \gamma(K'_{T_1}, K_{T_2}, K_{T_3}) = 1$$

Згадуємо (*) $\Rightarrow \gamma\left(p^{T_1}(k_{T_1}, k_{T_1}, k'_{T_1}),\ p^{T_2}(k_{T_2}, k'_{T_2}, k_{T_2}),\ p^{T_3}(k'_{T_3}, k_{T_3}, k_{T_3})\right) =$

$= \gamma(k_{T_1}, k_{T_2}, k_{T_3}) = \underline{1.}$

$$\gamma(k) = \{\text{I}, \text{II}\} = \gamma_{T_1 \cup T_2}(K_{T_1 \cup T_2}) \,\&\, \gamma_{T_2 \cup T_3}(K_{T_2 \cup T_3}) \,\&\, \gamma_{T_2 \cup T_3}(K_{T_1 \cup T_3})$$

Це дає нам можливість дробить проекції

$\gamma \rightarrow \gamma_{T_1 \cup T_3} \rightleftarrows$; $\gamma \downarrow \gamma_{T_1 \cup T_2}$; $\gamma \searrow \gamma_{T_2 \cup T_3}$

$\gamma_{T_1 \cup T_2}\ \downarrow\downarrow\downarrow$ ... ; $\gamma_{T_2 \cup T_3}\ \downarrow\downarrow\downarrow$

: До яких пір ми можемо дробити?

до тих поки T_i не стане одноелементною

$T_i = \{t_i\},\ T_j = \{t_j\}$.

Миговорилмо, що така проекція і є допустимістю дужки

$$\left(\gamma_{\{t_i\} \cup \{t_j\}} \overset{def}{=} g_{t_i t_j}\right)$$

$$\Rightarrow \left(\exists g: \tau \times k^2 \rightarrow \{0,1\}\right) : (\forall k: T \rightarrow K) : \gamma(k) = \mathop{\&}_{tt' \in \tau} g_{tt'}(k_t, k_{t'}) -$$

$\rightarrow$ якщо задача інв. відн. мат. оператора.

Виходить γ може бути представлена у вигляді графа,

а ваги дуг є проекціями ваг цієї задачі на ці дуги.

Scanned with CamScanner

зірки в симплекс

$\overline{Z} = \langle T,\ \tau \subseteq T^2,\ K,\ g: \tau \times K^2 \to \{0,1\} \rangle$

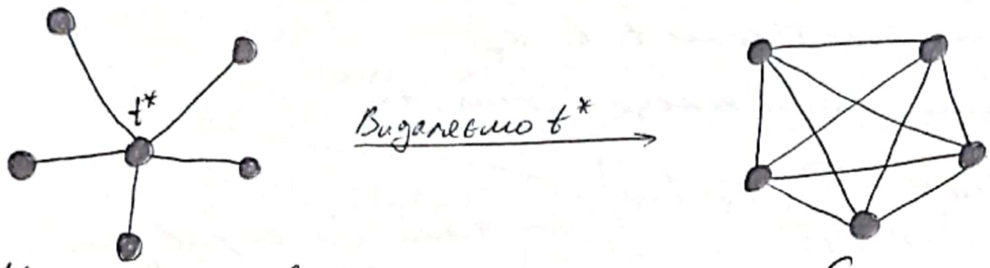
OR-AND задача: $G = \bigvee_{k: T \to K} \ \&_{tt' \in \tau} \ g_{tt'}(k_t, k_{t'})$: якщо $G = 1 \Rightarrow$ існує доп. розмітка

Задача пошуку розмітки: $k^* \in \arg \bigvee_{k: T \to K} \ \&_{tt' \in \tau} \ g_{tt'}(k_t, k_{t'})$

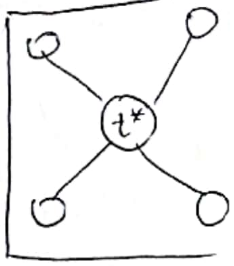
$\arg_{x \in X} \bigvee f(x) = \{x^* \in X : f(x^*) = \bigvee_{x \in X} f(x)\}$

Алгоритм, який дозволяє визначити допустимість розмітки для задачі, що інв. відносно мажоритарного оператора це алгоритм <u>зірки в симплекс</u>

Проекція — це операція, що дозволяє нам виріснити: проходить хоч якась доп. розмітка через якусь мн-ку.



Ми хочемо видалити об'єкт так, щоб це не вплинуло на розмітку. Ми рахуємо проекцію і це значить, що (із вл-ті проекції) отримана задача (після викреслення об'єкта) не перестає бути інв. відн. маж. оп $\Rightarrow$ вона як завжди може бути представлена сукупністю дуг.



$$\&_{\substack{t_1 \in N_{t^*},\\ t_2 \in N_{t^*}}} g_{t_1 t_2}(k_{t_1}, k_{t_2}) = \bigvee_{k^* \in K} \ \&_{t \in N_{t^*}} g_{t^* t}(k^*, k_t), \ \forall k: T \to K.$$

Ми знаємо, що якщо задача інв. відносно маж. оператора, то допустимість кожної її дуги рахується як проекція:

$$g_{tt'}(k_t, k_{t'}) = \bigvee_{k: T \setminus \{t, t'\} \to K} \delta(k, k_t, k_{t'})$$

Застосуємо це:

$$g'_{t_1 t_2}(k_{t_1}, k_{t_2}) \ominus \bigvee_{k: N_{t^*} \setminus \{t_1, t_2\} \to K} \ \bigvee_{k^* \in K} \ \&_{t \in N_{t^*}} g_{t^* t}(k^*, k_t): \ \forall t_1, t_2 \in N_{t^*} \ \forall k_{t_1}, k_{t_2} \in K.$$

$$\ominus \left\{ \begin{matrix} \text{винесем} \\ \text{з } t_1, t_2 \end{matrix} \right\} = \bigvee_{k: N_{t^*} \setminus \{t_1, t_2\} \to K} \ \bigvee_{k^* \in K} g_{t^* t_1}(k^*, k_{t_1}) \ \& \ g_{t^* t_2}(k^*, k_{t_2}) \ \& \ \&_{t \in N_{t^*} \setminus \{t_1, t_2\}} g_{t^* t}(k^*, k_t)$$

= { комутативність диз'юнкції }

$$= \bigvee_{k^* \in K} g_{t^* t_1}(k^*, k_{t_1}) \ \& \ g_{t^* t_2}(k^*, k_{t_2}) \ \& \ \bigvee_{k: N_{t^*} \setminus \{t_1, t_2\} \to K} \ \&_{t \in N_{t^*} \setminus \{t_1, t_2\}} g_{t^* t}(k^*, k_t) =$$

(#)

$g_{t^* t}$ не залежить від інших $g_{t^* t'}$

(#) $\Rightarrow$ можемо поміняти місцями

$$= \bigvee_{k^* \in K} g_{t^* t_1}(k^*, k_{t_1}) \,\&\, g_{t^* t_2}(k^*, k_{t_2}) \,\&\, \bigvee_{\substack{t \in N_{t^*} \setminus \{t_1, t_2\} \\ k : N_{t^*} \setminus \{t_1, t_2\} \to K}} g_{t^* t}(k^*, k_t)$$

Тепер нам не потрібно рахувати диз'юнкцію по всім розміткам

Тому ми можемо змінити місцями $\bigvee$ та $\&$

Давайте уявимо, що $g_{t^* t}(k^*, k_t) \overset{\#}{=} h^*_t(k_t)$

$\Rightarrow$ тепер це ніби унарна ф-я, добре вона від k^* залежить, але k^* визначається тут $\#$, і від t^* залежить, але t^* ми зафіксували $\Rightarrow$ змінюються тільки t і k_t

Уявимо собі якесь інше напівкільце: $(\max, +)$

$$\sum_{t' \in T} \max_{k : T} h_{t'}(k_{t'}) = \max_{k : T \to K} \sum_{t' \in T} h_{t'}(k_{t'})$$ ← така вл. справедлива для $\forall$ напівкільця якщо визначити нез-ні змінні.

$$|T| \cdot |K| \xleftrightarrow{\text{компоненти не залежні}} |K|^{|T|} \cdot |T|$$

Тому ми це зробили $\#$

$$\Leftrightarrow \bigvee_{k^* \in K} g_{t^* t_1}(k^*, k_{t_1}) \,\&\, g_{t^* t_2}(k^*, k_{t_2}) \,\&_{t \in N_{t^*} \setminus \{t_1, t_2\}} g_t(k^*) \quad \begin{matrix} \forall t_1, t_2 \in N_{t^*} \\ \forall k_{t_1}, k_{t_2} \in K \end{matrix}$$

← чому б нам не сказати, що $\bigvee_{k \in K} g_{t^* t}(k^*, k_t)$ це нова ф-я g.

Складність алгоритму: $|N_{t^*}|^2 \cdot |K|^2 |K| (|K| + |N_{t^*}|)$ – для одн. t^* !

В найгіршому випадку $|T|^3 \cdot |K|^3 + |T|^2 |K|^4$

Це алгоритм для одного об'єкта !

~~задачу, що інваріантна відносно мажоритарного оператора. Маємо задачу, який вірно відповідає на запитання "вага найкращої розмітки?". Навести ефективний алгоритм пошуку кращої розмітки та показати, що під час роботи цього алгоритму задача залишається інваріантною відносно мажоритарного оператора.~~

Мажоритарний оператор $p(k,k,k') = p(k,k';k) = p(k',k,k) = k$

Приклади маж. оп-ів: медіана, мода

$Z = \langle T, \tau \subseteq T^2, K, g: \tau \times K^2 \to \{0,1\} \rangle$

T — об'єкти, τ — структура сусідства, K — мітки, g — дуги

Ми вирішуємо задачу про перевірку не суперечності цієї системи обмежень

$$\bigvee_{k: T \to K} \ \mathop{\&}_{tt' \in \tau} g_{tt'}(k_t, k_{t'}) = \left\{ \begin{array}{l} \text{Виберемо довільним чином} \\ t^* \in T \text{ — центр зірки з променями (листками).} \\ N_{t^*} \subseteq T \end{array} \right\} =$$

$$= \bigvee_{k: T \setminus \{t^*\} \to K} \ \bigvee_{k_{t^*} \in K} \left(\mathop{\&}_{\substack{tt' \in \tau \\ t \neq t^* \\ t' \neq t}} g_{tt'}(k_t, k_{t'}) \ \& \mathop{\&}_{t \in N_{t^*}} g_{tt^*}(k_t, k_{t^*}) \right) =$$

Тільки ті пари, що не є променями зірки

це константа відносно k_{t^*}

не залежить від t^* => можна винести за V; дистрибутивність

$$= \bigvee_{k: T \setminus \{t^*\} \to K} \ \mathop{\&}_{\substack{tt' \in \tau \\ t \neq t^* \\ t' \neq t^*}} \left[g_{tt'}(k_t, k_{t'}) \ \& \left\{ \bigvee_{k_{t^*} \in K} \ \mathop{\&}_{t \in N_{t^*}} g_{tt^*}(k_t, k_{t^*}) \right\} \right] =$$

перетворення зірки на симплекс дає нам g'

$$= \bigvee_{k: T \setminus \{t^*\} \to K} \ \mathop{\&}_{\substack{tt' \in \tau \\ t \neq t^* \\ t' \neq t^*}} \left[g_{tt'}(k_t, k_{t'}) \ \& \left\{ \mathop{\&}_{\substack{tt' \in \tau \\ t \in N_{t^*} \\ t' \in N_{t^*}}} g'_{tt'}(k_t, k_{t'}) \right\} \right] =$$

$$= \left\{ g^*_{tt'}(k_t, k_{t'}) = \begin{cases} g_{tt'}(k_t, k_{t'}), & t \notin N_{t^*} \wedge t' \notin N_{t^*} \\ (g \,\&\, g')_{tt'}(k_t, k_{t'}), & t \in N_{t^*} \vee t' \in N_{t^*} \end{cases} \right.$$

Скориставшись комутативністю операції &, згрупували дуги

$$= \bigvee_{k: T \setminus \{t^*\} \to K} \ \&_{\substack{t,t' \in T \\ t \neq t^* \\ t' \neq t^*}} g^*_{tt'}(k_t, k_{t'}).$$

Алгоритм самоконтролю не порушує інваріантність задачі

Припустимо задача інваріантна відносно оператора p. Виконуємо крок алгоритму пошуку розмітки із самоконтролем.

Знаходимо k_t^*.

if $g_t(k_t^*) = 0 \Rightarrow$ йдемо до наступного k_t^*

if $g_t(k_t^*) = 1 \Rightarrow$ необхідно $g_t(p(k_t^*, \dots, k_t^*)) = 1$

Для цього оператора має виконуватись ідемпотентність.

а оператор мажоритарний є ідемпотентним.

$\forall$ задачі, що інв. відн. ідемпотентного оператора алгоритм пошуку розмітки з самоконтролем не порушує інваріантність

Ідемпотентність оператора p — це достатня умова інваріантності.

Ідемпотентність означає, якщо ми цьому оператору подали один і той самий аргумент декілька разів, то він його і видасть. $p(k, k) = k$, $x \circ x = x$.

для max-min

$$g_{tt'}(k_t, k_{t'}) = \max_{\substack{k_{t^*} \in K \\ t^* \neq t \\ t^* \neq t'}} \gamma(k) \ ; \ \gamma(k) = \min_{t'' \in T} g_{tt''}(k_t, k_{t''})$$

Вибираємо який завгодно об'єкт $t^* \in T$ в якості центра зірки.

$$\forall k: N_{t^*} \to K \quad \min_{\substack{t \in N_{t^*} \\ t' \in N_{t^*} \\ t \neq t'}} g'_{tt'}(k_t, k_{t'}) = \max_{k^* \in K} \min_{t \in N_{t^*}} g_{t^*t}(k^*, k_t)$$

$$g'_{tt'}(k_t, k_{t'}) = \max_{k^* \in K} \left[\min \left\{ g_{t^*t}(k^*, k_t), g_{t^*t'}(k^*, k_{t'}), \min_{t^\circ \in N_{t^*} \setminus \{t, t'\}} \max_{k \in K} g_{t^*t^\circ}(k^*, k) \right\} \right]$$

Зірку в симплекс ми повторюємо до тих пір поки не закінчаться об'єкти. $\Rightarrow$ відновлюємо центр останньої зірки $\Rightarrow$
$\Rightarrow$ відновлюємо центр передостанньої зірки.

Scanned with CamScanner

Навести алгоритм викреслювання третього порядку. Показати, що застосування алгоритму дає той самий результат, що й перетворення зірки в симплекс

OR-AND

Ініціалізація : $g^0_{tt'}(k,k') = g_{tt'}(k,k')$, $\forall tt' \in \tau, \forall k \in K, \forall k' \in K$.

$\tau' = \{ \{t,t'\} : t \neq t', t \in T, t' \in T \}$ – всі об'єкти сусіди. невпорядковані пари об'єктів.

$g^0_{tt'}(k,k') = 1 \quad \forall tt' \in \tau' \setminus \tau, \forall k \in K, \forall k' \in K$

$i = 1.$

$$g^i_{tt'}(k,k') = g^{i-1}_{tt'}(k,k') \,\&\, \underset{t^* \in T \setminus \{t,t'\}}{\&} \; \bigvee_{k^* \in K} \left[g^{i-1}_{tt^*}(k,k^*) \,\&\, g^{i-1}_{t't^*}(k',k^*) \right]$$

Ми хочемо, щоб в кожному сусіді крім tt' хоча б одна мітка приводила до допустимої дуги.

~~Доз~~ Припустимо що в нас була допустима розмітка, ми робимо всі об'єкти сусідами. і там де не було дуг ми додали дуги $\Rightarrow$ значить всі мітки які не були в сусідніх об'єкта t вони одна з одним узгоджені $\Rightarrow$ розмітка залишилась допустимою

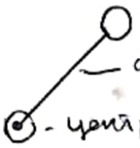
Далі: берем не доп. розмітку $\Rightarrow$ це значить є пара сусідніх об'єктів які з'єднані не допустимою дугою.

Згадаймо зірку в симплекс:

$$g^i_{tt'}(k,k') = g^{i-1}_{tt'}(k,k') \,\&\, \bigvee_{k^* \in K} g_{tt^*}(k,k^*) \,\&\, g_{t't^*}(k',k^*) \,\& \underset{t^0 \in N^{i-1}_{t^*} \setminus \{t,t'\}}{\&} \; \bigvee_{k^0 \in K} g_{t^*t^0}(k^*,k^0)$$

Це перетворення може занулити ті дуги, які раптом стали не допустимі через те, що в нас через зірку не проходить допустима. розмітка. А тепер ми зробили сусідами ~~більше~~ інші дуги в усіх дугах вага зможе змінитись тільки в тому випадку якщо до них були приєднані промені зірки, які не створюють "гарну" розмітку.

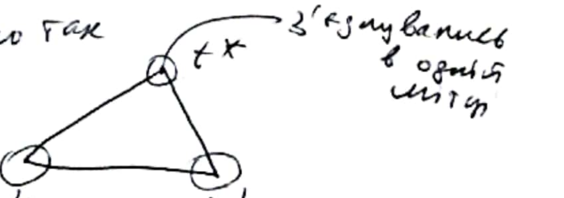
?



— фіксуємо дуга з'єднує недоп. мітки $\Rightarrow$ після викреслення нічого поганого не відбулось.

- центр

Викреслювання 3-го порядку хоче щоб було так



Логіка така: ~~Якщо у нас дуга з'єд~~

Якщо із якоїсь вершини виходять тільки нульові дуги в якийсь об'єкт то викреслювання #3 порядку обов'язково їх викреслить, до у нас не створюється трикутник.

$$\boxed{O(|K^2T^2| \cdot |T| \cdot |K|) \sim O(|KT|^3)}$$

оцінка ресурсної скл. зверху

Scanned with CamScanner

Для Max-Min.
Ініціалізація
$g^0_{tt'}(k,k') = g_{tt'}(k,k'), \forall tt' \in T, \forall k \in K, \forall k' \in K$

$T' = \{\{t,t'\}: t \neq t', t' \in T\}$ – всі об'єкти сусіди.
невпорядковані пари об'єктів.

$g^0_{tt'}(k,k') = +\infty$

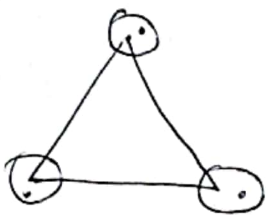
$$g^i_{tt'}(k,k') = \min\left\{ g^{i-1}_{tt'}(k,k'), \min_{t^* \in T \setminus \{t,t'\}} \max_{k^* \in K} \min\left[g^{i-1}_{tt^*}(k,k^*), g^{i-1}_{t't^*}(k',k^*)\right]\right\}$$

Нам потрібно, щоб вага розмітки не змінилась, а вага розмітки в першу чергу визначається найменшою дугою => тому ми ставимо $+\infty$.

Згадаємо зірку в симплекс.

$$g^i_{tt'}(k,k') = \min\left\{ g^{i-1}_{tt'}, \max_{k^* \in K}\left[\min\left(g_{tt^*}(k,k^*), g_{t't^*}(k',k^*), \min_{t^0 \in N^{i-1}_{t^*} \setminus \{t,t'\}} \max_{k^0 \in K} g_{t^*t^0}(k^*,k^0)\right)\right]\right\}$$

Зірка в симплекс перевіряє „допустимість Δ"
тобто щоб промені йшли в одну мітку



Викреслювання 3-го порядку
це теж робить те чого треба задача
Оскільки все, що викреслить викреслення 3-го порядку і викреслить зірка в симплекс.
Іншими словами коли ми повністю пропустимо викреслювання 3-го порядку зірка в симплекс не змінить жодної дуги.
По суті викреслювання 3-го порядку зберігає в собі частину зірки в симплекс. Друга частина перетворення зірки в симплекс це викреслювання 2-го порядку (перевірка на узгодженість дуг, що проходять через центр зірки). Що зазвичай вирішує викреслення 2-го порядку, але викреслення 3-го порядку працює краще за 2-ий порядок

Scanned with CamScanner