Eulercykel

Definition

En **Eulercykel** är en cykel som använder varje båge exakt en gång.

Definition

En nods **valens** är antalet bågar som ansluter till noden.

Sats

En sammanhängande oriktad graf har en Eulercykel om och endast om alla dess noder har jämn valens.

Kinesiska brevbärarproblemet

En *brevbärartur* är en tur som använder varje båge minst en gång.

Det kinesiska brevbärarproblemet är att finna en brevbärartur med minimal kostnad.

En Eulercykel är en optimal brevbärartur, om den finns.
Den finns om alla noder har jämna valens.

Om inte, får man gå i vissa bågar en extra gång.

Åtminstone i de bågar som går till noder med udda valens.

Metod:

Minimera extraarbetet, dvs. minimera kostnaden för de bågar som används mer än en gång.

Kinesiska brevbärarproblemet: Modell

Matematisk modell:

x_{ij}: antalet gånger båge (i,j) trafikeras.
z_i: antal gånger nod i passeras i turen.

$$\begin{aligned} \min \quad & \sum_i \sum_j c_{ij} x_{ij} & \\ \text{då} \quad & \sum_j (x_{ij} + x_{ji}) - 2z_i = 0 & \text{för alla } i \\ & x_{ij} \geq 1, \text{ heltal} & \text{för alla } i,j \\ & z_i \geq 1, \text{ heltal} & \text{för alla } i \end{aligned}$$

Kinesiska brevbärarproblemet: Metod

Att gå i en båge två gånger kan ses som att gå en gång i bågen och en gång i en dubblett till bågen.
Vi vill alltså dubblera vissa bågar.

Mål: Lägg till bågar mellan noder med udda valens, så att alla noder får jämn valens.

Eulercykeln bildar då en brevbärartur. Om extrabågarna lagts till på billigaste sätt, fås den billigaste turen.

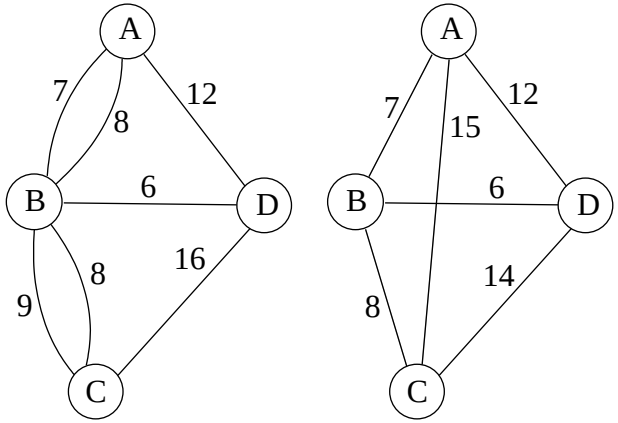
Hur? Förbind noderna med udda valens på billigaste sätt (mha billigaste-väg och minimal perfekt matchning).

Finn Eulercykeln.

Detta är en polynomisk optimerande metod.

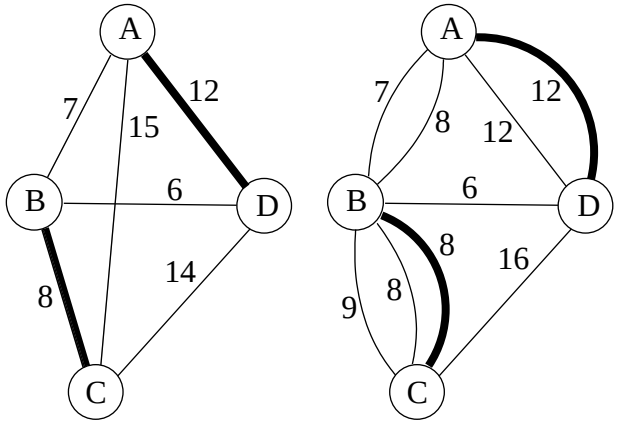
Kinesiska brevbärarproblemet

Exempel: Billigaste vägarna mellan alla noder:



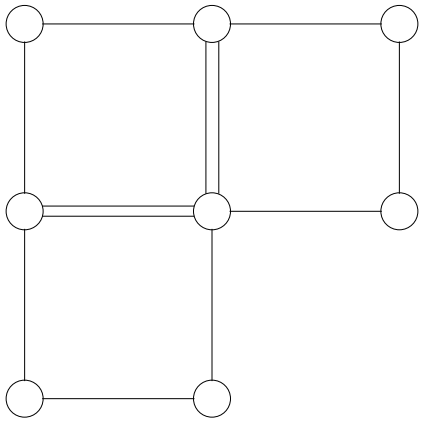
Kinesiska brevbärarproblemet

Addera bågarna i billigaste perfekta matchning.



Finn valfri Eulercykel.

Kinesiska brevbärarproblemet: Exempel



Beräkna valens. Udda noder. Finn billigaste matchning. Dubblera bågarna. Finn Eulertur.

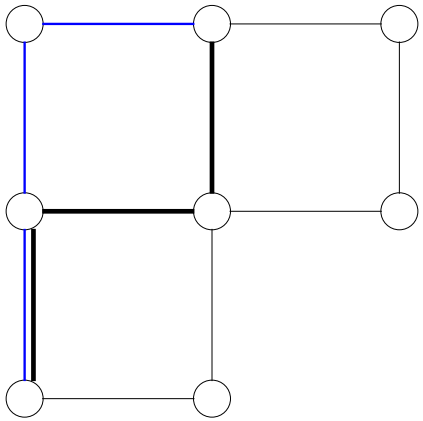
Lantbrevbärarproblemet

Brevbäraren behöver bara täcka vissa av bågarna (kallas nödvändiga, "required").

Lantbrevbäraren åker mellan småorter och delar ut post,
men vägarna mellan orterna används bara för transport.

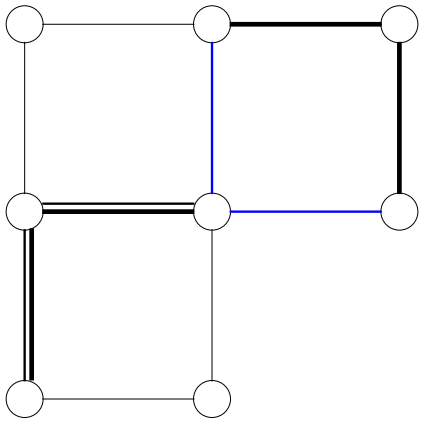
Kan vara lika lätt som kinesiska brevbärarproblemet,
men kan också vara betydligt svårare.

Lantbrevbärarproblemet: Exempel



Udda noder. Finn billigaste matchning. Dubblera nödvändiga bågar.
Eulertur.

Lantbrevbärarproblemet: Exempel 2



Ej sammanhängade. Måste koppla ihop.

K-brevbärarproblemet

Flera fordon delar på uppgiften att täcka alla bågar.

Vilket fordon ska täcka vilken båge?

Hur ska fordonen köra?

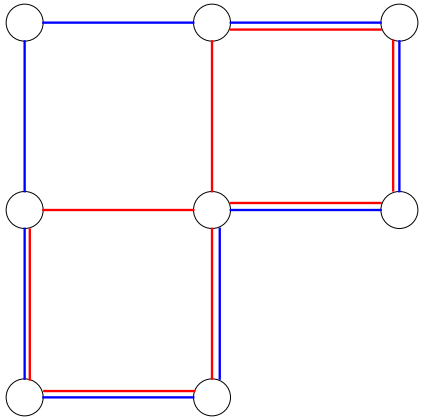
Gör allokeringen av bågar till fordon först.

Bestäm sedan hur fordonen ska köra
genom att lösa ett lantbrevbärarproblem för varje fordon.

Summera ihop kostnaderna.

Göra ev. om allokeringen och upprepa.

K-brevbärarproblemet: Exempel



En allokering. Röd: 5, blå: 5. Rundturer. Röd: 6, blå: 6.
En annan allokering. Röd: 4, blå: 6. Turer. Röd: 4, blå: 8.
En sämre allokering. Röd: 5, blå: 5. Blå rundtur: 8. Röd rundtur: 8.

Snöröjning

Vad kostar det?

Hur lång tid tar det?

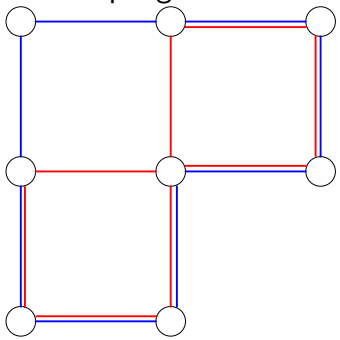
Kostnaden blir summan av kostnaderna för turerna,
plus fasta kostnader för varje fordon.

Tiden bestäms av det fordon som tar längst tid,
dvs. maximum av fordonens tider.

Sedan kan man addera tiden till kostnaden med lämplig vikt
för att få en siffra att jämföra.

Snöröjning: Exempel

Enkelt exempel: Varje båge kostar 1 (i både tid och pengar). Lika vikt på tid och pengar. Fast kostnad: 1.



En gör allt: Kostnad: 12+1. Tid: 12. Tot: 25.
Allok 1: Kostnad: 6+6+2. Tid: 6. Tot: 20.
Allok 2: Kostnad: 8+4+2. Tid: 8. Tot: 22.
Allok 3: Kostnad: 8+8+2. Tid: 8. Tot: 26.

Lokalsökning

Hoppa från en punkt till en bättre granne. Upprepa.

Definiera **omgivning** N (närliggande punkter, grannar).

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:

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

Klättringsstrategier:

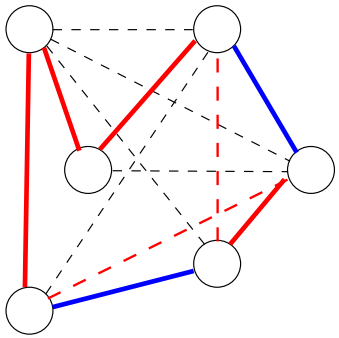
- Finn bästa punkten i hela omgivningen.
- Finn bästa punkten i delmängd av omgivningen.
- Finn första bättre punkten.
- Finn första bättre punkten i delmängd av omgivningen.

Lokalsökning: Omgivning

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

Exempel: Binärt kappsäcksproblem:

En lösningen är en vektor med nollor och ettor.

Möjliga ändringar:
Sätt en variabel till ett.
Sätt en variabel till noll.
Byt (dvs. sätt en variabel till noll och en annan till ett).

Den sista kan uppnås genom de två första,
men genom att göra det i ett steg, slipper man
otillåtenhet eller betydligt sämre lösning.

Lokalsökning: Omgivning

Exempel: Anläggningslokalisering:

Vi ska bygga en eller flera fabriker, som ska producera och transportera varor till kunder.

Två beslut: Vilka fabriker ska byggas? Hur ska varorna skickas?

Det första beslutet är viktigast.
Om vi vet vilka fabriker som ska byggas, återstår bara ett transportproblem (minkostnadsflödesproblem).

Omgivningar för fabriker:
Öppna en fabrik till.
Stäng en fabrik.
Byt (dvs. stäng en fabrik och öppna en annan).

Den sista kan uppnås genom de två första, men genom att göra det i ett steg, slipper man jämföra med målfunktionsvärdet i mellansteget.

Lokalsökning: Omgivning

Exempel: Ruttplanering:
Ett visst antal bilar ska förse kunder med varor.
Varje bil ska köra en rundtur (som börjar och slutar i en depot).

Om vi vet vilka kunder varje bil ska besöka, kan vi lösa ett handelsresandeproblem för varje bil, för att bestämma hur den ska köra.

Den övergripande optimeringen gäller vilka kunder varje bil ska besöka.

En möjlig omgivning är att flytta en kund från en bil till en annan.
(Man kan inte bara ta bort en kund från en bil, för då blir kunden utan.)
Man kan också flytta fler kunder från en bil till en annan.
Men ofta vill man då flytta en sammanhängade deltur.

Metaheuristiker

Utvidgning av lokalsökning.

Lokalsökning hittar ett lokalt optimum med hänsyn till den omgivning som används.

(Lokalsökning är alltså rationell och smart.)
Men ofta är detta lokala optimum inte speciellt bra (globalt sett).

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

Då kan man klättra över lokala max för att komma till bättre min.

Men man måste undvika cykling,
dvs. att man återvänder till en punkt man redan varit i.

Det finns olika sätt att uppnå detta.

Tabusökning

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

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.

Simulerad kylning

(Härma fysikaliskt förlopp: Långsam kylning av metall.)

Acceptera försämring med viss sannolikhet, $e^{-\Delta/T}$, där T är "temperaturen" som långsamt minskas.

1. Välj en startpunkt, $x^{(0)} \in X$. Sätt $k = 0$.
Välj en starttemperatur, T, och en kylfaktor, $r : 0 < r < 1$.
2. Sätt $l = 0$.
    3. Välj en slumpmässig granne, $\bar{x} \in N(x^{(k)})$.
    4. Beräkna ändringen $\Delta = f(\bar{x}) - f(x^{(k)})$.
    5. Om $\Delta \leq 0$ (bättre), sätt $x^{(k+1)} = \bar{x}$. Gå till 3.
    6. Om $\Delta > 0$ (sämre), sätt $x^{(k+1)} = \bar{x}$ med sannolikheten $e^{-\Delta/T}$.
    7. Sätt $l = l + 1$. Om $l \leq L$, gå till 3.
8. Reducera temperaturen: Sätt $T = rT$.
9. Stoppa om systemet är "fruset". Annars sätt $k = k + 1$, $l = 0$, gå till 2.

Tillägg

Det ideala stoppkriteriet är att stanna när ingen förflyttning kan ske,
dvs. när man hittat ett lokalt optimum,
och sänkt temperaturen så att man inte kan ta sig därifrån.

Men det kan ta lång tid att uppfylla.

Därför använder man oftast en maxgräns på antalet iterationer, I.

Eftersom man kan gå till sämre punkter, bör man komma ihåg den bästa man har funnit,
så att man kan plocka fram den när man har slutat.

Parametrar

Metoder som simulerad kylning har parametrar som påverkar effektiviteten mycket:

Starttemperaturen, T.

Kylfaktorn, r.

Antalet iterationer med samma temperatur, L.

Maximalt total antal iterationer, I.

Tanken är att metoden ska leta sig in i ett bra område

innan man börjar sänka sannolikheten att acceptera sämre lösningar.

Om man sänker temperaturen för snabbt, ökar risken att man hamnar i (dåliga) lokala optima.

Om man sänker temperaturen för långsamt, hoppar man omkring onödigt mycket.

Omgivning

Det kanske viktigaste för dessa metoder är hur man definierar sin omgivning.

En dålig omgivning kan göra det svårt att hitta bättre punkter.

Vissa punkter kanske inte kan nås alls.

Man kanske behöver gå via många dåliga punkter för att hitta en bättre.

Det är svårt för dessa metoder.

En stor omgivning ger många grannar.

Det ökar risken att man går till sämre grannar, även om det finns många bättre.

Det spelar också roll hur lång tid det tar att beräkna ett nytt målfunktionsvärde.

Har man tid att göra 10 000 iterationer?

Omgivning

Hur rationell (smart) får man vara?

Kom ihåg att metaheuristiker ska vara en förbättring av lokalsökning.

Om man är för smart, blir det kanske bara lokalsökning.

Dvs. man fastnar i första lokala optimum.

Därför måste stegen innehålla en hel del slump.

Det är slumpen (dåliga steg) som hjälper oss bort från lokala optima.

Kombinationer

Ofta är det bra att använda kombinationer av olika metoder.

Exempelvis kan man lägga till förbättringsfas, som försöker (mycket rationellt) förbättra nuvarande punkt.

Lyckas man, sparar man den, men går inte dit.

Lyckas man inte, struntar man bara i den.

Man kan göra vad som helst, bara man inte stör metodens iterationssekvens.

Detta gäller även Lagrangeheuristiker.

Kombinationer

Ett sätt att undvika nackdelen med en viss omgivning
är att byta omgivning då och då.

I metoden VNS ("variable neighborhood search") byter man omgivning med jämna mellanrum.

Om man t.ex. har hittat på tre omgivningar, men ingen av dem verkar ensam ge bra konvergens,
kan man byta mellan dessa tre,
och hoppas att få fördelarna med alla tre.

I tabusökning kan man byta omgivning, om ingen granne är bättre.

Ett lokalt optimum i en omgivning är kanske inte lokalt optimum i en annan omgivning.

Det globala optimat är dock lokalt optimum i alla omgivningar.

Relaxerad lokalsökning

Ibland kan man göra en lokalsökning utan att beräkna exakt målfunktionsvärde.

Exempel: Ruttplanering: Ett visst antal bilar ska förse kunder med varor.
Om vi vet vilka kunder varje bil ska besöka, kan vi lösa ett handelsresandeproblem för varje bil, för att bestämma hur den ska köra.

Den övergripande optimeringen gäller vilka kunder varje bil ska besöka.

En möjlig omgivning är att flytta en kund från en bil till en annan.
Man kan då beräkna "besparingen" som fås om man tar bort en kund från en tur och lägger till den till en annan.

Detta ger ett (approximativt) mått på förändringen av målfunktionsvärdet, och kan användas för att finna en bra uppdelning.

Till slut bör man dock lösa handelsresandeproblemen för att se hur det egentligen blev.

Snö

För snöröjningsproblemet kan följande ändringar göras:

En slumpmässig båge flyttas från ett fordon till ett annat. (Ofta dumt...)

En cykel flyttas från ett fordon till ett annat.

Speciellt om cykeln går genom en nod som det andra fordonet passerar.

Ingen ändring av totalkostnaden, men maxtiden kan minskas.

(Fler möjligheter i projektet.)