Wydział Podstawowych Problemów Techniki
Politechniki Wrocławskiej

Wybrane problemy odpornej optymalizacji dyskretnej z możliwością modyfikacji

Tomasz Strzałka

Praca magisterska napisana
pod kierunkiem
dr hab. Pawła Zielińskiego, prof. PWr

Politechnika
Wrocławska
Wrocław 2016

Podziękowania chciałbym skierować do mojego promotora,
dr hab. Pawła Zielińskiego, prof. nadzw. Politechniki Wrocławskiej,
za nakład włożonej pracy w korektę tekstu,
poświęcony na konsultacjach czas oraz zaangażowanie,
płynące z chęci pomocy w przygotowaniu jak najlepszej pracy dyplomowej.

X,
X
X
X.

Politechnika
Wrocławska

Spis treści

Wstęp

xxx

Część implementacyjna pracy magisterskiej została napisana w języku C++ zgodnie ze standardem ISO-/IEC 14882:2014 z wykorzystaniem środowiska programistycznego Eclipse w wersji 4.5.0 (Mars), debuggerów GDB oraz Valgrind (w wersji 3.10.1). Tekst poniższej pracy został złożony w systemie LaTeX , między innymi z wykorzystaniem podstawowych pakietów do reprezentacji kodu programistycznego: listings oraz algorithm2e. Do składu wszelkich ilustracji użyto aplikacji internetowej draw.io[1] oraz programu Inkscape w wersji 0.91. Do wygenerowania przedstawionych w pracy wykresów dwu- i trójwymiarowych zastosowano programy: Octave (w wersji 3.8.2) oraz croppdf (do korekty otrzymanych wykresów). Całość została skompilowana pod kontrolą systemu Linux Ubuntu 15.04 (Vivid Vervet).

- Trzeba zrobić wstęp o programowaniu liniowym.
  - Klasa problemów 0-1 ($\mathscr{C}$): kombinatroyczne (ścieżka, drzewo), plecak, pokrycie zbioru
  - opowiedzieć o unimodularności, relaksacji Lagrangiana, warunkach komplementarności
- Wstęp o odpornej optymalizacji.

[1] Adres strony internetowej: https://www.draw.io/.

Min–max and min–max regret versions of combinatorial optimization problems: A survey

---

- Paramatry (współczynniki) funkcji celu mogą być niepewne, podchodzimy do tego problemu z pomysłem scenariuszy: albo dyskretnych albo ciągłych (interval scenario case) na przedziałach, gdzie każdy parametr jest określony przez LB i UB.
- Mamy dwa główne modele: ostrożnej —(min-maxowej optymalizacji) i ostrożnej optymalizacji z kryterium żalu (regret criterion):
  - W przypadku dyskretnym mamy $|S| = m$ scenariuszy s, każdy jest reprezentowany wektorem współczynników funkcji celu $c^s = (c_1^s, ...c_n^s)$, gdzie każdy współczynnik jest naturalny (w przypadku problemu 0-1) $\forall s \in S c_i^s \in N, i \in 1, ..., n$.
  - W przypadku scenariuszy ciągłych (opartych na przedziałach) każdy współczynnik c_i może przyjąć wartość z przedziału $[\underline{x_i}, \overline{c_i}]$, gdzie $0 \leqslant \underline{c_i} \leqslant \overline{c_i}$ a zbiór scenariuszy jest produktem kartezjańskim wszystkich z nich.
  - Ekstremalne (krytyczne) scenariusze (extreme scenario) to oczywiście: $\underline{s} = (\underline{c_1}, ..., \underline{c_n})$ i $\overline{s}$, gdzie dla każdego $i \in 1, ..., n$ $c_i = \overline{c_i}$.
  - $c_s(x)$ — wartość rozwiązania dla wektora zmiennych decyzyjnych $x \in X$ dla danego scenariusza $s \in S$.
  - c_s^* — wartość optymalnego rozwiązania dla danego scenariusza s.
  - $x_{(s)}^*$ — wektor wartości zmiennych decyzyjnych dla optymalnego rozwiązania dla scenariusza s i $c_s^* = c_s(x_{(s)}^*)$.
  - W pierwszym przypadku dla dyskretnych scenariuszy po prostu liczymy każdy scenariusz i wybieramy takie rozwiązanie, które da nam minimalną wartość dla najgorszego z możliwych scenariuszy spośród optymalnych rozwiązań ($c(x^*) = \min_{x \in X} max_{s \in S} c_s(x)$). Bez optymalizacji musimy wziąć każde możliwe rozwiązanie $x \in X$, wyliczyć dla wszystkich sceniariuszy $s \in S$ wartości $c_s(x)$, wybrać z tak powstałych grup ($C_S^x = (c_{s_1}(x), ..., c_{s_m}(x))$ - m scenariuszy) max i z tak powstałej grupy: $C_X^{max} = (\max C_S^{x_1}, ..., \max C_S^{x_{2^k}})$ (k binarnych zmiennych decyzyjnych dla problemu $\mathscr{P} \in \mathscr{C}$) wybrać taki $x \in X$, że $c(x) = \min C_X^{max}$. Czyli po polsku liczymy dla wszystkich możliwych $x \in X$ dla którego scenariusza $s \in S$ $c_s(x)$ osiągnie największą wartość a potem wybieramy takiego x, dla którego ta wartość maksymalna jest najmniejsza spośród pozostałych. Czyli chcemy zminimalizować straty w najbardziej pesymistycznym przypadku
  - W przypadku min-max regret criteria nie chcemy minimalizować strat bez względu na dziejący się sceiariusz, lecz jesteśmy zainteresowani wyborem takiego rozwiązania, które sprawi, że będziemy jak najmniej stratni w przypadku wystąpienia dowolnego scenariusza. Np. mając dane rozwiązanie x, jeżeli jest ono dobrane pod dany scenariusz s to regret wyniesie 0. W przeciwnym przypadku najprawdopodobniej istnieje lepsze rozwiązanie dla danego scenariusza s'. Naszym celem jest wybranie takiego rozwiązania, które będzie tracić jak najmniej do optymalnego rozwiązania w przypadku scenariusza, w którym to dane rozwiązanie traci do optimum dla tego scenariusza najwięcej.

- $r_s(x) = c_s(x) - c_s^*$ — straty poniesione w wyniku wyboru danego rozwiązania $x \in X$ w przypadku wystąpienia sceniariusza $s \in S$ względem wyboru optymalnego rozwiązania pod dany scenariusz. Liczymy min-max, więc wartość optymalna dla problemu c_s^* będzie wartością najmniejszą: $\forall x \in X c_s(x) \geqslant c_s(x_s^*)$ dla każdego scenariusza (przez $c_s(x^*)$ często będziemy rozumieć $c_s(x_{(s)}^*)$, gdyż w takim kontekście stosowanie $x_{(s)}$ jest zbędne) — stąd też w $r_{max}(x) = \max s \in S c_s(x) - c_s^*$ dla każdego $s \in S$ $c_s(x) \geqslant c_s^*$. Im gorszy rezultat, tym różnica między tymi wartościami będzie większa.
- $r_{max}(x) = \max s \in S r_s(x)$ — maksymalna strata jaką poniesiemy w przypadku wyboru rozwiązania x względem wartości optymalnych dla poszczególnych scenariuszy.
- $\min x \in X r_{max}(x) = \min x \in X \max s \in S(c_s(x) - c_s^*)$ — chcemy zawsze być jak najbliżej optymalnego rozwiązania (czyli chcemy zminimalizować różnicę w stratach pomiędzy tym przypadkiem a przypadkiem gdybyśmy zawczasu znali mające wydarzyć się scenariusze i odpowiednio wybierali rozwiązania).
- Problemy max-min:
  - max-min — chcemy zmaksymalizować zysk bez względu na to jaki scenariusz się pojawi i jak źle mógłby on wpłynąć na przychody: $\max x \in X \min s \in S c_s(x)$,
  - max-min z żalem to NADAL min-max (bo chcemy minimalizować różnicę między rozwiązaniami optymalnymi a regret, jedyne co się zatem zmienia to sposób liczenia $r_s(x)$ z $r_s(x) = c_s(x) - c_s^*$ na $r_s(x) = c_s^* - c_s(x)$, bo chcemy zyski maksymalizować — w związku z tym wartość optymalna x_s^* zawsze będzie większa bądź równa $c_s(x)$, $r_{max}(x)$ wyznacza nam najgorsze możliwe przypadki dla wektora $x \in X$ niezależnie od scenariusza, tak więc im gorsze dobranie wektora i scenariusza, tym większa różnica pomiędzy c_s^* i $c_s(x)$, tym mniejsza wartość $c_s(x)$ względem c_s^*, tym większa wartość $r_{max}(x)$. Czyli MAX-MIN dla regret to: $\min x \in X r_{max}(x) = \min x \in X \max s \in S c_s^* - c_s(x)$.
  - Dla rozróżnienia możemy wprowadzić oznaczenia funkcji $r_{max}(x)$ dla problemów min-max i max-min:
    - MIN-MAX: $r_{max}^{*min} = c_s(x) - c_s^*$,
    - MAX-MIN: $r_{max}^{*max} = c_s^* - c_s(x)$.

- min-max z relatywnym żalem zmienia tylko sposób liczenia r_{max}^{*min} i r_{max}^{*max}:
  - MIN-MAX: $r_{max}^{\overset{*min}{*}} = \frac{r_{max}^{*min}}{c_s^*} = \frac{c_s(x) - c_s^*}{c_s^*}$,
  - MAX-MIN: $r_{max}^{\overset{*max}{*}} = \frac{r_{max}^{*max}}{c_s^*} = \frac{c_s^* - c_s(x)}{c_s^*}$.

- Można zauważyć, że jeżeli weźmiemy $\hat{r_{min}} = r_{max}^{\overset{*min}{*}} + 1$ to $\max s \in S \hat{r_{min}} = \frac{c_s(x) - c_s^*}{c_s^*} + \frac{c_s^*}{c_s^*} = \max s \in S \frac{c_s(x)}{c_s^*} = \max_{s \in S} \frac{c_s(x)}{\min_{y \in X} c_s(y)} = \max_{s \in S} \max_{y \in X} \frac{c_s(x)}{c_s(y)}$

- W przypadku min-max/max-min ze scenariuszami o rozmytych parametrach (przedziały) liczymy po prostu min/max po scenariuszu $\overline{s}/\underline{s}$, gdyż w przypadku min-max ($\min_{x \in X} \max_{s \in S} c_s(x)$) mamy pewność, że $\forall_{s \in S} c_{\overline{s}}(x) \geqslant c_s(x)$ (jako, że $\forall_{c_i \in \overline{s}} \forall_{s \neq \overline{s}} \forall_{c_i' \in s} c_i' \leqslant c_i$), więc $\min_{x \in X} \max_{s \in S} c_s(x) = \min_{x \in X} c_{\overline{s}}(x)$. Analogicznie w przypadku max-min.

- W przypadku min-max regret sytuacja też nie za wiele się zmienia

2.1 Przykłady

.... jaka jest różnica między klasycznym podejściem do problemu a minimaxowym? A weźmy sobie graf z $2n$ wierzchołkami, tak jak na rysunku Fig. 1. W klasycznym podejściu (w przypadku jednego scenariusza) odnaleźlibyśmy optymalne rozwiązanie dla tego scenariusza i zwrócili rozwiązanie. W przypadku kilku scenariuszy sam nasuwa się pomysł policzenia optymalnego rizwiązania dla każdego ze scenariuszy osobno

i	w_i	x_i
1	3	0–1
2	5	0–1
3	2	0–1
4	4	0–1
5	5	0–1
6	3	0–1

(a)

	$\underline{s}$	$\overline{s}$
c_1	3	5
c_2	2	6
c_3	2	5
c_4	2	3
c_5	3	9
c_6	1	6

(b)

	s_1	s_2	s_3
c_1	4	3	3
c_2	8	4	6
c_3	5	3	3
c_4	3	2	4
c_5	2	8	2
c_6	4	6	2

(c)

Rysunek 2.1: Wygenerowana ilustracja grafu przez skrypt makeGraphViz.bash. **(a)** Kod pośredni pomiędzy danymi wejściowymi w standardowym formacie DIMACS Implementation Challenge a finalnym rysunkiem. Kod pośredni jest zapisany w języku DOT. **(b)** Rysunek grafu wygenerowany przez program GraphViz.

i wybranie najlepszego z nich (o najmniejszej wartości). Jest to duży błąd, bo w takim przypadku dla 1 scenariusza (lewe wartości etykiet krawędzi) optymalnym rozwiązaniem jest ścieżka 1, $n+1$, ... $2n$, której długość wynosi 0. Dla drugiego scenariusza mamy sytuację taką samą - optymalnym rozwiązaniem w tym przypadku jest górna ścieżka, która także ma łączną długość 0.

Dla problemu minimaxowego natychmiast otrzymujemy, że dla każdego możliwego rozwiązania $x \in X$ ($|X| = 3$ - albo górą, albo dołem, albo środkiem) dla pierwszych dwóch przypadków (rozwiązań optymalnych dla osobnych scenariuszy) natychmiast otrzymujemy, że $max_{s\in S} c_s\left(x^*_{(1)}\right) = max\left\{0, \sum_{i=0}^{n-1} 2^{n-1}\right\} = 2^n - 1$ oraz $max_{s\in S} c_s\left(x^*_{(2)}\right) = max\left\{\sum_{i=0}^{n-1} 2^{n-1}, 0\right\} = 2^n - 1$ (gdzie $x^*_{(1)}$ i $x^*_{(2)}$ to optymalne rozwiązania odpowiednio dla scenariusza 1 i 2.), więc $min_{x\in\left\{x^*_{(1)}, x^*_{(2)}\right\}}\{2^n - 1, 2^n - 1\} = 2^n - 1$. Celowo nie uwzględniliśmy trzeciej możliwości: przejścia przez środek o wartościach $(1, 1)$. Tutaj, niezależnie od scenariusza maximum po trzecim rozwiązaniu (x_3) będzie wynosić 1, stąd $min_{x\in\left\{x^*_{(1)}, x^*_{(2)}, x_3\right\}}\{2^n - 1, 2^n - 1, 1\} = 1$. Stąd widzimy, że dla rozpatrywanych osobno scenariuszy różnica pomiędzy najlepszą (0) a najgorszą wartością (jeżeli zmieniłby się nagle scenariusz na przeciwny) wynosi aż $2^n - 2$. Za to w przypadku minimaxowym koszt podjęcia innej decyzji niż optymalna dla scenariusza (żal) wynosi co najwyżej 1. Różnica między tymi wartościami jest wykładnicza i wynosi $2^n - 2$. Podejście minimaxowe wydaje sie zatem o niebo lepsze.

Co z innym podejściem? Na przykład możemy przypuszczać, że zamiast rozpatrywać każdy scenariusz osobno, możemy wziąć „uśredniony" scenariusz tzn. taki, że każdy współczynnik będzie średnią arytmetyczną tego współczynnika ze wszystkich scenariuszy. Pokazać można niestety nierówność, która nie gwarantuje nam, że dla optymalnego rozwiązania dla problemu I', w przypadku dobrania najgorszego scenariusza, otrzymane rozwiązanie będzie równe optymalnemu rozwiązaniu problemu minimaxowego - będzie ono co najwyżej mniejsze od k krotności wartości tego rozwiązania, co dopuszcza otrzymywanie nawet k krotnie gorszych rozwiązań niż w przypadku minimaxowym. Dowód Propozycji 1 jest prosty. $val'(x,s) = \sum_{i=1}^{|X|} x_i * c'_i$, a skoro każde c'_i jest średnią arytmetyczną: $\sum_{s=1}^{k} \frac{c_i^s}{k}$ to $\frac{1}{k}$ możemy wyciągnąć przed $val(x,s)$, podobnie jak sumę $\left(\sum_{s\in S} \frac{1}{k} val\,(x,s) = val'(x,s)\right)$. Wystarczy rozpisać. $L = \min_{x\in X}\left(\frac{1}{k}\sum_{s\in S} val\,(x,s)\right)$ (chcemy minimalizować) oczywiście jest $\leqslant \min_{x\in X}\left(\frac{1}{k}\cdot k\cdot \max_{s\in S} val\,(x,s)\right) = opt\,(I)$.

Kolejnym pomysłem jest zastąpienie „średniego" scenariusza przez najgorszy scenariusz. Tutaj także pokażemy, że rozwiązanie optymalne dla instacji problemu z tak zdefiniowanym scenariuszem nie jest dobre, bo dla wyliczonego optymalnego rozwiązania x', wartość takiego rozwiązania w przypadku wzięcia najgorszego scenariusza ze zbioru scenariuszy problemu I $(\max_{s\in S} val\,(x', s))$ może być nawet k razy gorsza w porównaniu z optymalnym rozwiązaniem problemu minimaxowego. Skorzystamy tu z szeregu przekształceń: $\max_{s\in S}\sum_{i=1}^{n} c_i^s \cdot x'_i$ (bierzemy TYLKO wektor x_i jako optymalne rozwiązanie naszego wydumanego problemu). To jest oczywiście mniejsze od tego, gdy faktycznie weźmiemy współczynniki „najgorszego" scenariusza. $\sum_{i=1}^{n} c'_i \cdot x'_i$ jest optymalną wartością dla problemu z „najgorszym" scenariuszem, więc dla każdego innego x'a ta suma będzie lepsza lub taka sama (w szczególności dla x^*): $\leqslant \sum_{i=1}^{n} c'_i \cdot x^*_i$. To jest oczywiście mniejsze od $\sum_{i=1}^{n}\sum_{s\in S} c_i^s \cdot x^*_i = \sum_{s\in S}\sum_{i=1}^{n} c_i^s \cdot x^*_i$ (każde $c'_i = \max_{s\in S} c_i^s$, więc jest na pewno mniejsze od sumy tych współczynników z wszystkich scenariuszy), a to z kolei jest mniejsze od k krotnej wielokrotności maxa po scenariuszach: $k \cdot \max_{s\in S}\sum_{i=1}^{n} c_i^s \cdot x^*_i$. Tutaj mamy już wartość optymalną problemu I -

$\min_{x\in X}\max_{s\in S}\sum_{i=1}^{n}c_i^s\cdot x_i=\max_{s\in S}\sum_{i=1}^{n}c_i^s\cdot x_i^*$

Ta nierówność nie obowiązuje przy minimax regret, jak pokazano na Fig. 2. W myśl tego rysunku mamy dwa scenariusze + trzeci - pesymistyczny. W dwóch pierwszych przypadkach optymalnym rozwiązaniem jest ścieżka dolna - oczywistym jest, że $0+2^n<2^n+2^n$ (dla pierwszego scenariusza) oraz, że $2^n+1+0<2^n+2^n$ (dla drugiego). W związku z tym, że dla obydwu scenariuszy mamy to samo rozwiązanie, regret wynosi 0 $(min_{x\in X}\max_{s\in S}(val(x,s)-val_s^*)=min_{x\in X}\max_{s\in S}(c_s(x)-c_s^*)=\max_{s\in S}(c_s(x^*)-c_s^*)=\max_{s\in S}\{(2^n+1+0)-(2^n+1+0),(0+2^n)-(0+2^n)\}=\max_{s\in S}\{0,0\}=0$ W przypadku zaś „pesymistycznego" scenariusza koszt dolnej ścieżki jest większy niż górnej $(2^n+1+2^n>2\cdot 2^n)$ w związku z czym optymalnym rozwiązaniem będzie ścieżka górna. Jedna w przypadku tego rozwiązania)optymalnego dla pesymistycznego scenariusza), dla pozostałych przypadków to rozwiązanie będzie traciło do optimum $2\cdot 2^n-(2^n+1)$ i $2\cdot 2^n-2^n$, odpowiednio dla pierwszego oraz drugiego scenariusza. Zatem max regret wyniesie w tym przypadku 2^n (co jest znacznie gorszym wynikiem niż k razy).

Co ciekawe, można zauważyć, że optymalne rozwiązanie problemu z „pesymistycznym" scenariuszem jest rozwiązaniem problemu z wąskim gardłem tzn. zamiast fuckje celu definiować jako sumę i minimalizować $\min_{x\in X}\sum_{i=1}^{n}c_i\cdot x_i$ to minimalizuje się $\max_{i\in\{1,\dots,n\}}c_i\cdot x_i$ - wynik nie zależy od całości tylko od najbardziej obiążającej wynik pary współczynnik-wartość zmiennej - słowem od wąskiego gardła danego problemu. Stąd też klasy problemów: DISCRETE MIN–MAX (REGRET) BOTTLENECK $\mathscr{P}$ i INTERVAL MIN–MAX (REGRET) BOTTLENECK $\mathscr{P}$ od swoich standardowych odpowiedników różnią się tylko wartością $val(x,s)=c_s(x)$ ($\max_{i\in\{1,\dots,n\}}c_i\cdot x_i$ zamiast $\sum_{i=1}^{n}c_i\cdot x_i$). Łatwo to dowieść, gdyż: optymalna wartość problemu DISCRETE MIN–MAX (REGRET) BOTTLENECK $\mathscr{P}$ $x^*=\min_{x\in X}\max_{s\in S}val(x,s)=\min_{x\in X}\max_{s\in S}\left(\max_{i\in\{1,\dots,n\}}c_i^s\cdot x_i\right)$. Po zamienieniu kolejności maxowania i podstawieniu pod $max_{s\in S}c_i^s=c_i'$ (zgodnie z definicją współczynników dla „pesymistycznego" scenariusza) otrzymamy $\min_{x\in X}\max_{i\in\{1,\dots,n\}}c_i'\cdot x_i=\max_{i\in\{1,\dots,n\}}c_i'\cdot x_i^*$, co jest dokładnie wartością funkcji celu dla optymalnego rozwiązania problemu z „pesymistycznym" scenariuszem.

Proposition 4 pokazuje jak redukować przypadek z regret. Analogicznie. Raczej powinno być $\overline{c}_i^s=\max\left\{c_i^s-\frac{val_s^*}{x_i},0\right\}$, bo inaczej ten dowód się sypie.

Jest także naturalna pokusa myśleć o scenariuszu jak o specjalnym przypadku funkcji celu i chęci porównania minimaxa z problemem o wielu funkcjach celu. W tym wypadku, dla problemu P z k scenariuszami mamy k funkcji celu ze współczynnikami $c_1^h,\cdots c_n^h$ dla scenariusza/funkcji celu h. Wartość $h$tej funkcji celu dla rozwiązania x: $val(x,h)=\sum_{i=1}^{n}c_i^h\cdot x_i$. Możemy przyjąć, że wszystkie f. celu są zorientowane na minimalizację. Mówimy, że rozwiązanie x jest słabo zdominowane przez strategię (rozwiązanie) y, jeżeli dla każdego scenariusza $h=1,...,k$ zachodzi $val(y,h)\geqslant val(x,h)$ i przynajmniej w 1 przypadku jest to ostra nierówność. Mówimy, że x jest mocno zdominowane przez y jeżeli dla każdego scenariusza wartość funkcji celu dla x jest ostro mniejsza od wartości tej funkcji dla tego samego scenariusza dla rozwiązania y. Efektywnymi rozwiązaniami tego problemu są osiągalne rozwiązania dla których nie istnieje już inne osiągalne rozwiązanie, które by je dominowało.

Propozycja 5 sugeruje, że jeśli x dominuje y to oczywiście $\max_{s\in S}val(x,s)\leqslant\max_{s\in S}val(y,s)$ (minimalizujemy). Jako że x dominuje y to dla każdego scenariusza h zachodzi $val(x,h)\leqslant val(y,h)\leqslant\max_{s\in S}val(y,s)$. Zatem wartość funkcji celu dla strategii x dla dowolnego scenariusza jest mniejsza od $\max_{s\in S}val(y,s)$ (jako, że każda z osobna jest od tego wyrażenia mniejsza). W szczególności $\forall h\in\{1,\cdots,k\}\ \ val(x,h)\leqslant\max_{s\in S}val(x,s)\leqslant\max_{s\in S}val(y,s)$. Propozycja jest teraz taka, że jeśli jest choć jedno optymalne rozwiązanie dla DISCRETE MIN–MAX to jest ono efektywnym rozwiązaniem problemu z wieloma funkcjami celu, co pokazuje Fig. 3., na którym punkty przedstawiają rozwiązania x oraz wartości poszczególnych funkcji celu dla scenariuszy s_1 oraz s_2. Wartość optymalna dla DISCRETE MIN–MAX to oczywiście takie x, że minimalizuje ono wyrażenie $\max_{s\in S}val(x,s)$ (wynika bezpośrednio z definicji). Prowadząc zatem prostopadłe linie do obu osi współrzędnych, a przecinające się w punkcie gdzie $val(x,s_1)=val(x,s_2)$, zaczynając od początku układu współrzędnych, znajdujemy pierwsze takie rozwiązanie. Daje ono nam najmniejszą wartość dla scenariusza s_2. Wszystkie punkty poza kwadratem dają większą wartość funkcji celu (dla obu bądź tylko jednego scenariusza - jakiś punkt może dla 1 scenariusza przyjmować mniejszą od znalezionej wartość, ale że nas interesuje $\max_{s\in S}val(x,s)$), to ten sam punkt dla 2 scenariusza daje nam ostatecznie gorszy wynik). Zaznaczone trójkątami punkty symbolizują efektywne rozwiązania - dla żadnego z tych punktów nie istnieje inne rozwiązanie, które dawałoby mniejszą wartość funkcji celu dla WSZYSTKICH scenariuszy - co najwyżej może dominować dla jednego z dwóch scenariuszy lub dla żadnego.

Podobnie propozycja 6. sugeruje, że jeśli mamy przynajmniej jedno rozwiązanie optymalne dla DISCRETE

MIN–MAX REGRET, to na pewno jest ono jednym z rozwiązań efektywnych dla problemu z wieloma funkcjami celu. Oczywistym jest, że jeśli $x \in X$ dominuje $y \in X$ to z tego, że dla każdego $s \in S$ zachodzi $val(x,s) \leqslant val(y,s)$ to dla każdego $s \in S$ zachodzi $r_{(s)}(x) = val(x,s) - val_s^* \leqslant val(y,s) - val_s^* = r_{(s)}(y)$. Z rozumowania z poprzedniej propozycji mamy, że także zachodzi $\max_{s\in S} val(x,s) \leqslant \max_{s\in S} val(y,s)$, a więc i $R_{max}(x) = \max_{s\in S}(val(x,s) - val_s^*) \leqslant \max_{s\in S}(val(y,s) - val_s^*) = R_{max}(y)$. Tutaj także trzeba doprecyzować, że skoro dla każdego $s \in S$ zachodzi $r_{(s)}(x) \leqslant r_{(s)}(y)$ to szczególnie też zachodzi $r_{(s)}(x) \leqslant \max_{d\in S} r_{(s)}(y) = R_{max}(y)$. Skoro dla każdego $s \in S$ zachodzi $r_{(s)}(x) \leqslant R_{max}(y)$ to w szczególności dla $\max_{s\in S} r_{(s)}(x) = R_{max}(x)$ równanie to też jest prawdziwe. Czyli $R_{max}(x) \leqslant R_{max}(y)$. W związku z tym rozwiązaniem problemu DISCRETE MIN–MAX REGRET jest na pewno jedno z efektywnych rozwiązań problemu z wieloma funkcjami celu (czyli takie rozwiązania x dla których nie istnieje takie y, że $R_{max}(y)$ byłoby lepsze/mniejsze od $R_{max}(x)$). Pozostaje nam więc tylko znaleźć spośród tych rozwiązań efektywnych takie, które jest minimalne ($\min_{x\in X} R_{max}(x) = \min_{x\in X} \max_{s\in S}(val(x,s) - val_s^*)$).

Pokazuje to rysunek Fig. 4, gdzie nowym układem odniesienia jest układ rozpoczynający się w punkcie będącym odpowiednikiem optymalnych rozwiązań dla obu scenariuszy. Szukamy takiego rozwiązania, które minimalizuje nam $\max_{s\in\{s_1,s_2\}}(val(x,s) - val_s^*)$ (czyli rozwiązania będącego najbliżej zarówno jednej jak i drugiej niebieskiej osi). Widzimy także, że rozwiązania efektywne nie są dominowane przez żadne z pozostałych punktów dla obu scenariuszy naraz.

2.1.1 Interval scenario case

Dla INTERVAL MIN–MAX dla minimalizacji rozwiązaniem optymalnym było rozwiązanie scenariusza najgorszego (o najwyższych kosztach: $\overline{c} = \{\overline{c_i}, \cdots, \overline{c_n}\}$), a dla maksymalizacji - o najmniejszych.

Dla INTERVAL MIN–MAX REGRET propozycja 7 sugeruje, że aby policzyć maksymalny żal rozwiązania $x \in X$ problemu P, wystarczy rozpatrzyć najgorszy z dwóch skrajnych scenariuszy (odpowiednio $\overline{c}$ dla problemu minimalizacyjnego oraz $\underline{c}$ dla maksymalizacji).

Proponuje wziąć najgorszy scenariusz:

$$c_i^-(x) = \begin{cases} \overline{c_i} & \text{if} \quad x_i = 1, \\ \underline{c_i} & \text{if} \quad x_i = 0, \end{cases} \quad i = 1, \cdots, n. \tag{2.1}$$

Mając rozwiązanie $x \in X$ niech $\mathbb{1}(x) = \{i \in \{1, \cdots, n\} : x_i = 1\}$. Teraz dla każdego scenariusza pokażemy, że zachodzi $R(x,s) \leqslant R(x, c_i^-(x))$, czyli, że wartość regret jest największa dla najgorszego scenariusza. Innymi słowy, że dla tego scenariusza, $R(x, c_i^-(x)) = R_{max}(x)$. Optymalne rozwiązanie x zatem będzie to takie rozwiązanie, które spełni $\min_{x\in X} R_{max}(x)$, czyli będzie rozwiązaniem optymalnym problemu INTERVAL MIN–MAX REGRET.

$\forall s \in S R(x,s) = val(x,s) - val_s^* = val(x,s) - val(x_s^*, s)$. W propozycji 7 rozpatrujemy (tak jak i w całym dokumencie) problem binarny, gdzie $\forall i x_i \in \{0,1\}$ zatem $val(x,s) = \sum_{i:x_i=1} c_i \cdot 1 + \sum_{i:x_i=0} c_i \cdot 0 = \sum_{i:x_i=1} c_i$. Możemy zatem napisać, że $val(x,s) - val(x_s^*, s) = \sum_{i\in\mathbb{1}(x)} c_i^s - \sum_{i\in\mathbb{1}(x_s^*)} c_i^s$. Każdy współczynnik, który pojawił się w sumie miał przyporządkowanego $x_i = 1$, zgodnie z definicją $\mathbb{1}(x)$. Możemy zauważyć, że takie c_i^s, dla którego $i \in \mathbb{1}(x)$ oraz $i \in \mathbb{1}(x_s^*)$ skrócą się, więc możemy zapisać równoważnie: $\sum_{i\in\mathbb{1}(x)} c_i^s - \sum_{i\in\mathbb{1}(x_s^*)} c_i^s = \sum_{i\in\mathbb{1}(x)\setminus\mathbb{1}(x_s^*)} c_i^s - \sum_{i\in\mathbb{1}(x_s^*\setminus\mathbb{1}(x))} c_i^s$. To zaś na pewno jest mniejsze od $\sum_{i\in\mathbb{1}(x)\setminus\mathbb{1}(x_s^*)} \overline{c_i} - \sum_{i\in\mathbb{1}(x_s^*)\setminus\mathbb{1}(x)} \underline{c_i}$, jako że $\overline{c_i}$ to największe wartości spośród scenariuszy dla i, a $\underline{c_i}$ - najmniejsze, zaś zbiór po którym sumujemy się nie zmienia. Teraz zauważyć trzeba, że $\sum_{i\in\mathbb{1}(x)\setminus\mathbb{1}(x_s^*)} \overline{c_i} - \sum_{i\in\mathbb{1}(x_s^*)\setminus\mathbb{1}(x)} \underline{c_i} = val(x, c_i^-(x)) - val(x_s^*, c_i^-(x))$.

$val(x, c_i^-(x)) = \sum_{i\in\mathbb{1}(x)} c_i^-(x) + \sum_{i\notin\mathbb{1}(x)} 0 = \sum_{i\in\mathbb{1}(x)} \overline{c}_i = \sum_{i\in\mathbb{1}(x)\setminus\mathbb{1}(x_s^*)} \overline{c}_i + \sum_{i\in\mathbb{1}(x_s^*)} \overline{c}_i$. Przy czym część $\sum_{i\in\mathbb{1}(x_s^*)} \overline{c}_i$ odnosi się do tych elementów, które są częścią wspólną $\mathbb{1}(x)$ oraz $\mathbb{1}(x_s^*)$

$val(x_s^*, c_i^-(x)) = \sum_{i\in\mathbb{1}(x_s^*)} c_i^-(x) = \sum_{i\in\mathbb{1}(x_s^*)\setminus\mathbb{1}(x)} c_i^-(x) + \sum_{i\in\mathbb{1}(x)} c_i^-(x) = \sum_{i\in\mathbb{1}(x_s^*)\setminus\mathbb{1}(x)} \underline{c}_i + \sum_{i\in\mathbb{1}(x)} \overline{c}_i$. Przy czym część $\sum_{i\in\mathbb{1}(x)} \overline{c}_i$ odnosi się do tych elementów, które są częścią wspólną $\mathbb{1}(x)$ oraz $\mathbb{1}(x_s^*)$.

Zatem suma $val(x, c_i^-(x)) + val(x_s^*, c_i^-(x)) = \left[\sum_{i\in\mathbb{1}(x)\setminus\mathbb{1}(x_s^*)} \overline{c}_i + \sum_{i\in\mathbb{1}(x_s^*)} \overline{c}_i\right] + \left[\sum_{i\in\mathbb{1}(x_s^*)\setminus\mathbb{1}(x)} \underline{c}_i + \sum_{i\in\mathbb{1}(x)} \overline{c}_i\right] = \sum_{i\in\mathbb{1}(x)\setminus\mathbb{1}(x_s^*)} \overline{c_i} - \sum_{i\in\mathbb{1}(x_s^*)\setminus\mathbb{1}(x)} \underline{c_i}$, ponieważ $\sum_{i\in\mathbb{1}(x_s^*)} \overline{c}_i = \sum_{i\in\mathbb{1}(x)} \overline{c}_i$.

Stąd dostajemy ostatecznie $R(x,s) \leqslant val(x, c_i^-(x)) + val(x_s^*, c_i^-(x))$. Oczywiście $val(x, c_i^-(x)) + val(x_s^*, c_i^-(x)) \leqslant val(x, c_i^-(x)) + val\left(x^*_{c_i^-(x)}, c_i^-(x)\right)$, jako że $val\left(x^*_{c_i^-(x)}, c_i^-(x)\right) = val^*_{c_i^-(x)}$ (minimalizujemy), a to z kolei równa się już $R(x, c_i^-(x))$.

Pokazaliśmy zatem, że optymalne problemu INTERVAL MIN-MAX P ($\min_{x\in X}\max_{s\in S} R(x,s)$) jest optymalnym rozwiązaniem dla problemu minimalizacyjnego z najgorszym scenariuszem ($\min_{x\in X} R(x,c_{\bar{i}}(x)) = \min_{x\in X}\max_{s\in S} R(x,s)$), bo pokazliśmy, że dla każdego $s\in SR(x,s) \leqslant R(x,c_{\bar{i}}(x))$, czyli, że $R(x,c_{\bar{i}}(x)) = \max_{s\in S} R(x,s)$.

Propozycja 8 teraz sugeruje, że optymlane rozwiązanie x^* dla problemu INTERVAL MIN–MAX REGRET odpowiada rozwiązaniu problemu minimalizacyjnego dla choć jednego ekstremalnego scenariusza, w szczególności dla najlepszego, zdefiniowanego następująco:

$$c_i^+(x^*) = \begin{cases} \underline{c}_i & \text{if} \quad x_i^* = 1, \\ \overline{c}_i & \text{if} \quad x_i^* = 0, \end{cases} \quad i = 1,\cdots,n. \tag{2.2}$$

Weźmy optymalne rozwiązanie x^* dla problemu INTERVAL MIN–MAX REGRET. Według propozycji jest ono także optymalnym rozwiązaniem dla problemu z ekstremalnym scenariuszem. Weźmy także zbiór $\mathbb{1}(x) = \{i\in\{1,\cdots,n\} : x_i = 1\}$, tak jak w poprzedniej propozycji. Analogicznie do przypadku 7 możemy wyprowadzić ciąg nierówności, które doprowadzą nas do: $val(x^*,s) - val\left(x^*_{c^+(x^*)},s\right) = ... \geqslant val(x^*,c^+(x^*)) - val\left(x^*_{c^+(x^*)},c^+(x^*)\right)$. Zauważmy, że w założeniu rozwiązanie optymalne problemu INTERVAL MIN–MAX REGRET odpowiada rozwiązaniu problemu dla scenariusza $c^+(x^*)$ tak więc $val(x^*,c^+(x^*)) = val^*_{c^+(x^*)} = val\left(x^*_{c^+(x^*)},c^+(x^*)\right)$. Załóżmy teraz, że tak nie jest i optymalne rozwiązanie dla INTERVAL MIN–MAX REGRET nie jest opt. dla problemu z ekstremalnym scenariuszem. Wtedy $val(x^*,c^+(x^*)) \neq val^*_{c^+(x^*)}$. Teraz wracamy do $val(x^*,s) - val\left(x^*_{c^+(x^*)},s\right) = ... \geqslant val(x^*,c^+(x^*)) - val\left(x^*_{c^+(x^*)},c^+(x^*)\right)$.

$val(x^*,s) \geqslant val\left(x^*_{c^+(x^*)},s\right) + \left[val(x^*,c^+(x^*)) - val\left(x^*_{c^+(x^*)},c^+(x^*)\right)\right] > val\left(x^*_{c^+(x^*)},s\right)$

$val(x^*,s) > val\left(x^*_{c^+(x^*)},s\right)$

$val(x^*,s) - val^*_s > val\left(x^*_{c^+(x^*)},s\right) - val^*_s$ dla każdego $s\in S$. Zatem $\max_{s\in S} val(x^*,s) - val^*_s > \max_{s\in S} val\left(x^*_{c^+(x^*)},s\right) - val^*_s$, co stoi w sprzeczności z tym, żr x^* jest optymalnym rozwiązaniem dla INTERVAL MIN–MAX REGRET (nie istnieje mniejszy max dla innego rozwiązania).

Mamy zatem prosty przepis na rozwiązanie problemu INTERVAL MIN–MAX REGRET. Znajdujemy optymalne rozwiązania dla wszystkich scenariuszów ekstremalnych, liczymy $R_{max}(x^*_s)$, korzystając z właściwości udowodnionych w propozycji 7, a następnie wybieramy spośród nich minimum z tych maximów. Nie jest to jednak genialne rozwiązanie, bo takich ekstremalnych scenariuszy jest 2^n (ekstremalny scenariusz to taki, dla którego każde $c_i = \overline{c}_i$ lub $c_i = \underline{c}_i$ dla każdego $i\in\{1,\cdots,n\}$). No chyba, że większa liczba z wszystkich współczynników jest ustalona/znana ($\underline{c}_i = \overline{c}_i$) (niezdegenerowana) co efektywnie zmniejsza nam liczbę scenariuszy ekstremalnych.

W praktyce rozwiązanie problemu INTERVAL MIN–MAX ma max regret blisko optymalnej jego wartości/rozwiązania dla INTERVAL MIN–MAX REGRET, jednak są zadania, w których te wyniki są od siebie zupełnie różne. Np. Fig. 5, gdzie mamy przejść z węzła 1 do n. Mamy 2 rozwiązania: x_1, przechodzący bezpośrednio z 1 do n oraz x_2, który idzie na około. Mamy dwa scenariusze. $\max_{s\in S} val(x_1,s) = 2^n$, $\max_{s\in S} val(x_2,s) = 2^n + 1$, rozwiązaniem optymalnym jest zatem x_1. Żal wynosi 2^n (dla scenariusza s_1 idącego na około $val(x_1,s_1) - val^*_{s_1} = 2^n - 0$). Dla problemu INTERVAL MIN–MAX REGRET optymalnym rozwiązaniem jest x_2 - dla X_1 $\max_{s\in S}(2^n - 0, 2^n - 2^n)$, dla X_2 $\max_{s\in S}(0 - 0, 2^n + 1 - 2^n)$, więc $\min_{x\in\{x_1,x_2\}}\{2^n, 1\} = 1$ dla x_2. Najgorsza zatem wartość tego rozwiązania to $2^n + 1$ z żalem 1.

Propozycja 9 i 10 do dowodu.

A tabu search algorithm for the min-max regret minimum spanning tree problem

Dobra, trochę porządku w oznaczeniach:

- $G = (V, E)$, $|V| = n$ $|E| = m$, $E = \{e_i : i \in \{1, \ldots, m\}\}$
- rozmyty koszt $c_i = c_{e_i} = \left[\underline{c}_{e_i}, \overline{c}_{e_i}\right]$
- Zbiór wszystkich scenariuszy $\Gamma \equiv c_{e_1} \prod c_{e_2} \prod \cdots \prod c_{e_m}$, gdzie każdy scenariusz $S \in \Gamma$ jest wektorem $\left(c_e^S\right)_{e \in E}$.
- $T \in \mathcal{T}$ - drzewo rozpinające T należące do zbioru wszystkich acyklicznych podgrafów grafu G.
- $s \in S$ - scenariusz s w zbiorze scenariuszy S. Teraz $S \in \Gamma$ to scenariusz S należący do zbioru scenariuszy Γ
- $val\,(T, s) = F\,(T, S) = \sum_{e \in T} c_e^S$ - koszt drzewa rozpinającego T w scenariuszu S
- $val_s^* = F^*\,(S) = \min_{T \in \mathcal{T}} F\,(T, S)$ - optymalne rozwiązanie dla scenariusza S. Drzewo rozpinające o najmniejszym koszcie - minimalne drzewo rozpinające. Można policzyć Primem, Kraskalem lub Chazellem.
- $R_{max}\,(T) = Z\,(T) = \max_{S \in \Gamma} \{F\,(T, S) - F^*\,(S)\}$ - maksymalny żal dla danego drzewa rozpinającego (ile w najgorszym przypadku stracimy wybierając to rozwiązanie - najgorszy przypadek scenariusza dla drzewa rozpinającego T)
- Alternatywa dla najgorszego przypadku - drzewo T będące optymalnym rozwiązaniem dla najgorszego scenariusza.

3.1 Minmax regret minimum spanning tree problem

Patrz propozycja 7 w Paper0 (Min–max and min–max regret versions of combinatorial optimization problems: A survey) - tutaj zdefiniowana ta sama propozycja, tylko dla drzew rozpinających. Tak więc i wniosek jest taki sam, że rozwiązując klasyczny problem dla najgorszego scenariusza zorientowanego na wybrane drzewo T:

$$c^{S_T}\,(x) = \begin{cases} \overline{c}_e & \text{if} \quad e \in T, \\ \underline{c}_e & \text{if} \quad e \in E \setminus T \equiv e \notin T \end{cases} \tag{3.1}$$

dostajemy minimalne drzewo rozpinające T dla tego scenariusza (S_T), które jednocześnie równa się $Z\,(T)$, czyli maksymlanemu żalowi dla drzewa T. Aby rozwiązać problem minmax regret minimum spanning tree wystarczy teraz wziąć $\min_{T \in \mathcal{T}} Z\,(T)$.

Wzmianka o problemie centralnego drzewa rozpinającego jest chyba zbędna, bo nic nie wnosi, poza tym, że MINMAX dla MST jest silnie NP. $d\,(T_1, T_2) = |T_1 \setminus T_2| = |T_2 \setminus T_1|$. Np. $T_1 = \{e_a, e_b, e_c\}$, $T_2 = \{e_a, e_b, e_d\}$, $d\,(T_1, T_2) = |\{e_c\}| = 1$.

3.2 Previous methods of solving the problem

"Fortunately, there exists a simple and efficient approximation algorithm for the problem, proposed in Kasperski and Zieliński- mowa tu o propozycji 9 z Paper0 (Min–max and min–max regret versions of combinatorial optimization problems: A survey), tyle że dla MST.

Aproksymacja problemu, którego rozwiązanie wynosi OPT jest $f(n)$-aproksymacją, jeżeli rozwiązanie problemu aproksymacyjnego x spełnia $x \leqslant f(n) \cdot OPT$.

Jest to najlepszy algorytm aproksymacyjny do tej pory dla tego problemu, choć testy pokazują, że lepsze wyniki osiąga ten sam algorytm AM z lekką modyfikacją - AMU, która polega na policzeniu także klasycznego problemu dla najgorszego scenariusza i wzięciu mniejszego wyniku (czyli $\min\{R_{max}(x^*), R_{max}(y^*)\}$, gdzie $R_{max}(x^*) \leqslant 2 \cdot OPT$ a x^* jest optymalnym rozwiązaniem dla problemu ze scenariuszem S, gdzie każdy $c_{e_i}^S = \frac{\underline{c}_{e_i} + \overline{c}_{e_i}}{2}$, a y^* jest MST dla problemu ze scenariuszem najgorszym).

3.3 Solving the problem by local search

Algorytm sąsiadów w SP jest prosty. Bierzemy zbiór krawędzi, który nie należy do MSP i dla każdej z takich krawędzi (i, j) znajdujemy zbiór krawędzi $f = \{f_1, \cdots, f_k\}$, który prowadzi inną drogą z jednego jej końca na drugi. Dodając tę krawędź możemy bezkarnie usunąć jedną z takich krawędzi na ścieżce z i do j, bo właściwości drzewa rozpinającego będą zachowane.

3.3.1 Neighborhood function and local minimum

$N(T) = \{T_1 \in \mathcal{T} : |T \setminus T_1| = 1\}$ - sąsiadami drzewa rozpinającego T są wszystkie drzewa, które różnią się od T dokładnie jedną krawędzią i nadal są ST. Lokalne minimum to takie T_{min}, że nie istnieje inne $T \in N(T_{min})$ takie, że $Z(T) < Z(T_{min})$ - $Z(T_{min}) = \min_{T \in N(T_{min})} Z(T)$. Pokażemy teraz, że $Z(T_{min}) = 2 \cdot OPT$, więc że nie jest ono lepsze od rozwiązania dawanego przez AM lub AMU (gdzie rozwiązanie T^*: $Z(T^*) \leqslant 2 \cdot OPT$). Graf $G = (V, E)$ na Fig. 4 jest kompletny, ma $2 \cdot m - 2$ wierzchołki (to 6, 6', k, k', m, m' symbolizuje, że jest ich tam więcej, od 6 do m i od 6' do m', tyle że już ktoś zapomniał uzupełnić środek, żeby to nadal był graf pełny). Każda krawędź ma koszt $[0, 1]$. Pogrubione linie oznaczają wybrane rozwiązanie lokalne T, składające się z $2m-3$ węzłów. W związku z tym $F(T, S_T) = \sum_{e \in T} c_e^{S_T} = \sum_{e \in T} \overline{c}_e = \sum_{e \in T} 1 = 2 \cdot m - 3$. Przypominam:

$$c^{S_T}(x) = \begin{cases} \overline{c}_e & \text{if} \quad e \in T, \\ \underline{c}_e & \text{if} \quad e \in E \setminus T \equiv e \notin T \end{cases} \tag{3.2}$$

zaś w tym przypadku $\overline{c}_e = 1$, $\underline{c}_e = 0$. Teraz skonstruujmy dla scenariusza S_T alternatywne drzewo rozpinające T^* dla najgorszego scenariusza S_T. Zauważyć można, że jeśli skonstruujemy je z podzbioru krawędzi, które nie należą do T (a możemy to zrobić, bo - jak widać na Fig. 4b - bez krawędzi należących do T graf nadal jest spójny i istnieje drzewo go rozpinające) to $F(T^*, S_T) = \sum_{e \in T^*} c_e^{S_T} = \sum_{e \in T^* \setminus T} c_e^{S_T} = \sum_{e \in T^* \setminus T} \underline{c}_e = \sum_{e \in T^* \setminus T} 0 = 0$. Dlatego też oznaczyliśmy to drzewo rozpinające jako optymalne T_*. Widać stąd bezpośrednio, że żal dla scenariusza S_T wynosi $F(T^*, S_T) - 0 = 2 \cdot m - 3$ a zatem dla wszystkich scenariuszy $Z(T) = 2 \cdot m - 3$. Na pewno nie jest większy, bo koszt drzewa nie może być mniejszy od zera (współczynniki kosztów wynoszą co najmniej 0), ani też większy od $2 \cdot m - 3$ (bo drzewo rozpinające składa się DOKŁADNIE z $|V| - 1 = (2 \cdot m - 2) - 1 = 2 \cdot m - 3$ krawędzi, zaś koszt każdej wynosi maksymalnie 1). Krótko mówiąć nie musimy sprawdzać innych scenariuszy, bo na pewno maksymalna wartość $Z(T) = \max_{S \in \Gamma}\{F(T, S) - F^*(S)\} = F(T, S_T) - F^*(S) = F(T, S_T) - 0 = 2 \cdot m - 3$. Musimy teraz pokazać, że jest to lokalne minimum, czyli że nie istnieje inne drzewo T_1 w sąsiedztwie, które miałoby lepszą/mniejszą wartość $Z\{T_1\}$ od $Z(T)$. Stwórzmy zatem drzewo T_1 poprzez usunięcie jednej dowolnej krawędzi $f \in T$ i dodanie do niego dowolnej innej krawędzi $e \in E \setminus T$ (zgodnie z definicją sąsiedztwa $N(T)$). Pokażemy, że $Z(T_1) = Z(T) = 2 \cdot m - 3$. Tym samym pokażemy, że nie istnieje w sąsiedztwie drzewa T inne drzewo, które by miało mniejszą wartość od $Z(T)$ i że T jest faktycznie lokalnym optimum. Aby to pokazać, wystarczy zauważyć, że aby rozerwać graf z Fig. 4a należy usunąć więcej niż jedną krawędź. Każde inne MTS ma

oczywiście $|V|-1$ krawędzi, generalnie dla każdego T', nie tylko T_1 zachodzi $F(T', S_{T'}) = 2 \cdot m - 3$. Z kolei z faktu, że aby rozerwać graf należy usunąć więcej niż 1 krawędź wynika, że zawsze $F^*(S_{T'}) = 0$ (zawsze znajdziemy takie drzewo T'', że $T'' = \{e : e \notin T'\}$, a zamienioną krawędź T –¿ T_1 możemy zastąpić taką, która nie należy do T_1). Zatem w szczególności $F^*(S_{S_1}) = 0$, a z tego wynika, że $Z(T_1) = Z(T) = 2 \cdot m - 3$. Stąd drzewo T jest lokalnym minimum.

Znajdźmy teraz globalne optymalne rozwiązanie dla tego grafu. Chcemy zatem spełnić $\min_{T\in\mathcal{T}} \max_{S\in\Gamma} F((T,S) - F^*(S))$. Z propozycji 7 z Paper0 (Min–max and min–max regret versions of combinatorial optimization problems: A survey) wiemy, że masymalny regret otrzymamy dla najgorszego scenariusza. Znajdźmy teraz takie drzewo rozpinające R, że dla scenariusza S_R (najgorszy scenariusz dla drzewa R) max regret jest namniejszy z możliwych. Pokazaliśmy już wcześniej, że dla dowolnego drzewa rozpinającego T'' i dla odpowiadającemu mu najgorszego scenariusza $S_{T''}$ zdefiniowanego wyżej $F(T'', S_{T''}) = |V| - 1$. Tak więc szukamy. Nasze zadanie sprowadza się do znalezienia $\min_{T\in\mathcal{T}}((|V|-1) - F^*(S_R)) = \min_{T\in\mathcal{T}}(2 \cdot m - 3 - F(R^*, S_R))$, czyli takiego drzewa R, że $F(R^*, S_R)$ ma największą wartość. Skoro zaś współczynniki S_R dla rozwiązania R^* przyjmują wartość 1 tylko wtedy, gdy $e \in R^*$ należą także do R, zaś dla innych przyjmują wartość 0, dochodzimy do wniosku, że $F(R^*, S_R) = |R^* \cap R|$. Zatem jedynym sensownym wyborem R jest takie drzewo, które zawiera jak największą liczbę krawędzi będącymi koniecznymi do zachowania spójności grafu (tak, że R^* musi zawierać jak największą liczbę wspólnych krawędzi z R). Z rysunku odczytamy, że $F(R^*, S_R) = m - 2$. Stąd $Z(R) = 2 \cdot m - 3 - (m-2) = m - 1$. Jest to globalne optimum. Dzieląc otrzymane wcześniej maksymalne lokalne optimum przez globalne optimum otrzymujemy 2. No i mamy bezpieczną twierdzenie, że $Z(T_{min} \geqslant (2-\epsilon) \cdot OPT)$.

Dla problemu MINMAX spaning tree tak zdefiniowane otoczenie NIE jest dokładne (stąd też mamy taką nierówność, a nie $= OPT$).

3.3.2 Iterative improvement algorithm

Local search i Tabu Search. Wymaga doczytania o słabych i silnych krawędziach - jeśli jakaś krawędź jest silna to wtedy należy do drzewa rozpinającego bez względu na scenariusz, więc na pewno należy do rozwiązania optymalnego. Co więcej, takie rozwiązanie zawiera wszystkie takie krawędzie, zaś nie zawiera słabych. Wstępne rozwiazanie, od którego zaczynamy algorytm może być losowe, podane przez AMU lub zaburzony AMU (PMU) - i tak ponoć najlepiej jest wybrać losowe rozwiązanie. Tabu list i kryterium aspiracji - wiadomo. Pamięć długotrwała - za każdym restartem algorytmu bierzemy inny graf $G = (V, e^*)$, gdzie E^* składa się z łuków należących do najlepszej alternatywy dla najgorszego przypadku drzewa T, które polepsza obecne najlepsze rozwiązanie. Czyli:

- na początku dla wstępnego rozwiązania T liczymy T^* (najlepszą alternatywę dla najgorszego przypadku dla T - generujemy najgorszy scenariusz dla T, czyli (CHYBA?)

$$c^{S_T}(x) = \begin{cases} \overline{c}_e & \text{if} \quad e \in T, \\ \underline{c}_e & \text{if} \quad e \in E \setminus T \equiv e \notin T \end{cases} \tag{3.3}$$

  i znajdujemy dla tego scenariusza najlepsze drzewo rozpinające, które pewnie składa się z jak najmniejszej liczby wspólnych krawędzi z T) i dodajemy zbiór krawędzi należących do T^* do E^*, który jest na początku pusty,

- potem, już w pętli, znajdujemy najlepsze rozwiązanie w sąsiedztwie $Z(T_1)$ (dla przypomnienia - $Z(T_1) = \max_{S\in\Gamma}(F(T_1, S) - F^*(S)) = F(T_1, S_{T_1}) - F^*(S_{T_1})$, zaś pamiętać należy, że nie dla każdego i musi zachodzić $c_i = [0, 1]$. Inaczej bezcelowe byłoby liczenie za każdym razem $F(T_1, S_{T_1}) = |e : e \in T_1|$.)

- jeżeli okazuje się ono lepsze od obecnego Z_{best} wtedy aktualizujemy rozwiązanie oraz dodatejmy do zbioru E_* krawędzie należące do T_*, które generujemy tak samo jak w punkcie 1.

to nam gwarantuje dwie rzeczy:

- graf $G^* = (V, E^*)$ na pewno jest spójny (bo zawiera krawędzie dopuszczalnych rozwiązań),

- graf zawiera maksymalnie różne krawędzie od tych, które należały do znajdowanych rozwiązań, które polepszały ogólne rozwiązanie (sam sposób generowania T_* z T wymusza ażeby rozwiązanie T_* było jak

najbardziej odległe od T i tym samym by $|T \setminus T^*|$ była jak największa). Stąd też, restartując algorytm szukamy rozwiązań w potencjalnie innym lokalnym kawałku.

Najcięższym kawałkiem chleba jest policzenie $Z(T_i)$ dla każdego $T_i \in \overline{N}(T)$ by z kolei wybrać z tego minimum. W najprostszym podejściu możemy wziąć najgorszy scenariusz $\overline{c} = (\overline{c}_e)_{e \in E}$ i policzenie dla każdego sąsiedniego drzewa T_i wartości $Z(T_i)$ i wybrać z tego najlepsze drzewo.

Istnieje lepsza metoda, choć według mnie jest nieco wydumana. No ale OK. Weźmy drzewo T, będące aktualnym drzewem rozpinającym. T^* jest alternatywą dla najgorszego przypadku dla T, czyli optymalnym rozwiązaniem dla scenariusza S_T. Weźmy ze zbioru $\overline{N}(T) \subseteq N(T)$ dowolne drzewo T_1. T_1 powstało z usunięcia z T krawędzi $f \in T$ i dodaniu $e \in E \setminus T$. Jasnym jest zatem, że $T_1 = T \cup \{e\} \setminus \{f\}$ i to samo można powiedzieć o wartości funkcji celu: $F(T_1, S_{T_1}) = F(T, S_T) + \overline{c}_e - \overline{c}_f$.

$$c^{S_T}(x) = \begin{cases} \overline{c}_e & \text{if} \quad e \in T, \\ \underline{c}_e & \text{if} \quad e \in T_1 \setminus T \equiv e \notin T_1 \end{cases} \tag{3.4}$$

$$c^{S_{T_1}}(x) = \begin{cases} c^{S_T} & \text{if} \quad e \in T \cap T_1 \\ \overline{c}_e & \text{if} \quad e \in T_1 \setminus T \\ \underline{c}_e & \text{if} \quad e \in T \setminus T_1 \end{cases} \tag{3.5}$$

$F(T_1, S_{T_1}) = \sum_{e \in T_1} c_i^{S_{T_1}} = \sum_{e \in T_1 \cap T} c_i^{S_{T_1}} + \sum_{e \in T_1 \setminus T} c_i^{S_{T_1}} = \sum_{e \in T_1 \cap T} c_i^{S_T} + \sum_{e \in T_1 \setminus T} \overline{c}_{e_i} = \sum_{e \in T_i \cap T} c_i^{S_T} + \overline{c}_e = \left[\sum_{e \in T \cap T_i} c_i^{S_T} + \overline{c}_f\right] - \overline{c}_f + \overline{c}_e = \left[\sum_{e \in T \cap T_i} c_i^{S_T} + \sum_{e \in T \setminus T_i} c_i^{S_T}\right] + \overline{c}_e - \overline{c}_f = F(T, S_T) + \overline{c}_e - \overline{c}_f$

Naszym celem jest szybkie policzenie $Z(T_1)$ na na podstawie T^*. $Z(T_1) = F(T_1, S_{T_1}) - F(T_1^*, S_{T_1})$, zależność między poszczególnymi $F(T_i, S_{T_i})$ a $F(T, S_T)$ pokazaliśmy wyżej, więc naszym celem jest szybkie wyliczenie $F^*(S_{T_1})$. Aby to zrobić, chcemy umieć najpierw szybko generować optymalne rozwiązanie dla S_{T_1} na podstawie danego T^*, by móc wyliczyć $\sum_{e \in T_1^*} c_e^{S_{T_1}} = F(T_i^*, S_{T_1}) = F^*(S_{T_1})$. Drzewo T^* otrzymaliśmy wcześniej z jakiegoś algorytmu np. zachłannego. W celu jego policzenia musieliśmy uporządkować koszty wierzchołków niemalejąco - zbiór σ. Budujemy najgorszy scenariusz dla T_1. W tym celu koszt krawędzi dodawanej do T_1 w scenariuszu S_{T_1} rośnie z $\underline{c}_e$ (taki koszt miała w scenariuszu S_T) do $\overline{c}_e$, zaś koszt krawędzi usuwanej f z T zmieniamy z $\overline{c}_f$ na $\underline{c}_f$. W związku z tą zmianą kosztów krawędzi, aby otrzymać nowy niemalejący porządek σ' musimy przesunąć koszt krawędzi f w lewo ku początkowi ciągu, jako że jej koszt się zmniejszył (bądź pozostał w skrajnym przypadku taki sam), zaś koszt krawędzi e - w prawo ku końcowi ciągu. Teraz możemy policzyć minimalne drzewo rozpinające dla scenariusza S_{T1} przy pomocy alg. zachłannego i ciągu σ', otrzymując T_1^*. Skoro w ciągu σ' przesunęliśmy koszt krawędzi e prawo o 0 lub więcej pozycji, znaczy to, że krawędzi o mniejszym lub równym koszcie od c_e jest w nowym ciągu więcej lub tyle samo co w starym σ. Z jakiegoś tam twierdzenia mamy zatem, że jeśli zachodzi $pred(\sigma, e) \subseteq pred(\sigma', e)$ ($pred(\sigma, e)$ właśnie oznacza zbiór elementów w σ które poprzedzają - mają mniejszą lub równą wartość - e) to jeśli $e \notin T_1^*$ (ciąg σ') to wtedy także $e \notin T^*$. To prowadzi do następującego rozumowania:

- T i T_1 różnią się dwoma węzłami: $e \notin T$, $e \in T_1$, $f \in T$, $f \notin T_1$. Załóżmy, że $e \notin T_1^*$ zatem i $e \notin T^*$. Wówczas koszty w scenariuszach S_T i S_{T1} dla ścieżek T^* i T_i^* mogą różnić się tylko kosztem c_f (gdyż w obydwu nie ma krawędzi e, więc inny jej koszt w tych scenariuszach nie ma żadnego znaczenia). $c_f^{S_T} \geqslant c_f^{S_{T_1}}$ (zmniejszaliśmy koszt c_f w scenariuszu S_{T_1}). Jeżeli zatem $f \in T^*$ (nie było innej ścieżki g o mniejszym koszcie od f, którą można by było ją zastąpić i jeszcze zmniejszyć koszt rozwiązania T^*) to f na pewno $\in T_1^*$, jako, że kosz ścieżki f w scenariuszu S_{T_1} jest jeszcze mniejszy. Zatem otrzymujemy, że $T_1^* = T^*$, jako że jedyne krawędzie o zmienionych kosztach zgodnie należą ($f \in T^* \cap T_1^*$) lub nie należą ($e \notin T^*, e \notin T_1^*$) do obu z tych zbiorów. Koszt reszty się nie zmienia, więc musieliśmy rozpatrzyć tylko te 2 krawędzie.

- W przeciwnym wypadku, jeżeli $f \notin T^*$ (przy założeniu, że $e \notin T_1^*$ i $e \notin T^*$) to znaczy, że $T^* \cup \{f\}$ zawiera cykl: $\{f, g_1, \ldots, g_k\}$. Jeżeli wśród tego cyklu znajduje się krawędź g_i, która w scenariuszu S_{T1} ma większy koszt niż $\underline{c}_f$, wtedy zmiana krawędzi w T_1^* z g_i na f prowadzi do jeszcze mniejszej wartości funkcji celu (jako, że w T^* krawędź g_i była optymalna, bo należała do optymalnego rozwiązania, a nic nie zmienialiśmy poza kosztami e i f to należała także do T_1^*. Jako, że w wyniku zmniejszenia kosztu krawędzi f w scenariuszu S_{T_1} krawędź g_i przewyższyła ją kosztami, wtedy nie należy już ona do optymalnego rozwiązania, zaś $T_1^* = T^* \cup (f) \setminus \{g_i\}$ nadal tworzy drzewo rozpinające, jako że i f

i g_i należały do cyklu i usunięcie jednej z nich nie spowodowało rozerwania grafu.) Obliczymy wtedy $T_1^* \longleftarrow F^*(S_{T_1}) \longleftarrow Z(T_1)$ w czasie $O(|V|)$, który jest wymagany do odnalezienia wspomnianego cyklu. Policzenie sumy $F^*(S_{T_1})$ też w takim samym czasie, $Z(T_1)$ też.

- Jeśli w cyklu $\{f, g_1, \ldots, g_k\}$, pomimo zmniejszenia kosztu krawędzi f w scenariuszu S_{T_1} względem S_T, nadal f ma najwyższy koszt spośród nich wszystkich (a $e \notin T_1^*$ i $e \notin T^*$), to nadal nie należy ono do optymalnego rozwiązania - T_1^* by być drzewem rozpinającym weźmie jeden z węzłów g, zatem $f \notin T_1^*$. W związku z tym, skoro $e \notin T_1^*$ i $e \notin T^*$ oraz $f \notin T_1^*$ i $f \notin T^*$, reszta kosztów jest taka sama, a oczywiście $|T_1^*| = |T^*|$ to oczywiście $T_1^* = T^*$.
- Gdy $e \in T_1^*$ to musimy rozwiązać klasyczny problem z najgorszym scenariuszem by otrzymać T_1^*.

Zatem w 3/4 przypadków jesteśmy w stanie przyspieszyć z czasu potrzebnego na algorytm zachłanny $O(V \cdot E)$ do $O(|V|)$ lub nawet $O(|T_i^*|)$.

Robust recoverable and two-stage selection problems

Bierzemy na tapetę problem plecakowy z wagami przedmiotów 0–1 - czyli problem wyboru. Jak zwykle ustalmy na początku oznaczenia:

- $E = \{e_1, \ldots, e_n\}$ - zbiór przedmiotów
- w problemie wybieramy dokładnie p przedmiotów tak by zminimalizować $\sum_{e \in X} c_e$
- Γ - zbiór scenariuszy, zawierający wszystkie wektory możliwych kosztów przedmiotów,
- $\Phi = \{X \subseteq E : |X| = p\}$ - zbiór dopuszczalnych rozwiązań (z dokładnie p wybranymi przedmiotami),
- $\Phi_1 = \{X \subseteq E : |X| \leqslant p\}$ - zbiór rozwiązań, gdzie wybranych jest mniej niż p przedmiotów.
- $\Phi_X = \{Y \subseteq E \setminus X : |Y| = p - |X|\}$ - zbiór przedmiotów nie należących do wybranego rozwiązania $X \in \Phi_1$, uzupełniających to rozwiązanie do p przedmiotów.
- $\Phi_X^k = \{Y \subseteq E : |Y| = p, |Y \setminus X| \leqslant k\}, k \in \{0, \ldots, p\}$ - zbiór rozwiązań Y, które powstały z X w wyniku zamiany co najwyżej k elementów z X na inne, nie należące do tego rozwiązania,
- C_e- deterministyczny koszt przedmiotu $e \in E$ w pierwszej fazie algorytmu, gdy wybieramy początkowe rozwiązanie dla dostępnego, pojedynczego scenariusza,
- c_e^S wiadomo, $f(X, S)$ też wiadomo.

Dwufazowy algorytm będziemy budować następująco:

- na początku mamy pewien scenariusz, który determinuje początkowe koszty przedmiotów. Wartość tego rozwiązania to $\sum_{e \in X} C_e$ dla wybranego początkowo $X \in \Phi_1$,
- scenariusz się zmienia i chcemy, zachowując poprzednią decyzję o wyborze X, dobrać pozostałe przedmioty tak, aby mimo zmiany scenariusza wyjść na tym jak najlepiej - tak więc chcemy $\min_{Y \in \Phi_X} f(Y, S)$, gdzie $S \in \Gamma$ jest naszym nowym scenariuszem, zaś Y to możliwe dopełnienia zbioru przedmiotów X do p przedmiotów.
- Zatem koszt wyboru rozwiązania X w przypadku zmiany scenariusza z początkowego na S wynosi $f_1(X, S) = \sum_{e \in X} C_e + \min_{Y \in \Phi_X} f(Y, S)$
- w zagadnieniu odpornej optymalizacji dwufazowej (robust two-stage) szukamy oczywiście takiego początkowego wyboru X ażeby zminimalizować wartość funkcji celu w najgorszym przypadku (to jest dla scenariusza S w którym, nawet najlepszy wybór pozostałych przedmiotów $Y \in \Phi_X$ da nam wartość gorszą od wszystkich pozostałych scenariuszy dla początkowego rozwiązania X): $opt_1 = \min_{X \in \Phi_1} \max_{S \in \Gamma} f_1(X, S)$

Model regeneracyjny nieco się różni. Tutaj wybieramy na początek od razu kompletne rozwiązanie $X \; in \Phi$. Po pojawieniu się nowego scenariusza (S) jesteśmy za to w stanie wymienić co najwyżej k przedmiotów z pierwszego rozwiązania i zastąpić je innymi, nie należącymi wcześniej do X. Tak jak wcześniej, chcemy by ta zmiana jak najkorzystniej odbiła się na wartości funkcji celu dla nowego scenariusza: $f_2(X, S) =$

$\sum_{e\in X} C_e + \min_{Y\in\Phi_X^k} f(Y,S)$. Tak samo dla robust recoverable selection problem chcemy wybrać takie początkowe rozwiązanie, które minimalizowałoby funkcję celu dla najgorszego scenariusza jaki może się pojawić - chcemy znaleźć takie początkowe rozwiązanie X, które bez względu na nowy scenariusz pozwoli na jak największą optymalizację rozwiązanie względem tego scenariusza i każdego innego. Oczywiście z którejśtam propozycji w paper0 wynika, że scenariuszy z przedziałami (Γ^1) dla najgorszego scenariusza $\overline{S} = (\overline{c})_{e\in E}$ zachodzi zarówno $\max_{S\in\Gamma^1} f_1(X,S) = f_1\left(X,\overline{S}\right)$ jak i $\max_{S\in\Gamma^1} f_2(X,S) = f_1\left(X,\overline{S}\right)$, co prowadzi ponoć do prostego algorytmu rozwiązania.

Recoverable Selection: $opt_2^1 = \min_{X\in\Phi}\max_{S\in\Gamma^1} f_2(X,S) = \min_{X\in\Phi}\max_{S\in\Gamma^1}\left[\sum_{e\in X} C_e + \min_{Y\in\Phi_X^k} f(Y,S)\right] = \min_{X\in\Phi}\left[\sum_{e\in X} C_e + \min_{Y\in\Phi_X^k}\max_{S\in\Gamma^1}\sum_{e\in Y} c_e^S\right] = \min_{X\in\Phi}\left[\sum_{e\in X} C_e + \min_{Y\in\Phi_X^k}\sum_{e\in Y}\overline{c}_e^S\right]$. Model jest całkiem jasny i nie będę się tu o nim rozpisywać. Wartym zauważenia jest, że warunek $|X\cap Y| \geqslant p-k$ ogranicza rozwiązania Y tak aby nie różniły się od X większą liczbą przedmiotów niż k (muszą mieć co najmniej $p-k$ elementów wspólnych), czyli żeby $Y\in\Phi_X^k$, że możemy zdefiniować następująco decyzyjne zmienne binarne:

$$x_i(x) = \begin{cases} 0 & \text{if} \quad e_i \notin X, \\ 1 & \text{if} \quad e_i \in X \end{cases} \tag{4.1}$$

,

$$y_i(x) = \begin{cases} 0 & \text{if} \quad e_i \notin Y, \\ 1 & \text{if} \quad e_i \in Y \end{cases} \tag{4.2}$$

,

$$z_i(x) = \begin{cases} 0 & \text{if} \quad e_i \notin X\cap Y, \\ 1 & \text{if} \quad e_i \in X\cap Y, \end{cases} \tag{4.3}$$

,

że dany element e albo nie ograniczenia $x_i + z_i \leqslant 1$ i $y_i + z_i \leqslant 1$ zapewniają nam, że każdy element policzymy tylko raz , co pozwala na napisanie pozostałych ograniczeń. W unimodularność macierzy jestem w stanie uwierzyć, więc bez ograniczeń całkowitoliczbowych uzyskamy na pewno rozwiązanie całkowitoliczbowe (teorię przydałoby się wyłożyć).

Robust optimization with incremental recourse

Mamy początkowe kosztu d i odpowiadające im rozwiązanie x. Dostajemy inny scenariusz tożsamy z wektorem kosztów (scenariusz $s \in \mathcal{U} \equiv c^s \in \mathcal{U}$). Możemy w pewnym ograniczonym zakresie nagiąć rozwiazanie x tak, aby dobrze się sprawowało pod nowym scenariuszem. $S_x = \{y \in S : F(x,y) \leqslant K\}$ - zbiór scenariuszy nie różniący się od oryginalnego rozwiązania o więcej niż K. Podstawowy problem: $Z_{RobInc} = \min_{x\in S}\left[d^T \cdot x + \max_{c\in\mathcal{U}} \min_{y\in S_x} c^T \cdot y\right]$ - chcemy wybrać takie rozwiązanie x by móc je nagiąć tak, żeby zminimalizować wartość rozwiązania w przypadku pojawienia się najgorszego scenariusza (ogólnie jesteśmy zainteresowani minimalizacją

$\min_{x\in S}\left[d^T \cdot x + \min_{y\in S_x} c^T \cdot y\right]$ dla najgorszego scenariusza:

$\min_{x\in S} \max_{c\in\mathcal{U}}\left[d^T \cdot x + \min_{y\in S_x} c^T \cdot y\right]$, przy czym pierwszy człon nie zależy od scenariusza, więc

$\min_{x\in S} \max_{c\in\mathcal{U}}\left[d^T \cdot x + \min_{y\in S_x} c^T \cdot y\right] = \min_{x\in S}\left[d^T \cdot x + \max_{c\in\mathcal{U}}\left(\min_{y\in S_x} c^T \cdot y\right)\right])$.

Możemy wytyczyć dwa specyficzne przypadki:

- początkowe koszty $d = 0$ ($\min_{x\in S}\left[\max_{c\in\mathcal{U}}\left(\min_{y\in S_x} c^T \cdot y\right)\right]$) i $S_x = \{x\}$ (nie mamy żadnej możliwości modyfikacji rozwiązania). Skoro tak to $\min_{y\in S_x} c^T \cdot y = c^T \cdot x$, a więc $\min_{x\in S}\left[\max_{c\in\mathcal{U}}\left(\min_{y\in S_x} c^T \cdot y\right)\right] = \min_{x\in S}\left[\max_{c\in\mathcal{U}}\left(c^T \cdot x\right)\right] = \min_{x\in S}\max_{c\in\mathcal{U}} c^T \cdot x$.
- w drugim przypadku możemy móc całkowicie zmienić nasze rozwiązanie, niezależnie od początkowego wybranego $x \in S$ ($S_x = S$) tak więc, jeśli $d = 0$ to $\min_{x\in S}\left[\max_{e\in\mathcal{U}} \min_{y\in S_x} c^T \cdot y\right] = \min_{x\in S}\left[\max_{e\in\mathcal{U}} \min_{y\in S} c^T \cdot y\right]$, zatem od x nie zależy rozwiązanie ($\forall x \min_{x\in S}\left[\max_{e\in\mathcal{U}} \min_{y\in S} c^T \cdot y\right] = \max_{e\in\mathcal{U}} \min_{y\in S} c^T \cdot y$)

Dwa modele niepewności w zbiorach: $\mathcal{U}_1$ - zawiera wszystkie wektory kosztów, w których każdy z jego elementów c_i może przyjmować ciągłą wartość pomiędzy $\overline{c}_i, \overline{c}_i + \hat{c}_i$, gdzie $\overline{c}_i$ to najmniejsza wartość współczynnika c_i. Suma kosztów wektora nie może być większa od kosztu wektora $\overline{c} = (\overline{c}_i)$ niż o Γ (w sensie jeśli jeden czynnik c'_i wektora $c' \in \mathcal{U}_1$ wynosi $\overline{c}_i + \Gamma$ to reszta musi już spełniać $c'_i = \overline{c}_i$). Drugim modelem jest $\mathcal{U}_2$

The robust spanning tree problem with interval data

Modele LP dla MST:

Pierwszy model jest czysto teoretyczny, opiera się na definicji. W każdym podzbiorze krawędzi wierzchołków S o mocy $|S|$ ma występować dokładnie $|S|-1$ krawędzi. Czyli:

$$E(S) = \left\{ \{i,j\} : v_i \overset{1}{\rightsquigarrow} v_j \in E \ \wedge \ v_i \in S \ \wedge \ v_j \in S \right\} \tag{6.1}$$

$$S \subseteq V \tag{6.2}$$

$$E(V) = E \tag{6.3}$$

$$\sum_{e \in E} x_e = |V| - 1 \tag{6.4}$$

$$\sum_{e \in E(S)} x_e = |S| - 1 \ \forall S \subseteq V \tag{6.5}$$

$$\forall e \in E \ x_e \in \{0,1\} \tag{6.6}$$

Poniższe modele można śmiało kopiować do pracy:

$$\begin{aligned}
\min \quad & \sum_{e \in E} c_e \cdot x_e, && && && (6.7a)\\
\text{s.t.} \quad & \sum_{e_\in E} x_e = |V| - 1, && && && (6.7b)\\
& \sum_{e \in E(S)} x_e = |S| - 1, && S \subseteq V, && && (6.7c)\\
& x_e \geqslant 0, && e \in E. && && (6.7d)
\end{aligned}$$

lub w formie:

$$\begin{aligned}
\min \quad & \sum_{e \in E} c_e \cdot x_e, \\
\text{s.t.} \quad & \sum_{e_\in E} x_e = |V| - 1, \\
& \sum_{e \in E(S)} x_e = |S| - 1, && S \ \subseteq V, \\
& x_e \geqslant 0, && e \ \in E.
\end{aligned} \tag{6.8}$$

Drugi model opiera się na przepływie, w punkcie początkowym v_1 mamy $n-1$ jednostek, do każdego wierzchołka innego niż v_1. Dla każdego wierzchołka poza początkowym zatem suma jednostek wychodzących ma być większa o dokładnie 1 niż jednostek przychodzących (1 jednostkę wierzchołek sobie zatrzymuje, resztę

przekazuje dalej). Jeśli dana krawędź e_{ij} należy do rozwiązania to jej flow może wynosić max $n-1$. Graf jest nieskierowany, więc to samo dla f_{ij} i dla f_{ji}. Liczba krawędzi ma być równa $n-1$.

$$\min \quad \sum_{e\in E} c_e \cdot x_e, \tag{6.9a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in E} f_{ij} - \sum_{(j,i)\in E} f_{ji} = \begin{cases} n-1 & \text{jeżeli } i=1, \\ -1 & \text{w przeciwnym przypadku,} \end{cases} \tag{6.9b}$$

$$f_{ij} \leqslant (n-1)\cdot x_{ij}, \qquad \forall \{i,j\} \in E, \tag{6.9c}$$

$$f_{ji} \leqslant (n-1)\cdot x_{ij}, \qquad \forall \{i,j\} \in E, \tag{6.9d}$$

$$\sum_{e\in E} x_e = n-1, \tag{6.9e}$$

$$f_e \geqslant 0, \qquad \forall e \in E, \tag{6.9f}$$

$$x_e \in \{0,1\}, \qquad \forall e \in E. \tag{6.9g}$$

Trzeci schemat rozwarstwia graf. Teraz każda jednostka przepływu płynie na swojej warstwie. Dodanie jeszcze jednego wymiaru skutkuje w unimodularności macierzy.

Teraz każdy wierzchołek poza początkowym ma jedną konkretną jednostkę k, którą musi dostać. Zatem dla każdej takiej jednostki suma wychodzących jednostek ze źródła musi być większa o 1 niż jednostek wchodzących do źródła. Dla każdego wierzchołka pośredniego i, jeżeli to nie jego jednostka, różnica sumy wejścia do i i wyjścia z i ma być równa 0 — węzeł przepuszcza nie swoją jednostkę.

Analogicznie dla swojej jednostki. Dla każdej jednostki k, suma wchodzących jednostek do wierzchołka, dla którego ta jednostka jest przeznaczona ma być o 1 większa niż suma jednostek z niego wychodzących — węzeł zatrzymuje swoją jednostkę.

Każdy flow f_{ij}^k jest teraz zerojedynkowy.

Mamy podziała na y_{ij} i y_{ji}, bo poprzednio mieliśmy 2 warunki na flowy, tutaj tylko 1.

$$\min \quad \sum_{(i,j)\in E} c_{ij}\cdot (y_{ij}+y_{ji}), \tag{6.10a}$$

$$\text{s.t.} \quad \sum_{(j,s)\in E} f_{js}^k - \sum_{(s,j)\in E} f_{sj}^k = -1, \qquad \forall k \in V\setminus\{v_s\}, \tag{6.10b}$$

$$\sum_{(j,i)\in E} f_{ji}^k - \sum_{(i,j)\in E} f_{ij}^k = 0, \qquad \forall i,k \in V\setminus\{v_s\} \ \wedge \ i\neq k, \tag{6.10c}$$

$$\sum_{(j,k)\in E} f_{jk}^k - \sum_{(k,j)\in E} f_{kj}^k = 1, \qquad \forall k \in V\setminus\{v_s\}, \tag{6.10d}$$

$$f_{ij}^k \leqslant y_{ij}, \qquad \forall (i,j)\in E \ \wedge \ \forall k \in V\setminus\{v_s\}, \tag{6.10e}$$

$$\sum_{(i,j)\in E} y_{ij} = n-1, \tag{6.10f}$$

$$f_{ij} \geqslant 0, \qquad \forall (i,j)\in E, \tag{6.10g}$$

$$y_{ij} \geqslant 0, \qquad \forall (i,j)\in E. \tag{6.10h}$$

Jak to przekuć na incremental:

rozdzielić zmienne decyzyjne wierzchołków na te, które są w starym rozwiązaniu i dodać warunek $\sum_{e\in E} |x_e - y_e| \leqslant K$. Wartość bezwzględną można zrobić poprzez rozbicie zmiennych na dodatnie i ujemne np. tak:

$$\min \quad \sum_{e \in E} c_e \cdot y_e, \tag{6.11a}$$

$$\text{s.t.} \quad \sum_{(j,s) \in E} f_{js}^k - \sum_{(s,j) \in E} f_{sj}^k = -1, \qquad \forall k \in V \setminus \{v_s\}, \tag{6.11b}$$

$$\sum_{(j,i) \in E} f_{ji}^k - \sum_{(i,j) \in E} f_{ij}^k = 0, \qquad \forall i,k \in V \setminus \{v_s\} \ \wedge \ i \neq k, \tag{6.11c}$$

$$\sum_{(j,k) \in E} f_{jk}^k - \sum_{(k,j) \in E} f_{kj}^k = 1, \qquad \forall k \in V \setminus \{v_s\}, \tag{6.11d}$$

$$f_{ij}^k \leqslant y_{ij}, \qquad \forall (i,j) \in E \ \wedge \ \forall k \in V \setminus \{v_s\}, \tag{6.11e}$$

$$\sum_{(i,j) \in E} y_{ij} = n - 1, \tag{6.11f}$$

$$f_{ij} \geqslant 0, \qquad \forall (i,j) \in E, \tag{6.11g}$$

$$y_{ij} \geqslant 0, \qquad \forall (i,j) \in E. \tag{6.11h}$$

gdzie x_e w tym przypadku to albo rozwiązanie odpowiadające x_e z jednemu z pierwszych formuł albo $(y_{ij} + y_{ji})$ dla trzeciego, dla $e = (i,j)$.

w każdym razie optymalne zawsze ma albo z+i = 0 albo z-i=0 (bo albo dodajemy albo odejmujemy krawedz, nie robimy tego jednoczesnie).

Incremental Network Optimization: Theory and Algorithms

Incremental problem dla MST - mamy drzewo T, mamy jakieś wstępne rozwiązanie T^*, zmieniamy koszty, chcemy znaleźć inne rozwiązanie, które nie różni się zbytnio od pierwotnego.

Kroki:

- Mamy graf $G = (N, A)$, $|N| = n$, $|A| = m$, mamy początkowe drzewo rozpinające T^0,
- Zmieniamy koszt krawędzi w drzewie (nie mówiliśmy, że T^0 jest MST, bo po zmianie kosztów pewnie i tak już nim nie jest). Szukamy T^k, które ma najmniejszy koszt dla zmodyfikowanych wag i dodatkowo różni się co najwyżej k krawędziami (ma co najwyżej k krawędzi, których nie ma w T^0 — $f\left(T^0, T^k\right) = \left|T^0 \setminus T^k\right| = \left|T^k \setminus T^0\right| \leqslant k$). Czyli szukamy takiego drzewa T^k, którego koszt jest $c\left(T^k\right) = \min\left\{c\left(T\right) : f\left(T, T^k\right) \leqslant k\right\}$.
- Będziemy szukać naszego T^k na dość okrojonym grafie. Mamy bowiem lemat, który mówi, że dla dowolnego drzewa MST $T^* \in G = (N, A)$, gdzie $|N| = n$, $|A| = m$ istnieje IMST $T^k \in G^* = (N, A^*)$, gdzie $A^* = T^* \cup T^0$, co może nam efektywnie zredukować liczbę krawędzi (co jak się potem okaże znacznie przyspieszyć może cały algorytm).

  W najgorszym przypadku oczywiście zbiory wierzchołków należących do MST T^* (MST dla zmodyfikowanych kosztów, bez uwzględniania warunku $f\left(T, T^*\right) \leqslant k$) oraz do drzewa T^0 (drzewo początkowe, będące pewnie MST dla pierwotnych kosztów) mogą być całkowicie rozłączne, co prowadzi do ograniczenia $n - 1 \leqslant |A^*| \leqslant 2 \cdot (n-1)$. Jeżeli jednak $m \gg 2 \cdot (n-1)$ (np. w grafach pełnych, gdzie $m = \binom{n}{2} = \frac{n \cdot (n-1)}{2}$) to potencjalnie zyskujemy bardzo wiele, kosztem wyliczenia MST dla danych kosztów (co możemy zrobić dowolnym algorytmem, bo koszty są stałe i nie interesują nas dodatkowe ograniczenia).

  Czemu tak jest? Szukamy IMST, które od T^0 różni się o nie więcej niż k krawędzi. Niech T^k będzie takim IMST spośród wszystkich optymalnych rozwiązań, które ma maksymalną liczbę krawędzi w A^*. Jeżeli ma wszystkie ($\forall e \in T^k \implies e \in A^*$) to $T^k \in G^*$ i nie ma czego dowodzić.

  Załóżmy zatem, że tak nie jest i istnieje krawędź $e \in T^k \setminus A^*$. Usuńmy teraz tą krawędź z drzewa T^k. Otrzymamy podział na dwa poddrzewa: S oraz $\overline{S}$ a także zbiór $\mathcal{Q}\left(T^k, e\right)$, zawierający wszystkie takie krawędzie (i, j), gdzie $i \in S$ a $j \in \overline{S}$. Z własności optymalnego cięcia MST (drzewo jest MST wtedy i tylko wtedy, gdy należące do niego krawędzie e mają najmniejszy koszt spośród wszystkich innych krawędzi, które należą do zbioru $\mathcal{Q}\left(T^*, e\right)$) wiemy, że w zbiorze $\mathcal{Q}\left(T^k, e\right)$ jest krawędź $e' \in T^*$ (która ma najmniejszy koszt spośród wszystkich krawędzi, należących do tego cięcia). Wiemy, że e nie należy do A^*, więc też $e \notin T^*$, więc $e \neq e^*$. Stąd, tworząc nowe drzewo $T^{k\prime}$ poprzez usunięcie z niego krawędzi $e \in T^k \setminus A^*$, dodając do niego krawędź $e' \in T^* \in A^*$, otrzymujemy drzewo o koszcie na pewno nie większym niż drzewo T^k. Skoro T^k było MST, więc nowe drzewo jest takim także (z $c\left(T^{k\prime}\right) \leqslant c\left(T^k\right)$ oraz z optymalności T^k mamy, że $c\left(T^{k\prime}\right) = c\left(T^k\right)$). A skoro $T^{k\prime}$ też jest MST to mamy sprzeczność, że T^k jest MST i ma największą liczbę krawędzi w A^*, bo $T^{k\prime}$ ma ich o jedną więcej.

  Nowe drzewo $T^{k\prime}$ spełnia też w oczywisty sposób $f\left(T^0, T^{k\prime}\right) \leqslant k$ — $T^{k\prime} = T^k - e + e'$, gdzie $e \notin A^* \equiv e \notin T^* \cup T^0 \equiv e \notin T^* \wedge e \notin T^0$, stąd po usunięciu z drzewa T^k krawędzi e ($T^k \setminus \{e\}$), ma ono mniej krawędzi nie będących w pierwotnym drzewie T^0, jako że usunięta krawędź na pewno do niego nie należała. Stąd $f\left(T^0, T^k \setminus \{e\}\right) < k$ oraz $f\left(T^0, T^{k\prime}\right) \leqslant k$.

- Następnie formułujemy sam problem. Celem jest wykorzystanie własności relaksacji Lagrangian'a do sprowadzenia modelu IMST do MST ze zmodyfikowanymi kosztami krawędzi, zależnie od λ. Jest oczywistym, że zwykły problem MST możemy rozwiązać nie tyle co LP, co innymi algorytmami, takimi jak Kruskala, Prima, czy Sollin'a. Przy wykorzystaniu zaawansowanych struktur (Johnson's data structure), możemy zejść z czasem poszukiwania MST dla konkretnych danych do $O\left(m \cdot \log\left(\log\left(C\right)\right)\right)$ (choć pewnie samo tworzenie takiej struktury jest koszmarnie drogie). Weźmy rozpatrzmy model LP dla MST:

  Pierwszy z modeli jest najprostszy i wynika bezpośrednio z definicji bycia drzewem rozpinającym.

$$\sum_{e \in E} x_e = |V| - 1 \tag{7.1}$$

$$\sum_{e \in E(S)} x_e = |S| - 1 \; \forall S \subseteq V, \tag{7.2}$$

  gdzie $G = (V, E)$, $\forall S \subseteq V \; E\left(S\right) = \{e = (i, j) : e \in E, i, j \in S\}$ (zbiór krawędzi łączących ze sobą dany podzbiór wierzchołków grafu G), zaś $x_e = 1 \iff e \in MST$. Minimalne drzewo rozpinające musi mieć dokładnie $|V| - 1$ krawędzi, nie może mieć cykli (czyli każdy podzbiór $S \subseteq V$ musi być drzewem - mieć $|S| - 1$ krawędzi). Każda krawędź e ma koszt c_e, oczywiście chcemy minimalizować $\sum_{e \in E} c_e \cdot x_e$.

  Ten model ma jednak $P\left(V\right) + 1 = 2^{|V|} + 1$ ograniczeń, co go raczej dyskwalifikuje. Na plus na pewno jest to, że zmienne decyzyjne nie muszą być całkowitymi - z unimodularności macierzy mamy, że wynik i tak będzie całkowity.

  Drugiego i trzeciego modelu nie będę tutaj przedstawiać, bo nie o to chodzi.

  Co ważne to jest to, że przedstawiony model MST teraz chcielibyśmy przekształcić do modelu IMST:

$$\begin{aligned}
\min \quad & \sum_{e_i \in E} c_i \cdot x_i && && (7.3a)\\
\text{s.t.} \quad & \sum_{e_i \in E} x_i = |V| - 1, && && (7.3b)\\
& \sum_{e_i \in E(S)} x_i = |S| - 1, && S \subseteq V, && (7.3c)\\
& x_i \geqslant 0, && e_i \in E && (7.3d)
\end{aligned}$$

(a)

$$\begin{aligned}
\min \quad & \sum_{e_i \in E} c_i \cdot x_i && && (7.4a)\\
\text{s.t.} \quad & \sum_{e_i \in E} x_i = |V| - 1, && && (7.4b)\\
& \sum_{e_i \in E(S)} x_i = |S| - 1, && S \subseteq V, && (7.4c)\\
& x_i \geqslant 0, && e_i \in E && (7.4d)\\
& \sum_{e_i \in T^* \setminus T^0} x_i \leqslant k, && && (7.4e)\\
& x_i = 0, && e_i \notin A^* && (7.4f)
\end{aligned}$$

(b)

  Do pierwszego modelu dodaliśmy ograniczenie, że dla początkowego drzewa rozpinającego T^0, po zmianie kosztów krawędzi, nowe optymalne rozwiązanie T^* nie może się różnić od T^0 większą liczbą niż k krawędzi. Dodatkowo pokazaliśmy, że nowe drzewo MST na pewno będzie się znajdować w $G^* = (V, E^*)$, gdzie $E^* = T^* \cup T^0$ (T^* to dowolne MST dla nowych kosztów w grafie), więc śmiało możemy wszystkie zmienne decyzyjne, które decydują o przynależności do MST krawędzi spoza zbioru E^*, ustawić na 0.

  Teraz, poza warunkiem 7.4e, model jest ciągle prostym modelem na MST - dodatkowy warunek, że niektóre krawędzie nie znajdą się w MST, możemy wymusić na algorytmie Prima/Kruskala itd. po prostu dając mu do rozwiązania G^* zamiast G. Zrelaksujmy zatem ograniczenie 7.4e. Po drodze zauważmy, że warunek $\sum_{e_i \in T^* \setminus T^0} x_i \leqslant k$ jest równoważny $\sum_{e_i \in T^0} x_i \geqslant (n-1) - k$ ($|e : e \in T^*| - \left|e : e \in T^0\right| = \left|e : e \in T^* \setminus T^0\right| \leqslant k$ z pierwszego założenia i $|e : e \in T^*| = n - 1$, więc $(n-1) - k \leqslant \left|e : e \in T^0\right|$).

$$\min \quad \sum_{e_i \in E^*} c_i \cdot x_i \tag{7.5a}$$

$$\text{s.t.} \quad \sum_{e_i \in E^*} x_i = |V| - 1, \tag{7.5b}$$

$$\sum_{e_i \in E^*(S)} x_i = |S| - 1, \qquad S \subseteq V, \tag{7.5c}$$

$$x_i \geqslant 0, \qquad e_i \in E^*, \tag{7.5d}$$

$$\sum_{e_i \in T^0} x_i \geqslant n - 1 - k. \tag{7.5e}$$

Rysunek 7.2

$$\min \quad \sum_{e_i \in E^*} c_i \cdot x_i + \lambda \cdot \left((n-1-k) - \sum_{e_i \in T^0} x_i \right) \tag{7.6a}$$

$$\text{s.t.} \quad \sum_{e_i \in E^*} x_i = |V| - 1, \tag{7.6b}$$

$$\sum_{e_i \in E^*(S)} x_i = |S| - 1, \qquad S \subseteq V, \tag{7.6c}$$

$$x_i \geqslant 0, \qquad e_i \in E^*. \tag{7.6d}$$

Rysunek 7.3

Po zrelaksowaniu, teraz od parametru λ zależy jakość rozwiązania. Jeżeli będzie ono dopuszczalne dla niezrelaksowanego problemu to nic się nie dzieje. Jeżeli nie będzie dopuszczalne (czyli $\sum_{e_i \in T^0} x_i < n-1-k$), wtedy wartość funkcji celu odpowiednio się pogorszy, wzrośnie.

$$L(\lambda, x) = \sum_{e_i \in E^*} c_i \cdot x_i + \lambda \cdot \left((n-1-k) - \sum_{e_i \in T^0} x_i \right) = \tag{7.7}$$

$$\sum_{e_i \in T^* \cup T^0} c_i \cdot x_i - \lambda \sum_{e_i \in T^0} x_i + \lambda \cdot (n-1-k) = \tag{7.8}$$

$$\left(\sum_{e_i \in T^* \setminus T^0} c_i \cdot x_i + \sum_{e_i \in T^0} c_i \cdot x_i \right) - \lambda \sum_{e_i \in T^0} x_i + \lambda \cdot (n-1-k) = \tag{7.9}$$

$$\sum_{e_i \in T^* \setminus T^0} c_i \cdot x_i + \sum_{e_i \in T^0} (c_i - \lambda) \cdot x_i + \lambda \cdot (n-1-k) \tag{7.10}$$

$$L(\lambda) = \min_{x \in X} L(\lambda, x) = \min \left\{ \sum_{e_i \in T^* \setminus T^0} c_i \cdot x_i + \sum_{e_i \in T^0} (c_i - \lambda) \cdot x_i + \lambda \cdot (n-1-k) \right\} = \tag{7.11}$$

$$= \min \left\{ \sum_{e_i \in T^* \setminus T^0} c_i \cdot x_i + \sum_{e_i \in T^0} (c_i - \lambda) \cdot x_i \right\} \tag{7.12}$$

.

Po kolei, co się zadziało: przekształciliśmy funkcję $L(\lambda, x)$ zgodnie z operacjami na zbiorach (wystarczy rozrysować), skoro $\forall e_i \notin E^* x_i = 0$, więc nie ma sensu w wartości funkcji celu uwzględniać tych krawędzi, bo dla nich $c_i \cdot x_i = 0$. Parametry n i k są stałe, więc $L(\lambda)$ zależy tylko od x_i, C_i i λ (oczywiście wartość $L(\lambda)$ już od k i n zależy).

I tutaj dochodzimy do sedna, bo z relaksacji Lagrangea otrzymaliśmy znowu model na MST, tyle że z nieco innymi kosztami, które teraz zależą od parametru λ. Ustalając go sobie odpowiednio, dostajemy niczym nie różniący się od podstawowego model MST, który będziemy rozwiązywać standardowymi algorytmami (koszty krawędzi należących do starego rozwiązania będą wynosić $c_i - \lambda$, wszystkie pozostałe będą kosztować po staremu (czyli po nowemu, ale bez kolejnych zmian, wynikających z relaksacji problemu)).

- dla tak utworzonych danych odpalamy algorytm, ale wcześniej słów parę czemu to działa.

Jak widzimy, minimalizujemy $\sum_{e_i \in T^* \setminus T^0} c_i \cdot x_i + \sum_{e_i \in T^0} (c_i - \lambda) \cdot x_i$. Im większą lambdę weźmiemy, tym mniejszy koszt będą miały krawędzie należące do starego rozwiązania. Czyli kiedyś, przy dostatecznie dużym λ, uda nam się skonstruować optymalne rozwiązanie dla tego λ, które będzie miało nie więcej niż k nowych krawędzi, a resztę wspólną.

7.0.1 Lagrangian

Będziemy teraz chcieli określić warunki, przy których się zatrzymamy, bo osiągniemy optimum.
Weźmy ogólny problem optymalizacyjny:

$$\begin{aligned} z^* = \min \quad & cx && (7.13a)\\ \text{s.t.} \quad & Ax = b, && (7.13b)\\ & x_i \in X. && (7.13c) \end{aligned}$$

Rysunek 7.4

Relaksując, dostajemy taki model:

$$\begin{aligned} L(\lambda) = \min \quad & cx + \lambda (Ax - b) && (7.14a)\\ \text{s.t.} \quad & Ax = b, && (7.14b)\\ & x_i \in X. && (7.14c) \end{aligned}$$

Rysunek 7.5

Oczywiście, $z^* = \min_{x \in X} \{cx : Ax = b\} \geqslant \min_{x \in X} \{cx + \lambda (Ax - b)\} = L(\lambda)$. Wystarczy chociażby sobie rozrysować zbiory rozwiązań dopuszczalnych, gdzie oczywiście pierwszy (z ograniczeniem) zawiera się w drugim (bez ograniczeń) i po jego poszerzeniu (przy pozbyciu się ograniczenia) możemy znaleźć lepsze rozwiązanie, niż to, które jest w pierwszym. Jeśli będą same większe rozwiązania, to minimum nadal pozostanie rozwiązanie sprzed relaksacji, a więc zawsze, dla każdego λ $L(\lambda) \leqslant z^*$.

Skoro $L(\lambda)$ jest zawsze dolnym ograniczeniem na optymalne rozwiązanie z^*, to chcielibyśmy mieć je jak najdokładniejsze, czyli znaleźć $L^* = \max_\lambda L(\lambda)$. Oczywiście, skoro dla każdego λ $L(\lambda) \leqslant z^*$ to i $L^* \leqslant z^*$. Z

drugiej strony dla dowolnego λ $L(\lambda) \leqslant \max_\lambda L(\lambda) = L^*$. Stąd mamy $L(\lambda) \leqslant L^* \leqslant z^*$. z^* jest optymalnym rozwiązaniem dla oryginalnego problemu, więc wartość funkcji celu dla tego rozwiązania jest na pewno nie większa (minimalizujemy) niż dla wszystkich dopuszczalnych rozwiązań: $z^* = c \cdot x^* \leqslant cx$, gdzie $x \in X$.

Tym samym: $L(\lambda) \leqslant L^* \leqslant z^* = c \cdot x^* \leqslant c \cdot x$.

Teraz, jeżeli mamy wektor λ i dopuszczalne rozwiązanie x oryginalnego problemu, które dodatkowo spełnia $L(\lambda) = c \cdot x$ (wszystkie zrelaksowane ograniczenia zaszły w postaci równości lub $\lambda = 0$), to z $L(\lambda) = c \cdot x$ mamy $L(\lambda) = L^* = z^* = c \cdot x^* = c \cdot x$, a stąd mamy, że nasze dopuszczalne rozwiązanie x jest rozwiązaniem optymalnym dla oryginalnego problemu. Dodatkowo tak wybrane λ jest optymalnym rozwiązaniem dla problemu mnożników Lagrangiana, gdyż $L^* = \max_\lambda L(\lambda)$, a najwyższym możliwym dolnym ograniczeniem dla z^* $(\forall \lambda L(\lambda) \leqslant z^*)$ jest właśnie $L(\lambda) = z^*$.

Aby zaszło $L(\lambda) = c \cdot x$, albo wszystkie ograniczenia musiałyby zajść w postaci równości, albo $\lambda = 0$ — w przypadku tej drugiej opcji wracamy do punktu wyjścia: $L(0) = \min cx + 0(Ax - b) = \min cx$. Ba, nawet jest gorzej, bo musimy wstrzelić się w dopuszczalne rozwiązanie x dla oryginalnego problemu bez podania ograniczenia, które zniknęło nam w relaksacji, tak więc algorytm rozwiązujący nowy model będzie wypluwał też rozwiązania niedopuszczalne. W pierwszym przypadku sprawa natomiast jest jasna - wszystkie ograniczenia zaszły w postaci równości, rozwiązanie x jest zatem dopuszczalne, funkcja celu zmienia się z $L(\lambda) = \min cx + \lambda(Ax - b) = \min cx$ na $L(\lambda) = \min cx$, co tak naprawdę jest tym samym, co funkcja celu dla oryginalnego problemu.

Dzięki nierównościom $L(\lambda) \leqslant L^* \leqslant z^* = c \cdot x^* \leqslant c \cdot x$ i wiedzy, że optymalne rozwiązanie będzie wtedy, gdy to wszystko będzie równe, możemy szacować jak daleko od rozwiązania aktualnie jesteśmy dla DOWOLNEGO x (nie znając optymalnej wartości): $\frac{cx - L(\lambda)}{L(\lambda)}$.

Zatem wreszcie dochodzimy do ostatecznego wniosku: że gdy zrelaksowany problem ma rozwiązanie x i spełnia warunek różnic dopełniających $\lambda(Ax - b) = 0$ i jest dopuszczalnym rozwiązaniem w oryginalnym problemie, wtedy x jest optymalny dla tego problemu.

7.0.2 I dalej w algorytm

- Z powyższego wynika podstawowe twierdzenie 2.5, które daje nam algorytm:

  Drzewo rozpinające T jest IMST wtedy i tylko wtedy, gdy istnieje $\lambda \geqslant 0$ takie, że T jest optymalne dla $L(\lambda)$ i zachodzi warunek komplementarności: $\lambda \cdot \left(\sum_{e_i \in T \setminus T^0} x_i - k \right) = 0$, czyli gdy:

  - $f\left(T, T^0\right) = k \wedge \lambda = 0$ lub
  - $f\left(T, T^0\right) < k \wedge \lambda = 0$ lub
  - $f\left(T, T^0\right) = k \wedge \lambda \neq 0$

  po skróceniu:

  - $f\left(T, T^0\right) \leqslant k \wedge \lambda = 0$ lub
  - $f\left(T, T^0\right) = k \wedge \lambda \neq 0$

  Łatwo to pokazać: T— IMST $\iff \exists \lambda \geqslant 0 : L(\lambda) = L^* \wedge \lambda \cdot \left(f\left(T, T^0\right) - k \right) = 0$

  Część pierwsza (A if B –¿ B if A) wynika prosto z poprzedniego twierdzenia. A only if B –¿ -B -¿ -A –¿ A -¿ B:

  Jeśli T jest IMST to jest też MST i na pewno jest optymalnym rozwiązaniem $L(0) = z^*$ i $f\left(T, T^0\right) \leqslant k$ (bo T jest IMST). Załóżmy, że T jest IMST, ale nie zaszedł pierwszy przypadek (czyli, że $f\left(T, T^0\right) = t > k$ lub $\lambda \neq 0$). Weźmy $\lambda \geqslant 0$, dla którego drugi z opcjonalnych warunków ma być prawdziwy. Patrząc na koszty krawędzi: $\sum_{e_i \in T^* \setminus T^0} c_i \cdot x_i + \sum_{e_i \in T^0} (c_i - \lambda) \cdot x_i$ widzimy, że wraz ze zwiększaniem λ koszty krawędzi należących do T^0 spadają, więc w jakimś momencie, dla λ_1, $c_{e_0} - \lambda_1 = c_{e_0}(\lambda_1) = c_{e_0^*}$, gdzie e_0 to krawędź należąca do rozwiązania, a należąca do starego drzewa T^0, zaś $e_0^* \in T(\lambda_0) \setminus T^0$. Skoro doszliśmy do punktu, gdzie sparametryzowane koszty tych krawędzi są równe, to możemy je ze sobą zamienić miejscami, nie powodując żadnej zmiany kosztów w drzewie, a nowe drzewo $T(\lambda_1)$ będzie spełniało równość $f\left(T(\lambda_1), T^0\right) = t - 1$. Ogólnie, przy zwiększaniu λ, zawsze będzie zachodzić:

$f\left(T\left(\lambda_i\right), T^0\right) = f\left(T\left(\lambda_{i-1}\right), T^0\right) - 1 = f\left(T\left(\lambda_0\right) = T, T^0\right) - i = t - i$. Dla wygody dowodzenia zakłada się, że wszystkie koszty krawędzi są różne i że otrzymane w ten sposób drzewo jest wtedy unikalne. Bez tego założenia też wszystko śmiga, z tym że mogą się zdarzyć sytuacje, w których moglibyśmy od razu zamienić ze sobą miejscami kilka krawędzi, co nie wpływa na cały proces budowania drzewa.

Z powyższego wynika, że funkcja $f\left(T\left(\lambda\right), T\right)$ jest monotonicznie malejąca, zaś z $f\left(T\left(\lambda_i\right), T^0\right) = t - i$ mamy, że $\lambda^* = \lambda_{t-k}$ ($f\left(T\left(\lambda_{t-k}\right), T^0\right) = t - (t-k) = k$), zaś jej wartość to $\lambda^* = \lambda_{t-k} = c_{e_{e-k}} - c_{e^*_{t-k}}$ (bo wtedy $c_{e_{t-k}}\left(\lambda_{t-k}\right) = c_{e_{t-k}} - \lambda_{t-k} = c_{e_{t-k}} - \left(c_{e_{e-k}} - c_{e^*_{t-k}}\right) = c_{e^*_{t-k}}$), gdzie $e_{t-k} \in T^0 \setminus T\left(\lambda_{t-k-1}\right)$ (krawędź należąca do zbioru krawędzi T^0, ale nie należąca jeszcze do dotychczas skonstruowanego drzewa), a $e^*_{t-k} \in T\left(\lambda_{t-k-1}\right) \setminus T^0$ (krawędź, która wylatuje z obecnego rozwiązania, a nie należy do T^0 — czyli jej pozbycie się redukuje liczbę niewspólnych krawędzi).

- Aby zapewnić unikalność rozwiązania, można albo zastosować pewien porządek na krawędziach albo zmodyfikować ich koszty tak, aby każda krawędź miała inne przy nadal zachowanym porządku ich wielkości.

  Weźmy zatem zmodyfikujmy koszty: $c'_{e_i} = c_{e_i} + \phi\left(i\right)$, gdzie $\phi\left(i\right) = \frac{m \cdot i^2 + i}{(m+1)^3}$.

  Pokażemy, że:

  P1 $c_{e_i} < c'_{e_i} < c_{e_i} + 1$

  P2 Dla każdego i, j, k, l, gdzie $i \neq j$ oraz $i \neq k$ mamy $c'_{e_i} - c'_{e_j} \neq c'_{e_k} - c'_{e_l}$

  Zaczynając od P1:

  Najpierw pokażmy, że $\phi\left(i\right) > 0$. $i > 0$, bo numerujemy krawędzie od 1 do $|E|$. $m = |E|$. Działa, sprawdzone WolframAlphą (dla $0 < n \leqslant m$).

  Drugie: wziąłem $j = l$, by było prościej i w zasadzie wolfram pokazał, że nie: $c_{e_i} + \phi\left(i\right) \neq c_{e_k} + \phi\left(k\right) \equiv \phi\left(i\right) - \phi\left(k\right) \neq c_{e_k} - c_{e_i}$, $c_e \in \mathbb{N}$, $c_{e_k} - c_{e_i} \in \mathbb{Z}$, a jedyne całkowite rozwiązania dla $\phi\left(i\right) - \phi\left(k\right)$ są wtedy, gdy $k < 0$ lub $m < -2$, co pozwala posunąć się do stwierdzenia, że $c_{e_i} + \phi\left(i\right) \neq c_{e_k} + \phi\left(k\right)$. Co do pełnej wersji nierówności to nie wiem. Pełna wersja przyda się później, gdzie dzięki niej będziemy mieli pewność, że kolejne λ, które z Tw 2.5 maja wartość równą różnicy kosztów krawędzi, są różne od siebie.

  Oczywiście, graf z takimi kosztami wygeneruje dokładnie takie samo drzewo, chociażby z racji, że w zachłannym algorytmie najpierw sortujemy krawędzie po kosztach, a ich kolejność się nie zmieniła (ze względu na własność P1). Oczywiście jest drobny haczyk, jeżeli istnieje kilka krawędzi $e_{i_1}, \cdots, e_{i_j}$ o tym samym koszcie, gdzie $\forall k\ i_k < i_{k+1}$, to zachodzi wtedy $\forall k \in \{1, \cdots, j\}\ c_{e_{i_k}} < c'_{e_{i_1}} < \cdots < c'_{e_{i_j}} < c_{e_{i_k}} + 1$. Jeśli teraz w MST $T^{*\prime}$, otrzymanym za pomocą zachłannego algorytmu z G', znajdzie się tylko część krawędzi, które w oryginalnym drzewie mają ten sam koszt ($\{e_{i_k} : 1 \leqslant k < j\} \in T^{*\prime}$) to nie mamy gwarancji, że ten sam podzbiór krawędzi znajdzie się również w MST T^*, które zostanie skonstruowane dla grafu G, gdzie koszty krawędzi $e_{i_1}, \cdots, e_{i_j}$ są takie same. Możemy tylko powiedzieć, że drzewo $T^{*\prime}$ ma ten sam koszt co drzewo T^* i że mogą, ale nie muszą, być to te same drzewa. Chyba, że umówimy się, że naturalnym porządkiem dla krawędzi $e_{i_1}, \cdots, e_{i_j}$ w posortowanym ciągu dla algorytmu zachłannego jest porządek zgodny z ich rosnącymi indeksami - wtedy OK.

- Sam algorytm szuka takiego λ aby zapewnić $f\left(T\left(\lambda\right), T^0\right) = k \wedge \lambda \neq 0$ (jeden z warunków komplementarności), gdzie T^0 to rozwiązanie stare, dla starych kosztów, zaś $T\left(\lambda\right)$ to drzewo znalezione przez algorytm do MST dla kosztów $c_i - \lambda$ dla $\forall i\ :\ e_i \in T^0$ i c_i dla pozostałych (czyli $T^* \setminus T^0$).

- Posiłkujemy się teorią o tym, że nasze rozwiązanie znajduje się na pewno w $G^* = (V, E^*)$, gdzie $E^* = \{e \in T^*\} \cup \{e \in T^0\}$. by ustalić ten zbiór E^*, najpierw musimy mieć stare rozwiązanie T^0, dostać informację o zmienionych kosztach, dla których chcemy znaleźć T^k i policzyć MST T^* dla tych kosztów, na razie bez uwzględnienia ograniczenia $f\left(T^*, T^0\right) \leqslant k$.

- Mając już G^*, możemy zacząć szukać λ^*. Bierzemy ograniczenia na λ^*: $L \leqslant \lambda^* \geqslant U$ i zaczynamy szukać. Z każdą iteracją dostajemy jakieś λ, które po policzeniu MST dla kosztów generowanych przez λ daje nam albo $f\left(T^*, T^0\right) < k$, albo $f\left(T^*, T^0\right) > k$, albo $f\left(T^*, T^0\right) = k$. W tym trzecim przypadku znaleźliśmy optymalną wartość λ i kończymy, jako że spełniliśmy warunek komplementarności i nasze

rozwiązanie jest optymalne. Jeśli $f\left(T^*, T^0\right) > k$, to znaczy, że nasza λ jest za mała (pokazaliśmy, że $f\left(T^*, T^0\right)$ maleje wraz ze wzrostem λ). Odpalamy zatem nasz algorytm od nowa, tym razem z nowym oszacowaniem dla λ^*: $\lambda \leqslant \lambda^* \geqslant U$. Jeśli $f\left(T^*, T^0\right) < k$, to znaczy, że nasza λ jest za duża, więc odpalamy zatem nasz algorytm od nowa, tym razem z nowym oszacowaniem dla λ^*: $L \leqslant \lambda^* \geqslant \lambda$. Tak powtarzamy dopóki ograniczenia dla λ^* nie będą już na tyle ciasne, by możliwych wartości dla λ^* pozostało na tyle mało, byśmy mogli spokojnie posłużyć się przeszukiwaniem binarnym. Znaczy cały czas szukaliśmy „binarnie", ale nieco przyspieszonym sposobem, robiąc duże przesiewy w każdej iteracji, teraz możemy zejść na mniejsze kroczki.

- Bierzemy pozostałe możliwe wartości dla λ^*, sortujemy je i szukamy odpowiedniego λ tak samo jak wyżej, dzieląc zbiór do przeszukania za każdym razem po połowie (binnary search w końcu). W skończonym czasie znajdziemy odpowiednie λ.

- W detalach: Nasze możliwe λ tworzymy w ten sposób, że $\lambda(i,j) = d(i,j) = c_{e_i} - c_{e_j^*}$ (z teorii mamy, że λ to różnica kosztu krawędzi $e \in T^0 \setminus T^*$ i $e^* \in T^* \setminus T^0$). Sortując pierwsze krawędzie rosnąco po kosztach, drugie zaś malejąco otrzymamy, że $d(i,j)$ rośnie wraz ze zwiększaniem i lub j. Stąd, gdy szukamy odpowiedniego zbioru λ, które spełniają $L \leqslant \lambda \geqslant U$, możemy skorzystać z tej właściwości monotoniczności $d(i,j)$. Wiemy, że $MaxIndex(i) \geqslant MaxIndex(i+1)$ ($\max\{j : d(i,j) \leqslant U\} \geqslant \max\{j : d(i+1,j) \leqslant U\}$, bo $d(i+1,j) \geqslant d(i,j)$, tym samym maksymalne j dla $MaxIndex(i+1)$ na pewno nie będzie większe niż dla $MaxIndex(i)$ (bo $d(i, MaxIndex(i)) \leqslant U$, ale $d(i+1, MaxIndex(i)) \leqslant U$ już niekoniecznie musi zachodzić)). Tak więc ciąg $MaxIndex(1) \geqslant MaxIndex(2) \geqslant MaxIndex(n)$. Stąd też każdy następny $MaxIndex(i+1)$ możemy zaczynać szukać od $j = MaxIndex(i)$, który albo od razu będzie rozwiązaniem, albo będzie mniejszy, czyli $MaxIndex(i+1) \leqslant MaxIndex(i)$. Będziemy tak sobie schodzić z j dla każdego i, więc cała ta zabawa zajmie $O(n)$.

- Podobnie z $MinIndex(i) = min\{j : d(i,j) \geqslant L\} \geqslant min\{j : d(i+1,j) \geqslant L\} = MinIndex(i+1)$

- Co więcej, zachodzi $L \leqslant d(i, MinIndex(i)) \leqslant d(i, MinIndex(i)+1) \leqslant d(i, MaxIndex(i)-1) \leqslant d(i, MaxIndex(i)) \leqslant U$, więc liczba tych dopuszczalnych j dla i jest równa $MaxIndex(i) - MinIndex(i) + 1$ lub 0.

- $|H|$ jest oczywiście ograniczone z dołu przez $6n$ ($\frac{TotalCount}{K} = \frac{TotalCount}{\left\lfloor \frac{TotalCount}{6n} \right\rfloor} = 6n$).

- Z góry jest nieco bardziej skomplikowana sprawa: $|H| \leqslant 9n$. Najpierw zauważmy, że $K \geqslant 2$, gdyż $TotalCount > 12n$. Iloraz $\frac{TotalCount}{\left\lfloor \frac{TotalCount}{6n} \right\rfloor}$ będzie największy wtedy, gdy $TotalCount = 6 \cdot k \cdot n - 1$ — wtedy $\left\lfloor \frac{TotalCount}{6n} \right\rfloor = k - 1$. $TotalCount > 12 \cdot n$ więc weźmy $\frac{TotalCount}{K} = \frac{12 \cdot n + (6 \cdot k \cdot n - 1)}{\left\lfloor \frac{12 \cdot n + (6 \cdot k \cdot n - 1)}{6n} \right\rfloor} = \frac{12 \cdot n + 6 \cdot k \cdot n - 1}{2 + \left\lfloor \frac{6 \cdot k \cdot n - 1}{6n} \right\rfloor} = \frac{12 \cdot n + 6 \cdot k \cdot n - 1}{2 + (k-1)}$. Zatem szukamy takiego l, że $l = \min\left\{ l' : \frac{12 \cdot n + 6 \cdot k \cdot n - 1}{k+1} \leqslant l' \cdot n \right\}$. Z wolframaAlphy wychodzi, że $l = 9$, gdzie nierówność jest spełniona dla $k \geqslant 1$ i $n > 0$. Dla $k < 1$ ($k \in N$) otrzymujemy natomiast $TotalCount = 12 \cdot n - 1 \leqslant 12 \cdot n$, tak więc dla $k = 0$ w ogóle nie rozpatrujemy, bo bylibyśmy w przypadku, gdzie odpalamy BS.

- Jeśli dla znalezionej mediany $\overbrace{\lambda}$ znaleźliśmy MST, które spełnia $f\left(T\left(\overbrace{\lambda}\right), T^0\right) = k$ wtedy się cieszymy i kończymy algorytm ze znalezionym optymalnym IMST. Jeśli nie to mamy dwa przypadki:

  - $f\left(T\left(\overbrace{\lambda}\right), T^0\right) > k$, czyli wybrana $\overbrace{\lambda} < \lambda$. Skoro $\overbrace{\lambda}$ jest medianą z równomiernie wybranych dopuszczalnych par $d(i,j) \in H$, to co najmniej połowa par z H spełnia $d(i,j) \leqslant \overbrace{\lambda}$. Mamy ograniczenie: $6 \cdot n \leqslant |H|$, więc tych par jest co najmniej $3 \cdot n$. Pamiętamy, że do zbioru h braliśmy co K'ty element ($|H| = \frac{TotalCount}{K}$), więc wszystkich osiągalnych elementów $d(i,j)$ takich, że $d(i,j) \leqslant \overbrace{\lambda}$ jest co najmniej $3 \cdot n \cdot K = 3 \cdot n \cdot \left\lfloor \frac{TotalCount}{6n} \right\rfloor \geqslant \frac{TotalCount}{2} - 3 \cdot n \geqslant \frac{TotalCount}{4}$ (jako, że pamiętamy, że $TotalCount > 12 \cdot n$, więc dla granicznego przypadku jest równość, dla większych wartości tym bardziej jest OK). W związku z tym możemy odrzucić co najmniej $\frac{1}{4}$ z dostępnych osiągalnych par (i,j), bo na pewno dla nich $d(i,j) <= \overbrace{\lambda} < \lambda^*$.

– $f\left(T\left(\widehat{\lambda}\right), T^0\right) < k$, czyli wybrana $\widehat{\lambda} > \lambda$. Skoro $\widehat{\lambda}$ jest medianą z równomiernie wybranych dopuszczalnych par $d(i,j) \in H$, to co najmniej połowa par z H spełnia $d(i,j) \geqslant \widehat{\lambda}$. Mamy ograniczenie: $6 \cdot n \leqslant |H|$, więc tych par jest co najmniej $3 \cdot n$. ....yyyy....

Rzeczy, których nie potrafię wyjaśnić:

– Czemu takie dziwne ograniczenie $\frac{1}{4}n^2 < \lambda^* \leqslant C$?
– Skąd się wzięły te pierwsze złożoności, np. $O\left(n^2 k\right)$?
– Czy P2 faktycznie tak łatwo udowodnić?
– Czemu w drugim przypadku mamy $2nK$??
– Czy w 4 równaniu nie ma czasem błędu? Czemu tam jest minus? U mnie wyszedł plus.
– Czasy działania z odwróconą funkcją Ackermana można sobie darować, zostawić na później ewentualnie.
– Czy przy sortowaniu nie można zrobić tego jakoś lepiej niż w $O(n \log(n))$? Np. przesortować tylko pierwszy wiersz lub kolumnę — reszta w kolumnie lub wierszu jest tak samo oddalona, więc wystarczy naprzemiennie brać kolejne elementy z kolumny lub wiersza.

WAŻNE: Modyfikacje kosztów są jednak istotne, gdyż w przeciwnym wypadku możemy natknąć się na sytuacje, w której będziemy mogli wybrać dobre rozwiązanie IMST, ale algorytm do MST wybierze złe rozwiązanie — gdy np. dobijemy przy przeszukiwaniu binarnym do końca, a zmodyfikowane koszta na krawędziach będą takie same — MST może ale nie musi wtedy wybrać IMST z ograniczeniem k i... krach.

Orlin Network Flows

Jeśli w funkcji celu potrzebujemy iloczyn kosztów krawędzi to można te koszty do modelu zlogarytmować w myśl równości:

$$\log(x \cdot y) = \log(x) + \log(y) \tag{8.1}$$

Jeśli mamy problem minimaxowy (koszt ścieżki z i do j jest równy maksymalnemu kosztowi krawędzi na tej ścieżce), gdzie chcemy zminimalizować koszt danej ścieżki to budując drzewo MST możemy natychmiastowo rozwiązać ten problem. Niech P będzie ścieżką w MST od p do q i niech jej koszt wynosi c_{ij} (krawędź (i,j)) jest na ścieżce i ma największy koszt spośród ścieżek należących do P). Usuwając z MST krawędź (i,j) tworzą się dwa osobne zbiory S i $\overline{S}$ (robimy cięcie na drzewie — teraz początek ścieżki $p \in S$ a $q \in \overline{S}$). Dodatkowo dla każdego łuku (k,l), gdzie $k \in S$ i $l \in \overline{S}$ zachodzi $c_{ij} \leqslant c_{kl}$ (bo jeśli nie to w MST poddrzewa S i $\overline{S}$ byłyby połączone inną krawędzią, której koszt jest mniejszy od (i,j) - więc pierwotne MST nie byłoby minimalne). Teraz wystarczy zauważyć, że krawędź (i,j) to właśnie szukany minimalny maksymalny koszt danej ścieżki — koszt krawędzi (i,j) w ścieżce P jest największy, lecz jest on najmniejszy spośród wszystkich krawędzi, które łączyć by mogły poddrzewa s oraz $\overline{S}$ (powodując, że da się dojść z p do q). Stąd wniosek, że MST zawiera rozwiązanie problemu minimaksowego.

Warunki optymalności:

- T^* jest MST wtedy i tylko wtedy, gdy dla każdej krawędzi $(i,j) \in T^*$, dla każdego $k \in S$, każdego $l \in \overline{S}$ takich, że (k,l) zachodzi $c_{ij} \leqslant c_{kl}$, gdzie S i $\overline{S}$ są podrzewami stworzonymi przez usunięcie z drzewa T^* krawędzi (i,j) (poprzez wykonanie cięcia na drzewie T^*).

  W jedną stronę to wiadomo, gdyby tak nie było to istniałaby taka krawędź (k,l), która ma mniejszy koszt niż (i,j), więc po wyrzuceniu krawędzi (i,j), stworzeniu dwóch poddrzew S i $\overline{S}$ oraz po ponownym połączeniu ich w jedno drzewo za pomocą krawędzi (k,l) otrzymalibyśmy drzewo o mniejszym koszcie niż pierwotne T^*, co przeczy optymalności T^*.

  Żeby pokazać w drugą stronę (że jeżeli drzewo T_0 spełnia warunek optymalnego cięcia to jest MST) załóżmy, że spełnia warunek optymalności i nie jest MST. Niech T^* będzie MST i $T_0 \neq T^*$ (T_0 różni się od T_* co najmniej jedną krawędzią). Z T_0 usuńmy krawędź (i,j), której nie ma w T^*, tworząc jednocześnie podział T_0 na dwa poddrzewa S i $\overline{S}$. Dodając tą krawędź do drzewa T_* stworzymy w nim cykl (własność drzewa, że dodanie jednej krawędzi stworzy cykl), do którego należeć będzie krawędź (i,j), gdzie $i \in S$, $j \in \overline{S}$ i w szczególności krawędź (k,l), gdzie $k \in \overline{S}$ i $l \in S$ (kolejność nie ma znaczenia, jako że są to grafy nieskierowane, lecz podkreśla to fakt konieczności istnienia krawędzi, która pozwala z powrotem przejść do zbioru S i zamknąć cykl). T^* jest MST, więc $c_{kl} \leqslant c_{ij}$ (zamienienie krawędzi $(k,l) \in T^*$ na inną krawędź, także łączącą ze sobą poddrzewa S i $\overline{S}$ nie polepszy nam kosztu). Z założenia natomiast T_0 spełnia warunek optymalnego cięcia, więc $c_{ij} \leqslant c_{kl}$. Z dwóch powyższych nierówności dostajemy $c_{ij} = c_{kl}$. Stąd mamy wniosek, że zamiana krawędzi $(i,j) \in T_0$ ($(i,j) \notin T^*$) na krawędź $(k,l) \in T^*$ nie zmieni kosztu całego drzewa. Nazwijmy tak powstałe drzewo T_1. W oczywisty sposób drzewo T_1 ma więcej krawędzi wspólnych z drzewem T^* niż drzewo T_0 ($|T_0 \setminus T^*| - 1 = |T_1 \setminus T^*|$) przy jednoczesnym zachowaniu $C_{T_0} = C_{T_1}$. Powtarzając ten proces dojdziemy do tego, że dla pewnego $k > 0$ zachodzić będzie $|T_k \setminus T^*| = 0$ przy zachowaniu $C_{T_0} = C_{T_1} = \ldots C_{T_k} = C_{T^*}$, co pokazuje, drzewo T_0 jest także MST (choć niekoniecznie tym samym).

- T^* jest MST wtedy i tylko wtedy, gdy dla każdej krawędzi (k,l) nie będącej w tym drzewie, dla każdej krawędzi (i,j) znajdującej się na ścieżce z k do l zachodzi $c_{ij} \leqslant c_{kl}$.

W jedną stronę jest znowu prosto. Zakładamy, że T^* jest MST i istnieje krawędź nienależąca do T^* (k, l) taka, że $c_{ij} > c_{kl}$. Gdy dołączymy krawędź (k, l) do drzewa, powstanie wtedy cykl (no bo wszystkie krawędzie, które są na ścieżce z k do l wraz z tą krawędzią tworzą po prostu cykl), w którym nowo dodana krawędź będzie miała mniejszy koszt od przynajmniej jednej z pozostałych krawędzi z tego cyklu, więc pozbywając się tej krawędzi otrzymujemy drzewo o mniejszym koszcie, mamy sprzeczność.

W drugą stronę też jest prosto. Bierzemy drzewo T^*, które jest MST, tworzymy dwa poddrzewa S i $\overline{S}$ poprzez usunięcie z drzewa T^* krawędzi (i, j). Weźmy teraz dowolną ścieżkę z k do l taką, że $k \in S$ oraz $l \in \overline{S}$. Jasnym jest, że skoro usuwając krawędź z (i, j) drzewa T^* rozdzieliliśmy drzewo na dwa zbiory, gdzie początek ścieżki jest w jednym a jej koniec w drugim, to ta krawędź w tym drzewie na pewno należy do ścieżki z k do l. Stąd z założenia mamy, że dowolna krawędź (k, l), która nie należy do drzewa ma większy koszt od c_{ij} i tym samym krawędź (i, j) ma najmniejszy koszt spośród wszystkich krawędzi łączących poddrzewo S z $\overline{S}$, co jest warunkiem optymalnego cięcia, zaś drzewo je spełniające jest MST.

Bibliografia

Biblioteka: Take Me Home

```
/
├── Presentations
├── References
├── Scripts
├── Thesis
├── TMH_Examples
├── TMH_Library
├── TMH_GraphHelper
├── clear.bash
└── README.md
/
├── clear.bash ...
└── Scripts
    ├── compileProject.bash ... Jak omawialiśmy
    │                            wcześniej --- służy do
    │                            kompilacji całego projektu.
    ├── generatePlot.bash ... Generuje wykresy na
    │                         podstawie danych,
    │                         wygenerowanych przez
    │                         przykładowy program
    │                         TMH_Examples.
    └── makeGraphViz.bash ... Generuje wizualizację
                              grafu zdefiniowanego
                              w pliku zgodnym z opisanym
                              standardem.
```

```
/
├── Scripts
│   └── randDistance.pl ... Pozwala na szybką
│                           modyfikację wszystkich
│                           kosztów ścieżek w pliku,
│                           definiującym graf.
└── TMH_GraphHelper... Pomocniczy projekt do
                       generowania danych
                       wejściowych (napisany
                       w języku Java).
```

/Scripts/compileProject.bash

CRTL + ALT + T

R_USATest_DKA

(ang. *Directed acyclic graph*)

```
pcname@user:/Scripts$ bash compileProject.bash
```

Rysunek A.1: Wygenerowany wykres dwóch funkcji dla przykładowego wywołania skryptu GENERATEPLOT.BASH.

```
digraph G {
        rankdir=LR;
        node [shape = circle];
        1 -> 2 [ label = "2"];
        1 -> 3 [ label = "6"];
        2 -> 3 [ label = "3"];
        2 -> 4 [ label = "4"];
        2 -> 5 [ label = "5"];
        3 -> 5 [ label = "1"];
        5 -> 4 [ label = "2"];
}
```

(a) (b)

Rysunek A.2: Wygenerowana ilustracja grafu przez skrypt MAKEGRAPHVIZ.BASH. **(a)** Kod pośredni pomiędzy danymi wejściowymi w standardowym formacie DIMACS IMPLEMENTATION CHALLENGE a finalnym rysunkiem. Kod pośredni jest zapisany w języku DOT. **(b)** Rysunek grafu wygenerowany przez program GRAPHVIZ.

http://algs4.cs.princeton.edu/home/ na licencji GNU GPLv3

$$\begin{aligned} f(x) + g(x) &= \cos(x) + \sin(x) + h(x) \\ g(x) &= \sin(x) \\ f(x) &= \cos(x) + h(x) \\ s(x) &= \arcsin(x) \end{aligned}$$