INF-MAT-5380

http://www.uio.no/studier/emner/matnat/ifi/INF-MAT5380/

Leksjon 7

GA - Oppsummering

- Viktige karakteristika
  - populasjon av løsninger
  - domeneuavhengighet – enkoding
  - mangel på utnyttelse av struktur
  - iboende parallellitet – skjema, vokabular
  - robusthet
  - gode mekanismer for intensifisering
  - mangler noe diversifisering
- Hybrid med lokalsøk: Memetic algorithms

Nye trender
GA- Evolusjonsmetoder

- Mer direkte representasjon av løsninger
  - mer naturlig enkoding av løsning
  - spesielle krysnings- og mutasjonsoperatorer
- Mer intensifisering
  - lokalsøk til lokalt optimum (spesielle mutasjonsoperatorer)
  - Memetiske algoritmer

Maurkolonioptimering for DOP - Ant Colony Optimization (ACO)

- Introdusert av Dorigo, Maniezzo & Colorni 1992
- Populasjonsbasert metaheuristikk
- Hver „maur“ i populasjonen **konstruerer** en løsning
- Når alle er ferdige oppdateres et **minne** (kunstig feromon)
- Løsningskonstruksjon og minneoppdatering repeteres inntil stoppkriterium er oppfylt

Maursystem (forts.)

- Virker bare sånn passe ...
- Biologisk analogi delvis forlatt
- 3 videreutviklinger
  - Ant Colony System – (ACS, Maurkolonioptimering)
  - Max-Min Ant System
  - Rank Based Ant System
- Alle inkluderer lokalsøk for å forbedre løsning
- Ant Colony System
  - Modifisert global og lokal oppdatering
  - Globalt og lokalt feromon
  - Elitisme: kun „beste maur“ får oppdatere globalt feromon
  - Endret randomisering, sannsynlighet i to trinn
    - velge grådig
    - velge probabilistisk blant alle tillatte alternativer

Leksjon 6 - Oppsummering

- Genetiske algoritmer (GA)
  - inspirert av biologisk evolusjon
  - populasjon av løsninger (kromosomer)
  - rekombinering ved krysning
  - mutasjon, seleksjon
- Øvrige populasjonsbaserte metoder
  - Memetiske algoritmer (”GA+lokalsøk”)
  - Maurkolonioptimering

Leksjon 7 - Oversikt

- Konstruktive heuristikker
- Multi-start baserte metaheuristikker
  - Variabelt nabolagssøk (VND/VNS)
  - Grådig Adaptivt Randomisert Søk (Greedy Randomized Adaptive Search, GRASP)
  - Iterert lokalsøk (Iterated Local Search, ILS)
- Kategorisering av metaheuristikker
- GUT (Grand Unifying Theory)?
- Hyperheuristikker

Konstruktive heuristikker

- Grådig konstruksjon
  - grad av lokalitet i beslutningene
  - informerte valg vs. regnetid
- Rekkefølge-basert konstruksjon
  - Kritikalitet
  - Statisk vs. dynamisk evaluering
- Parallell vs. sekvensiell konstruksjon
- Sæd (seeds)
- Kombinasjon med søk
  - lokalsøk
  - systematisk
- ”Squeaky Wheel” optimering

Enkleste multi-start metode

- Tilfeldig restart (Random Restart, RR)
- Lokalsøk (f. eks. med Steepest Descent)
- Iterasjon
- Tilfeldig startløsning

- Når virker RR bra?

Alternativ pseudokode

Lokalsøk med ”Steepest Descent”

```
Procedure Local_Search_SD(init_sol,N,f)
current:=init_sol
new_current:=Best_Neighbor(current,N,f)
*/ Assuming Best_Neighbor returns current
*/ if there is no improving move
while not f(new_current)=f(current) do
    current:=new_current
    new_current:= Best_Neighbor(current,N,f)
od
return current                    ; Local optimum
```

Tilfeldig restart (Random Restart – RR)

```
Procedure Random_Restart (S,N,f,Stop_Criterion)
current=Init_Solution(S)
incumbent:=current            */ best solution until now
while not Stop_Criterion() do
        local_optimum:=Local_Search_SD(current,N,f)
        if f(local_optimum) < f(incumbent) then
                incumbent:= local_optimum
        fi
        current:=Random_Init_Solution(S)
od
return incumbent     */ best solution until now
```

GRASP
Greedy Randomized Adaptive Search

- Variant av Tilfeldig restart (RR)
- Konstruksjon med tilfeldighet
- Ligner Maurkoloni-optimering (ACO)
- Begrenset kandidatliste for utvidelse av del-løsning

GRASP

```
Procedure GRASP (Max_Iterations)
incumbent:=Bad_Solution()
for k:=1 to Max_Iterations do
        current:=Local_Search(.....
                        Greedy_Randomized_Construction(...))
        if f(current) < f(incumbent) then
                incumbent:= current
        fi
od
return incumbent    */ best solution until now
```

GRASP – Randomized Construction

```
Procedure Greedy_Randomized_Construction(...)
solution := Empty_Solution()
while not Complete(solution) do
    Restricted_Candidate_List
                    :=Evaluate_Incremental_Costs(solution)
    partial_solution:=Extend_Solution(solution,Random_Elem
    ent(Restricted_Candidate_List))
od
return solution
```

GRASP
– Restricted Candidate List

- Restriksjon kan være
  - Kardinalitetsbasert (”de 10 beste alternativene”)
  - Verdibasert (”alle elementer med inkrementell kostnad ikke større enn gitt verdi”)

$$c(e) \in \left[c^{min}, c^{min} + \alpha\left(c^{max} - c^{min}\right)\right], \quad \alpha \in [0,1]$$

- Reaktiv GRASP: automatisk justering
- Perturbasjon av kostnadsfunksjon (noising)
- Utvidelse med Path Relinking
- Preprosessor til GA

”Squeaky Wheel Optimization”

- I en pipende maskin smøres først de deler som piper
- Basert på konstruktiv heuristikk der
  - Løsning bygges opp ved suksessiv utvidelse med elementer
  - Sekvens av elementer er viktig (prioritet av elementer)
  - Mål på hvor ”fornøyd” et element er i den endelige løsning
- Endring av rekkefølgen av elementer
- Repetisjon

Squeaky Wheel Optimization

```
Procedure Squeaky_Wheel_Optimization(f)
Element_Priority_List
    :=Determine_Element_Priorities() ; Static prioritization
incumbent:=Some_Feasible_Solution()
while not Stop_Criterion() do
    solution:= Empty_Solution()
    while not Complete(solution) do
          solution:=Augment_Solution(solution,
                                          Element_Priority_List)
    od
    if f(solution)< f(incumbent) then incumbent:=solution
     Element_Priority_List
          :=Update_Element_Priorities(Element_Priority_List,solution)
od
return solution
```

Variabelt nabolagssøk (VNS)

- Lokalt optimum er relativt til nabolag
- Et lokalt optimum m.h.t. ett nabolag er ikke nødvendigvis lokalt optimum m.h.t et annet
- Et globalt optimum er et lokalt optimum for alle nabolag
- Lokale optima er ofte nær hverandre
- Grunnleggende idé i VNS:
  - Systematisk variasjon av nabolag
- Nabolagsstruktur $N_k, \quad k = 1, \ldots k_{max}$
- Gjerne ordnet etter størrelse

Variable Neighborhood Descent (VND)

```
Procedure VND (N[1..kmax])
incumbent:=Initial_Solution() ; best solution until now
while not Stop() do
    restart: for k:=1 to kmax do
        local_optimum:=Some_Local_Search(N(k),incumbent)
        ; Variants: First Improve, Best Neighbor, SA, TS ...
        if f(local_optimum)< f(incumbent) then
            incumbent:=local_optimum
            if not Stop() goto restart fi
        fi
    od
od
return incumbent
```

Variable Neighborhood Search (VNS)

```
Procedure VNS (N[1..kmax])
incumbent:=Initial_Solution()
while not Stop() do
    restart: for k:=1 to kmax do
        current:=Random_Element(N(k),incumbent)
        local_optimum:=Some_Local_Search(N(k),current)
        ; Variants: First Improve, Best Neighbor, SA, TS, VND ...
        if f(local_optimum)< f(incumbent) then
            incumbent:=local_optimum
            if not Stop() goto restart
        fi
    od
od
return incumbent
```

VNS

- Lokalsøk i VNS kan byttes ut med VND
- VNS er beregningskrevende for store instanser
- Redusert lokalsøk: VNDS
- Hybridisering
  - Tabusøk
  - GRASP
  - ...

Iterert lokalsøk

- Basert på (variant av) lokalsøk
- Lokalsøk gir avbildning

$$LS : S \rightarrow \hat{S}$$

- Iterasjon med ulike startløsninger gir ulike lokale optima
- Tilfeldig restart
- Hvorfor ikke rekursjon

$$LS : \hat{S} \rightarrow \hat{\hat{S}}$$

Iterert lokalsøk

- Prøver å iterere slik at vi unngår å restarte i samme ”Basin of attraction”
- Gjør dette ved perturbasjon av beste løsning
- Diversifisering

Iterert lokalsøk

```
Procedure Iterated_Local_Search(N,f)
incumbent:=current:=Local_Search(Initial_Solution(),N,f)
while not Stop_Criterion() do
    new_start:=Perturbation(current,history)
    new_local_optimum:= Local_Search(new_start,N,f)
    if f(new_local_optimum)<f(incumbent) then
        incumbent:=new_local_optimum
    fi
    if Accept(new_local_optimum,incumbent,history) then
        current:= new_local_optimum
    fi
od
return incumbent
```

Itererert lokalsøk - Perturbasjon

- Flytt i høyordens nabolag
- Fjerning og gjenoppbygging
  - ”Ruin and Recreate” (NB! patentert i USA!)
- Tilfeldig perturbasjon
- Fokusert perturbasjon
- ”Noising” – endre problem og finn lokalt optimum
- Distansemål
  - sjekke avstand til tidligere startløsninger
  - sjekke distanse til lokale optima

Itererert lokalsøk

- Hvor stor skal perturbasjonen være?
  - for liten perturbasjon: fare for å havne i samme ”Basin of attraction”
  - for mye perturbasjon: ”Random restart”
  - variabel perturbasjon
- ”Ruin and Recreate”
- Very Large Neighborhood Search

Kategorisering av metaheuristikker

- Individ vs. populasjon
- Probabilistiske vs. deterministiske
- Minneløse vs. minnerike
- Modifikasjon av kostnadsfunksjon
- Nabolagsbaserte vs. Multi-start baserte

Nabolagsbaserte vs. Multi-start baserte metaheuristikker

- Nabolagsbaserte
  - Essensielt varianter av lokalsøk
  - SA, TA (grunnleggende), GLS
- Multi-start baserte
  - Iterativ generering av startløsninger for lokalsøk
  - Startløsningene tas videre med lokalsøk
  - GA, Evolusjonsalgoritmer, Memetic Algorithms
  - Maurkolonialgoritmer, ACO
  - GRASP
  - Variabelt nabolagssøk
  - Iterert lokalsøk

”GUT” of Metaheuristics?

- Hvilke mekanismer virker?
  - lokalsøk
  - restart
  - tilfeldighet
  - oppoverflytt
  - minne
  - straff
  - diversifisering

  - populasjon
  - ....
- Hva med å lage en ”universell” metaheuristikk?
- ”No free lunch”-teoremet

”No Free Lunch”-teoremet (Wolpert & MacReady, 1995)

Uformell beskrivelse:

Når vi midler over alle probleminstanser i et gitt søkerom så har alle søkealgoritmer samme gjennomsnittlige ytelse.

For enhver algoritme, så betales enhver fordel m.h.t. løsning av en klasse av problemer med tilsvarende ulempe for en annen klasse.

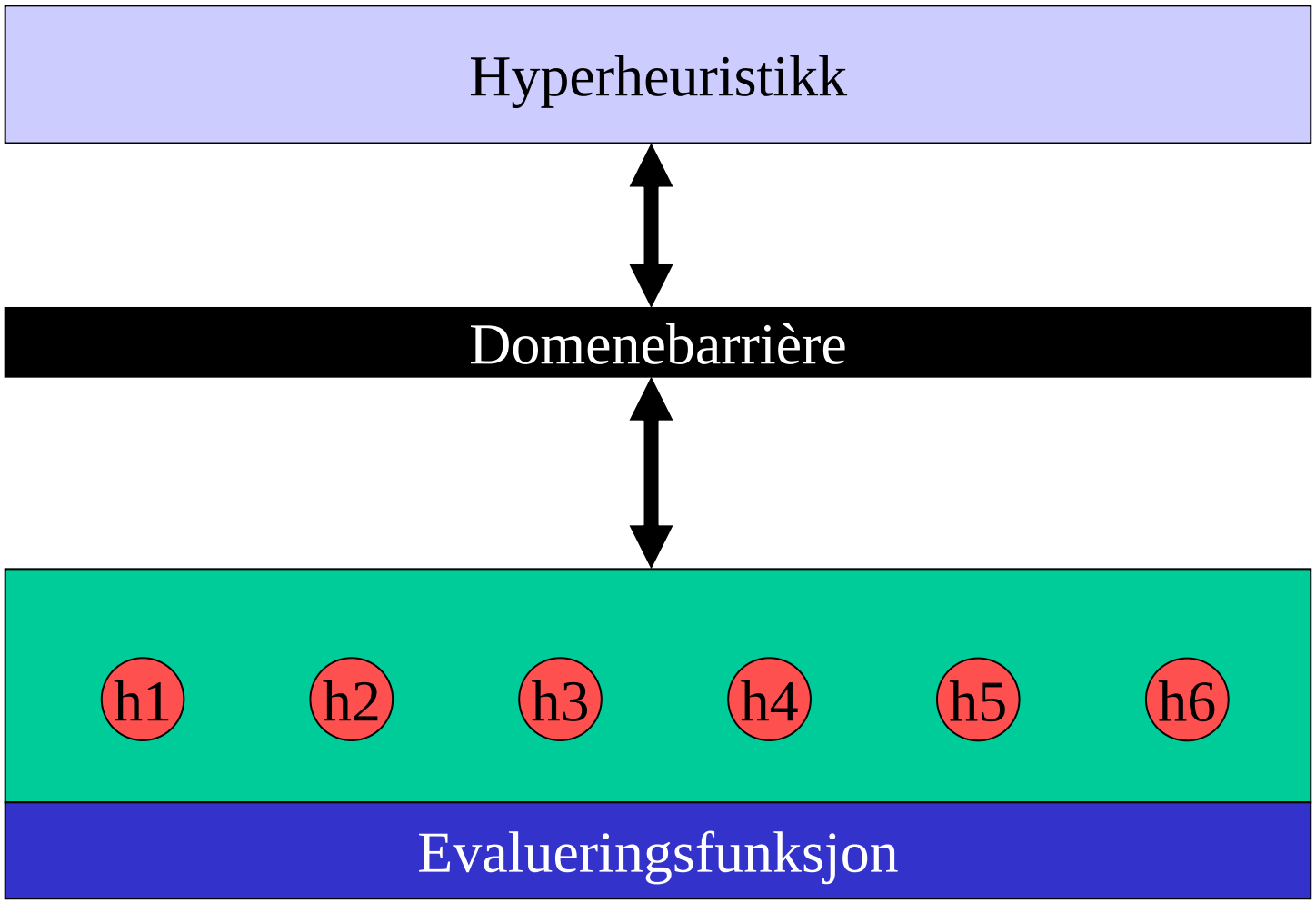
GUT for metaheuristikker

- Det er liten vits i å ”sause sammen” alle metaheuristikker til en metode
- Opportunistisk resonnering
  - analyse av problemet / problemløsningsstatus
  - valg av aktuell teknikk
  - læring
  - styring av søkeprosess etter kjølvannet

Hyperheuristikker

- Metaheuristikker er ikke generelle
- Det fins ingen “beste metaheuristikk”
- “No free lunch”-teoremet
- Hyperheuristikker
  - Generelle søketeknikker
  - Basert på (meta)heuristikker
  - Bruk av (meta)heuristikker for å velge (meta)heuristikk under søk
  - bruker ikke domenekunnskap

Hyperheuristikker



Hyperheuristikk

- Informasjon til hyperheuristikk fra hver (meta)heuristikk
  - CPU-tid brukt
  - Endring i objektfunksjon
  - Effektivitet
  - Hvor lenge siden er det siden heuristikken sist ble kalt

Hyperheuristikker

- Kvalifisert valg av heuristikk for å løse delproblem
- ”Læring” under søk
- Probabilistisk valg av heuristikk
- Rulett-seleksjon

Leksjon 7 - Oppsummering

- Kategorisering av metaheuristikker
- Konstruktive heuristikker
- Multi-start baserte metaheuristikker
  - Tilfeldig Restart (Random Restart, RR)
  - Variabelt nabolagssøk (VND/VNS)
  - Grådig Adaptivt Randomisert Søk (Greedy Randomized Adaptive Search, GRASP)
  - Iterert lokalsøk (Iterated Local Search, ILS)
- GUT (Grand Unifying Theory)?
- Hyperheuristikker