

Komplexitet

Teoretisk bas för frågorna:

- Är en viss metod bra eller dålig?
- Är ett visst problem lätt eller svårt?

Teori och praktik inte alltid överens, men i stort sett. . .

Algoritmkomplexitet:

Hur många operationer krävs, som funktion av indatas storlek, i värsta fall?
(Problemstorlek $\rightarrow \infty$, alla möjliga koefficienter.)

Definition

En **polynomisk algoritm** har polynomisk tidkomplexitet som funktion av indatastorleken. (Bra, dvs. snabb)

En **exponentiell algoritm** har exponentiell tidkomplexitet som funktion av indatastorleken. (Dålig, dvs. långsam)

En **pseudopolynomisk algoritm** har polynomisk tidkomplexitet som funktion av indatastorleken och en (största) *konstant* i indata.

Problemkomplexitet

Vilken tidskomplexitet har den **bästa kända metoden** för problemet?

Optimeringsversionen: Finn optimala (tillåtna) lösningen.

Evalueringversionen: Finn kostnaden för den optimala lösningen.

Vittnesversionen: Finn en tillåten lösning som har $c^T x \leq L$.

Igenkänningsversionen: Finns det en tillåten lösning som har $c^T x \leq L$?
(Svar: Ja eller nej.)

Algoritmkomplexitet

Exempel 1: Dijkstras metod i en graf med n noder:

Varje nod märks permanent *en* gång, alltså behövs n permanenta märkningar.

För varje permanent märkning, behöver man löpa igenom alla n noderna (2 gånger).

Komplexiteten blir $O(n^2)$, och algoritmen är polynomisk.

Exempel 2: Genomlöpa alla hörn i en hyperkub i n dimensioner:
 $O(2^n)$. Exponentiell algoritm.

Exempel 3: Edmonds-Karps metod för maxflödesproblemet:

I varje iteration: Dijkstras metod: $O(n^2)$.

Hur många iterationer?

Minst en enhet till i varje iteration. Alltså högst f^* iterationer.

Uppskattning av f^* : $\bar{f} = \sum_{(i,j) \in B} u_{ij}$. Komplexitet: $O(\bar{f} n^2)$.

Pseudopolynomisk metod.

Problemkomplexitet

Sats

Om igenkänningsversionen inte är polynomiskt lösbar, så är inte vittnes-, evaluering- eller optimeringsversionen det.

Sats

Om igenkänningsversionen är polynomiskt lösbar, så är även vittnesversionen det.

Sats

Om igenkänningsversionen är polynomiskt lösbar, och $0 \leq c^T x^* \leq 2^{p(n)}$, där $p(n)$ är ett polynom, så är även evaluering- och optimeringsversionen polynomiskt lösbara.

(Använd intervallhalvering på $L = c^T x^*$.)

Fokusera på igenkänningsversionen.

Definition

P är den klass av igenkänningsproblem som kan lösas (besvaras) av en polynomisk algoritm.

Definition

NP är den klass av igenkänningsproblem där ett ja-svar kan bekräftas på polynomisk tid.

Om svaret till igenkänningsproblemet är ja, så måste det finnas en lösning, som kan kontrolleras på polynomisk tid.

Exempel på problem i NP :

- Finns en handelsresandetur (Hamiltoncykel) med kostnaden mindre eller lika med L ?
- Finns en delmängd S av $\{1, 2, \dots, n\}$ sådan att $\sum_{j \in S} a_j = b$?

Komplexitet: Polynomiska relationer

Polynomisk reduktion från A till B : Lös problem A genom att anropa en subrutin, som löser B , ett polynomiskt antal gånger.

Sats

Om det finns en polynomisk algoritm för B , och A kan reduceras polynomiskt till B , så finns det en polynomisk algoritm för A .

Polynomisk transformation från A till B : Konstruera ett exempel av B ur ett exempel av A på polynomisk tid, så att exemplet av B ger ja-svar om och endast om exemplet av A ger ja-svar.

Definition

Ett igenkänningsproblem $A \in NP$ är **NP -fullständigt** om alla andra problem i NP har en polynomisk transformation till A .

Sats

$P \subseteq NP$.

Är $P = NP$? Njae.

Många har försökt bevisa det, men ingen har lyckats.

Bl.a. V. F. Romanov, som ger en polynomisk algoritm för 3-SAT-problemet, vilket skulle betyda att $P = NP$.

Artikeln innehåller dock flera oklarheter.

Ett *bevis* för att $P \neq NP$ vore mindre överraskande, eftersom alla tror det.

Vinay Deolalikar säger sig, i augusti 2010, ha bevisat detta.

Man har dock hittat några möjligtvis fatala fel i artikeln.

Komplexitet: NP -fullständighet

För att bevisa att ett problem, A , är NP -fullständigt:

1. Visa att problemet A tillhör NP .
2. Visa att alla andra problem i NP kan transformeras polynomiskt till A .

(Men hur gör man det?)

2: Visa att ett NP -fullständigt problem kan transformeras polynomiskt till A .

Obs: Om man skulle hitta en polynomisk algoritm för **ett** enda NP -fullständigt problem, så har man bevisat att $P = NP$.

Slutsats

Det finns ingen polynomisk algoritm för något NP -fullständigt problem (om inte $P = NP$).

Definition

Ett problem A kallas *NP-svårt*, om något *NP*-fullständigt problem kan transformeras polynomiskt till A , men vi ej kan visa att $A \in NP$.

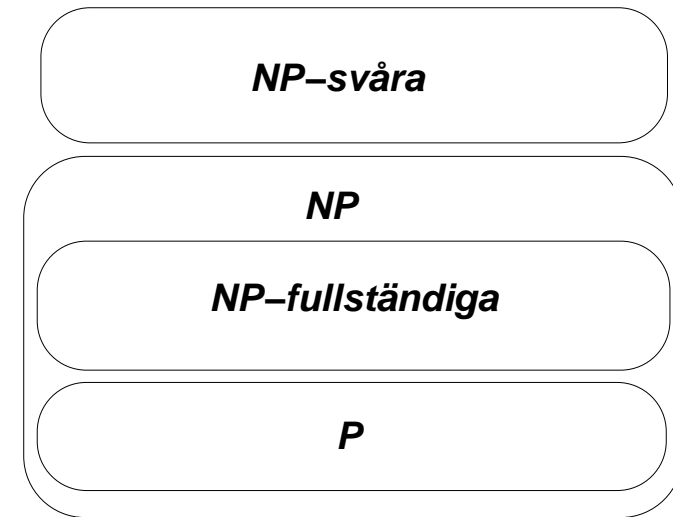
NP-svår minst lika svår som *NP*-fullständig.

Om $0 \leq c^T x^* \leq 2^{p(n)}$ inte gäller, kan man ej visa att optimeringsversionen tillhör *NP*.

Ofta: Igenkänningsversionen *NP*-fullständig, optimeringsversionen *NP*-svår.

Slutsats

Det finns ingen polynomisk algoritm för något *NP*-svårt problem (om inte $P = NP$).

Vilka problem är *NP*-fullständiga/*NP*-svåra?

Satisfieringsproblemet: (SAT) Finns en tilldelning av värden till n logiska variabler, så att en boolsk formel med m satser blir *sann*?

Bevisat *NP*-fullständigt av Cook 1971.

Binära heltalsproblemet: (BHP) Finns en binär n -vektor, x , som uppfyller $Ax \geq b$?

SAT kan transformeras polynomiskt till BHP.

Allmänna heltalsproblemet: (HP) Finns en ickenegativ, heltalig n -vektor, x , som uppfyller $Ax \geq b$, $x \leq u$?

Transformation från BHP: Ersätt heltalsvariabler med binära, $x_j = \sum_i 2^i y_{ij}$.

Flera *NP*-fullständiga/*NP*-svåra problem

- Existerar en (o)riktad Hamiltonväg/cykel??
- Handelsresandeproblemet (olika varianter).?
- Kappsäcksproblemet.?
- Valensbegränsade billigaste uppspännande träd-problem (VMST): Finns det ett uppspännande träd med $c^T x \leq L$ där ingen nod har valens större än K ??
- Kan noderna färgas med högst K olika färger, så att inga två närliggande noder har samma färg??
- Finns det ett snitt med *minst* kapacitet K i ett nätverk med kapaciteter??
- Innehåller en graf en "klick" (fullständig subgraf) med minst K noder??
- Billigaste-enkla-vägproblemet: Finns en enkel väg från start- till slutnod med $c^T x \leq L$?
- ... samt ca 3000 till.

Heuristiker

“Smarta” metoder som ger “skapliga” lösningar.

Ger ej garanterat optimum.

Men kan göra det om man har tur. (Fast det vet man inte.)

Snabbare än optimerande metoder.

Enda möjligheten för riktigt stora svåra problem.

Ofta: *NP*-svårt problem, polynomisk heuristik.

Heuristiker

$\max z = 3x_1 + 4x_2 + 5x_3$ då $2x_1 + 3x_2 + 4x_3 \leq 6$, $x_j \in \{0, 1\}$ för alla j

Kvoter: $x_1 : 3/2$, $x_2 : 4/3$, $x_3 : 5/4$

ger lösningen $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, och $z = 7$.

Optimum: $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $z = 8$.

$\max z = 3x_1 + 100x_2$ då $2x_1 + 99x_2 \leq 100$, $x_j \in \{0, 1\}$ för alla j

Kvoter: $x_1 : 3/2$, $x_2 : 100/99$ ger lösningen $x_1 = 1$, $x_2 = 0$ och $z = 3$.

Optimum: $x_1 = 0$, $x_2 = 1$, $z = 100$.

Heuristiker: Giriga metoder

Giriga metoder är snabba, men löser bara vissa enkla problem (matroider) optimalt.

(Kruskal löser MST.)

De kan dock användas som heuristik för många problem.

Kappsäcksproblemet:

Sortera kvoterna c_j/a_j .

Fyll på med bästa först.

Löser LP-relaxationen exakt.

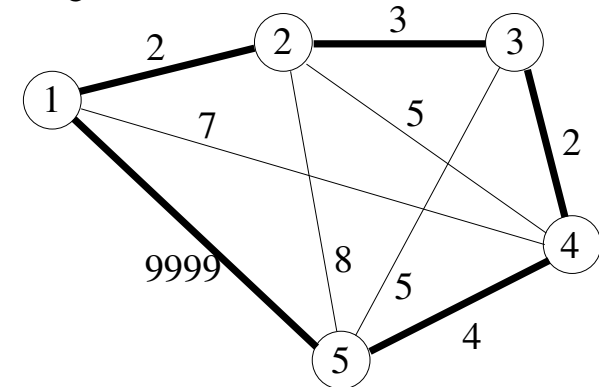
Samma metod på heltalsproblemet (dvs avrunda neråt), kan ge upp till 100% relativt fel.

Heuristiker för handelsresandeproblemet

Närmaste granne-algoritmen:

Partiell handelsresandetur, utöka med en nod i taget.

Lägg till närmaste granne till sista noden i turen.



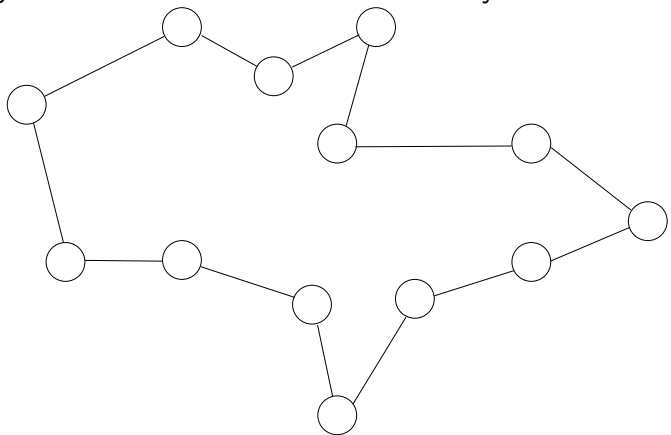
Kan falla helt för icke fullständig graf.

Närmaste sammanslagnings-algoritmen:

Flera partiella turer som successivt slås samman.

Slå ihop de turer som ligger närmast varandra: $\min_{k,l} (\min_{i \in T_k, j \in T_l} c_{ij})$,

och välj bägar att ta bort så att kostnaden för nya turen blir minimal.



Specialutvecklade (ad hoc) algoritmer

Utnyttja problemets struktur.

Trädalgoritmen för handelsresandeproblemet Δ TSP:

1. Finn billigaste uppspännande träd.
2. Konstruera en graf med parallella bågar genom att dubblera varje båge i trädet.
3. Finn en Eulercykel.
4. Transformera Eulercykeln till en Hamiltoncykel (genom att ta genvägar vid återbesök).

Komplexitet: $O(n^2)$ (Prim, steg 1).

Målfunktionsvärdet: $z_{MST} \leq z^*$, $z_E = 2z_{MST}$, $z_H \leq z_E \Rightarrow z_H \leq 2z^*$.

Sämrre för TSP utan Δ .

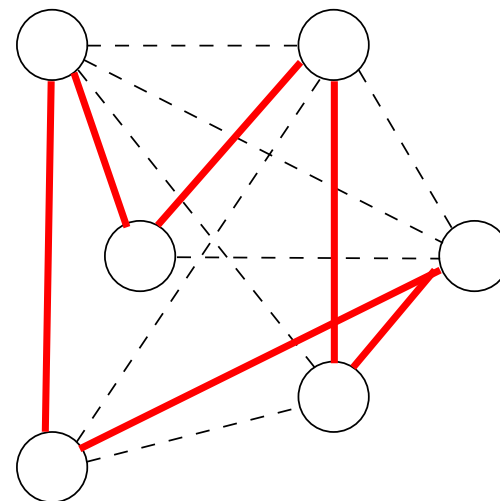
Insättnings-algoritmer

(en tur som utökas med en nod i taget):

- Närmaste additions-algoritmen, $O(n^2)$.
(Lägg till en viss ny nod intill en given nod i turen.)
- Närmaste insättnings-algoritmen, $O(n^2)$.
(Lägg till en viss ny nod på bästa ställe i turen.)
- Billigaste additions-algoritmen, $O(n^2 \log_2(n))$.
(Lägg till den nya nod som ger lägsta kostnad.)

$$z_H \leq 2z^*.$$

Trädalgoritmen för handelsresandeproblemet

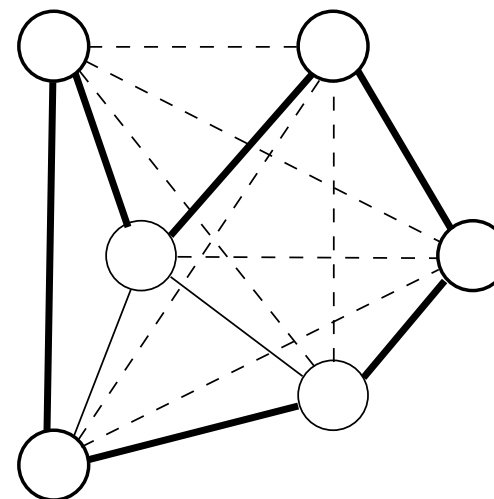


Christofides algoritm: Addera bara bågar till noder med udda valens.

1. Finn billigaste uppspännande träd T .
2. Finn alla noder som har udda valens i T och finn den billigaste perfekta matchningen M i den fullständiga grafen med dessa noder.
3. Finn en Eulercykel i $T + M$ och den handelsresandetur den innehåller.

Steg 2: Matchningsproblem, $O(n^4)$.

$$z_H \leq 3/2z^* \text{ för } \Delta\text{TSP.}$$



Heuristiker för hinkpackningsproblemet

Första plats ("First fit"): Placera nästa objekt i den hink längst till vänster där den får plats.

Sista plats ("Last fit"): Placera nästa objekt i den hink längst till höger där den får plats.

Nästa plats ("Next fit"): Placera nästa objekt i hinken längst till höger (även om en ny krävs).

Bästa plats ("Best fit"): Placera nästa objekt i den fullaste hink där den får plats.

Värsta plats ("Worst fit"): Placera nästa objekt i den tomaste använda hinken.

Näst värsta plats ("Almost worst fit"): Placera nästa objekt i den näst tomaste hinken.

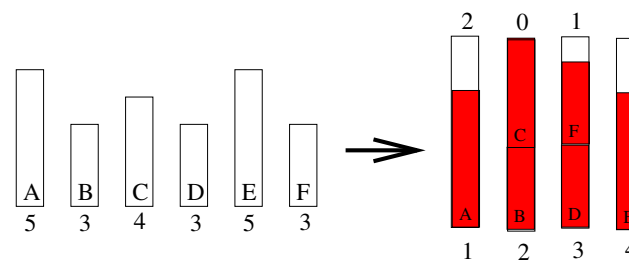
Förbättring: Sortera först objekten, så att de tyngsta kommer först.

Första/bästa plats: $z_H \leq 17/10 z^* + 2$.

Sorterad första/bästa plats: $z_H \leq 11/9 z^* + 4$.

Heuristiker för hinkpackningsproblemet

Första plats



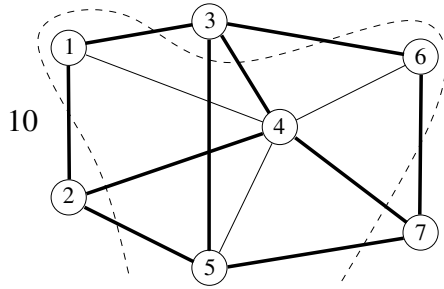
Undre gräns. $\lceil \frac{23}{7} \rceil = 4$. Optimum.

Heuristiker för maxsnittsproblemet

Maxsnittsproblemet: Finn ett snitt genom en oriktad graf så att antalet bågar som passerar snittet är maximalt.

1. Starta med valfritt snitt.
2. Flytta över en nod till andra sidan snittet om detta förbättrar lösningen (dvs. ökar antalet bågar över snittet).
3. Upprepa tills ingen flyttning av en enskilda nod ger någon förbättring.

$$z^* \leq 2z_H.$$



Heuristiker för andra grafproblem

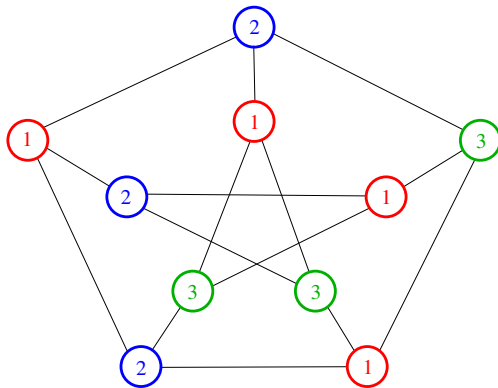
Ofta ganska självklara:

Ta noderna/bågarna i nummerordning. Bygg upp lösningen girigt.

- Nodfärgning med min antal färger: Färga nästa nod med lägsta möjliga färg.
- Bågfärgning med min antal färger: Färga nästa båge med lägsta möjliga färg.

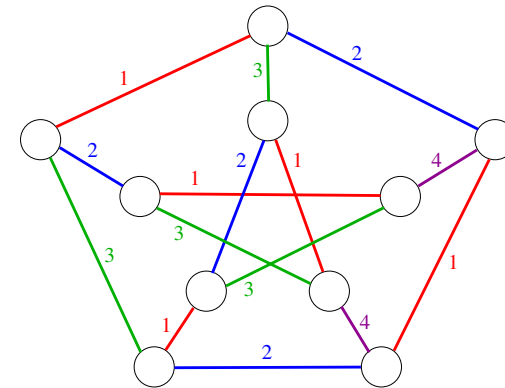
Heuristik för minimal nodfärgning

Färga nästa nod med lägsta möjliga färg.



Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



Lokalsökning

Hoppa från en punkt till en bättre granne. Upprepa.

Definiera **omgivning** N (närliggande punkter, grannar), mha avståndsmått samt avståndsgräns.

Exempel:

- Euklidiskt närmaste punkterna.
- Utbyte av element, t ex bågar, noder.
- Byte av 0 mot 1 och vv.

Genomsök omgivningen efter en bättre punkt.

0. Finn startpunkt $x^{(k)} \in X$. Sätt $k = 0$.

1. Genomsök $N(x^{(k)})$ efter en punkt \bar{x} med $f(\bar{x}) < f(x^{(k)})$.

2. Om ingen finns: Stopp, $x^{(k)}$ är lokalt optimum.

3. Annars sätt $x^{(k+1)} = \bar{x}$. Sätt $k = k + 1$ och gå till 1.

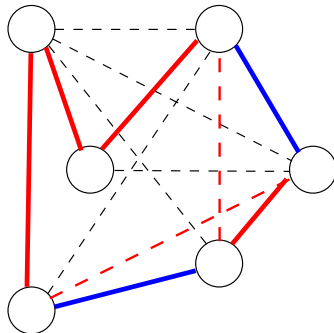
Lokalsökning

Omgivning för handelsresandeproblemet:

2-byte: Ta bort två bågar och sätt dit två nya.

3-byte: Ta bort tre bågar och sätt dit tre nya.

Def: Lokalt optimum map viss omgivning. (T ex 2-bytes-optimum.)



Lokalsökning

Varianter för bivillkor:

- Endast tillåtna punkter.
- Även otillåtna punkter. Lägg till strafffunktion till målfunktionen.

Klättringsstrategier:

- Finn bästa punkten i hela omgivningen.
- Finn bästa punkten i delmängd av omgivningen.
- Finn första bättre punkten.
- Finn första bättre punkten i delmängd av omgivningen.

Tabusökning

Utvidgning av lokalsökning.

För att ej fastna i lokalt optimum: Tillåt **temporär försämring**.

Förbjud förflyttningar till nyss besökt grannar. Spara tidigare förflyttningar (besökta punkter) i en **tabulista**, av viss längd (kö).

1. Välj startpunkt, $x^{(0)} \in X$. Sätt $k = 0$. Börja med en tom tabulista.
2. Välj en sökmängd $A(x^{(k)}) \subseteq N(x^{(k)})$ av punkter **som ej är tabu**.
3. Finn nästa punkt, $x^{(k+1)}$, mha $\min_{x \in A(x^{(k)})} f(x)$.
4. Uppdatera tabulistan: Tillför $x^{(k)}$. Om listan är full, ta bort äldsta punkten.
5. Stoppa om $k > K$. Annars sätt $k = k + 1$ och gå till 2.

Lokalsökning

$$\begin{aligned} \max z &= 2x_1 + 3x_2 \\ \text{då} \quad 4x_1 + x_2 &\leq 13 \\ 3x_1 + 5x_2 &\leq 16 \\ x_1, x_2 &\geq 0, \text{ heltal} \end{aligned}$$

Omgivning (grannar): ändra en variabel med en enhet.

Starta i $x^{(0)} = (0, 0)$ med målfunktionsvärde $z(x^{(0)}) = 0$.

Iteration 1. Omgivning:

- $x = (1, 0)$, $z = 2$, tillåten
- $x = (0, 1)$, $z = 3$, tillåten
- $x = (-1, 0)$, $z = -2$, ej tillåten
- $x = (0, -1)$, $z = -3$, ej tillåten

Välj $x^{(1)} = (0, 1)$, med $z(x^{(1)}) = 3$, uppdatera bästa funna lösning.

Lokalsökning

$$\begin{aligned} \max z &= 2x_1 + 3x_2 \\ \text{då} \quad 4x_1 + x_2 &\leq 13 \\ 3x_1 + 5x_2 &\leq 16 \\ x_1, x_2 &\geq 0, \text{ heltal} \end{aligned}$$

Fortsätt i $x^{(1)} = (0, 1)$ med målfunktionsvärde $z(x^{(1)}) = 3$.

Iteration 2. Omgivning: Omgivning:

- $x = (1, 1)$, $z = 5$, tillåten
- $x = (0, 2)$, $z = 6$, tillåten
- $x = (-1, 1)$, $z = -1$, ej tillåten
- $x = (0, 0)$, $z = 0$, tillåten

Välj $x^{(2)} = (0, 2)$, med $z(x^{(2)}) = 6$, uppdatera bästa funna lösning.

Lokalsökning

$$\begin{aligned} \max z &= 2x_1 + 3x_2 \\ \text{då} \quad 4x_1 + x_2 &\leq 13 \\ 3x_1 + 5x_2 &\leq 16 \\ x_1, x_2 &\geq 0, \text{ heltal} \end{aligned}$$

Fortsätt i $x^{(2)} = (0, 2)$ med målfunktionsvärde $z(x^{(2)}) = 6$.

Iteration 3. Omgivning:

- $x = (1, 2)$, $z = 8$, tillåten
- $x = (0, 3)$, $z = 9$, tillåten
- $x = (-1, 2)$, $z = 4$, ej tillåten
- $x = (0, 1)$, $z = 3$, tillåten

Välj $x^{(3)} = (0, 3)$, med $z(x^{(3)}) = 9$, uppdatera bästa funna lösning.

Lokalsökning

$$\begin{aligned} \max z &= 2x_1 + 3x_2 \\ \text{då} \quad 4x_1 + x_2 &\leq 13 \\ 3x_1 + 5x_2 &\leq 16 \\ x_1, x_2 &\geq 0, \text{ heltal} \end{aligned}$$

Fortsätt i $x^{(3)} = (0, 3)$ med målfunktionsvärde $z(x^{(3)}) = 9$.

Iteration 4. Omgivning:

- $x = (1, 3)$, $z = 11$, ej tillåten
- $x = (0, 4)$, $z = 12$, ej tillåten
- $x = (-1, 3)$, $z = 7$, ej tillåten
- $x = (0, 2)$, $z = 6$, tillåten

Finns inte någon tillåten punkt som förbättrar målfunktionsvärdet - STANNA.

Bästa lösning med lokalsökning är $x^{(3)} = (0, 3)$ med målfunktionsvärde $z(x^{(3)}) = 9$.

Tabusökning

$$\begin{aligned} \max z &= 2x_1 + 3x_2 \\ \text{då} \quad 4x_1 + x_2 &\leq 13 \\ 3x_1 + 5x_2 &\leq 16 \\ x_1, x_2 &\geq 0, \text{ heltal} \end{aligned}$$

Tabulista: Innehåller de två senaste punkterna.

Avbrottskriterium: 4 iterationer

Starta i $x^{(0)} = (0, 3)$ från lokalsökningen, med målfunktionsvärde $z(x^{(0)}) = 9$.

Tabulista: tom

Iteration 1. Omgivning:

- $x = (1, 3)$, $z = 11$, ej tillåten
- $x = (0, 4)$, $z = 12$, ej tillåten
- $x = (-1, 3)$, $z = 7$, ej tillåten
- $x = (0, 2)$, $z = 6$, tillåten

Välj $x^{(1)} = (0, 2)$, med $z(x^{(1)}) = 6$, bästa funna lösning är $z(x^{(0)}) = 9$.

Tabusökning

$$\begin{aligned} \max z &= 2x_1 + 3x_2 \\ \text{då} \quad 4x_1 + x_2 &\leq 13 \\ 3x_1 + 5x_2 &\leq 16 \\ x_1, x_2 &\geq 0, \text{ heltal} \end{aligned}$$

Fortsätt i $x^{(1)} = (0, 2)$ med målfunktionsvärde $z(x^{(1)}) = 6$.

Tabulista: $\{(0, 3)\}$

Iteration 2. Omgivning:

- $x = (1, 2)$, $z = 8$, tillåten
- $x = (0, 3)$, $z = 9$, tillåten, tabu
- $x = (-1, 2)$, $z = 4$, ej tillåten
- $x = (0, 1)$, $z = 3$, tillåten

Välj $x^{(2)} = (1, 2)$, med $z(x^{(2)}) = 8$, bästa funna lösning är fortfarande $z(x^{(0)}) = 9$.

Tabusökning

$$\begin{aligned} \max z &= 2x_1 + 3x_2 \\ \text{då} \quad 4x_1 + x_2 &\leq 13 \\ 3x_1 + 5x_2 &\leq 16 \\ x_1, x_2 &\geq 0, \text{ heltal} \end{aligned}$$

Fortsätt i $x^{(2)} = (1, 2)$ med målfunktionsvärde $z(x^{(2)}) = 8$.

Tabulista: $\{(0, 3), (0, 2)\}$

Iteration 3. Omgivning:

- $x = (2, 2)$, $z = 10$, tillåten
- $x = (0, 2)$, $z = 6$, tillåten, tabu
- $x = (1, 3)$, $z = 11$, ej tillåten
- $x = (1, 1)$, $z = 5$, tillåten

Välj $x^{(3)} = (2, 2)$, med $z(x^{(3)}) = 10$, uppdatera bästa funna lösning.

Tabusökning

$$\begin{aligned} \max z &= 2x_1 + 3x_2 \\ \text{då} \quad 4x_1 + x_2 &\leq 13 \\ 3x_1 + 5x_2 &\leq 16 \\ x_1, x_2 &\geq 0, \text{ heltal} \end{aligned}$$

Fortsätt i $x^{(3)} = (2, 2)$ med målfunktionsvärde $z(x^{(3)}) = 10$.

Tabulista: $\{(0, 2), (1, 2)\}$

Iteration 4. Omgivning:

- $x = (3, 2)$, $z = 12$, ej tillåten
- $x = (1, 2)$, $z = 8$, tillåten, tabu
- $x = (2, 3)$, $z = 13$, ej tillåten
- $x = (2, 1)$, $z = 7$, tillåten

Välj $x^{(4)} = (2, 1)$, med $z(x^{(4)}) = 7$, bästa funna lösning är fortfarande $z(x^{(3)}) = 10$.

Avbrottskriterium uppfyllt, vi har gjort 4 iterationer. Avsluta.

Relaxationsmetoder

Relaxera bivillkor, ger lättare problem.

Försök återskapa relaxerade bivillkor m.h.a. linjära straffkostnader och heuristiker. (Teori: Lagrangedualitet.)

Exempel: Lagrangeheuristik för TSP:

Relaxation: 1-träd (relaxerat nodvalens ≤ 2).

Nodstraff: Addera θ till kostnaden för alla bågar som ansluter till en nod med för hög valens.

Försök få en tillåten lösning genom att flytta bågar från noder med för hög valens till noder med för låg.

Försök fixera variabler m.h.a. övre och undre gränser.

Använd närmaste granne, två-byte eller annan heuristik då och då.

Starta gärna heuristiken med delar av subproblemlösningen.