Komplexitet

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. . .

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 $\to \infty$, alla möjliga koefficienter.)

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 $\to \infty$, alla möjliga koefficienter.)

Definition

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

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 $\to \infty$, 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)

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 $\to \infty$, 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.

Algoritmkomplexitet

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

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.

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:

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.

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:

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

Vilken tidskomplexitet har den bästa kända metoden för problemet?

Problemkomplexitet

Vilken tidskomplexitet har den bästa kända metoden för problemet?

Optimeringsversionen: Finn optimala (tillåtna) lösningen.

Problemkomplexitet

Vilken tidskomplexitet har den bästa kända metoden för problemet?

Optimeringsversionen: Finn optimala (tillåtna) lösningen.

Evalueringsversionen: Finn kostnaden för den optimala lösningen.

Problemkomplexitet

Vilken tidskomplexitet har den bästa kända metoden för problemet?

Optimeringsversionen: Finn optimala (tillåtna) lösningen.

Evalueringsversionen: Finn kostnaden för den optimala lösningen.

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

Problemkomplexitet

Vilken tidskomplexitet har den bästa kända metoden för problemet?

Optimeringsversionen: Finn optimala (tillåtna) lösningen.

Evalueringsversionen: Finn kostnaden för den optimala lösningen.

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

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

Problemkomplexitet

Sats

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

Problemkomplexitet

Sats

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

Sats

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

Problemkomplexitet

Sats

Om igenkänningsversionen inte är polynomiskt lösbar, så är inte vittnes-, evaluerings- 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 evaluerings- och optimeringsversionen polynomiskt lösbara.

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

Fokusera på igenkänningsversionen.

Komplexitet: *P* och *NP*

Definition

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

Komplexitet: *P* och *NP*

Definition

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

Definition

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

Komplexitet: *P* och *NP*

Definition

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

Definition

NP är den klass av igenkänningproblem 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.

Komplexitet: *P* och *NP*

Definition

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

Definition

NP är den klass av igenkänningproblem 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*:

Komplexitet: P och NP

Definition

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

Definition

NP är den klass av igenkänningproblem 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?

Komplexitet: P och NP

Definition

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

Definition

NP är den klass av igenkänningproblem 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 $\{1, 2, \ldots, n\}$ sådan att $\sum_{j \in S} a_j = b$?

Komplexitet: P och NP

Sats

$P \subseteq NP$.

Komplexitet: P och NP

Sats

$P \subseteq NP$.

Är $P = NP$?

Komplexitet: P och NP

Sats

$P \subseteq NP$.

Är $P = NP$? Njae.

Många har försökt bevisa det, men ingen har lyckats.

Komplexitet: P och NP

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.

Komplexitet: P och NP

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.

Komplexitet: P och NP

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: Polynomiska relationer

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

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.

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.

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.

Komplexitet: *NP*-fullständighet

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

Komplexitet: *NP*-fullständighet

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

1. Visa att problemet A tillhör *NP*.

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.

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?)

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.

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 $P = NP$.

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 $P = NP$).

Komplexitet: *NP*-svårt

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*.

Komplexitet: *NP*-svårt

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.

Komplexitet: *NP*-svårt

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*.

Komplexitet: *NP*-svårt

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.

Komplexitet: *NP*-svårt

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.

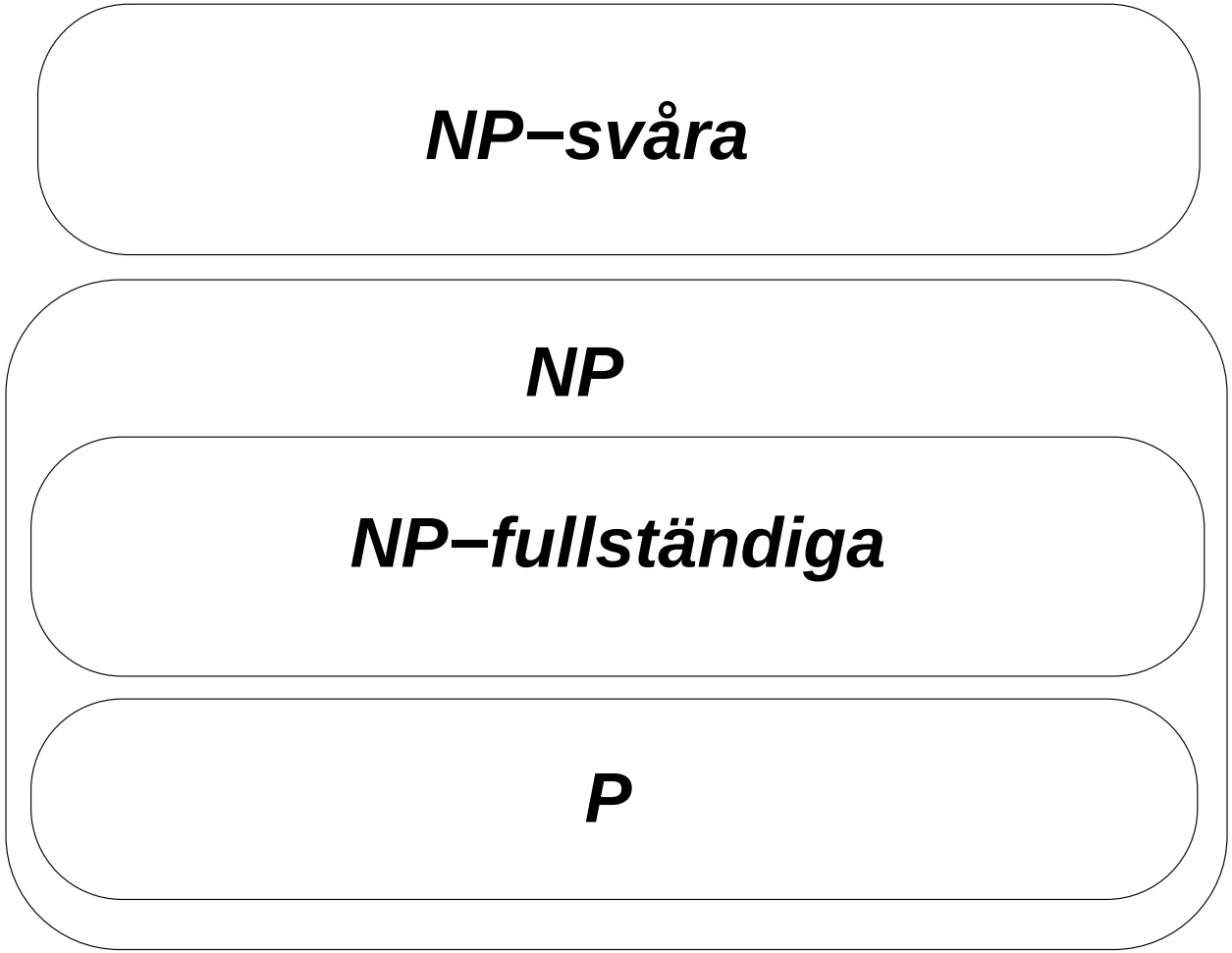
Om $0 \leq c^T x^* \leq 2^{p(n)}$ *inte* gäller, kan man ej visa att optimeringsversionen tillhör *NP*.

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

Slutsats

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

Komplexitet



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

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*?

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.

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 $Ax \geq b$?

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.

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 $Ax \geq b$, $x \leq u$?

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

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

- Existerar en (o)riktad Hamiltonväg/cykel??

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

- Existerar en (o)riktad Hamiltonväg/cykel??
- Handelsresandeproblemet (olika varianter).?

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

- Existerar en (o)riktad Hamiltonväg/cykel??
- Handelsresandeproblemet (olika varianter).?
- Kappsäcksproblemet.?

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??

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??

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??

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??

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 $c^T x \leq L$?

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.

Heuristiker

"Smarta" metoder som ger "skapliga" lösningar.

Ger ej garanterat optimum.

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

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.

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.

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: Giriga metoder

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

(Kruskal löser MST.)

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.

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.

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

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

Heuristiker

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

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

Heuristiker

$\max z = 3x_1 + 4x_2 + 5x_3 \quad \text{då} \quad 2x_1 + 3x_2 + 4x_3 \leq 6, \quad x_j \in \{0, 1\}$ 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 $z = 7$.

Heuristiker

$\max z = 3x_1 + 4x_2 + 5x_3 \quad$ då $\quad 2x_1 + 3x_2 + 4x_3 \leq 6, \quad x_j \in \{0, 1\}$ 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: $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $z = 8$.

Heuristiker

$\max z = 3x_1 + 4x_2 + 5x_3 \quad$ då $\quad 2x_1 + 3x_2 + 4x_3 \leq 6, \quad x_j \in \{0, 1\}$ 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: $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $z = 8$.

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

Heuristiker

$\max z = 3x_1 + 4x_2 + 5x_3 \quad$ då $\quad 2x_1 + 3x_2 + 4x_3 \leq 6, \quad x_j \in \{0, 1\}$ 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: $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $z = 8$.

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

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

Heuristiker

$\max z = 3x_1 + 4x_2 + 5x_3 \quad$ då $\quad 2x_1 + 3x_2 + 4x_3 \leq 6, \quad x_j \in \{0, 1\}$ 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: $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $z = 8$.

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

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

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

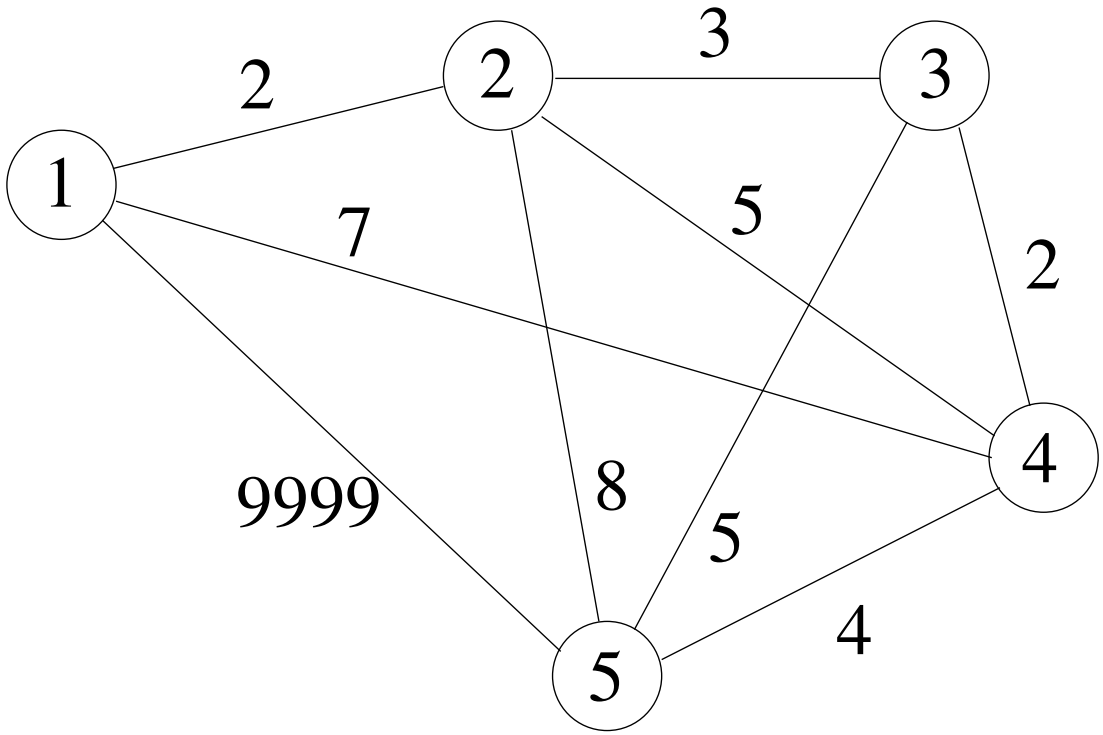
Heuristiker för handelsresandeproblemet

Närmaste granne-algoritmen:

Heuristiker för handelsresandeproblemet

Närmaste granne-algoritmen:

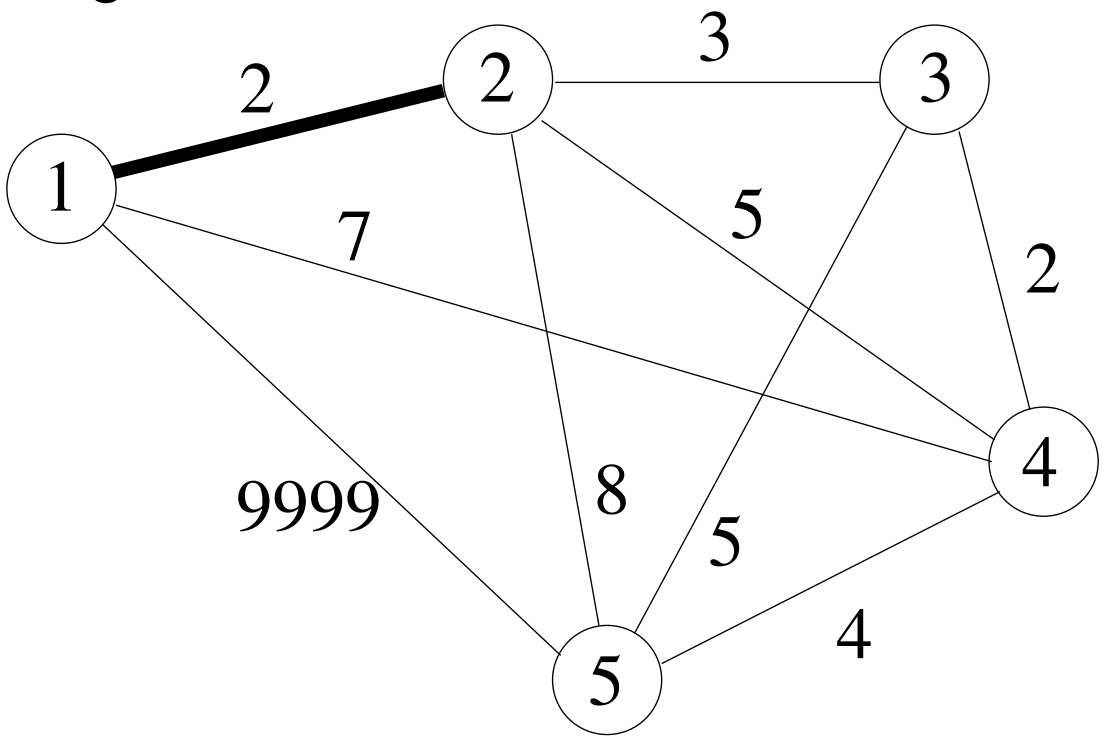
Partiell handelsresandetur, utöka med en nod i taget.
Lägg till närmaste grannen till sista noden i turen.



Heuristiker för handelsresandeproblemet

Närmaste granne-algoritmen:

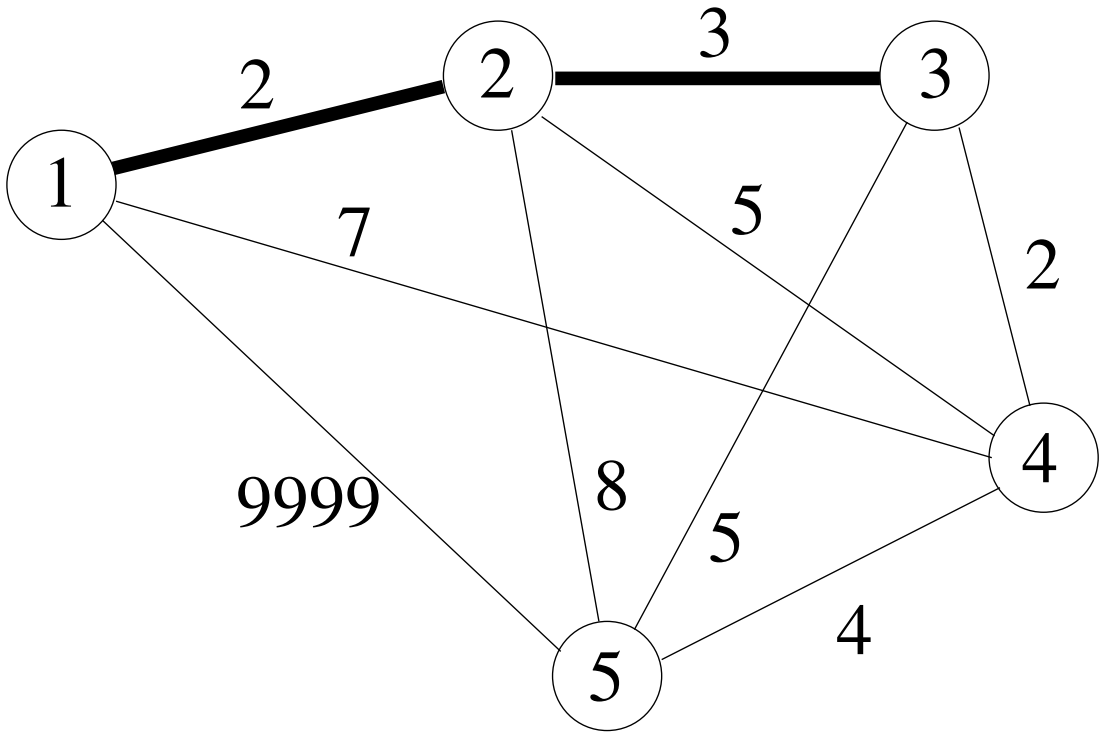
Partiell handelsresandetur, utöka med en nod i taget.
Lägg till närmaste grannen till sista noden i turen.



Heuristiker för handelsresandeproblemet

Närmaste granne-algoritmen:

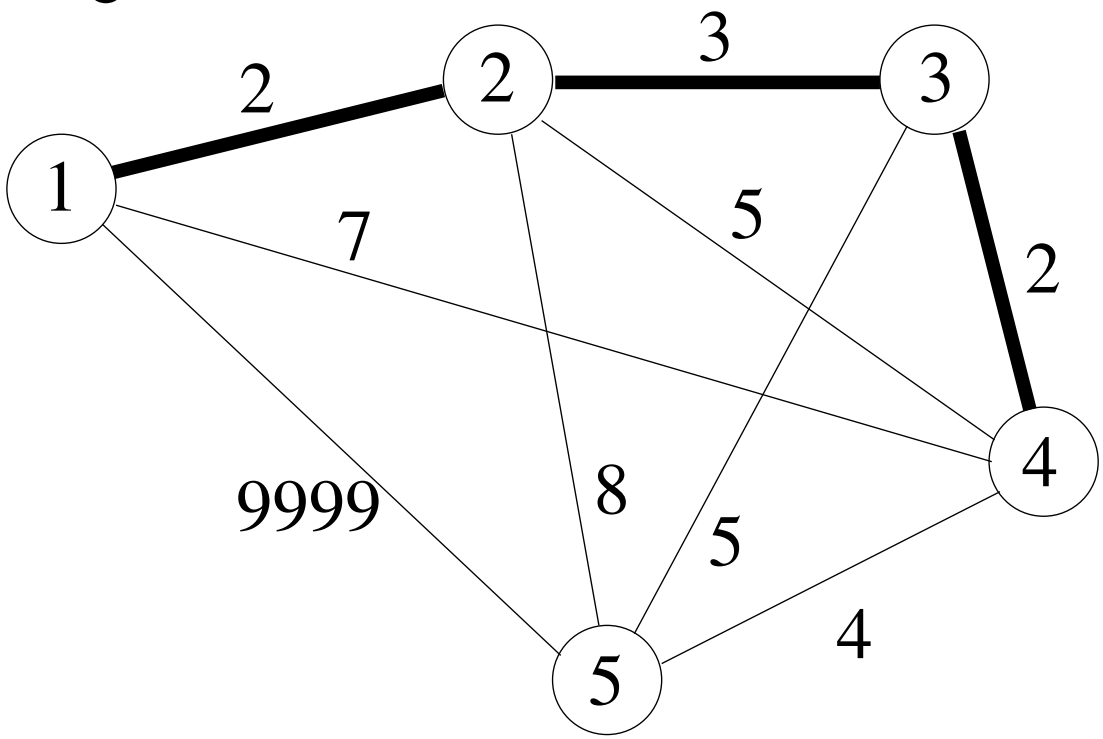
Partiell handelsresandetur, utöka med en nod i taget.
Lägg till närmaste grannen till sista noden i turen.



Heuristiker för handelsresandeproblemet

Närmaste granne-algoritmen:

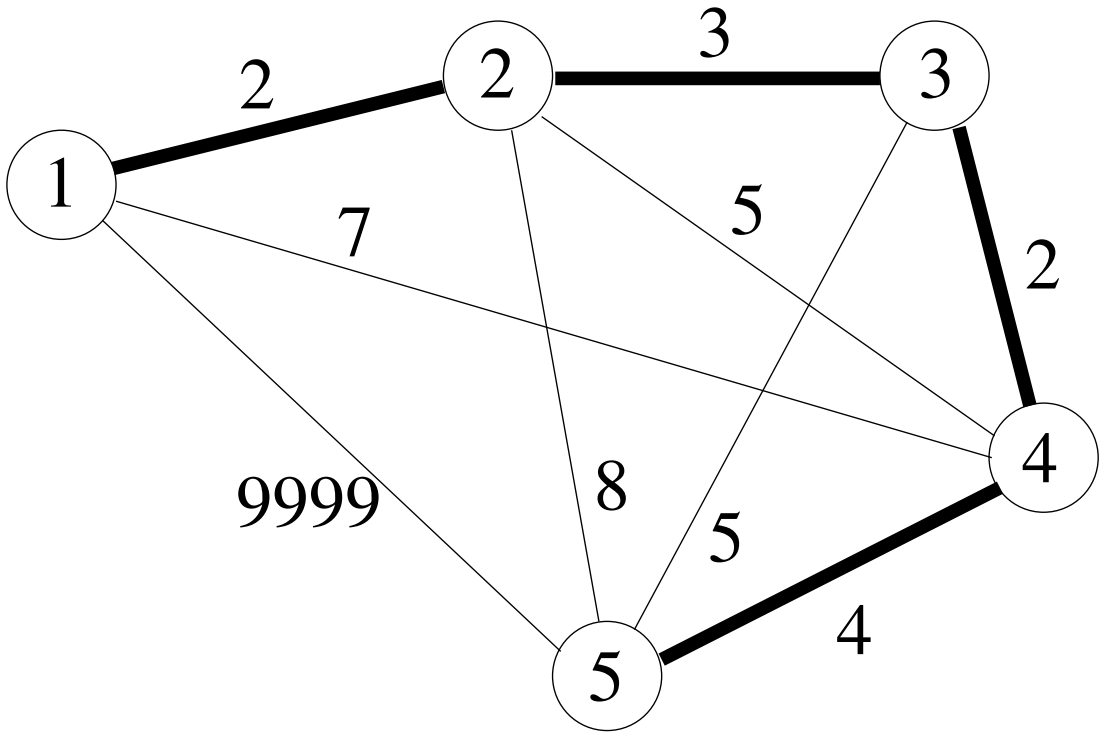
Partiell handelsresandetur, utöka med en nod i taget.
Lägg till närmaste grannen till sista noden i turen.



Heuristiker för handelsresandeproblemet

Närmaste granne-algoritmen:

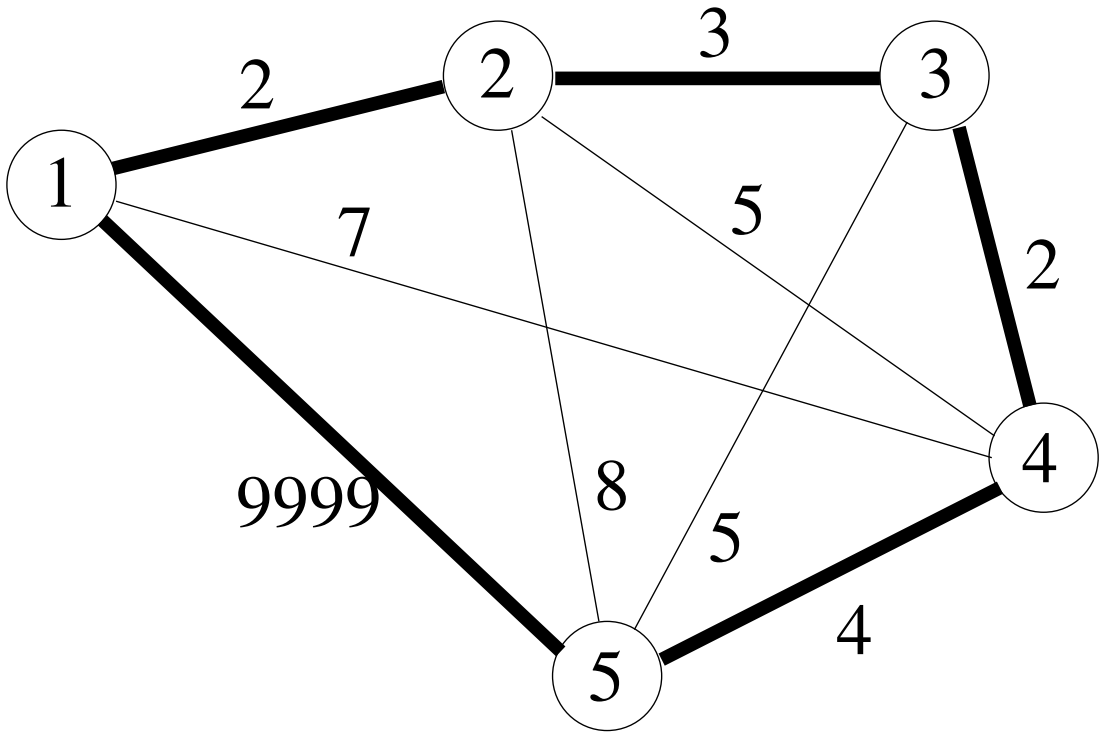
Partiell handelsresandetur, utöka med en nod i taget.
Lägg till närmaste grannen till sista noden i turen.



Heuristiker för handelsresandeproblemet

Närmaste granne-algoritmen:

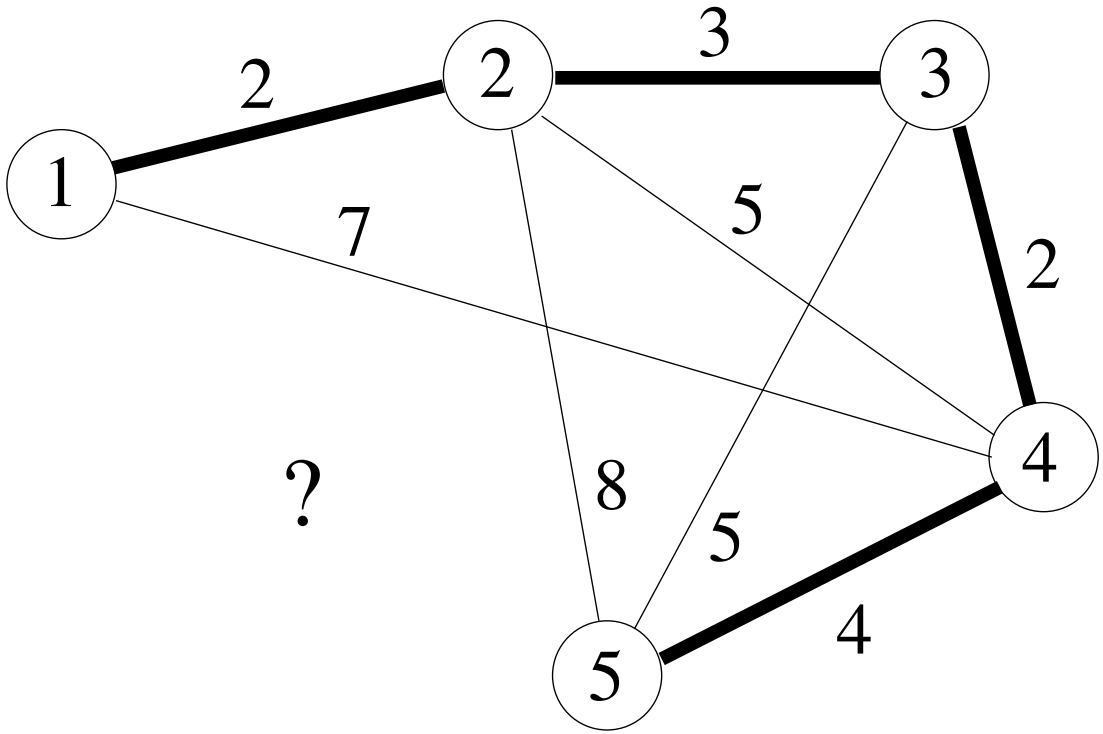
Partiell handelsresandetur, utöka med en nod i taget.
Lägg till närmaste grannen till sista noden i turen.



Heuristiker för handelsresandeproblemet

Närmaste granne-algoritmen:

Partiell handelsresandetur, utöka med en nod i taget.
Lägg till närmaste grannen till sista noden i turen.



Kan fallera helt för icke fullständig graf.

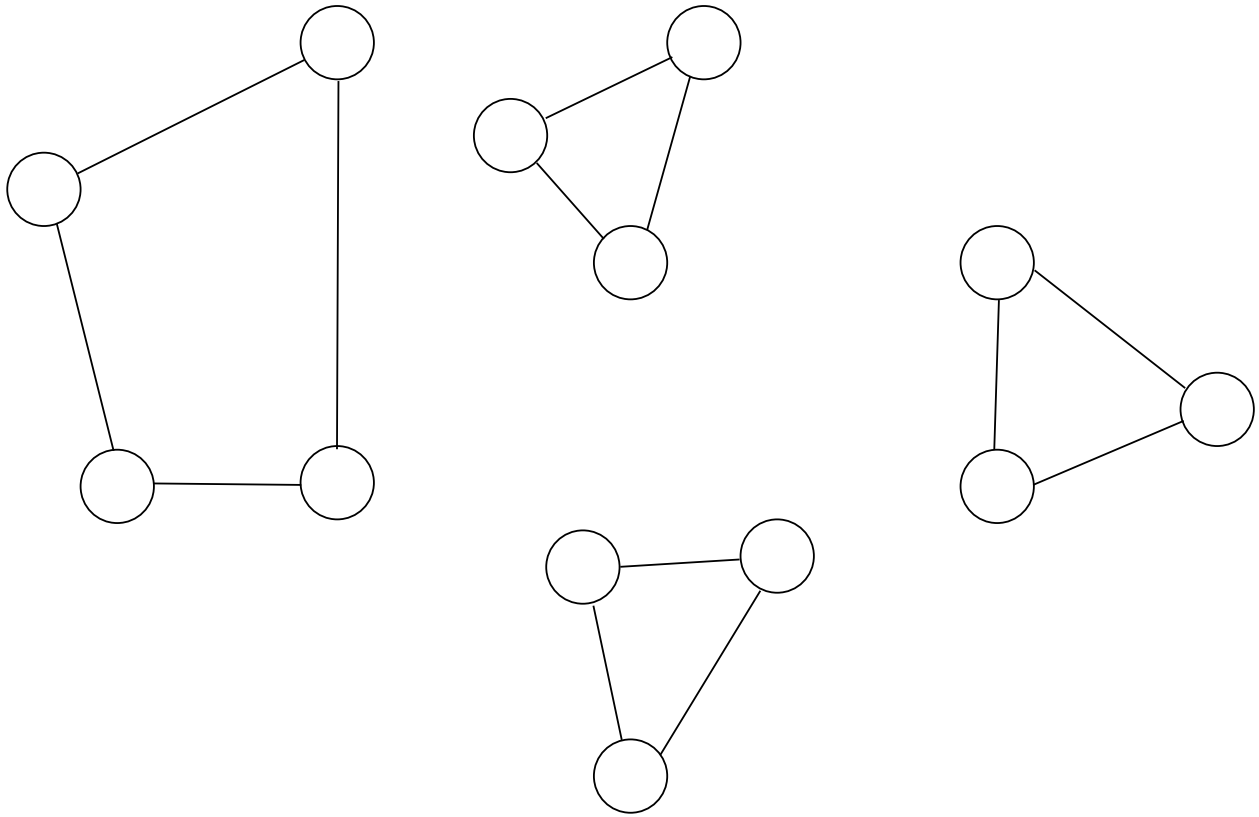
Heuristiker för handelsresandeproblemet

Närmaste sammanslagnings-algoritmen:

Heuristiker för handelsresandeproblemet

Närmaste sammanslagnings-algoritmen:

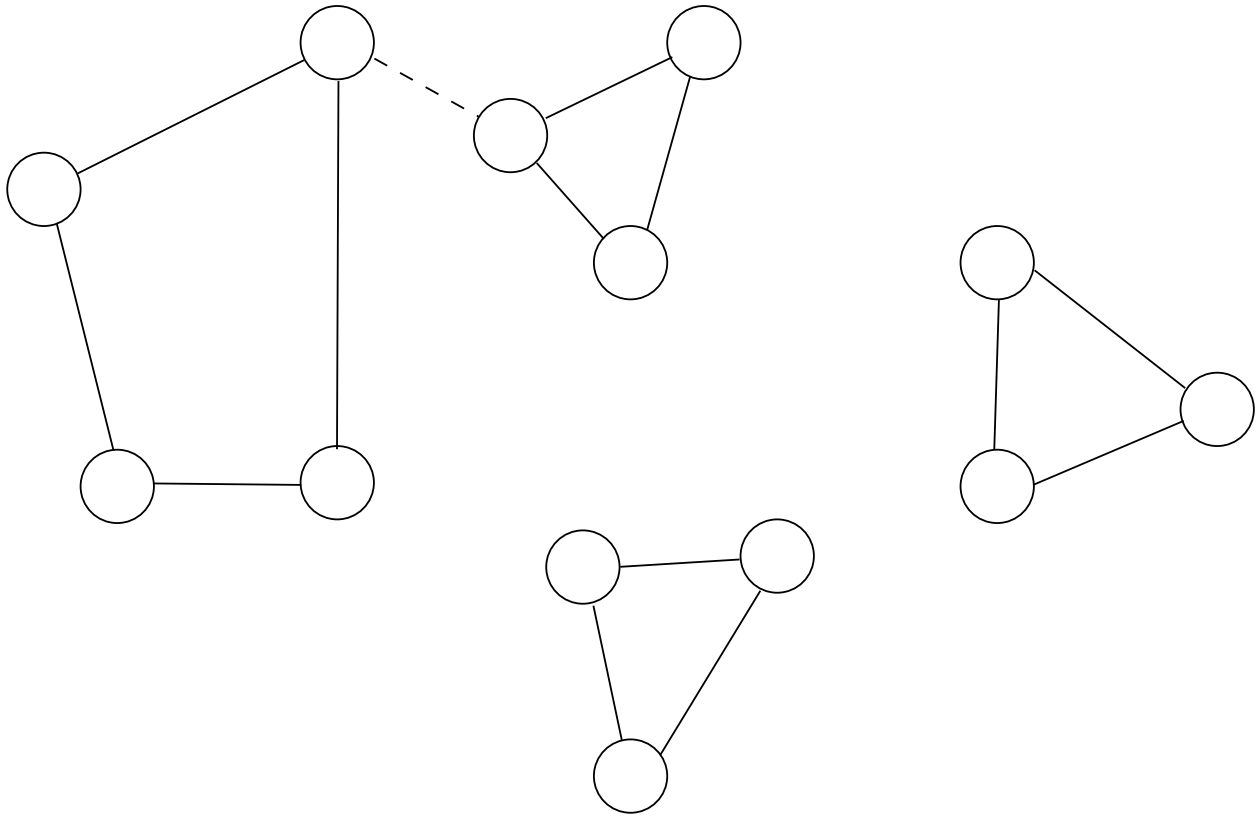
Flera partiella turer som successivt slås samman.
Slå ihop de turer som ligger närmast varandra: $\min_{k,l}(\min_{i \in T_k, j \in T_l} c_{ij})$,
och välj bågar att ta bort så att kostnaden för nya turen blir minimal.



Heuristiker för handelsresandeproblemet

Närmaste sammanslagnings-algoritmen:

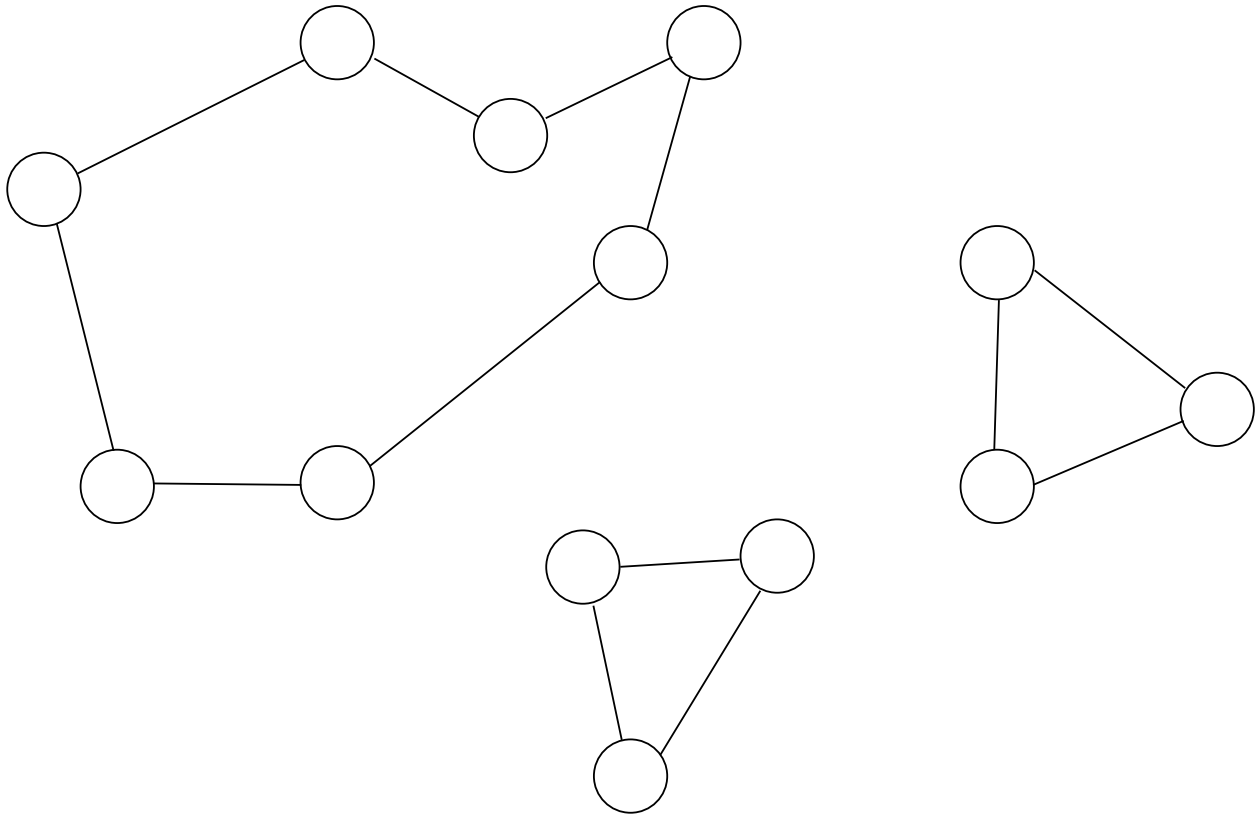
Flera partiella turer som successivt slås samman.
Slå ihop de turer som ligger närmast varandra: $\min_{k,l}(\min_{i \in T_k, j \in T_l} c_{ij})$,
och välj bågar att ta bort så att kostnaden för nya turen blir minimal.



Heuristiker för handelsresandeproblemet

Närmaste sammanslagnings-algoritmen:

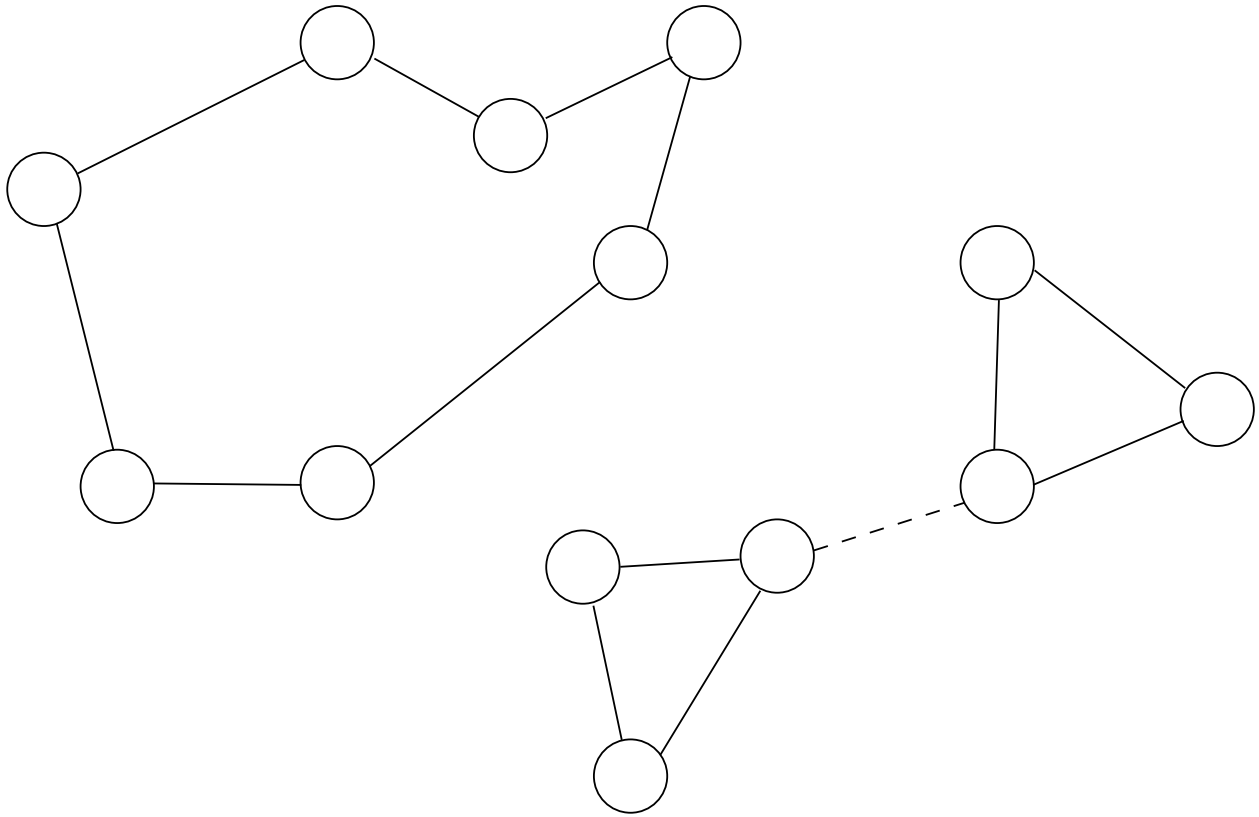
Flera partiella turer som successivt slås samman.
Slå ihop de turer som ligger närmast varandra: $\min_{k,l}(\min_{i \in T_k, j \in T_l} c_{ij})$,
och välj bågar att ta bort så att kostnaden för nya turen blir minimal.



Heuristiker för handelsresandeproblemet

Närmaste sammanslagnings-algoritmen:

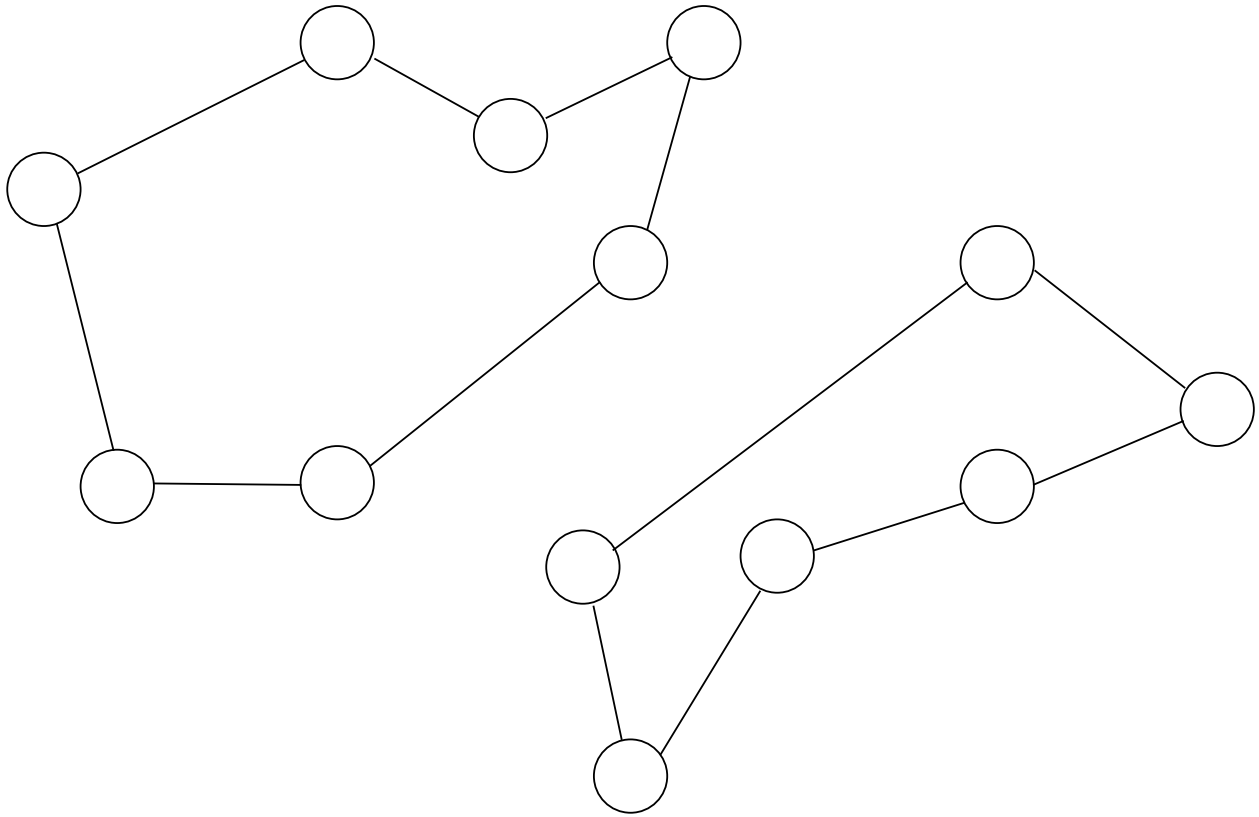
Flera partiella turer som successivt slås samman.
Slå ihop de turer som ligger närmast varandra: $\min_{k,l}(\min_{i \in T_k, j \in T_l} c_{ij})$,
och välj bågar att ta bort så att kostnaden för nya turen blir minimal.



Heuristiker för handelsresandeproblemet

Närmaste sammanslagnings-algoritmen:

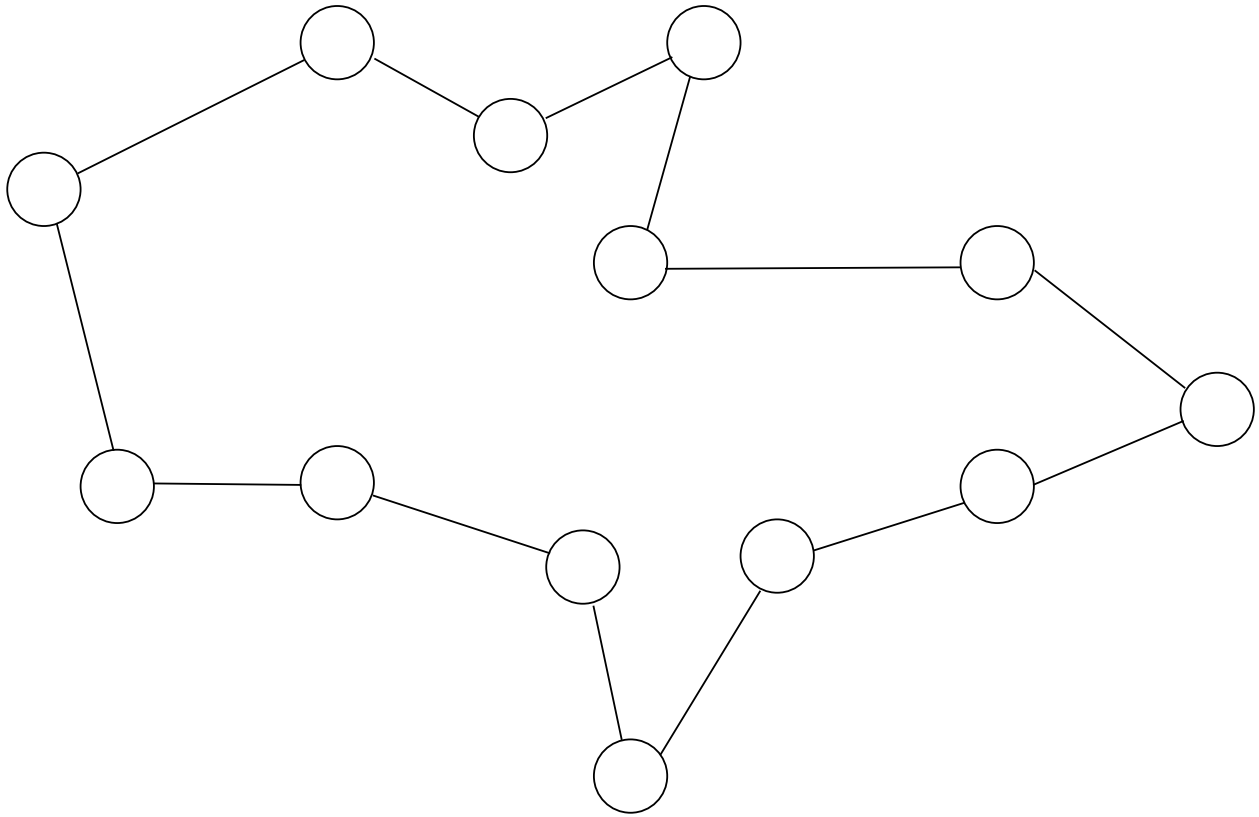
Flera partiella turer som successivt slås samman.
Slå ihop de turer som ligger närmast varandra: $\min_{k,l}(\min_{i \in T_k, j \in T_l} c_{ij})$,
och välj bågar att ta bort så att kostnaden för nya turen blir minimal.



Heuristiker för handelsresandeproblemet

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 bågar att ta bort så att kostnaden för nya turen blir minimal.



Heuristiker för handelsresandeproblemet

Insättnings-algoritmer
(en tur som utökas med en nod i taget):

Heuristiker för handelsresandeproblemet

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.)

Heuristiker för handelsresandeproblemet

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.)

Heuristiker för handelsresandeproblemet

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.)

Heuristiker för handelsresandeproblemet

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.)

$z_H \leq 2z^*$.

Specialutvecklade (ad hoc) algoritmer

Utnyttja problemets struktur.

Specialutvecklade (ad hoc) algoritmer

Utnyttja problemets struktur.

Trädalgoritmen för handelsresandeproblemet ΔTSP:

Specialutvecklade (ad hoc) algoritmer

Utnyttja problemets struktur.

Trädalgoritmen för handelsresandeproblemet ΔTSP:

1. Finn billigaste uppspännande träd.

Specialutvecklade (ad hoc) algoritmer

Utnyttja problemets struktur.

Trädalgoritmen för handelsresandeproblemet ΔTSP:

1. Finn billigaste uppspännande träd.
2. Konstruera en graf med parallella bågar genom att dubblera varje båge i trädet.

Specialutvecklade (ad hoc) algoritmer

Utnyttja problemets struktur.

Trädalgoritmen för handelsresandeproblemet ΔTSP:

1. Finn billigaste uppspännande träd.
2. Konstruera en graf med parallella bågar genom att dubblera varje båge i trädet.
3. Finn en Eulercykel.

Specialutvecklade (ad hoc) algoritmer

Utnyttja problemets struktur.

Trädalgoritmen för handelsresandeproblemet ΔTSP:

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).

Specialutvecklade (ad hoc) algoritmer

Utnyttja problemets struktur.

Trädalgoritmen för handelsresandeproblemet ΔTSP:

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).

Specialutvecklade (ad hoc) algoritmer

Utnyttja problemets struktur.

Trädalgoritmen för handelsresandeproblemet ΔTSP:

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

Specialutvecklade (ad hoc) algoritmer

Utnyttja problemets struktur.

Trädalgoritmen för handelsresandeproblemet ΔTSP:

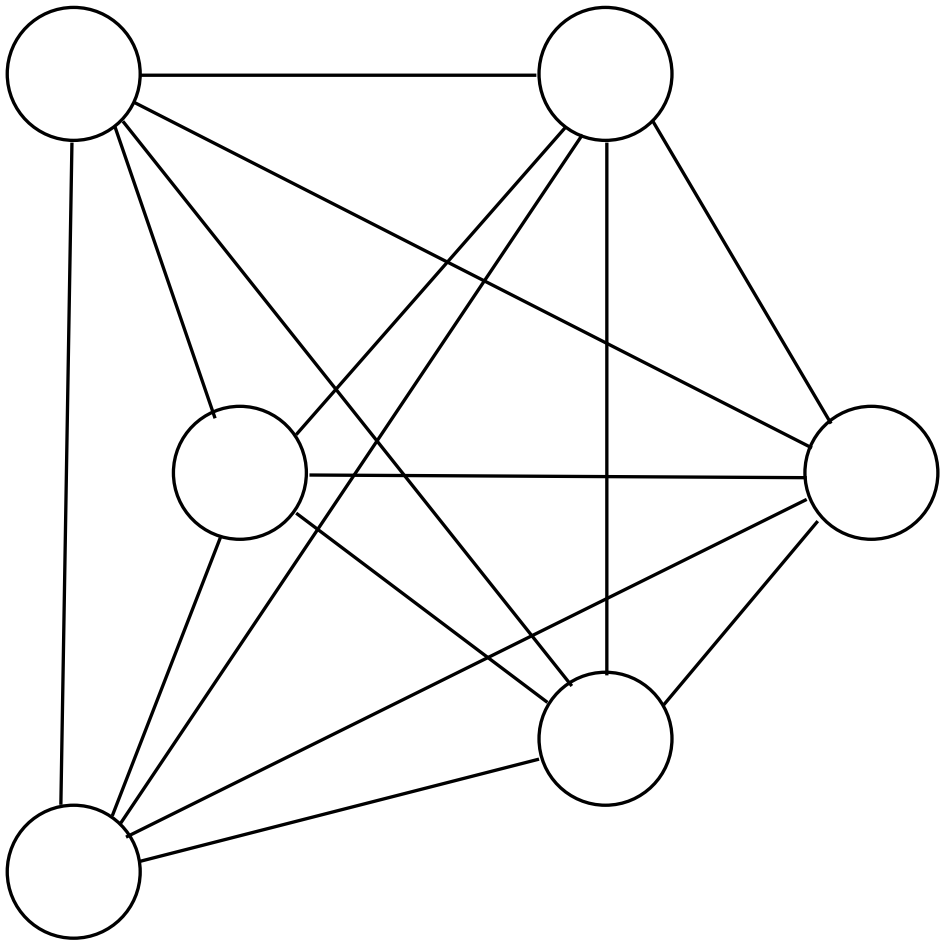
1. Finn billigaste uppspännande träd.
2. Konstruera en graf med parallella bågar genom att dubblera varje båge i trädet.
3. Finn en Eulercykel.
4. Transformera Eulercykeln till en Hamiltoncykel (genom att ta genvägar vid återbesök).

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

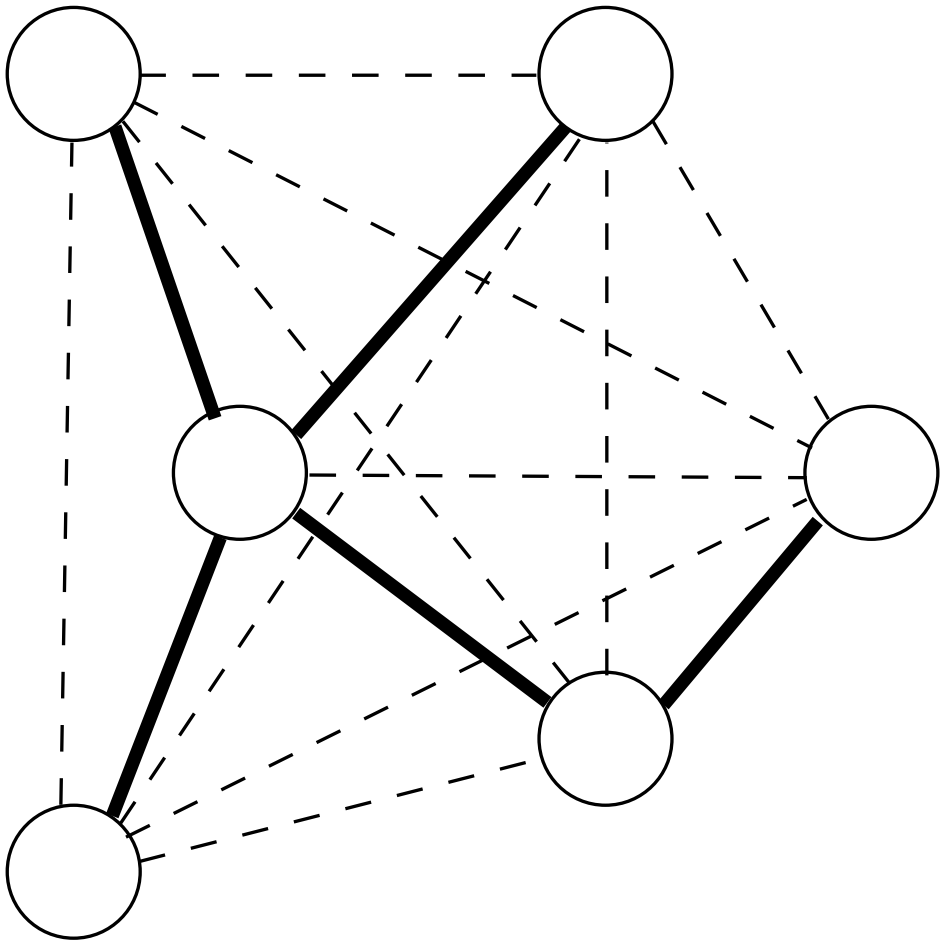
Målfunktionsvärdet: $z_{MST} \leq z^*$, $z_E = 2z_{MST}$, $z_H \leq z_E \Rightarrow z_H \leq 2z^*$.

Sämre för TSP utan Δ.

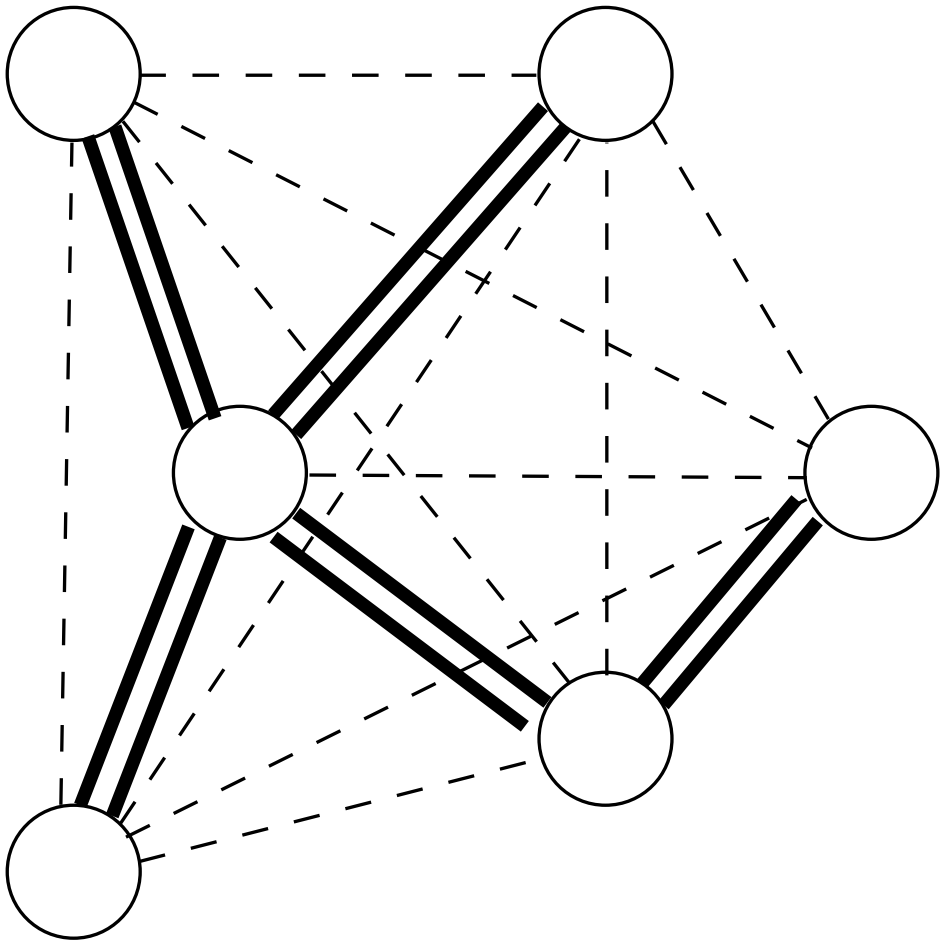
Trädalgoritmen för handelsresandeproblemet



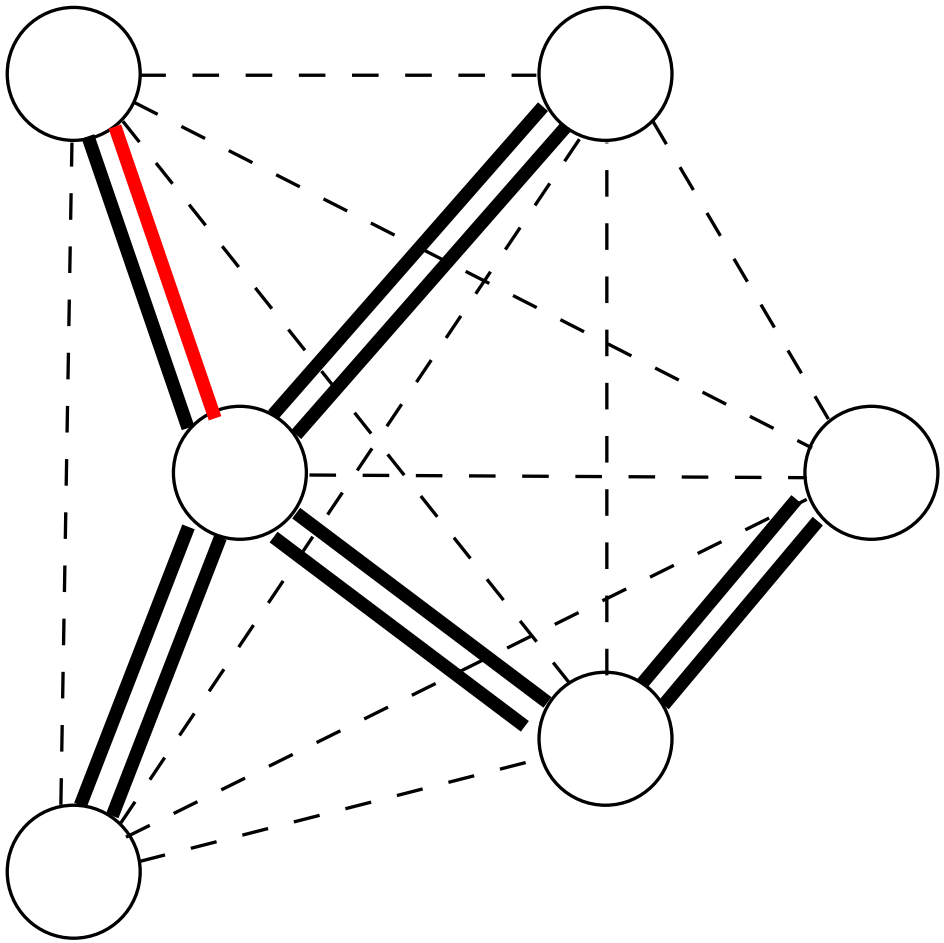
Trädalgoritmen för handelsresandeproblemet



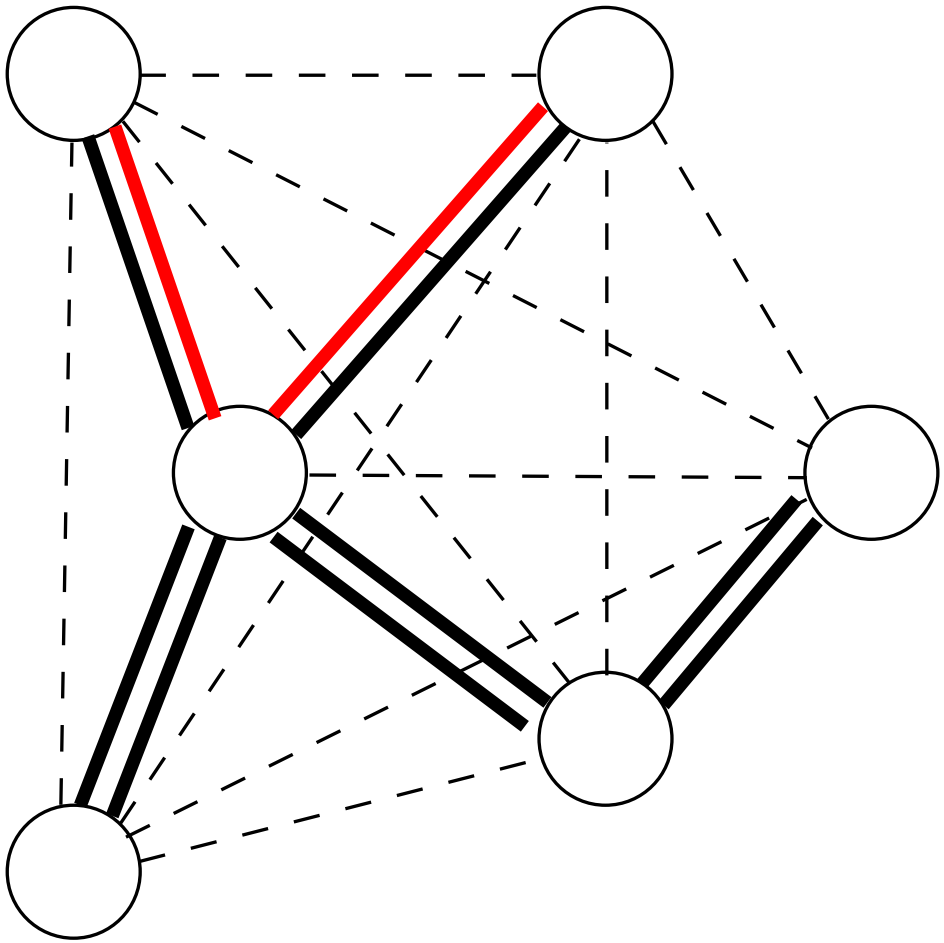
Trädalgoritmen för handelsresandeproblemet



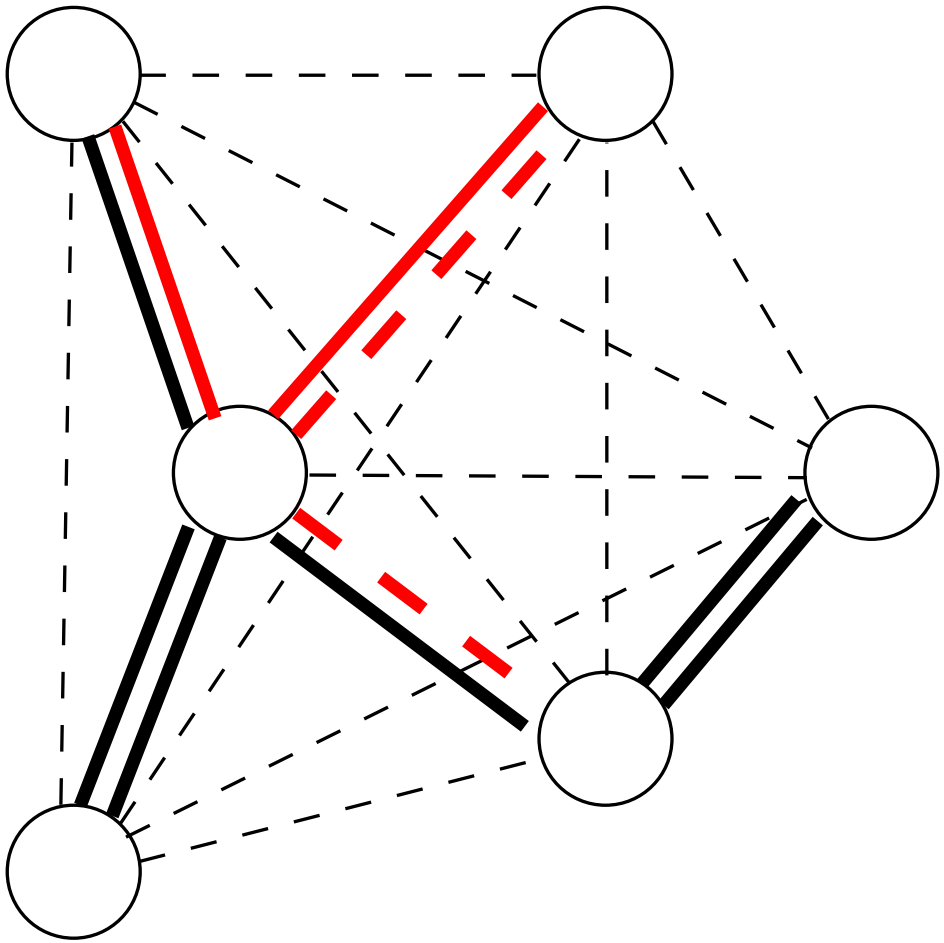
Trädalgoritmen för handelsresandeproblemet



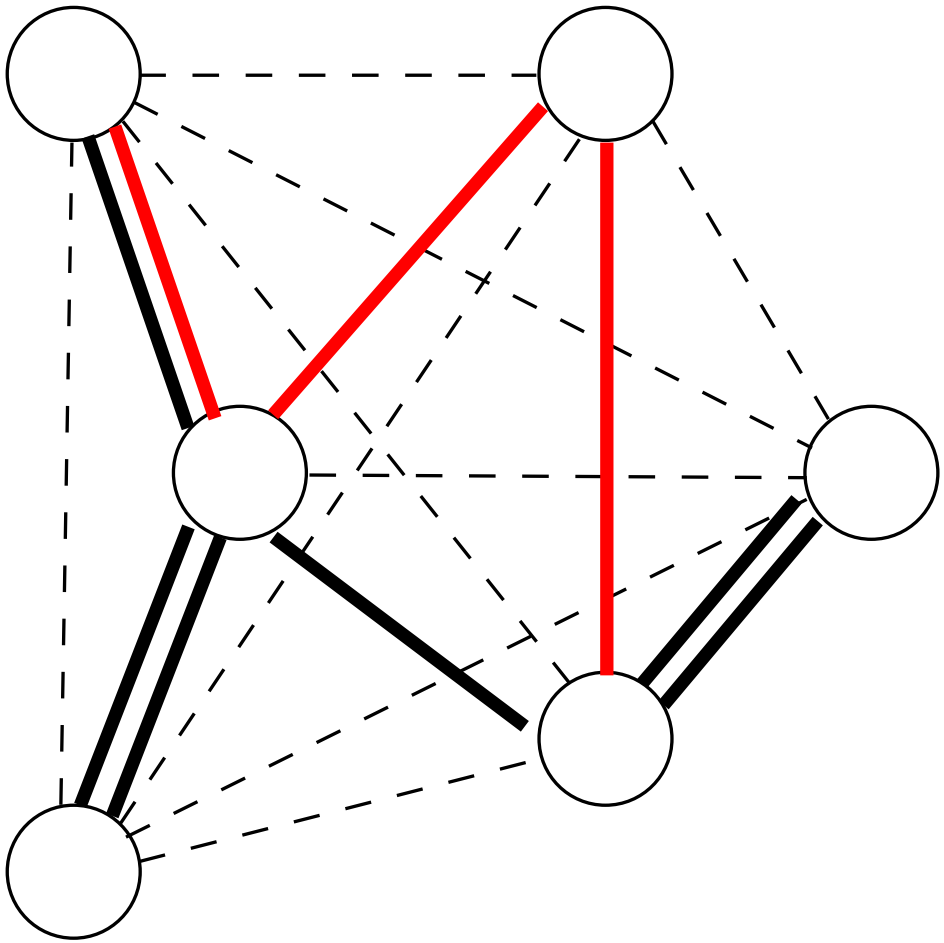
Trädalgoritmen för handelsresandeproblemet



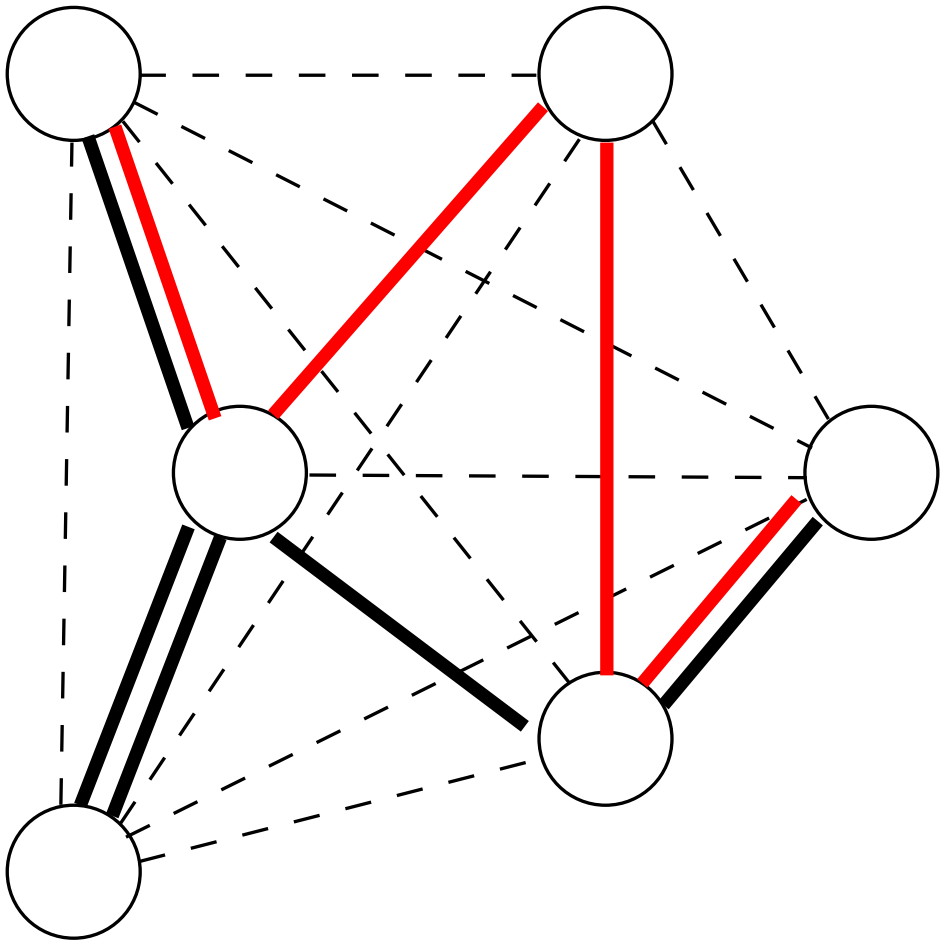
Trädalgoritmen för handelsresandeproblemet



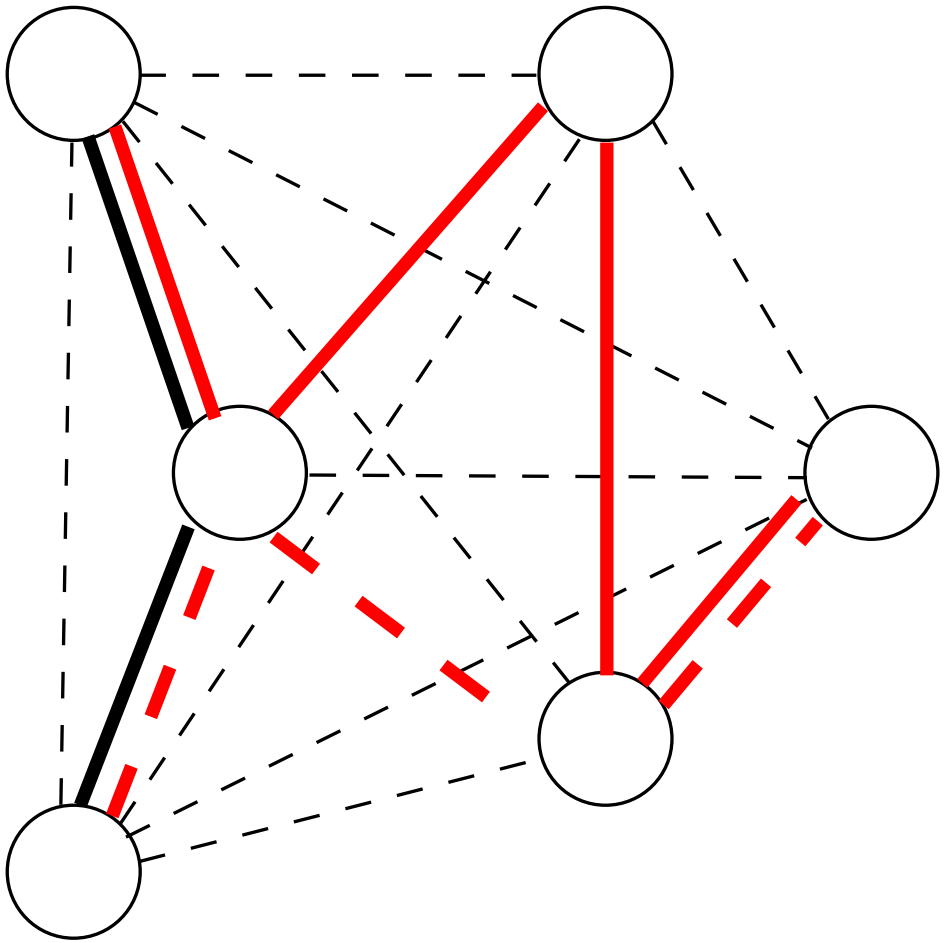
Trädalgoritmen för handelsresandeproblemet



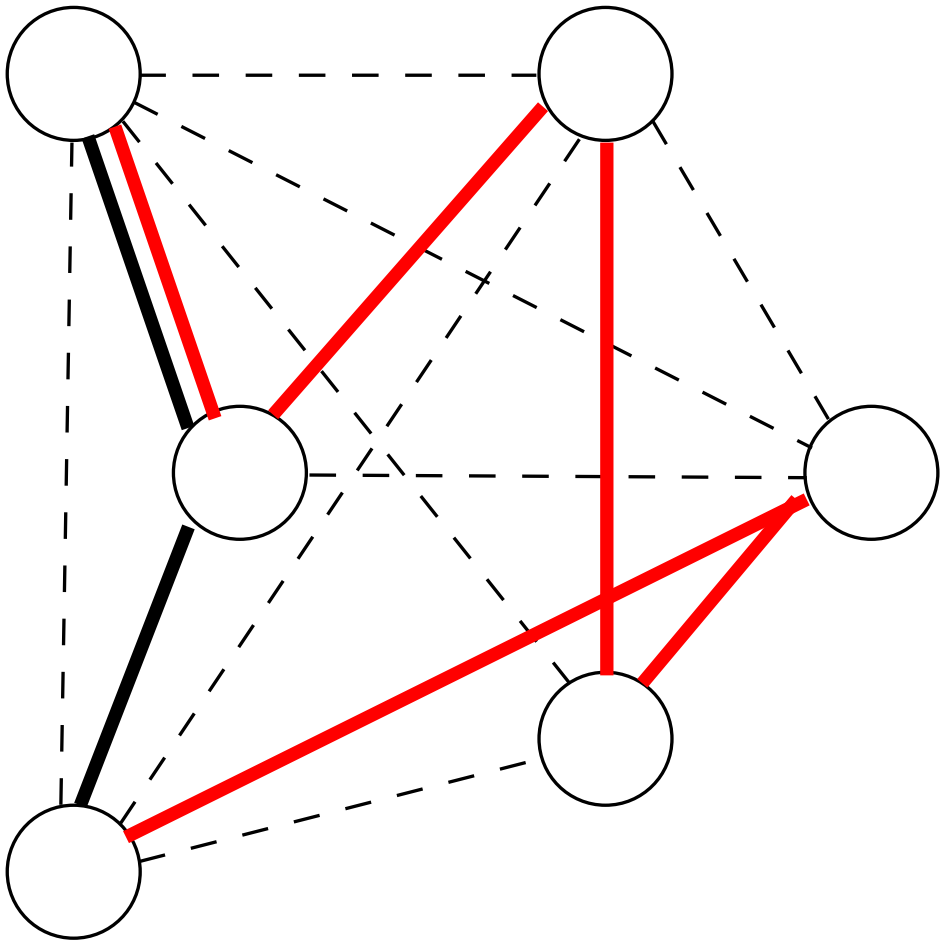
Trädalgoritmen för handelsresandeproblemet



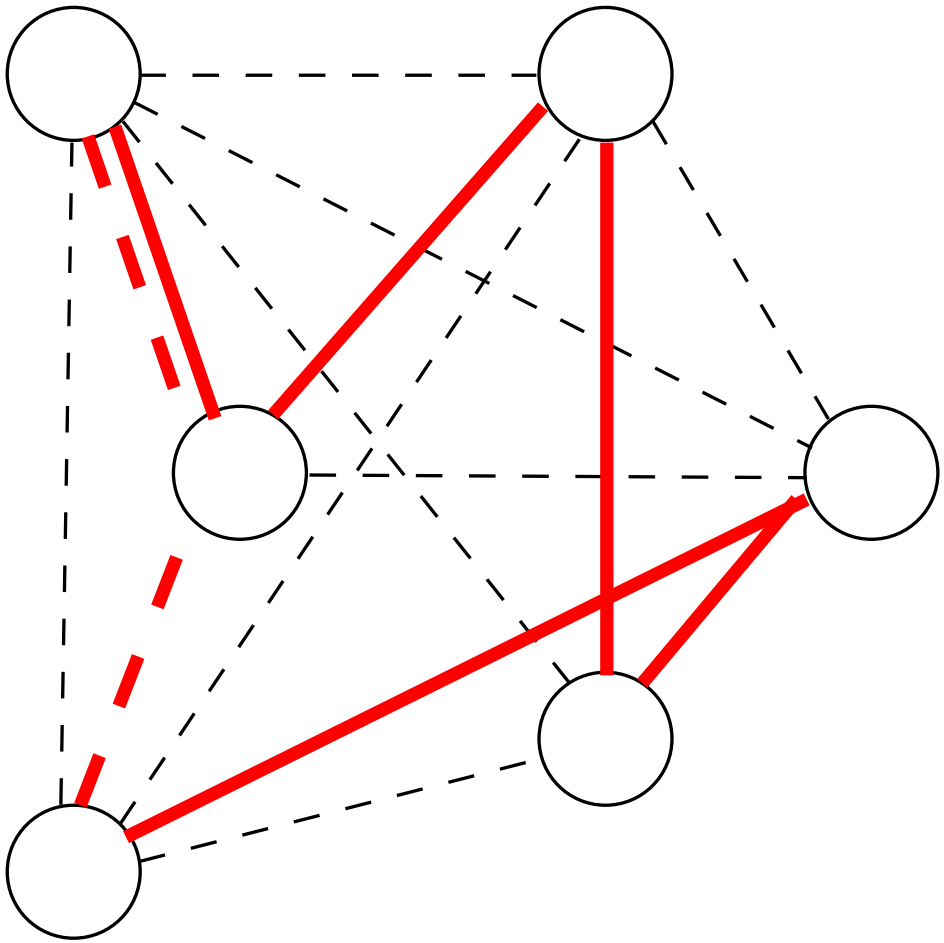
Trädalgoritmen för handelsresandeproblemet



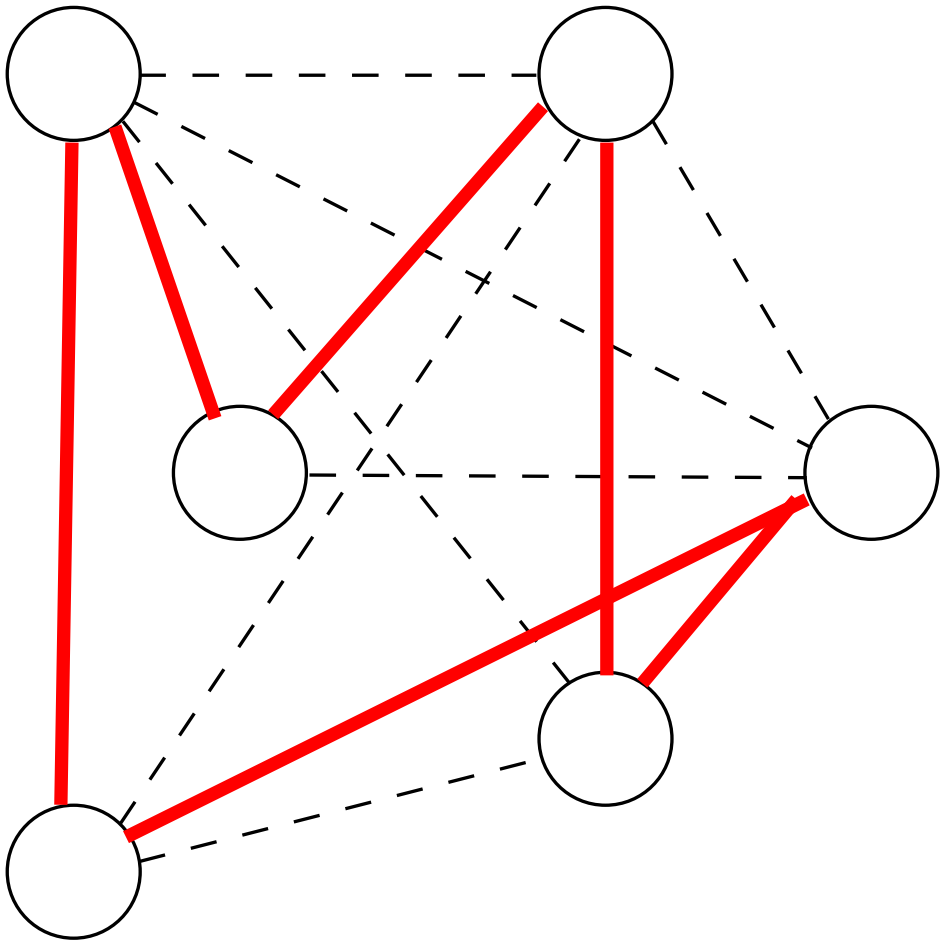
Trädalgoritmen för handelsresandeproblemet



Trädalgoritmen för handelsresandeproblemet



Trädalgoritmen för handelsresandeproblemet



Heuristiker för handelsresandeproblemet

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

Heuristiker för handelsresandeproblemet

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

1. Finn billigaste uppspännande träd T.

Heuristiker 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.

Heuristiker 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.

Heuristiker 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, $O(n^4)$.

Heuristiker för handelsresandeproblemet

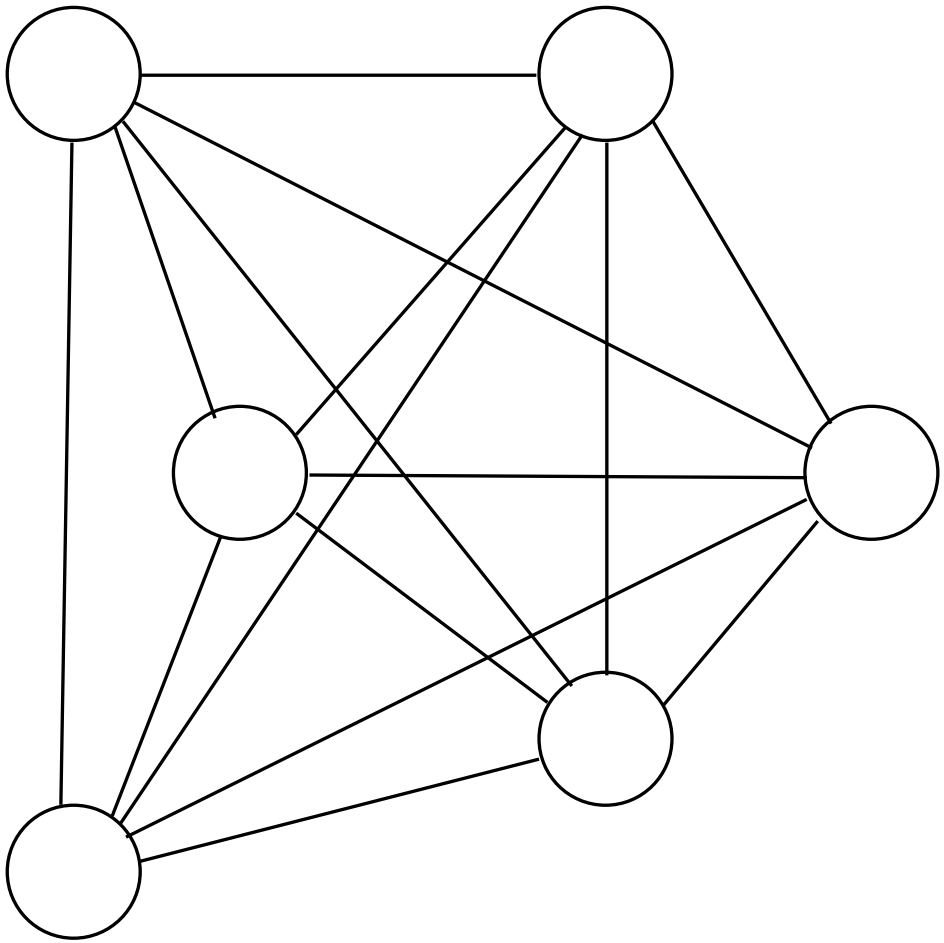
Christofides algoritm: Addera bara bågar till noder med udda valens.

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.

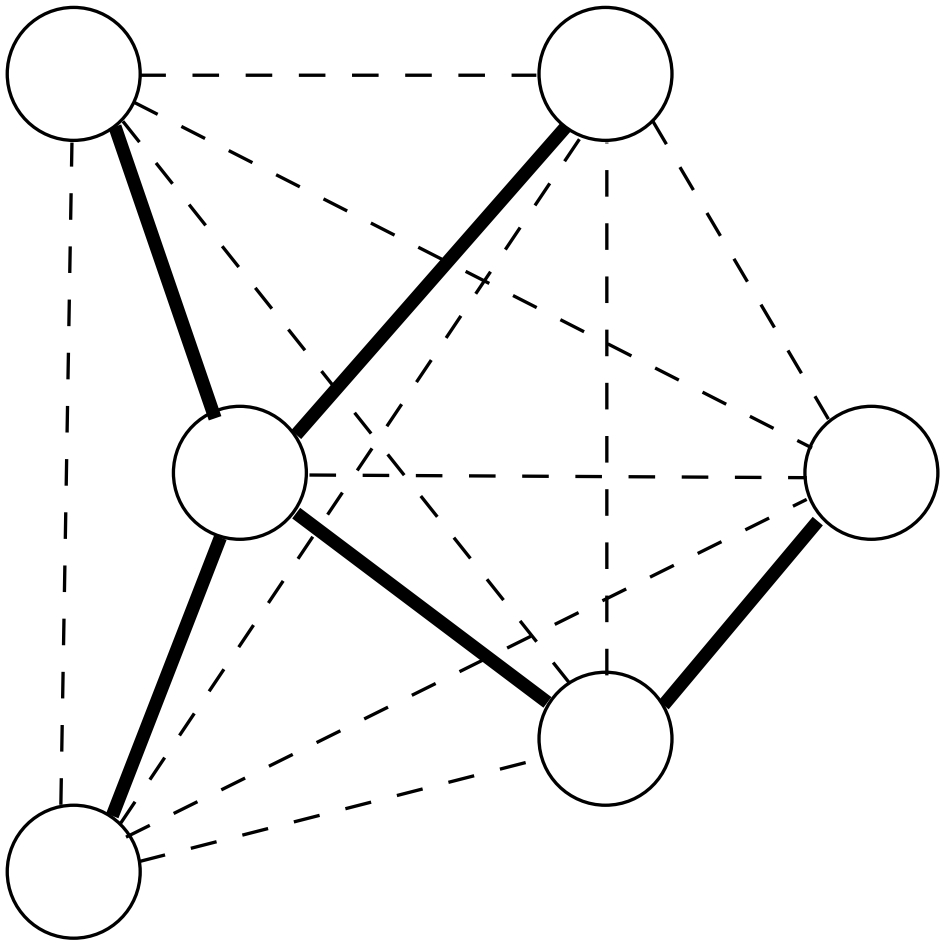
Steg 2: Matchningsproblem, $O(n^4)$.

$z_H \leq 3/2 z^*$ för ΔTSP.

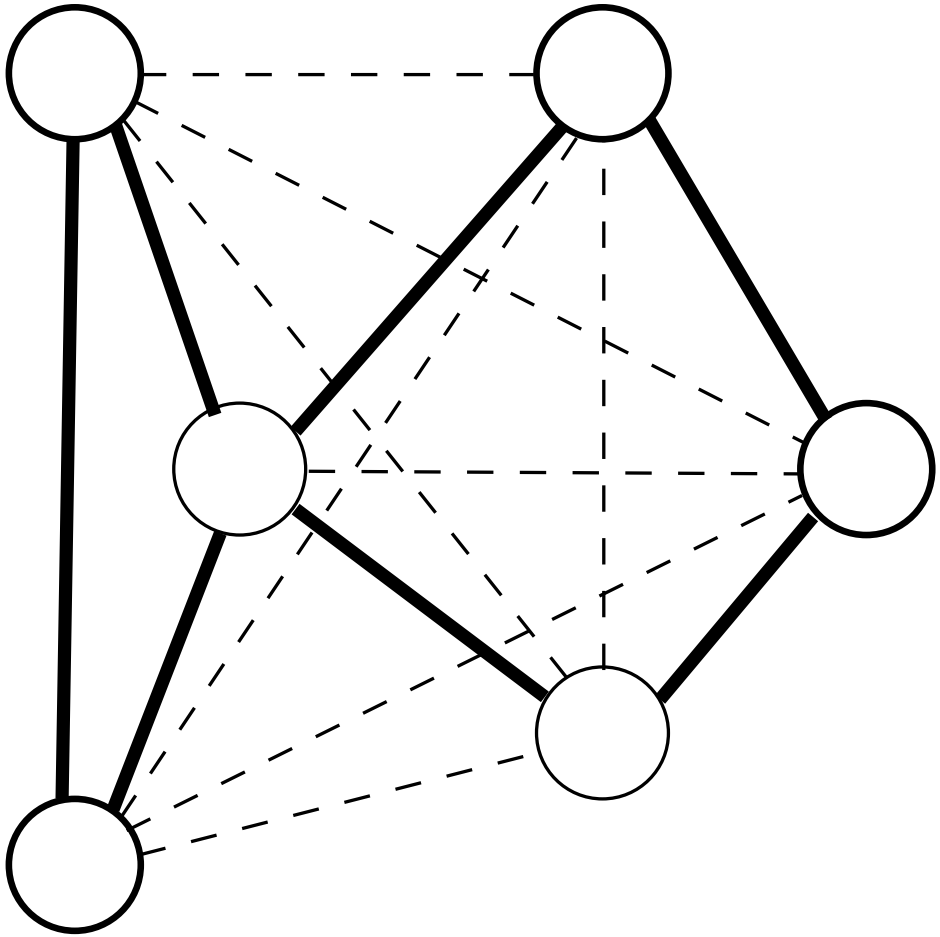
Heuristiker för handelsresandeproblemet



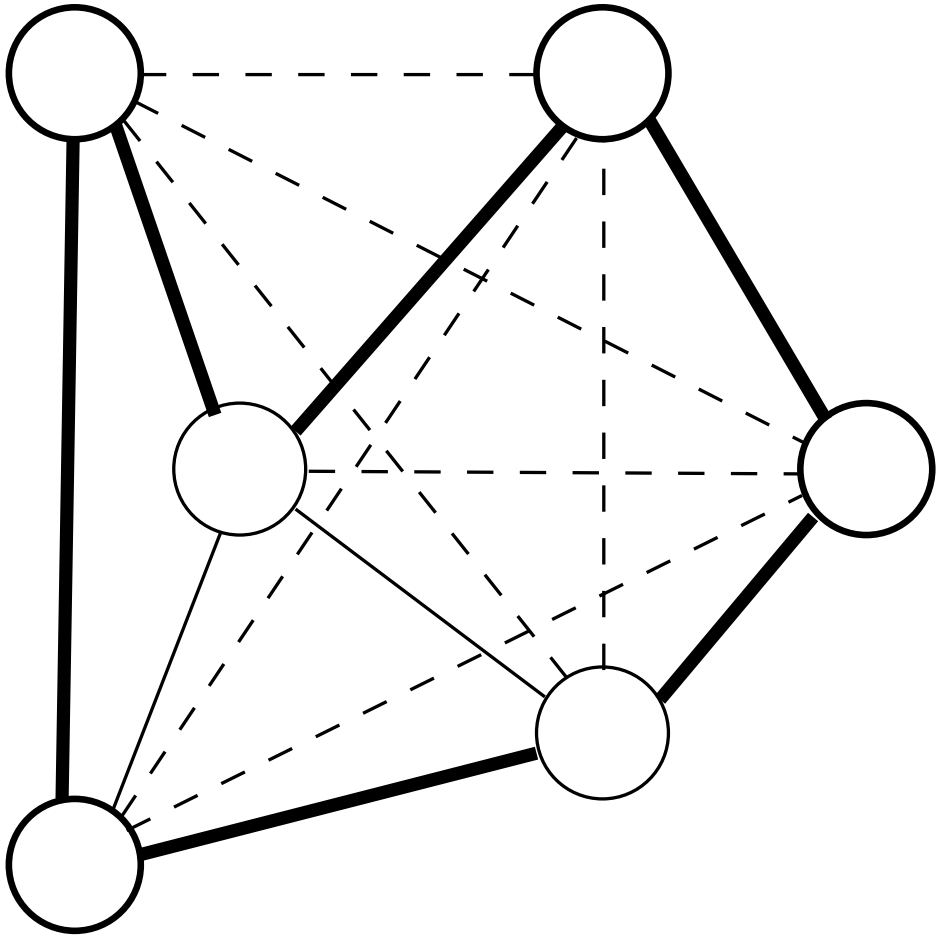
Heuristiker för handelsresandeproblemet



Heuristiker för handelsresandeproblemet



Heuristiker för handelsresandeproblemet



Heuristiker för hinkpackningsproblemet

Heuristiker för hinkpackningsproblemet

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

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.

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).

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.

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 tommaste användna hinken.

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 tommaste användna hinken.

Näst värsta plats (‘Almost worst fit”) Placera nästa objekt i den näst tommaste hinken.

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 tommaste användna hinken.

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

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

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 tommaste användna hinken.

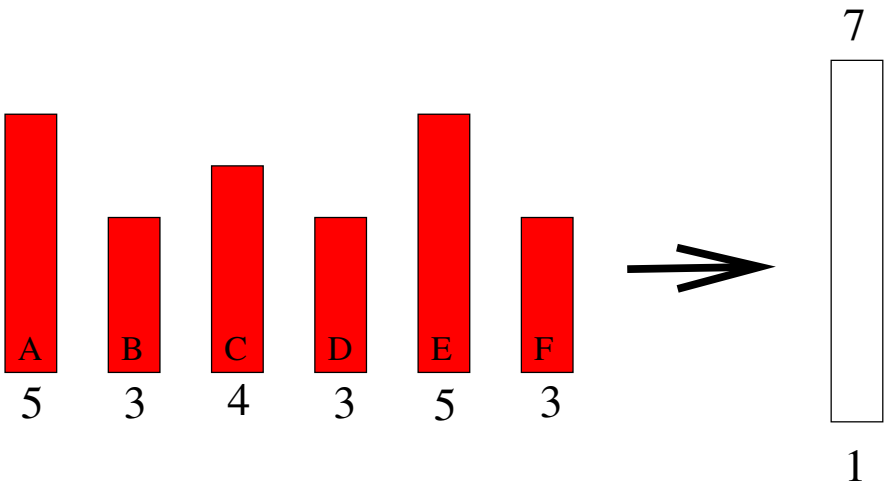
Näst värsta plats ('Almost worst fit") Placera nästa objekt i den näst tommaste hinken.

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

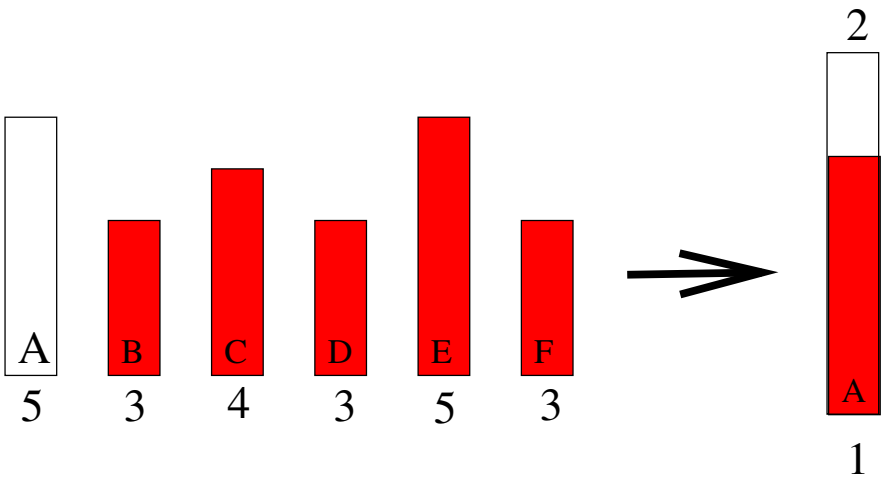
Första plats



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

Heuristiker för hinkpackningsproblemet

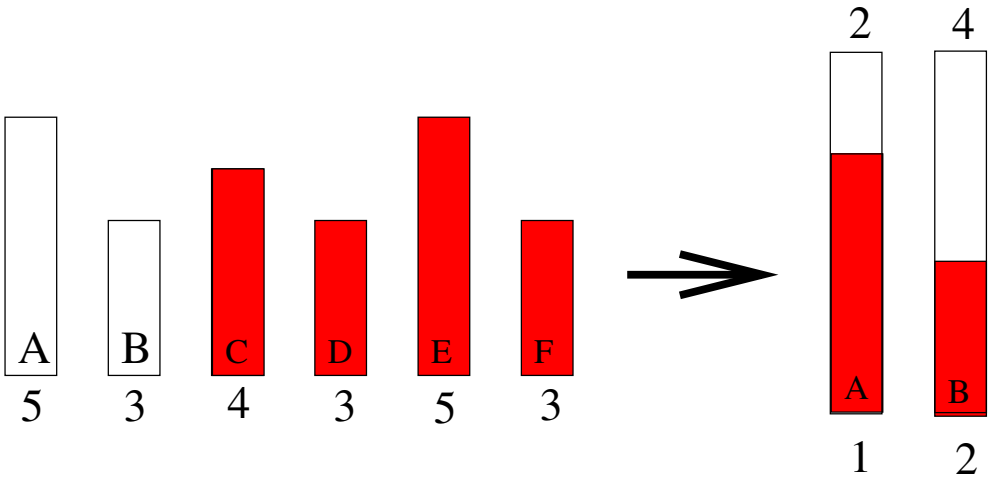
Första plats



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

Heuristiker för hinkpackningsproblemet

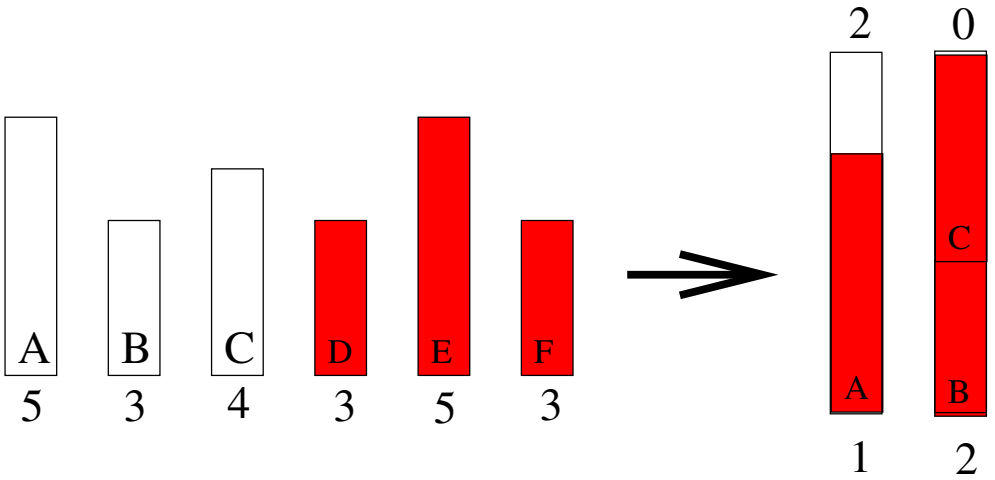
Första plats



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

Heuristiker för hinkpackningsproblemet

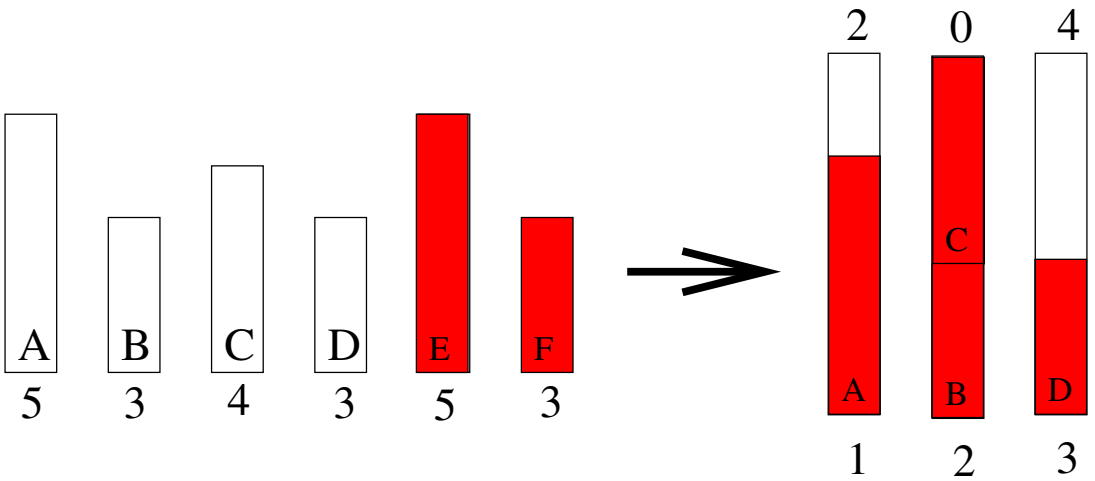
Första plats



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

Heuristiker för hinkpackningsproblemet

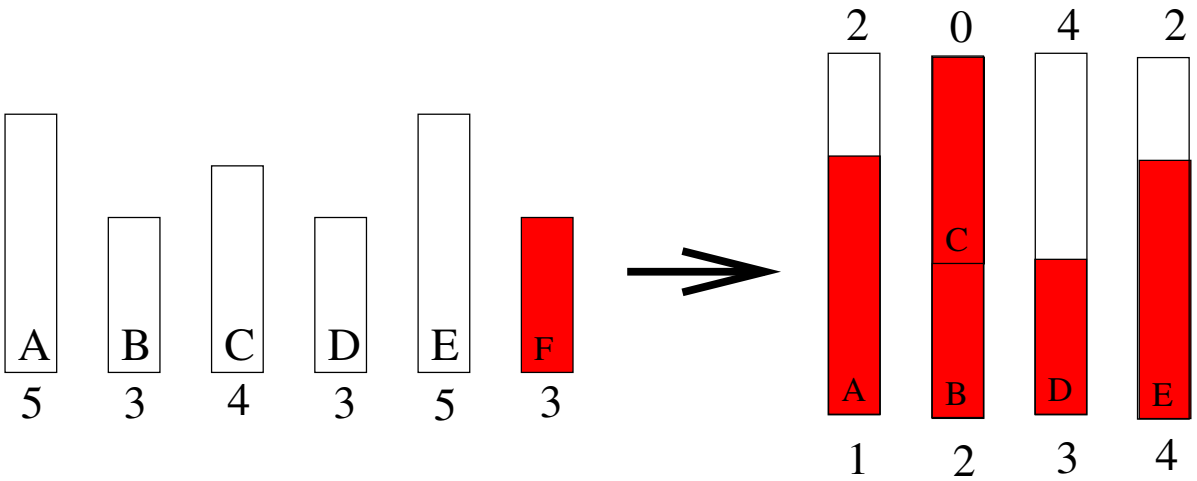
Första plats



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

Heuristiker för hinkpackningsproblemet

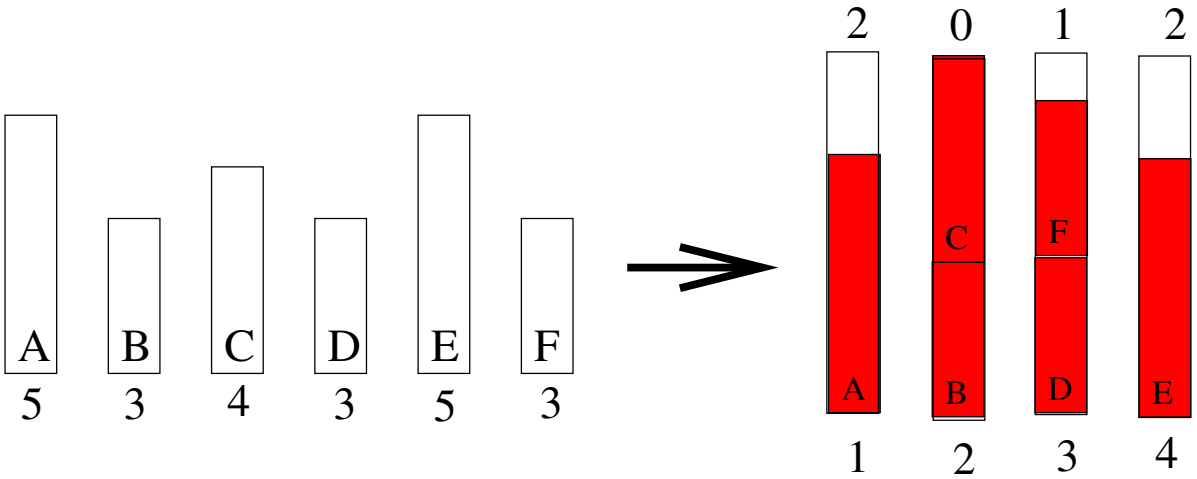
Första plats



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

Heuristiker för hinkpackningsproblemet

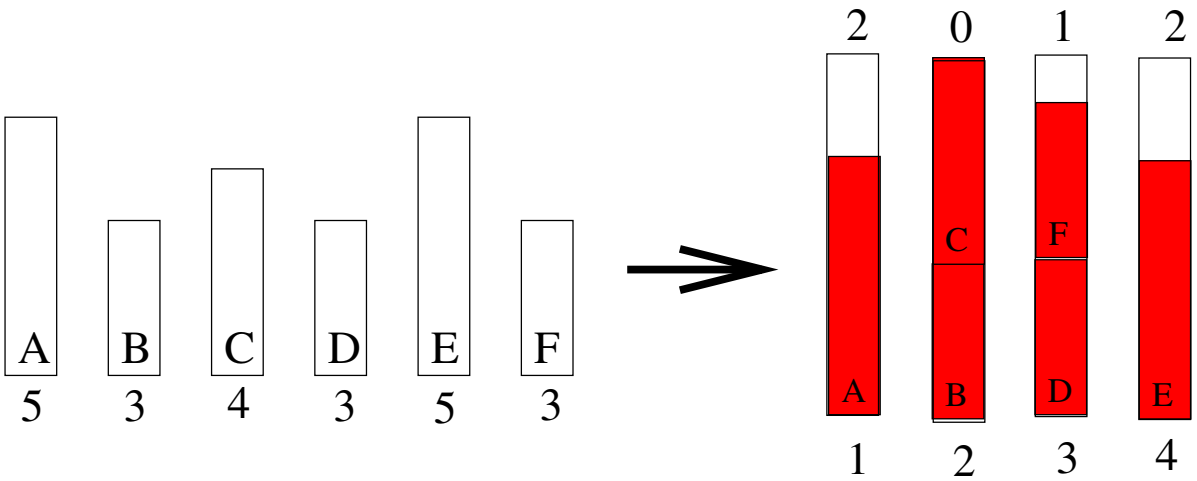
Första plats



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

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.

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.

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).

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 enstaka nod ger någon förbättring.

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 enstaka nod ger någon förbättring.

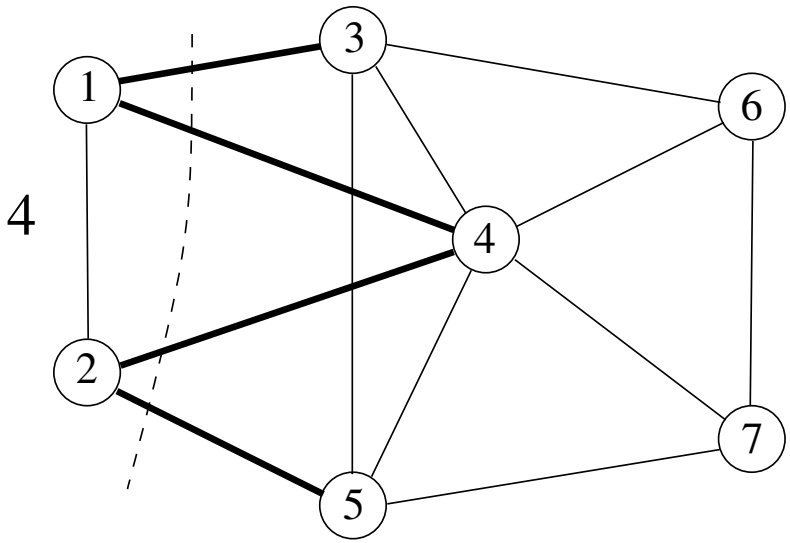
$z^* \leq 2z_H$.

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 enstaka nod ger någon förbättring.

$z^* \leq 2z_H$.

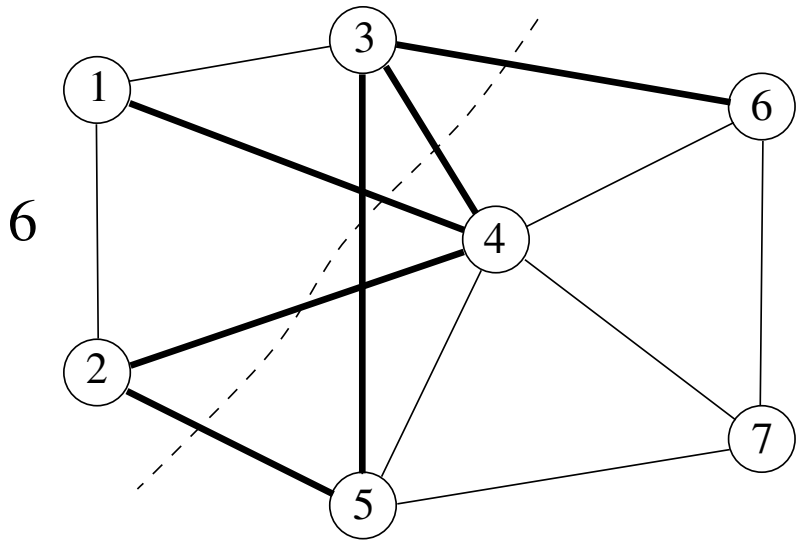


Heuristiker för maxsnittsproblemet

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 enstaka nod ger någon förbättring.

$z^* \leq 2z_H$.

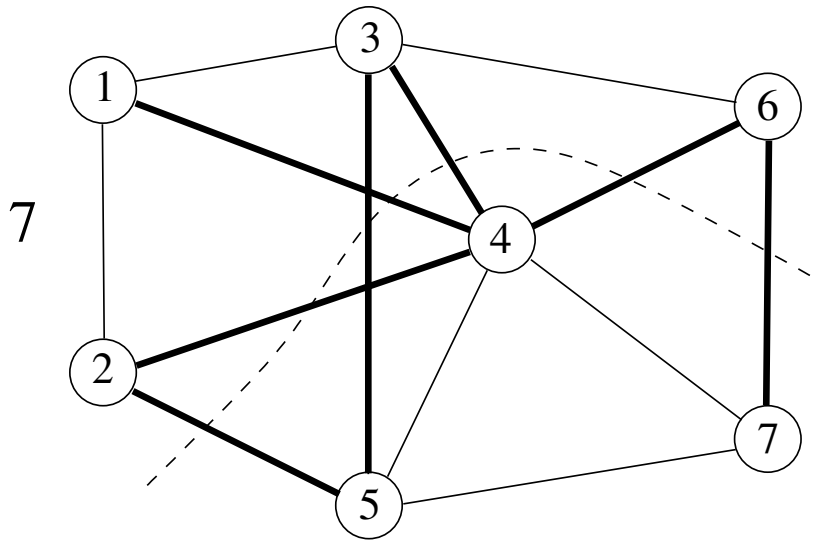


Heuristiker för maxsnittsproblemet

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 enstaka nod ger någon förbättring.

$z^* \leq 2z_H$.



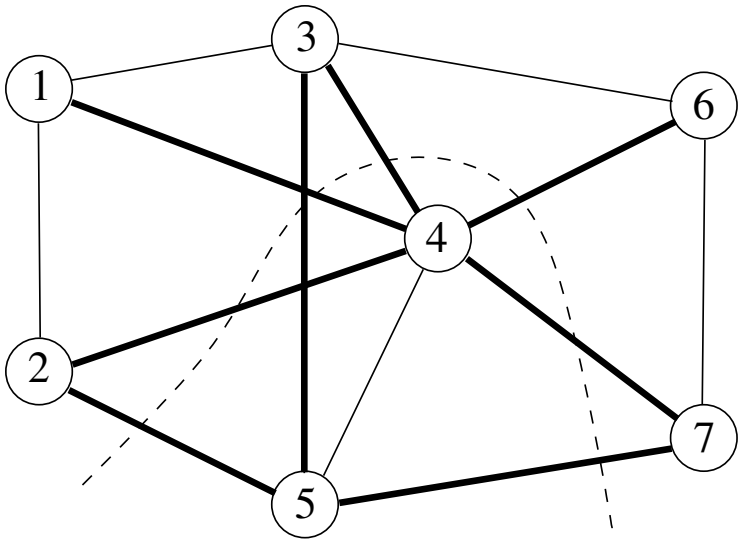
Heuristiker för maxsnittsproblemet

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 enstaka nod ger någon förbättring.

$z^* \leq 2z_H$.

8

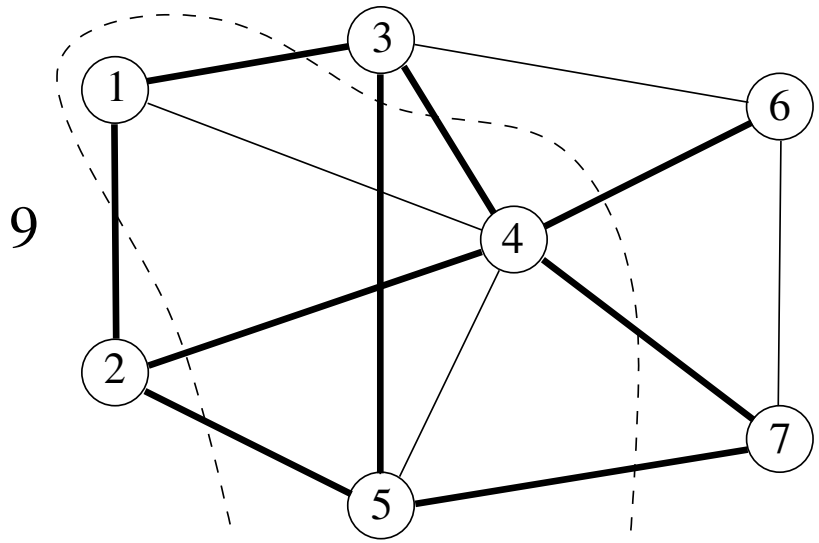


Heuristiker för maxsnittsproblemet

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 enstaka nod ger någon förbättring.

$z^* \leq 2z_H$.

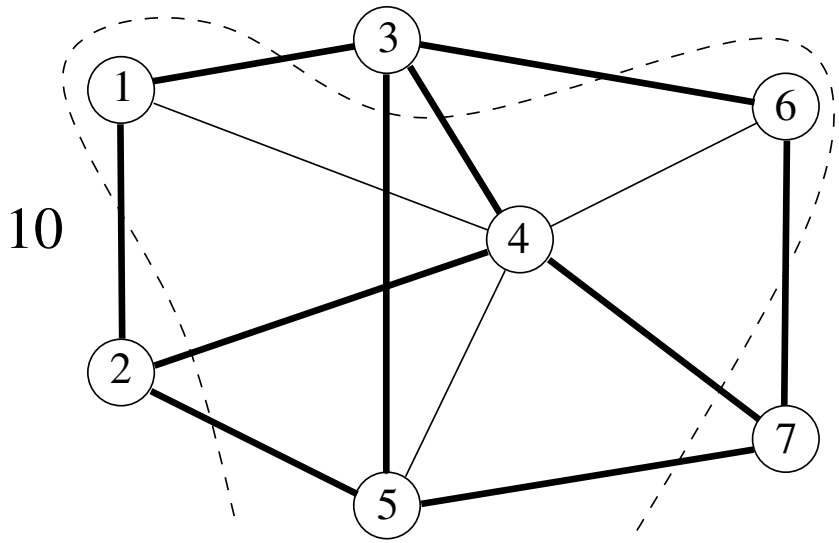


Heuristiker för maxsnittsproblemet

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 enstaka nod ger någon förbättring.

$z^* \leq 2z_H$.



Heuristiker för andra grafproblem

Ofta ganska självklara:

Heuristiker för andra grafproblem

Ofta ganska självklara:

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

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.

Heuristiker för andra grafproblem

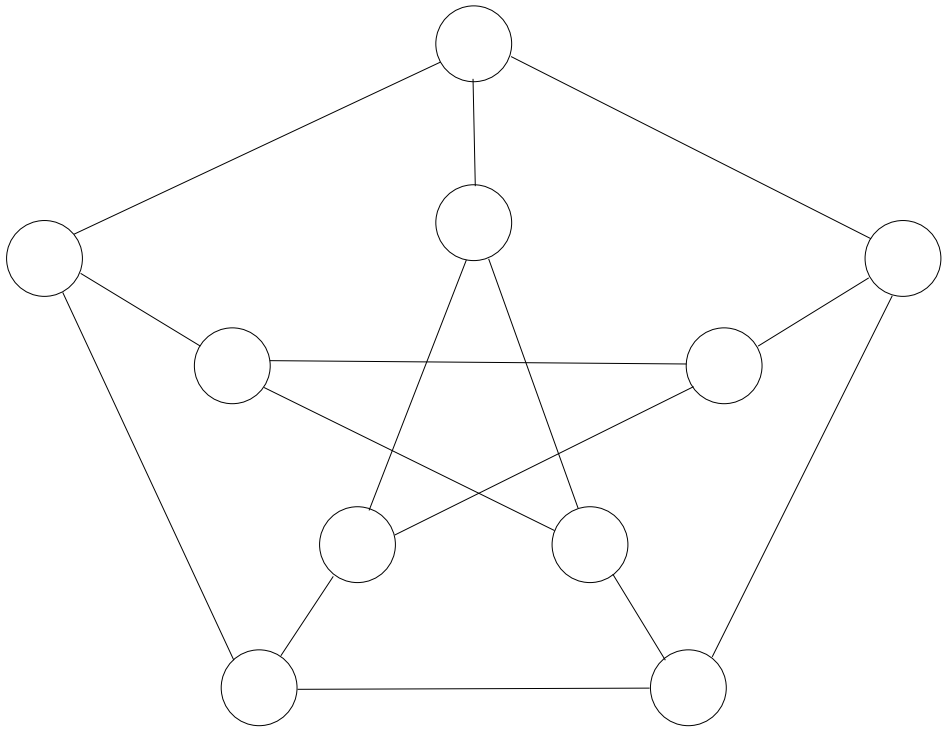
Ofta ganska självklara:

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.

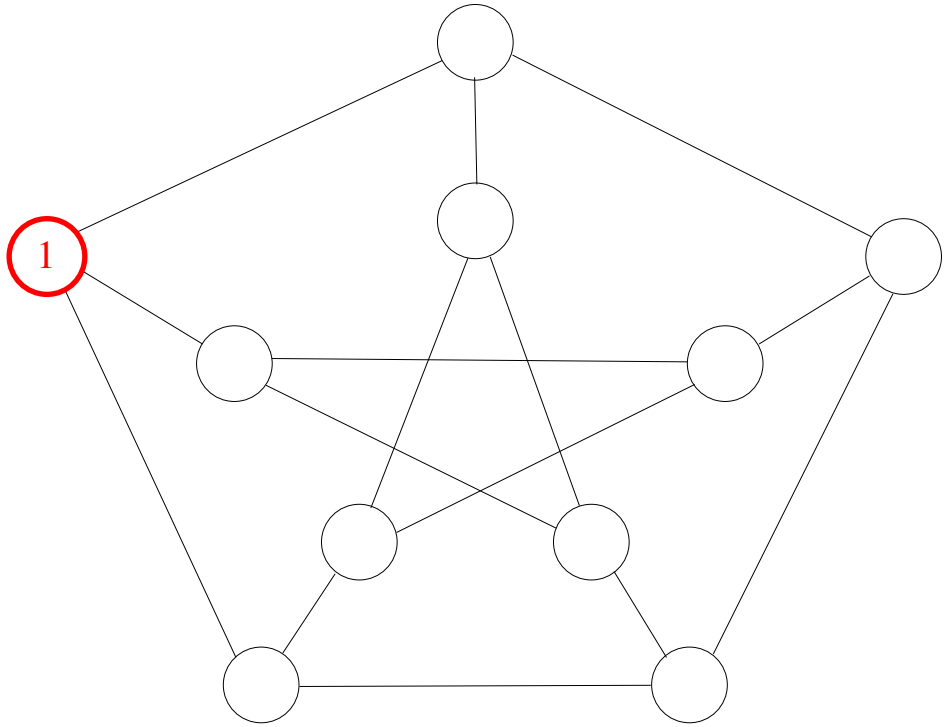
Heuristik för minimal nodfärgning

Färga nästa nod med lägsta möjliga färg.



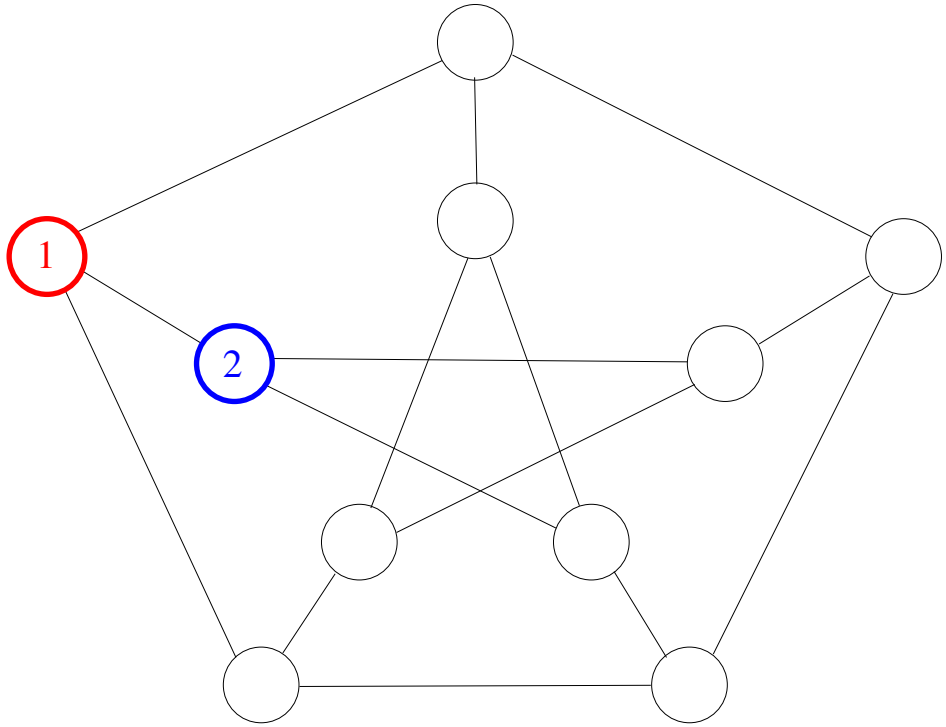
Heuristik för minimal nodfärgning

Färga nästa nod med lägsta möjliga färg.



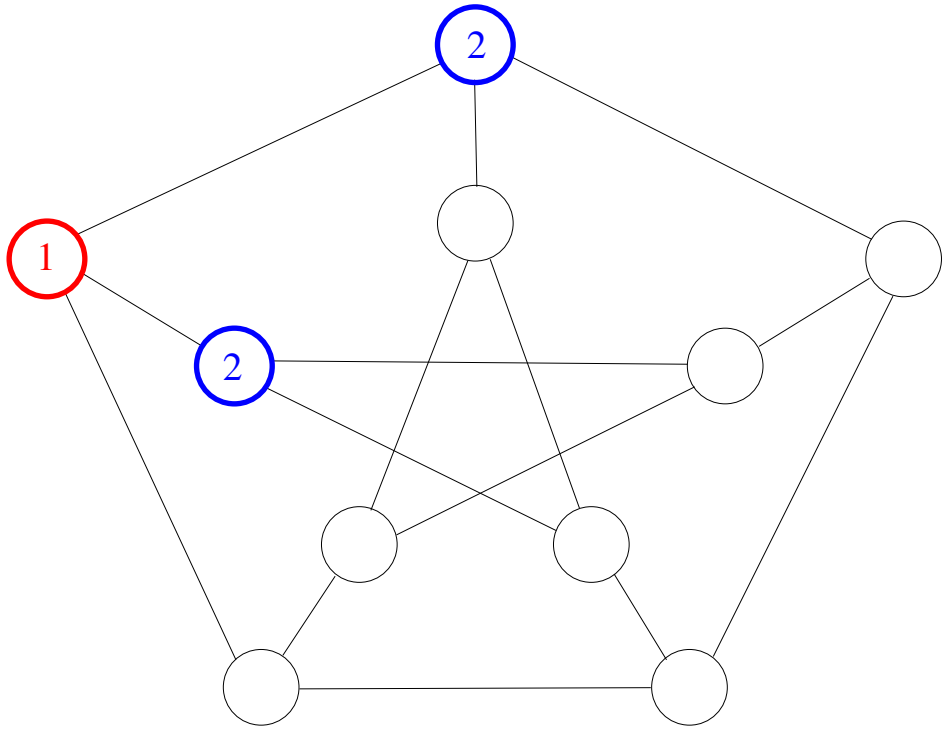
Heuristik för minimal nodfärgning

Färga nästa nod med lägsta möjliga färg.



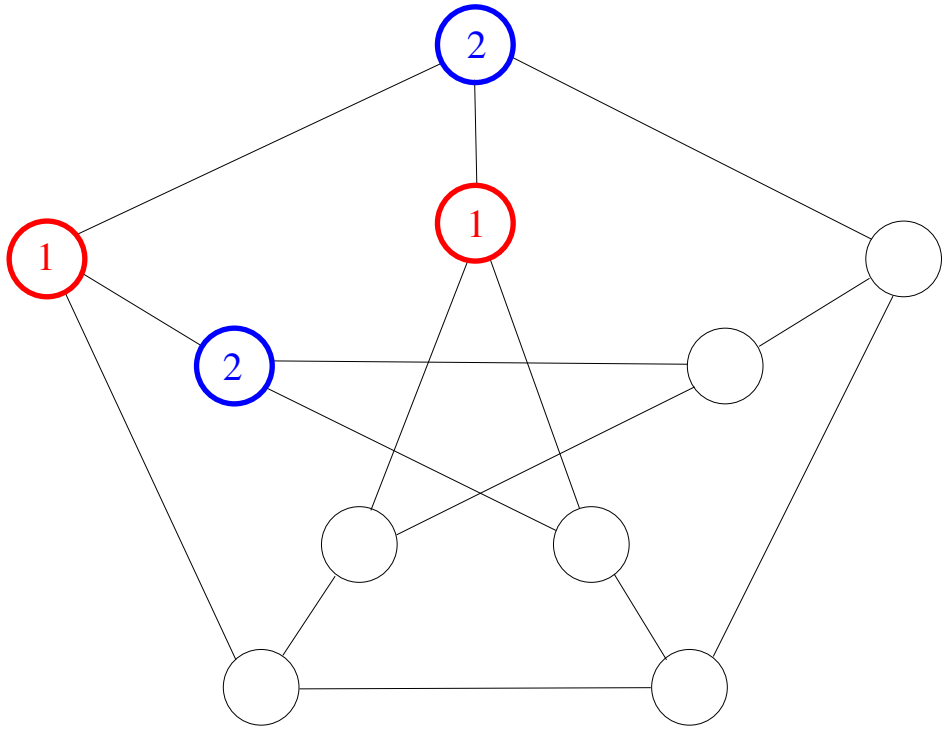
Heuristik för minimal nodfärgning

Färga nästa nod med lägsta möjliga färg.



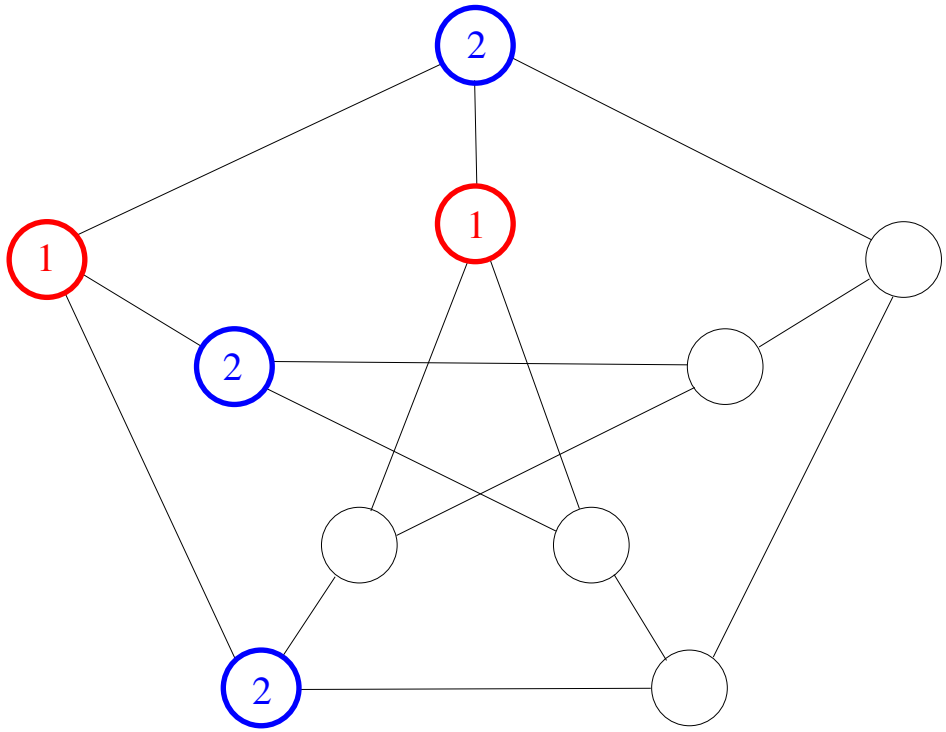
Heuristik för minimal nodfärgning

Färga nästa nod med lägsta möjliga färg.



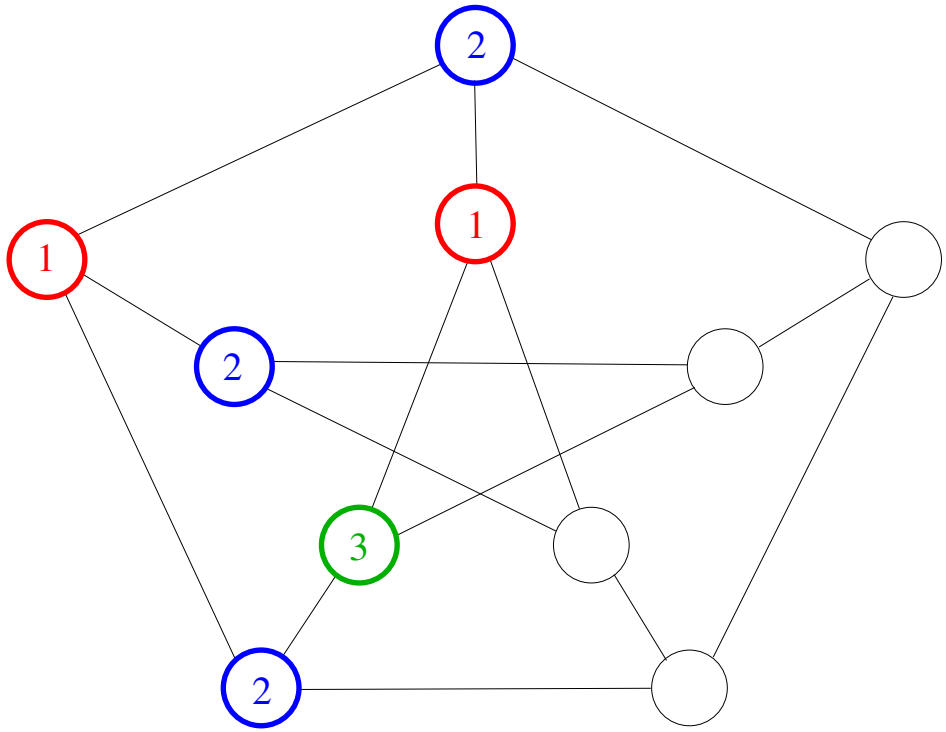
Heuristik för minimal nodfärgning

Färga nästa nod med lägsta möjliga färg.



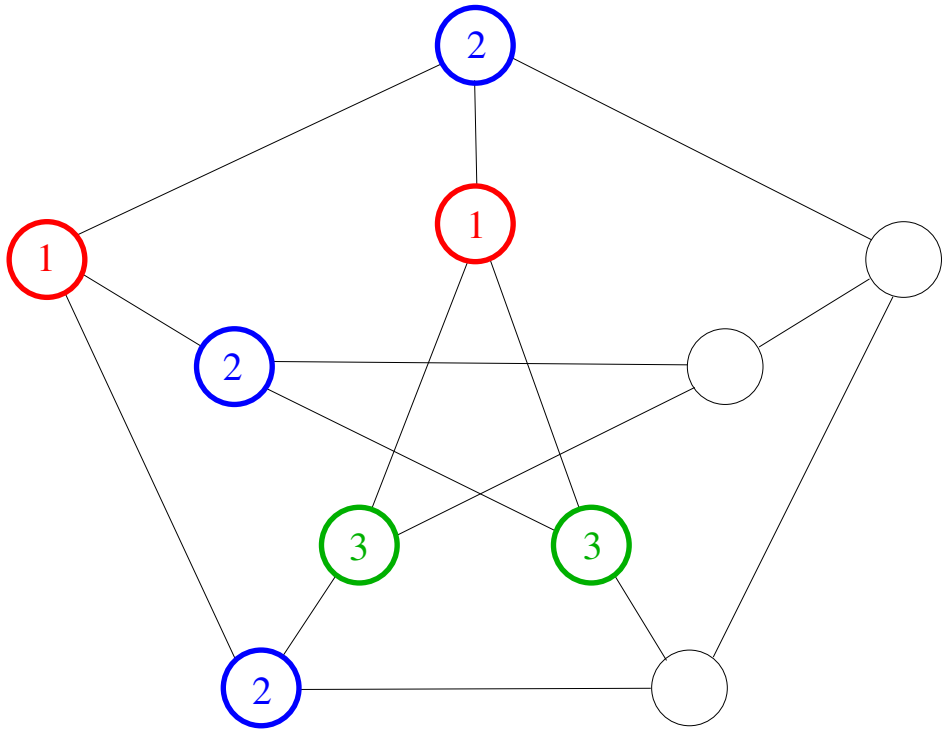
Heuristik för minimal nodfärgning

Färga nästa nod med lägsta möjliga färg.



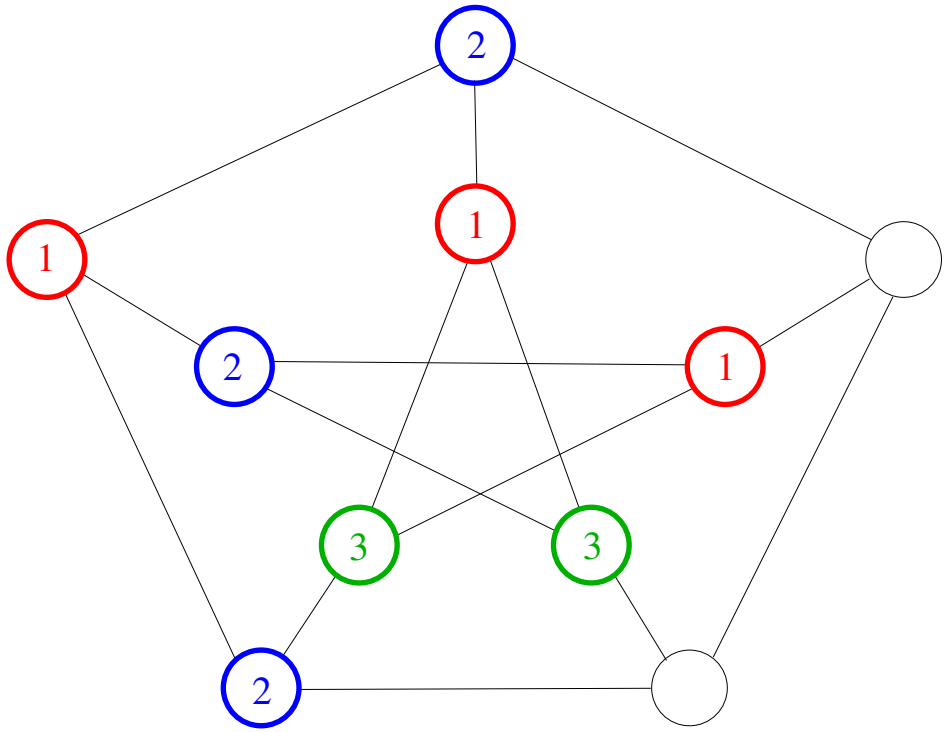
Heuristik för minimal nodfärgning

Färga nästa nod med lägsta möjliga färg.



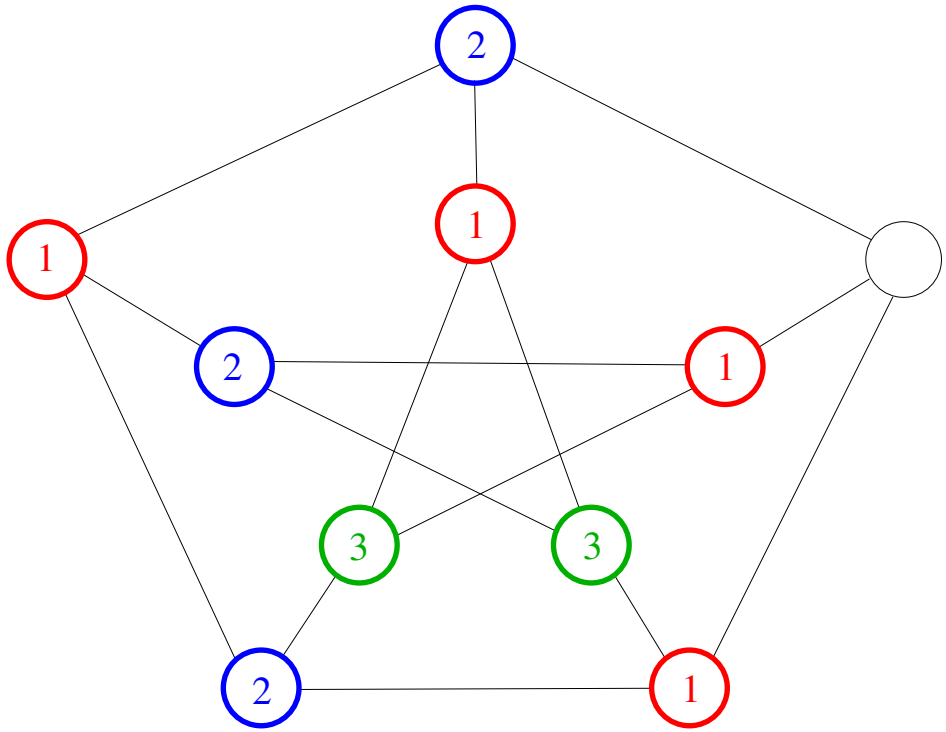
Heuristik för minimal nodfärgning

Färga nästa nod med lägsta möjliga färg.



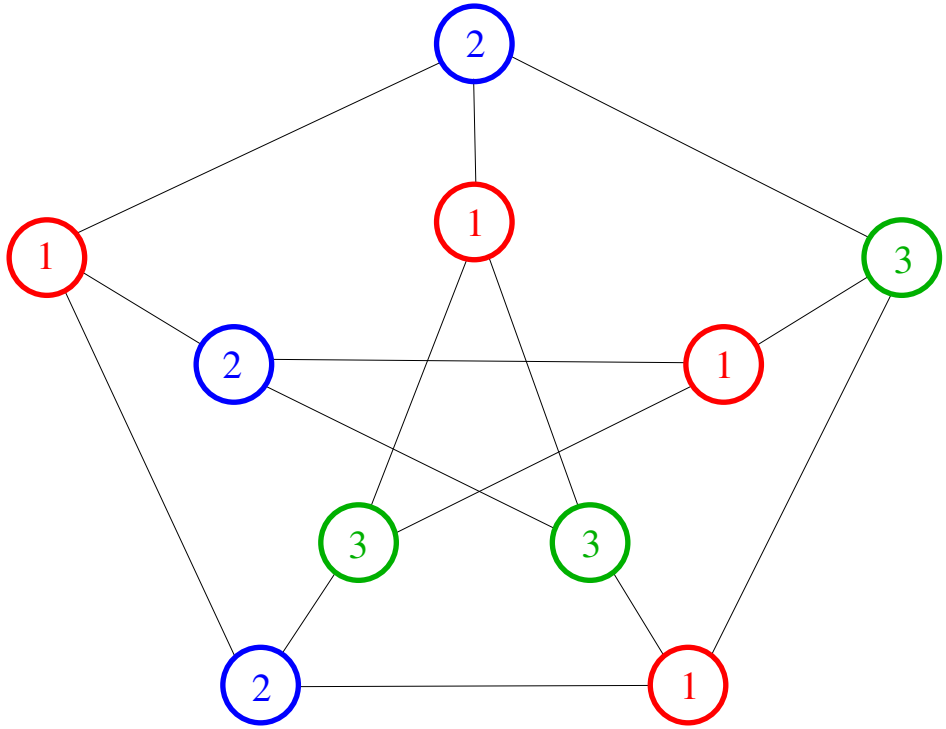
Heuristik för minimal nodfärgning

Färga nästa nod med lägsta möjliga färg.



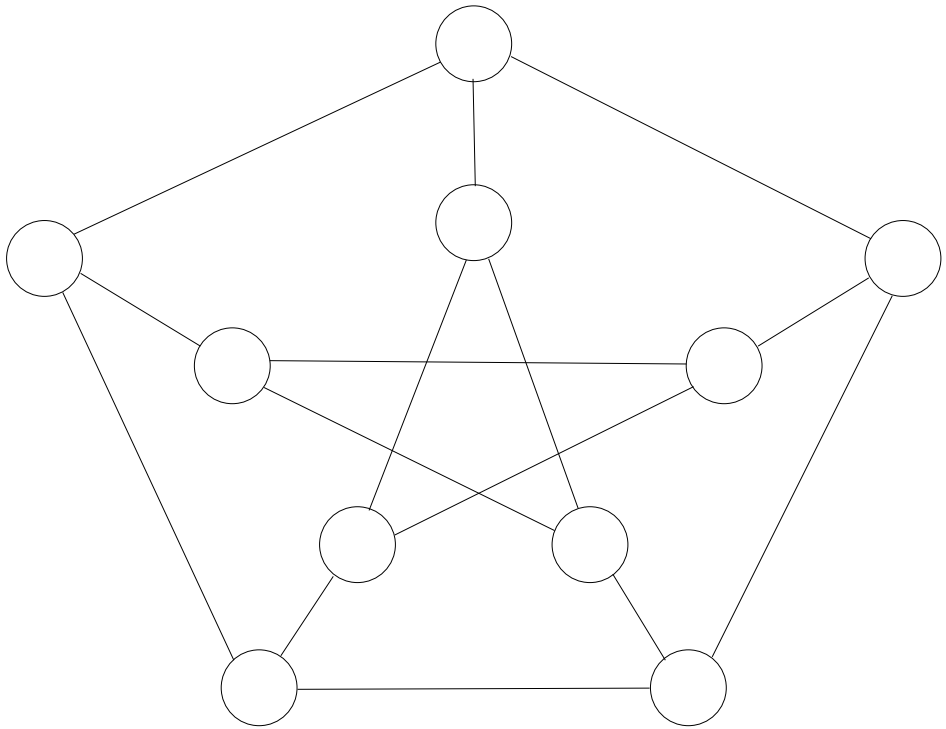
Heuristik för minimal nodfärgning

Färga nästa nod med lägsta möjliga färg.



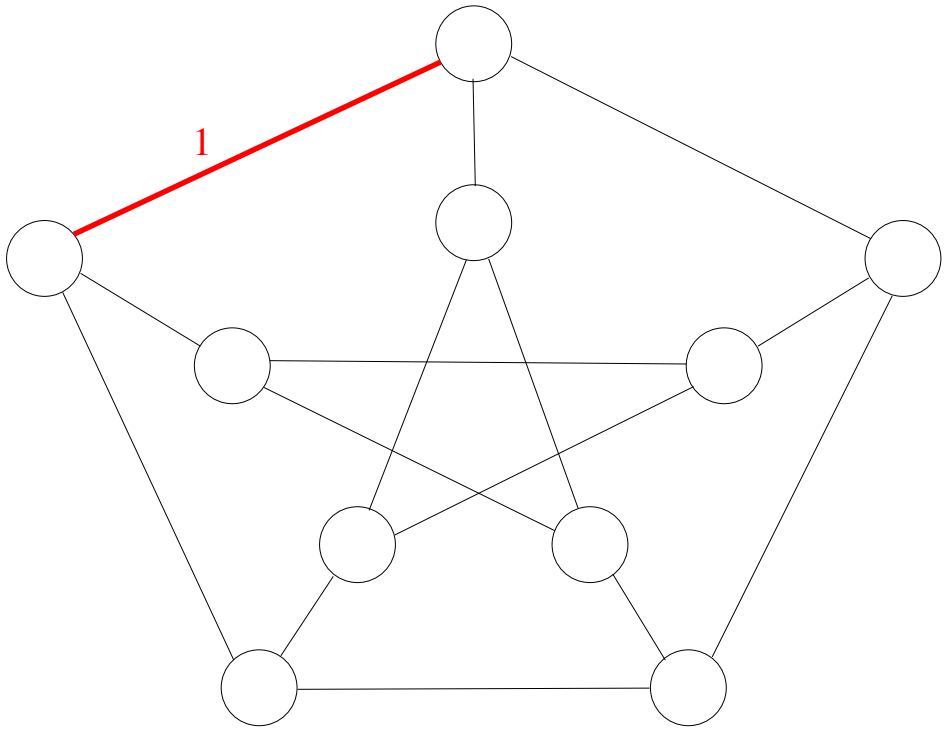
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



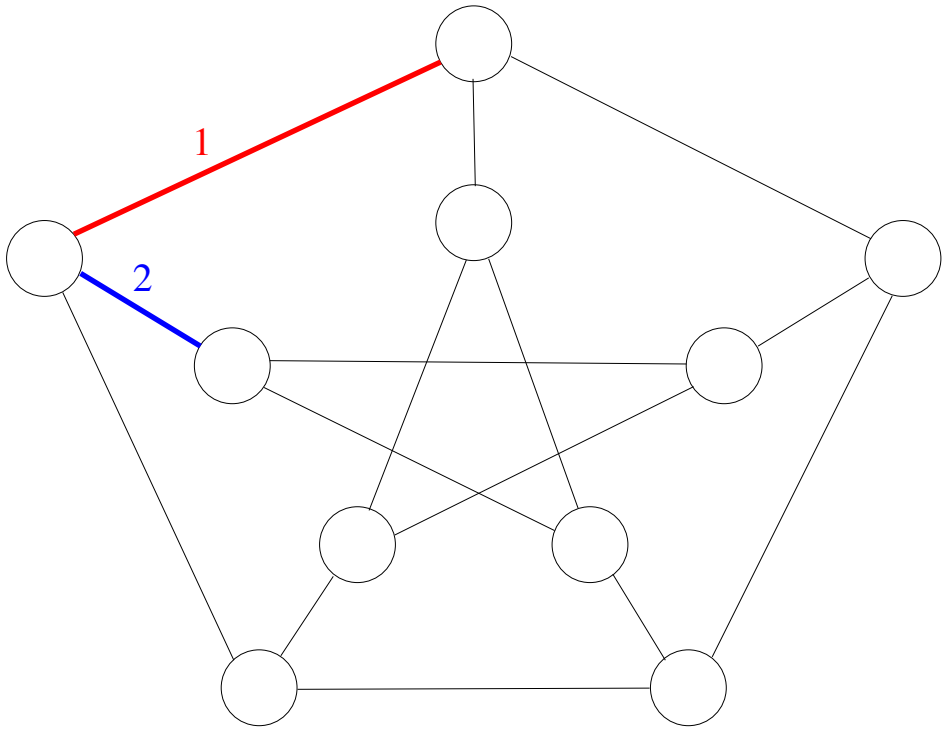
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



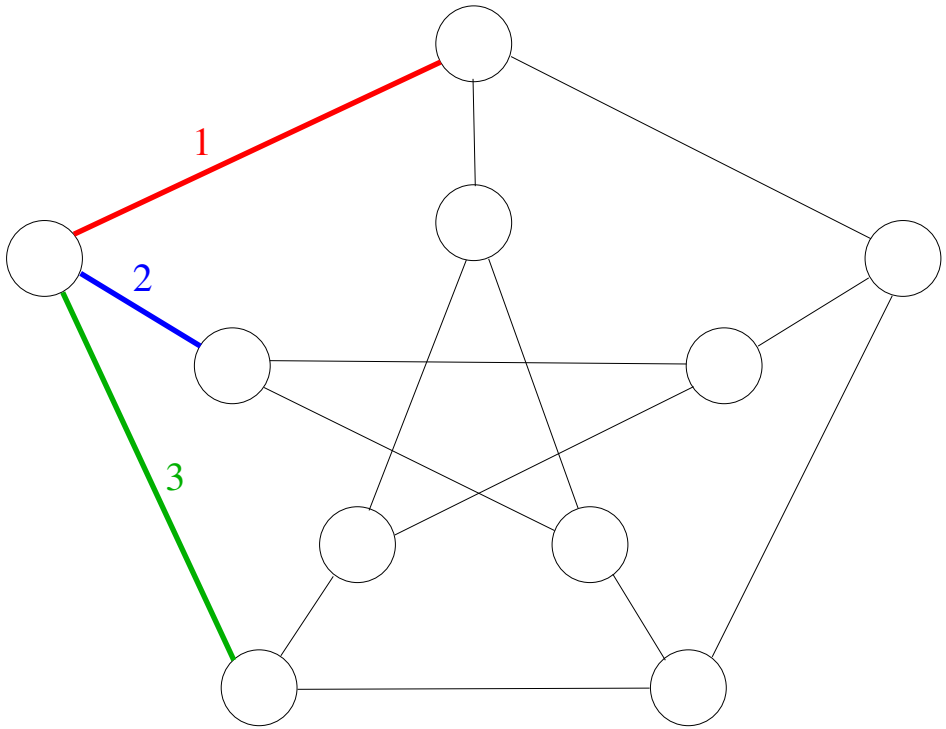
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



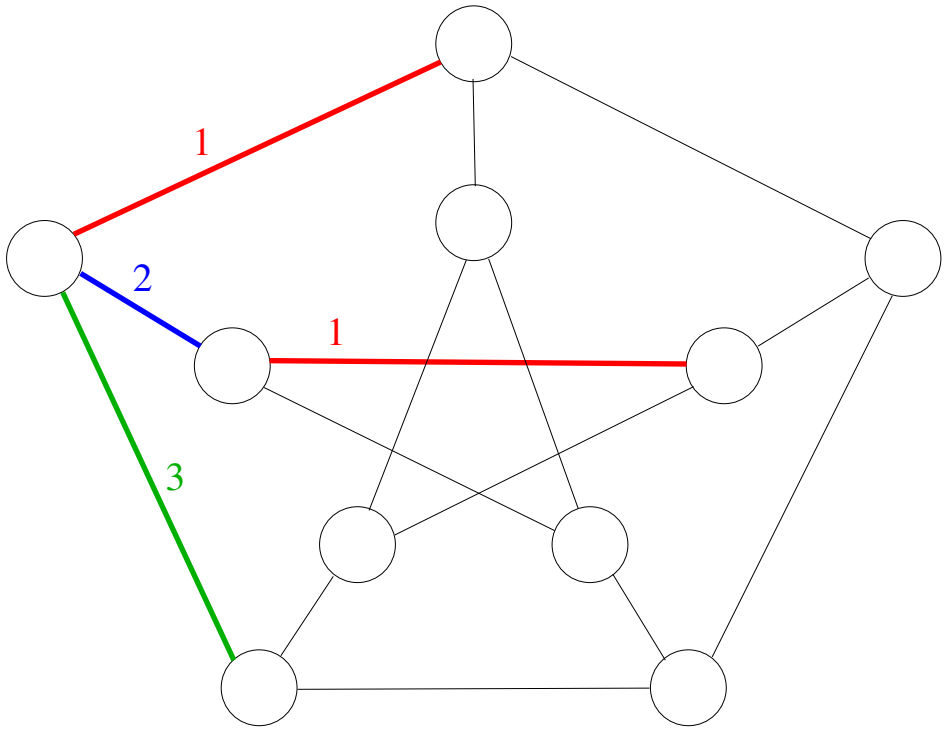
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



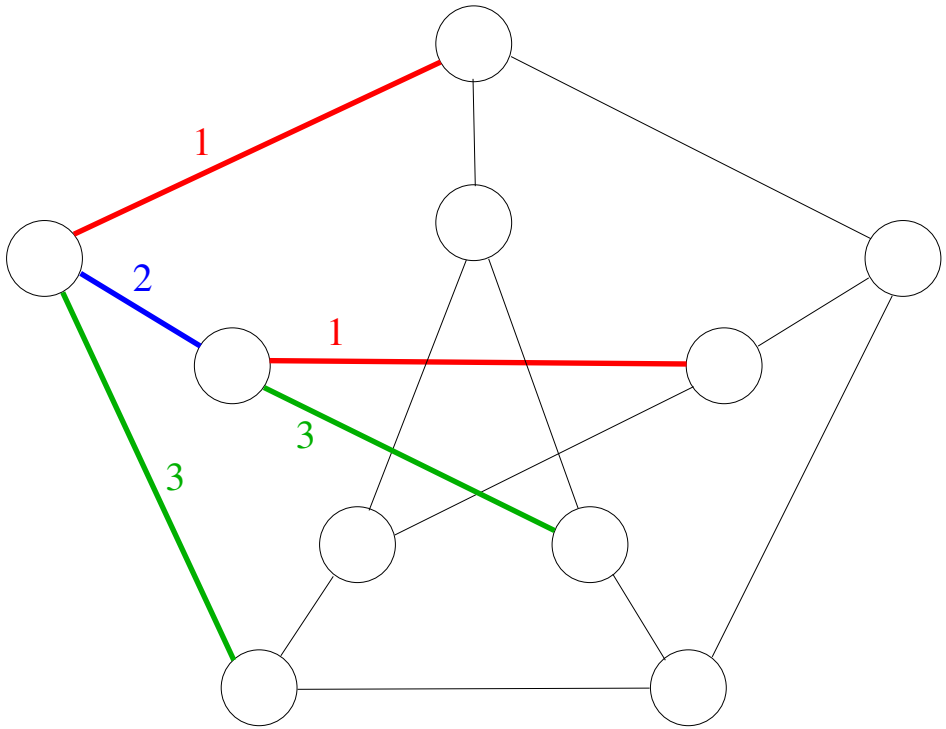
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



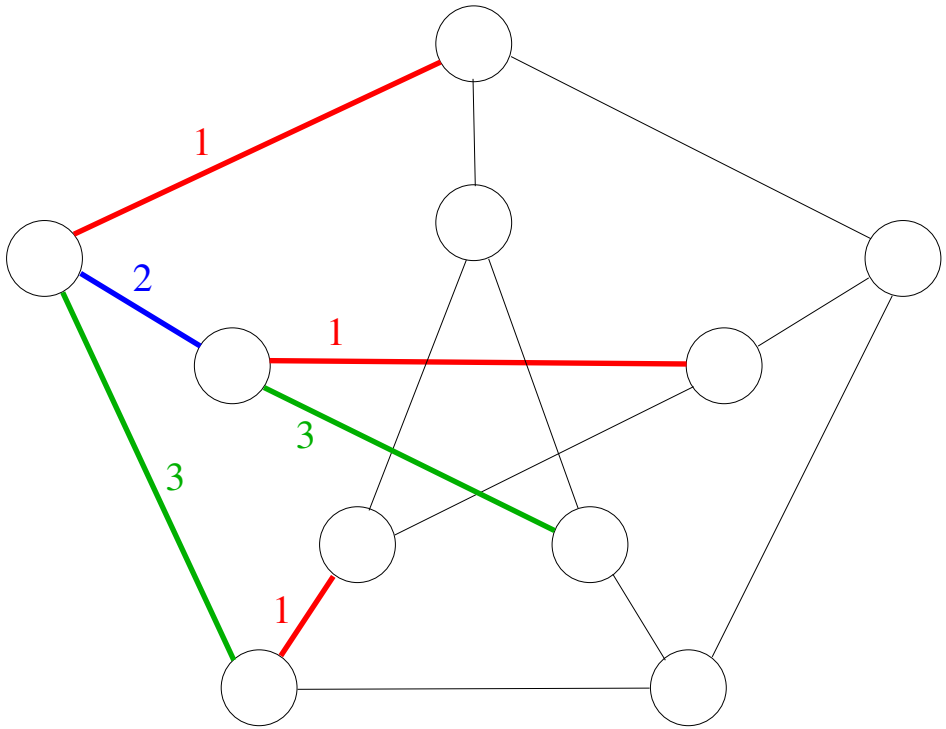
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



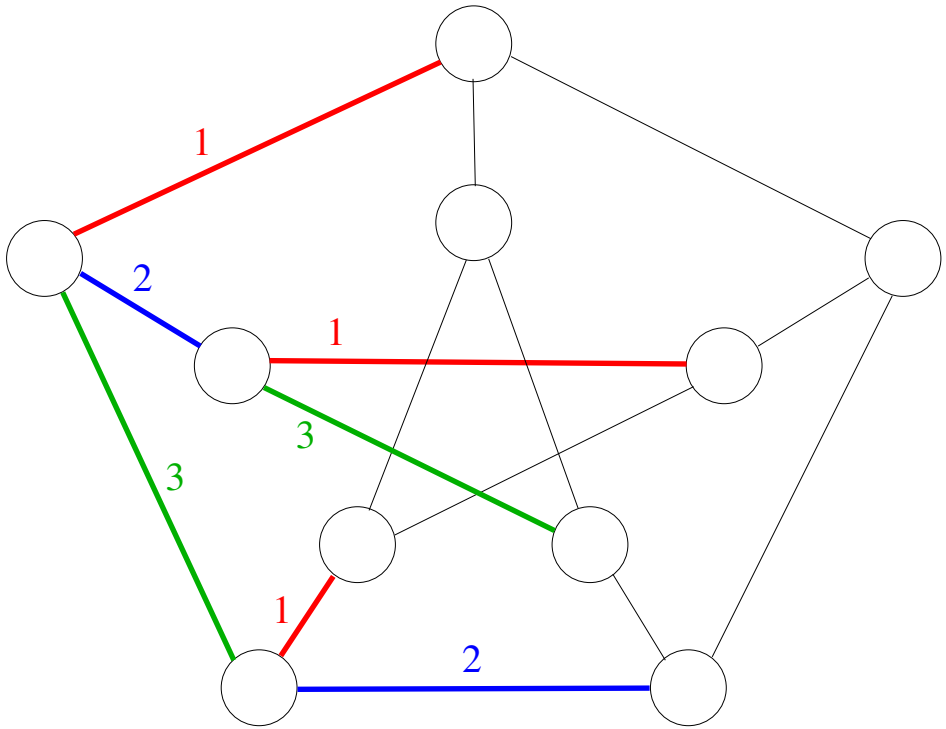
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



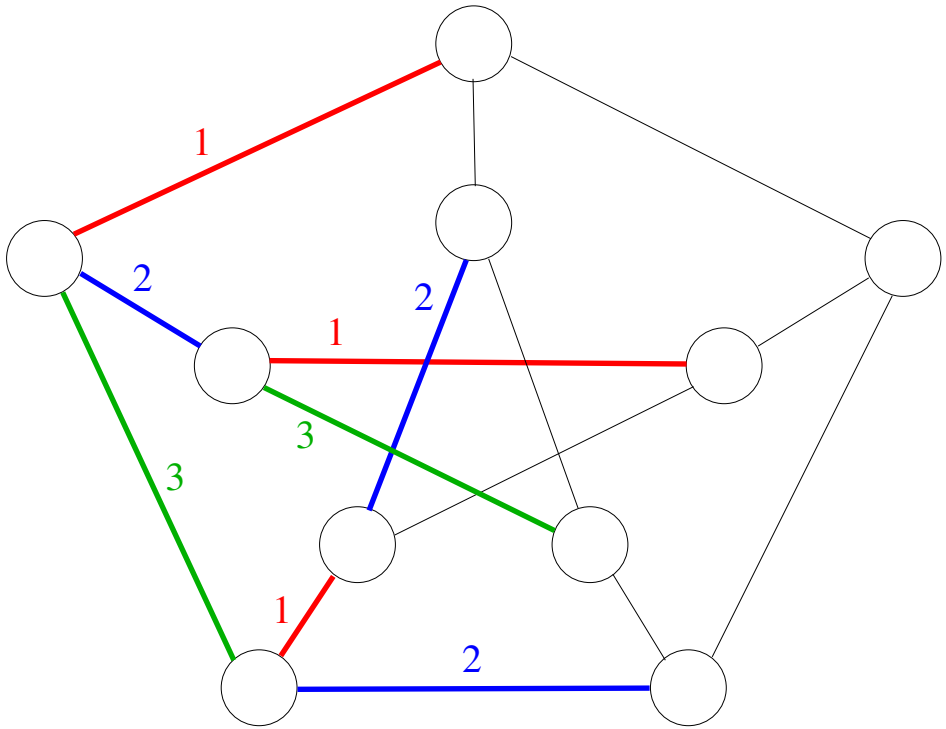
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



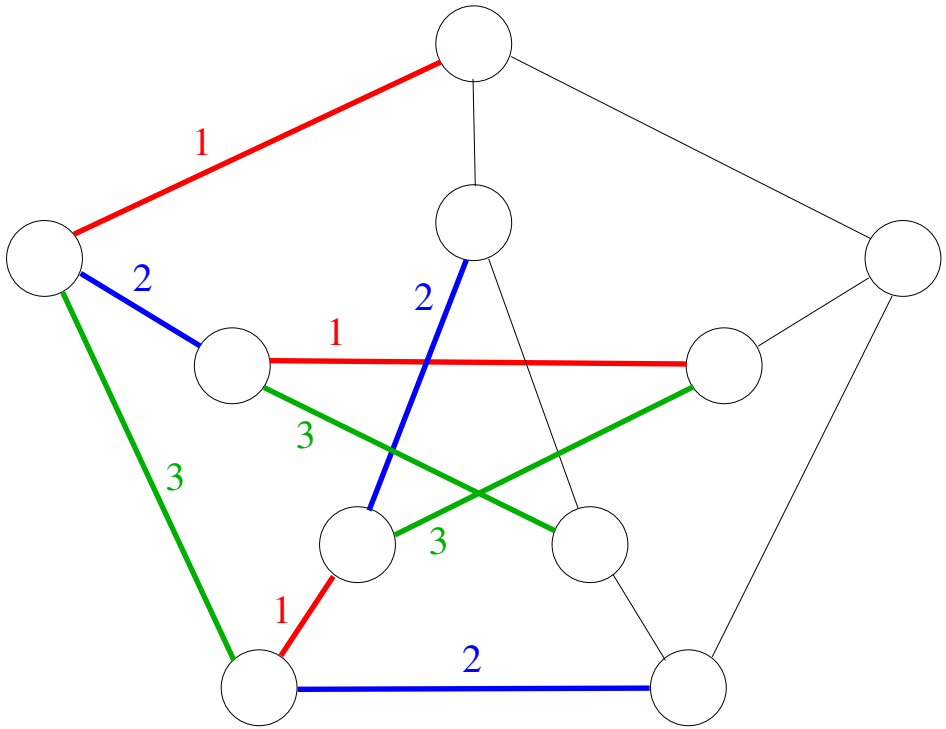
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



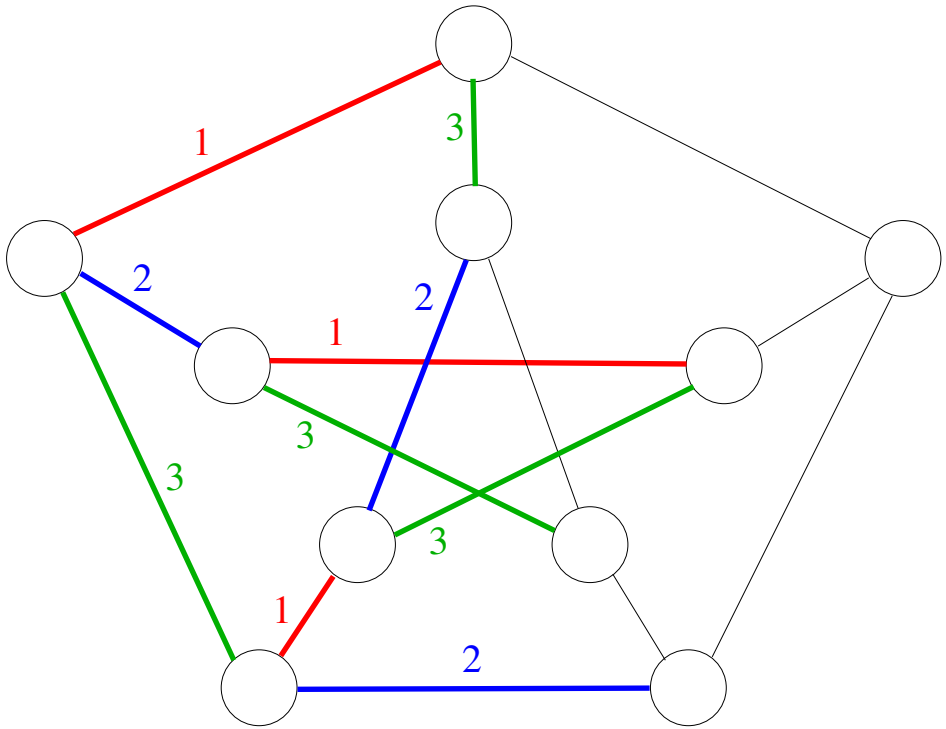
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



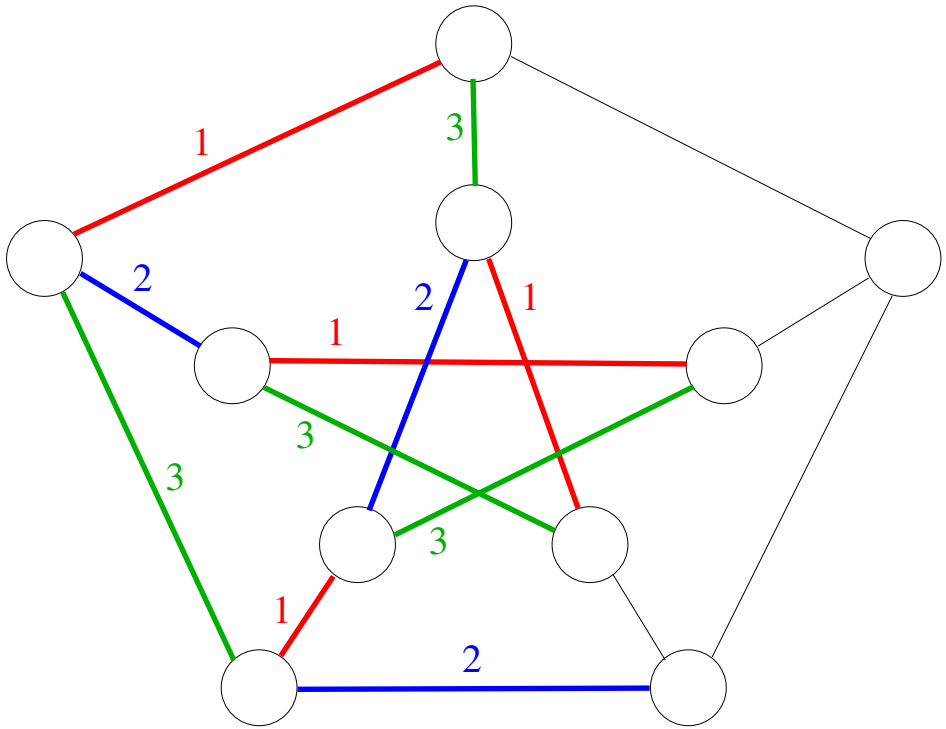
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



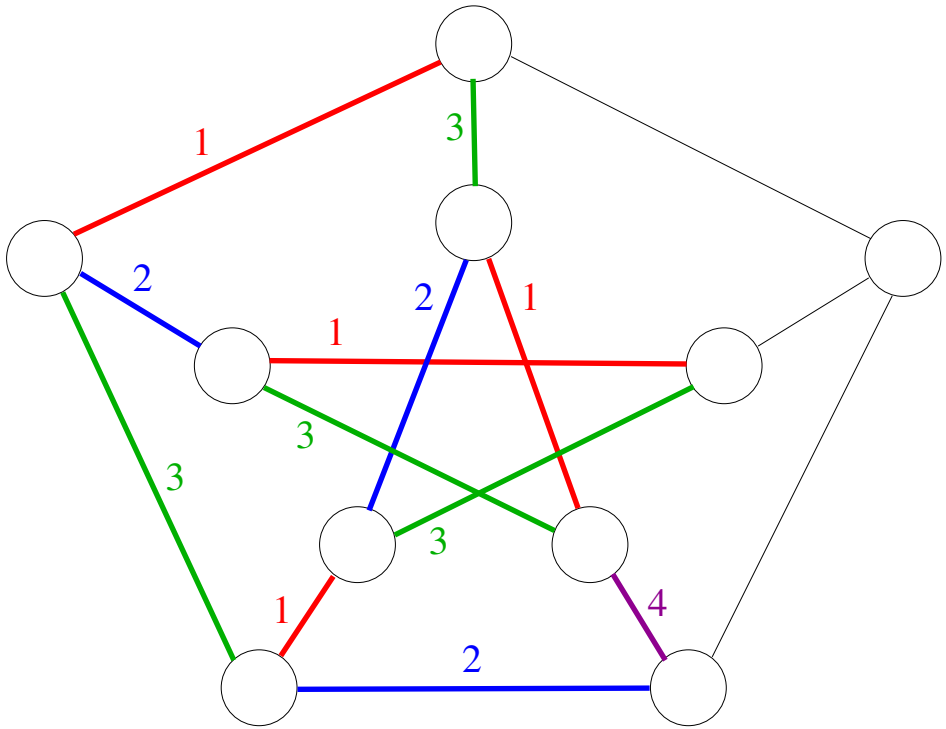
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



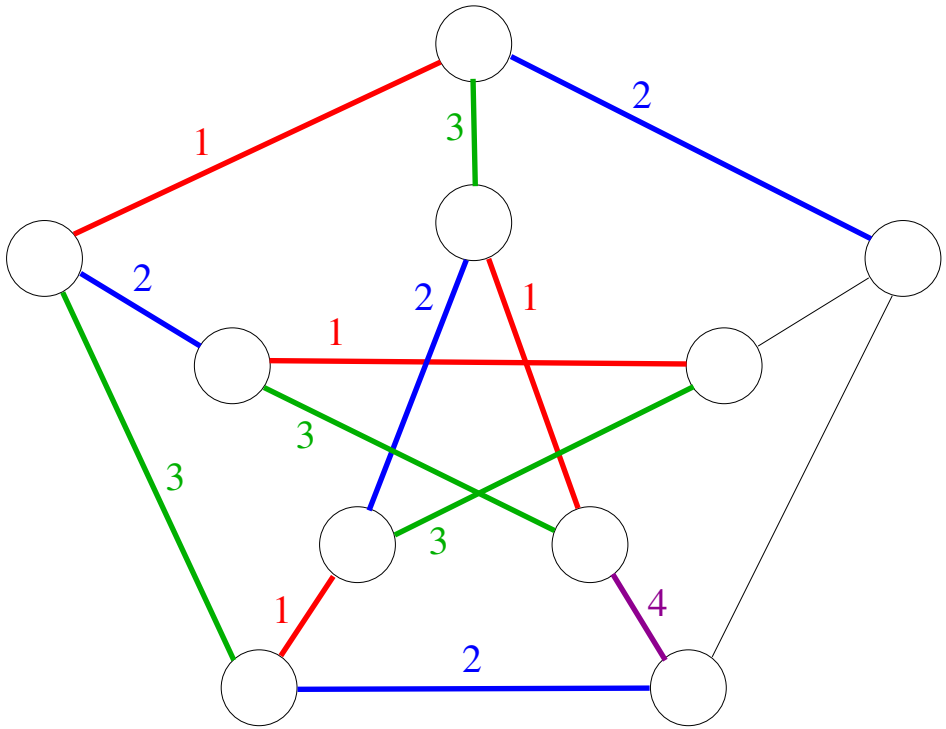
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



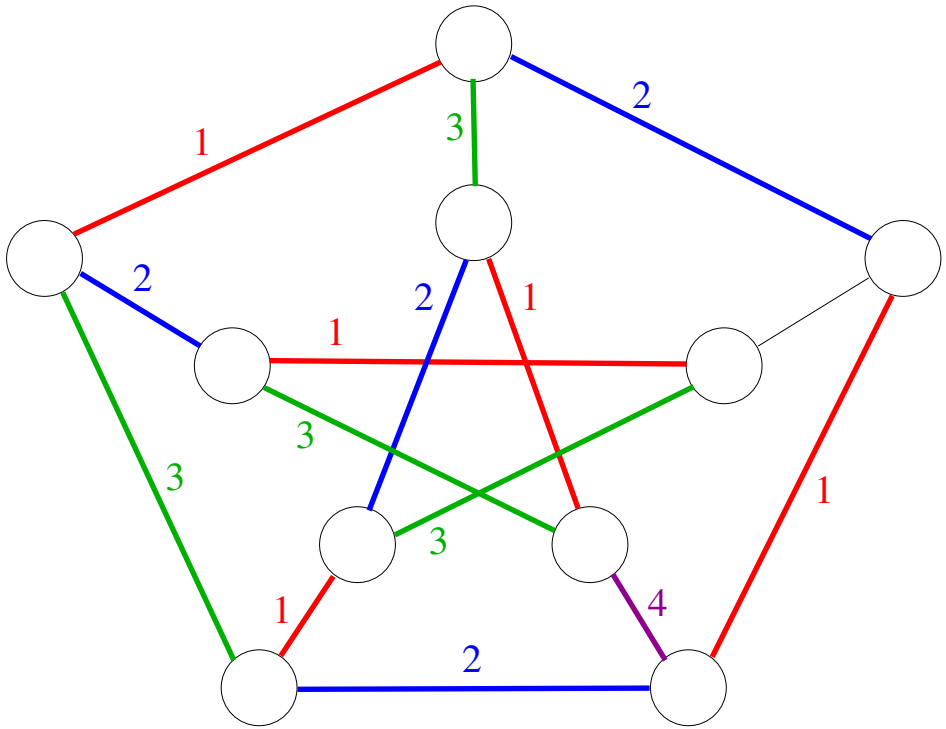
Heuristik för minimal bågfärgning

Färga nästa båge med lägsta möjliga färg.



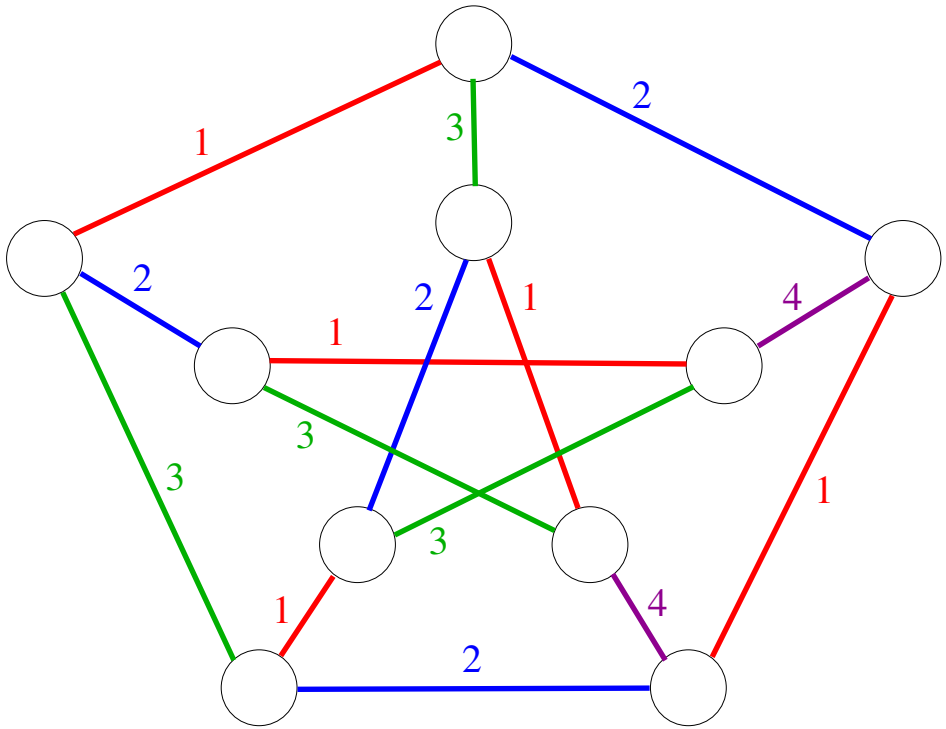
Heuristik för minimal bågfärgning

Färga nästa båge 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.

Lokalsökning

Hoppa från en punkt till en bättre granne. Upprepa.

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

Lokalsökning

Hoppa från en punkt till en bättre granne. Upprepa.

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

Exempel:
- Euklidiskt närmaste punkterna.

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.

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.

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.

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 $k = 0$.

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 $f(\bar{x}) < f(x^{(k)})$.

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.

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

Varianter för bivillkor:

Lokalsökning

Varianter för bivillkor:

- Endast tillåtna punkter.

Lokalsökning

Varianter för bivillkor:

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

Lokalsökning

Varianter för bivillkor:

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

Klättringsstrategier:

Lokalsökning

Varianter för bivillkor:

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

Klättringsstrategier:

- Finn bästa punkten i hela omgivningen.

Lokalsökning

Varianter för bivillkor:

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

Klättringsstrategier:

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

Lokalsökning

Varianter för bivillkor:

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

Klättringsstrategier:

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

Lokalsökning

Varianter för bivillkor:

- Endast tillåtna punkter.
- Även otillåtna punkter. Lägg till straffunktion 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.

Lokalsökning

Omgivning för handelsresandeproblemet:

Lokalsökning

Omgivning för handelsresandeproblemet:

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

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.

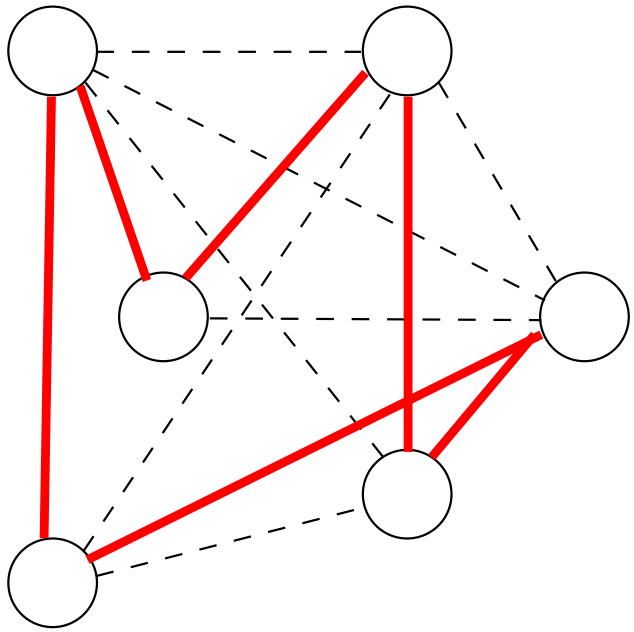
Lokalsökning

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.)



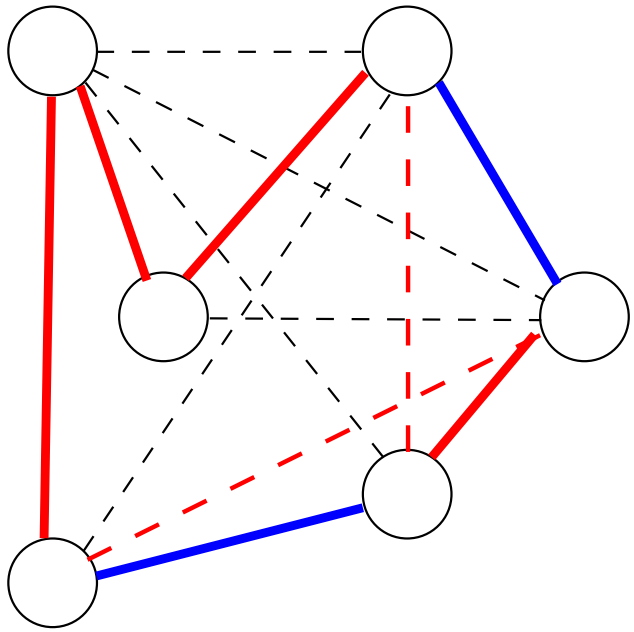
Lokalsökning

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.)



Tabusökning

Tabusökning

Utvidgning av lokalsökning.

Tabusökning

Utvidgning av lokalsökning.

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

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).

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.

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.

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 $\min_{x \in A(x^{(k)})} f(x)$.

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.

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

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

Lokalsökning

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

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

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

Lokalsökning

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

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

Lokalsökning

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

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

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

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

Lokalsökning

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

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

Lokalsökning

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

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

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

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

Lokalsökning

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

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

Lokalsökning

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

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

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

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

Lokalsökning

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

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

Lokalsökning

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

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.

Lokalsökning

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

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

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

Tabusökning

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

Tabulista: Innehåller de två senaste punkterna.
Avbrottskriterium: 4 iterationer
Starta i $x^{(0)} = (0, 3)$ från lokalsökningen, med målfunktionsvärde $z(x^{(0)}) = 9$.

Tabusökning

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

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

Tabusökning

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

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

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

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

Tabusökning

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

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

Tabusökning

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

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

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

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

Tabusökning

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

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

Tabusökning

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

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

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

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

Tabusökning

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

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

Tabusökning

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

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 $z(x^{(3)}) = 10$.

Tabusökning

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

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

Relaxationsmetoder

Relaxera bivillkor, ger lättare problem.

Relaxationsmetoder

Relaxera bivillkor, ger lättare problem.

Försök återskapa relaxerade bivillkor m.h.a. linjära straffkostnader

Relaxationsmetoder

Relaxera bivillkor, ger lättare problem.

Försök återskapa relaxerade bivillkor m.h.a. linjära straffkostnader och heuristiker.

Relaxationsmetoder

Relaxera bivillkor, ger lättare problem.

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

Relaxationsmetoder

Relaxera bivillkor, ger lättare problem.

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

Exempel: Lagrangeheuristik för TSP:

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).

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.

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.

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.

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.

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.