



Influence maps

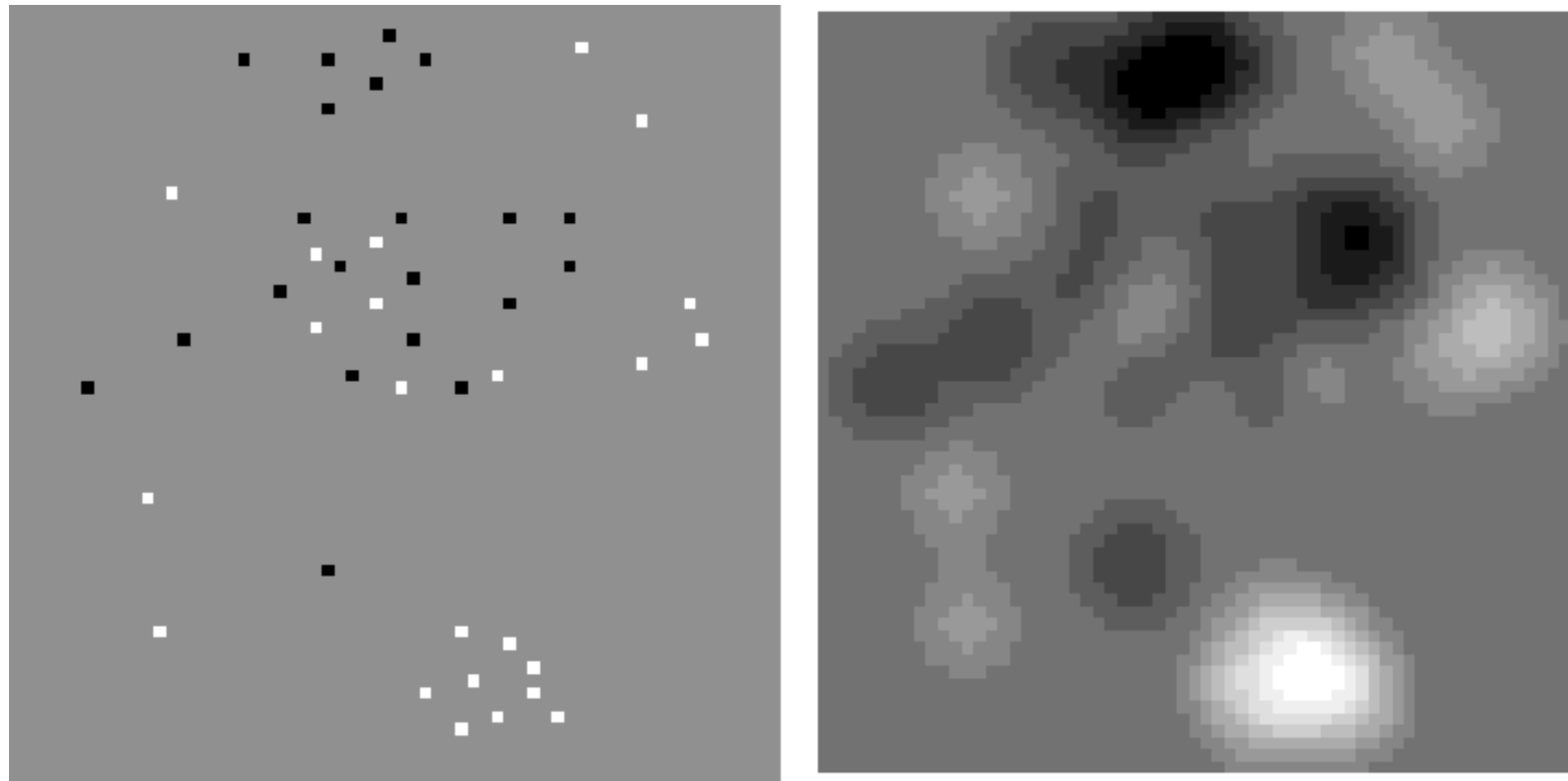
Kraftfull representation av information om spelsituation!

Lågpasfilter av karta av spelagenter förenklar analysen.

- Vem “äger” rutan?
- Är området säkert, inga fiender nära?
 - Vilken annan medspelare står fri?
 - Var är motståndaren som svagast?

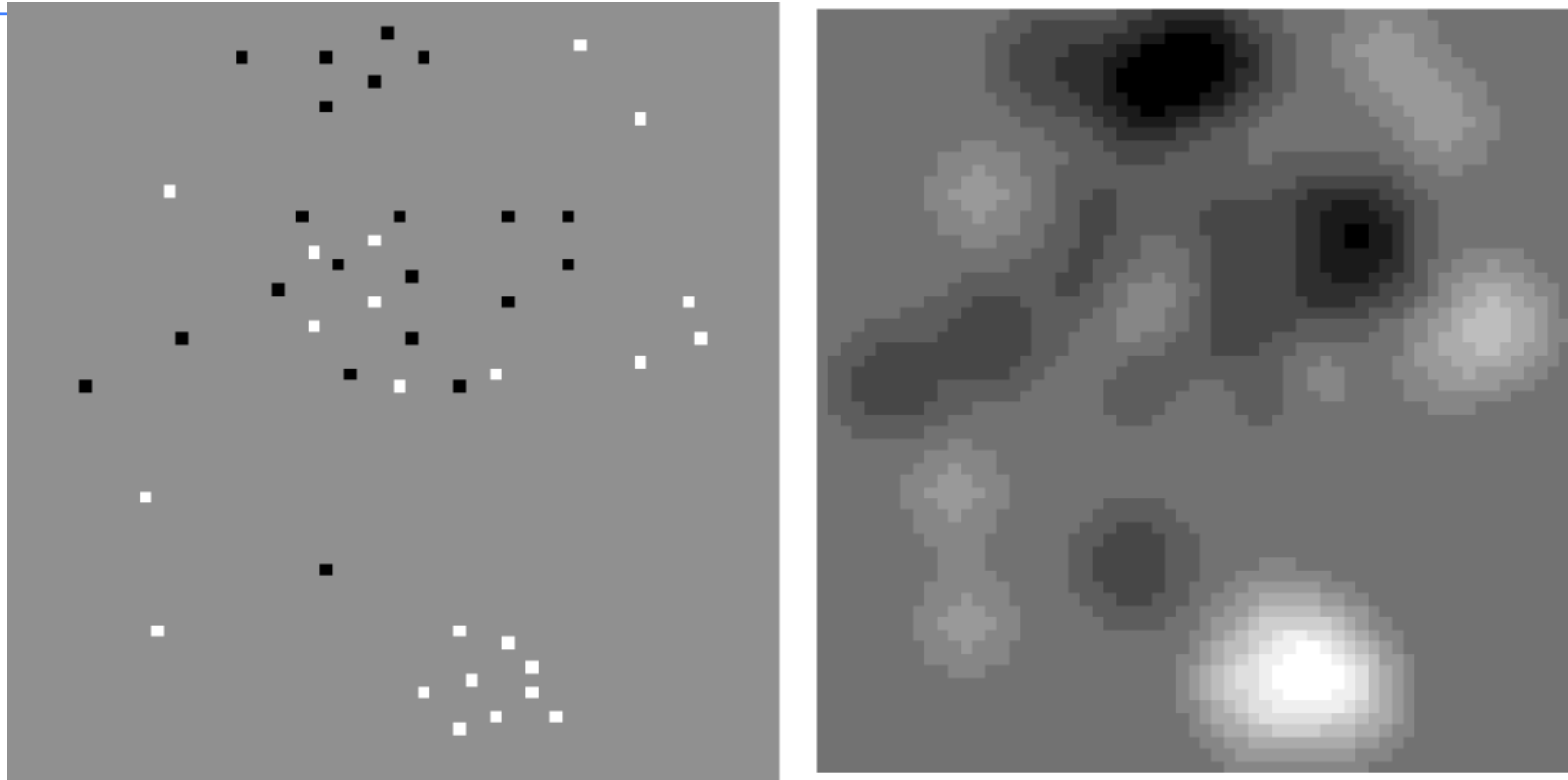


Influence maps, exempel





Information Coding / Computer Graphics, ISY, LiTH



- Fatta beslut beroende på influensvärdet
- Anfall svaga punkter
 - Passa till obevakade medspelare



Varianter

- Terrängberoende filtrering, långsammare spridning i svår terräng
 - Flera influence maps för olika data
 - Olika värde i kartan för olika pjästyper

Varianterna styrs i stort av speltypen, men grundprincipen är generellt användbar till många problem.



Exempel

En influence map för egen styrka. Vad äger jag?

En influence map för motståndaren.

En influence map för värdefulla punkter (resurser, vinstpoäng, baser...)

En karta över terrängen. Låt filtrering eller vägsökning påverkas av denna.

Angrip punkt med högt värde/(motstånd-styrka)



Exempel

Projekt 2016 (Dennis Ljung & Sebastian Sundin): AI som söker efter spelaren.

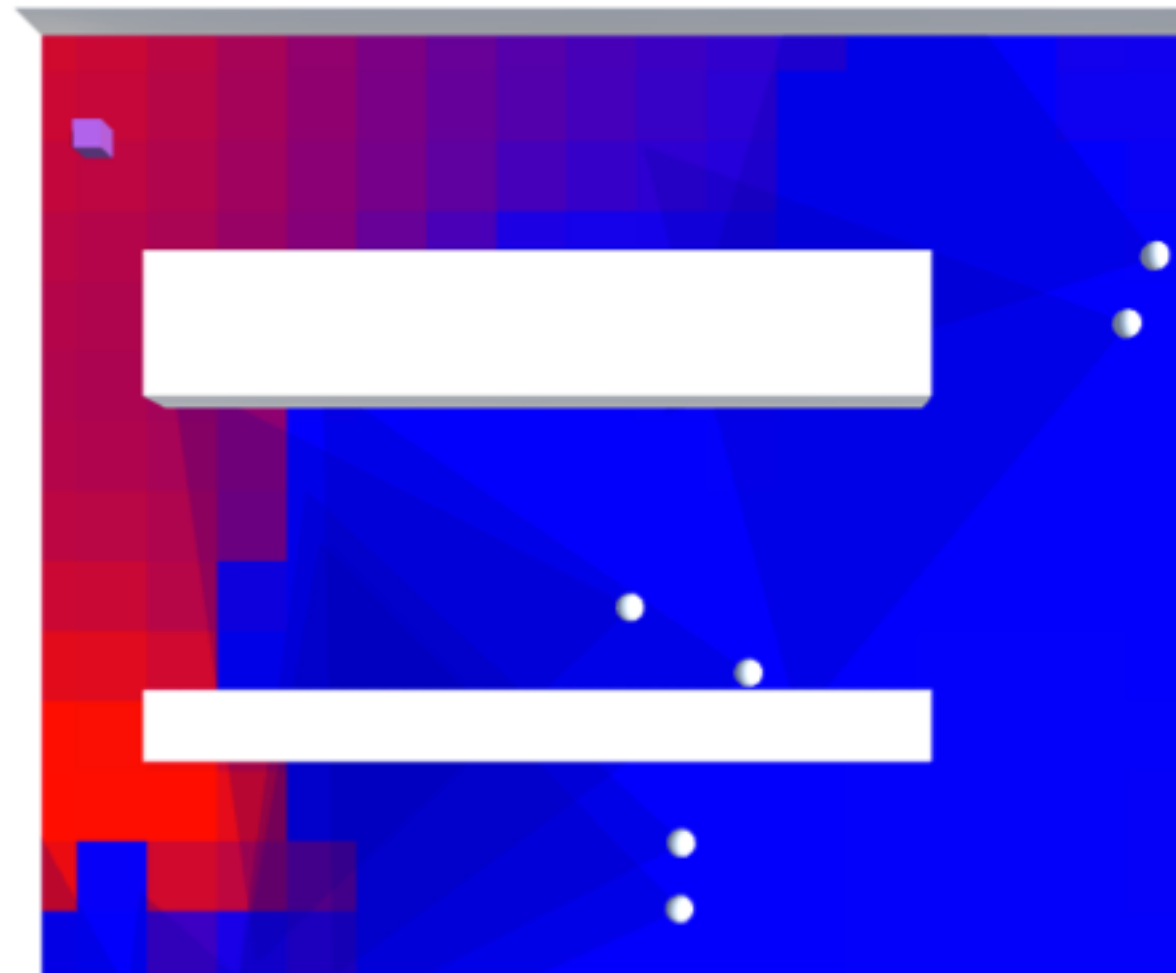
Influence map som "minns" var den sökt!

Om AI-agenten ser spelaren: Stark ökning kring den platsen.

Dessutom modifierad flocking för att sprida AI-agenterna.
(Kunde också varit influence map!)



Rött: Områden där AI-agenterna vet att spelaren är eller varit relativt nyligen.





Influence maps och bildanalys

En influence map är en bild av den spatiella situationen.

Bildanalys är ett välutforskat ämne - många metoder finns för att analysera innehållet.

- Segmentering, watersheds
 - Linjära och olinjära filter
 - Formanalys, skelettering

Analysen kan lätt snabbas upp på GPU:n med vanliga grafikmetoder

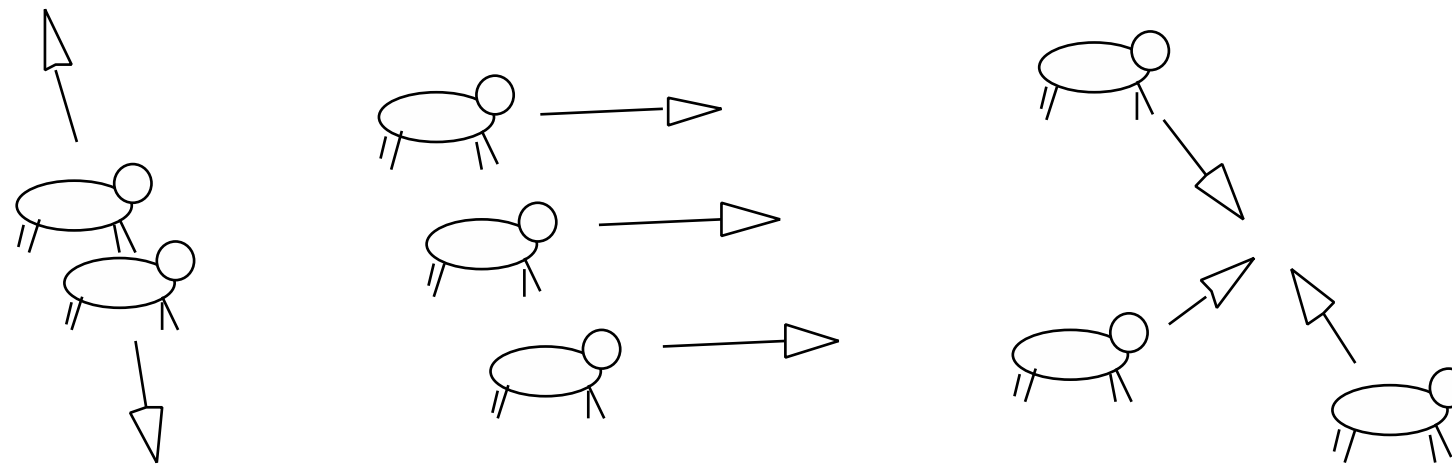


Flocking - Boids

Gruppbeteenden kan simuleras med enkla medel.

Craig W Reynolds "boids", baserade på tre enkla regler:

- Separation - undvik att gå för nära varandra
 - Alignment - sträva åt samma håll
- Cohesion - sträva mot mitten av gruppen





Boids, implementation

Partikelsystem

Separation: Kraft bort från närmaste boid

Alignment: Beräkna medelhastighet, drag egen hastighet mot denna

Cohesion: Beräkna medelposition, kraft mot denna



Boids

```
for all boids i
  for all boids j except i
    if dist(i, j) < kMaxDistance
      count += 1
      accumulate speed difference
      accumulate average position
      accumulate avoidance vector

for all boids i
  if count > 0 then
    divide accumulated data by count
  affect speed by
    speed diff (alignment)
    average pos (cohesion)
    avoidance vectors (avoidance)
  position += speed
```



Viktiga parametrar att ”skruva på”

Vilken vikt på varje komponent?

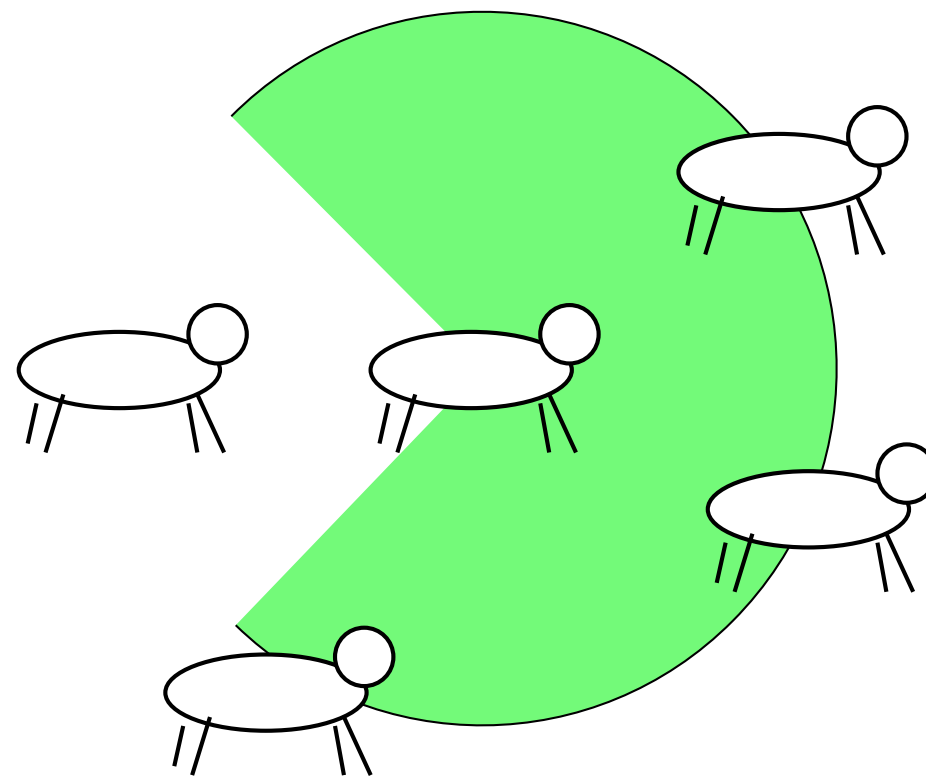
Hur stor omgivning skall räknas?

Avståndsberoende funktioner för separation och cohesion?



Boids, synfält

En boid kanske inte bör ha "ögon i nacken"



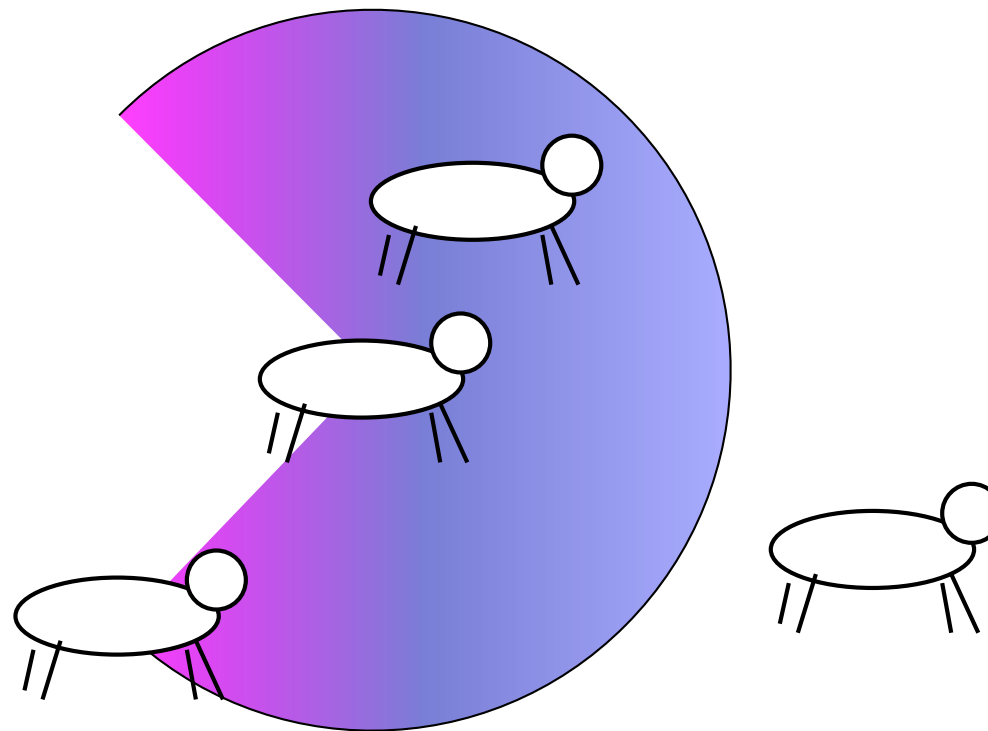
Vilka andra boids skall räknas?

Begränsa på vinkel såväl som avstånd.



Undvik diskontinuiteter

En enkel tröskelfunktion kan ge ryckig animation



Mjukt avtagande funktion?

För stark separation kan ge våldsamma krafter om funktionen blir för stor nära



Boids är exempel på A-life

Imitation av verkliga beteenden

Optimala lösningar är inte målet!

En A-life-vägsökning får gärna vela och tramsa, om det stämmer med agentens typ.

Optimal lösning för ett djur på vägen vs verkligt beteende?



Boids är också exempel på *gruppbeteenden*

Gruppbeteenden är ofta icke-triviala, intressanta problem.

Hur delar man upp grupper för bästa resultat?

Hur undviker man att två grupper hindrar vägen för varandra?

Skall individer tilldelas sin "plats i ledet" eller hitta dem själva?



Information Coding / Computer Graphics, ISY, LiTH

Flockinglabben på fredag

Baseras på enkel 2D-spritemotor

Implementera boids-algoritmen

Något tillägg för att få ett "rikare" system.

3 timmar, 10-13, på grund av schemaproblemen.



Lärande system

System som lär sig är ett intressant ämne! Mest kända metoderna:

- Nervnät
- Genetiska algoritmer

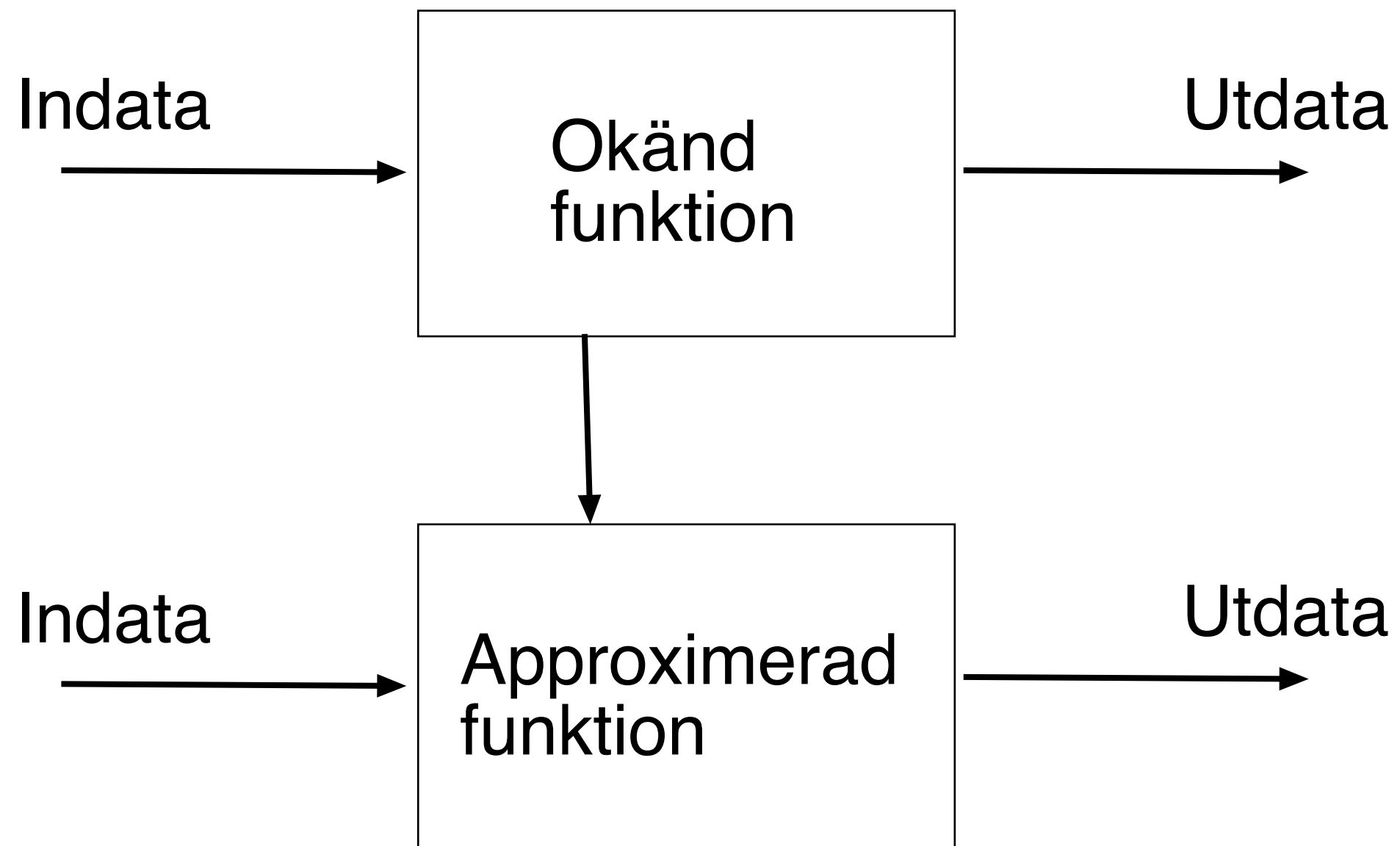
Aningen mindre kända:

- Simulated annealing
 - Hill climbing
- Heuristiska metoder

Men observera: Allt är i grund och botten *optimering!*



Optimering?





Information Coding / Computer Graphics, ISY, LiTH

Grundmetoder för lärande system/funktionsoptimering

Hill climbing

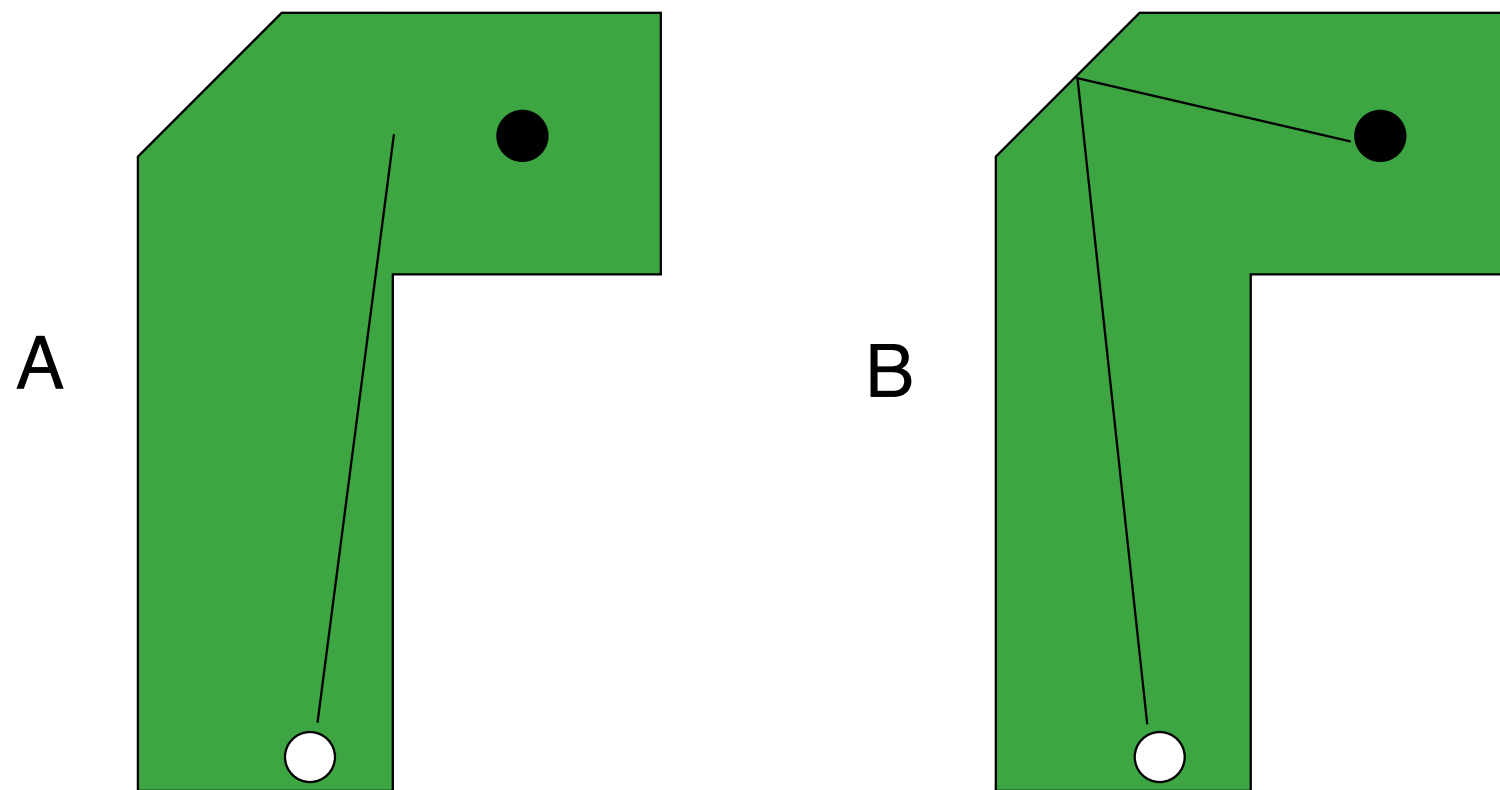
Simulated annealing

Genetic algorithms



Lokala minima måste undvikas

Exempel: minigolf



Små förändringar kring A ger sämre lägen; större avstånd till hålet



Lärande system/ funktionsoptimering, exempel

Minigolf: Slå bollen, få den att träffa så nära hålet
som möjligt

Indata: Vinkel, utgångshastighet

Utdata: Bollens slutposition

Målfunktion: Avstånd mellan utfall och mål.

Problem: Hitta rätt indataparametrar.



Hur appliceras respektive metod?

Hill climbing: Ändra parametrarna åt det håll som borde bli bättre.

Simulated Annealing: Ändra parametrarna slumpvis, behåll de bästa. Minska slumpstegen efter hand.

Genetic algorithms: Kombinera par av försök till nya, behåll de bästa.



Information Coding / Computer Graphics, ISY, LiTH

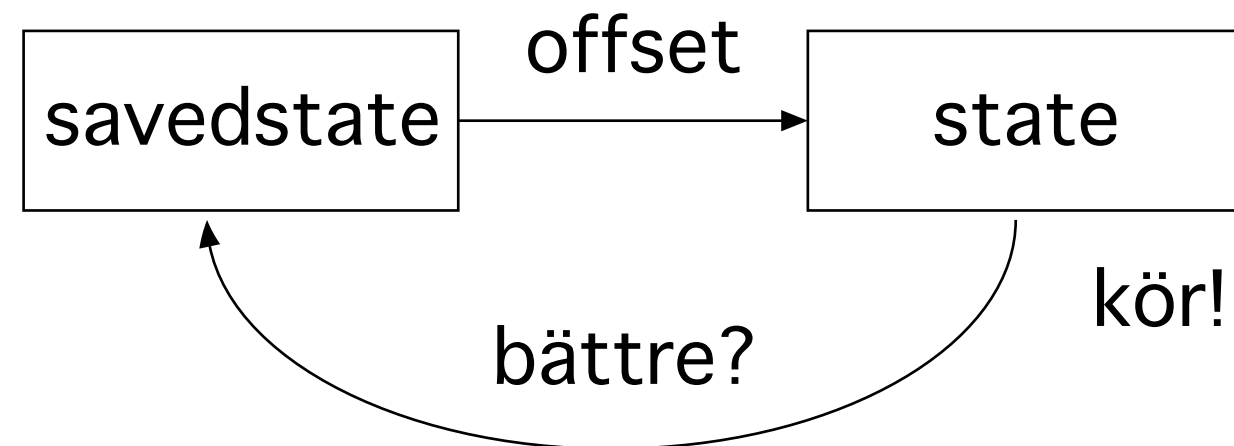
Data: Tillståndsvektor (startposition, utslagskraft, vinkel, resultat)

Hill climbing: Ändra parametrarna åt det håll som borde bli bättre.

Enkel metod:

$state = savedstate + \text{slumpmässig offset}$

Kör! Spara till savedstate om det är bättre



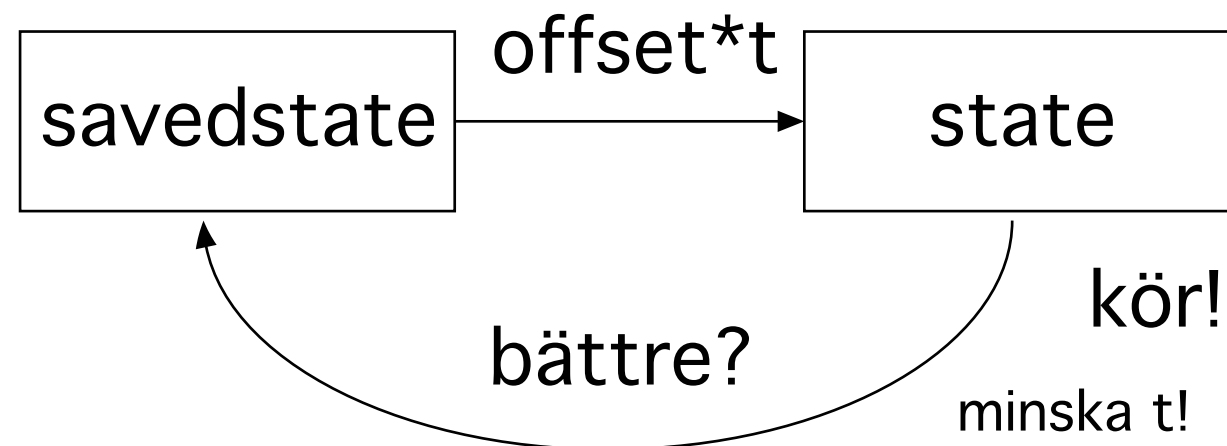


Simulated annealing: Samma sak men plus en temperaturparameter

$\text{state} = \text{savedstate} + \text{slumpmässig offset} * \text{temperatur}$

Kör! Spara till savedstate om det är bättre

Sänk temperaturen efter hand





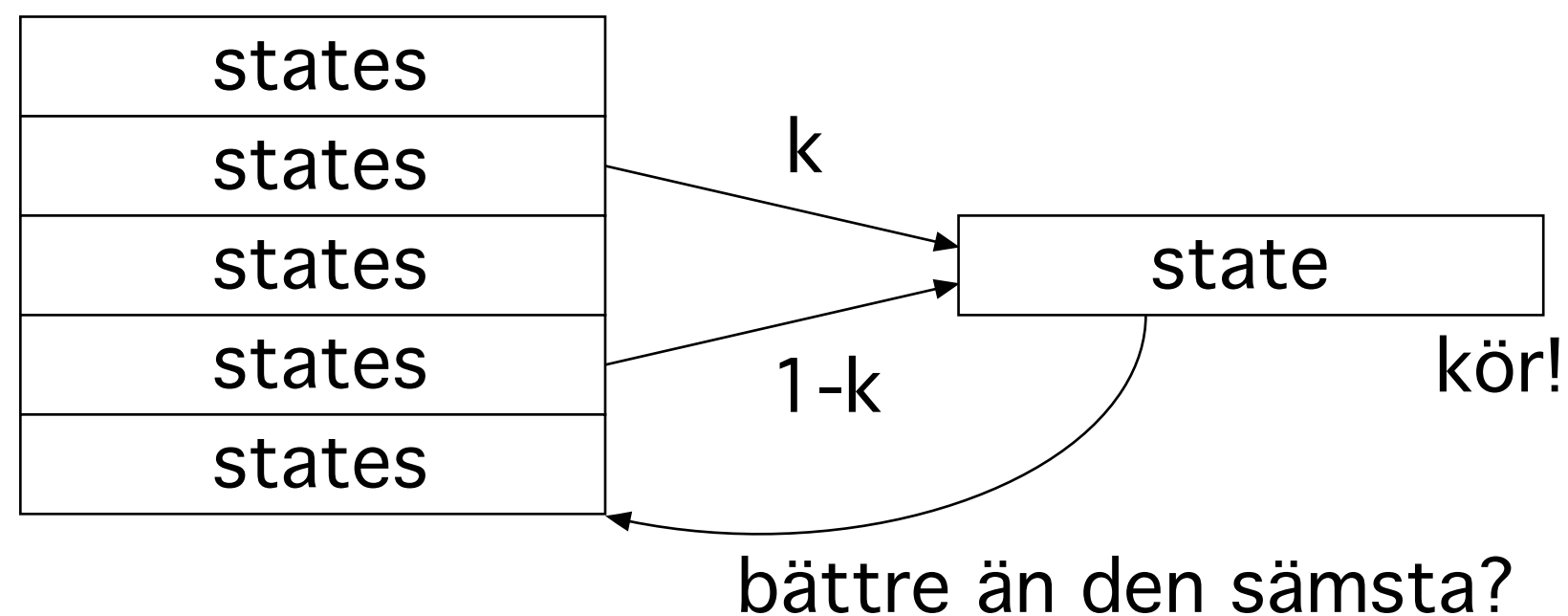
Genetic algorithms:

Spara ett antal tillstånd, t.ex. `states[10]`

$$\text{state} = \text{states}[i] * k + \text{states}[j] * (1-k)$$

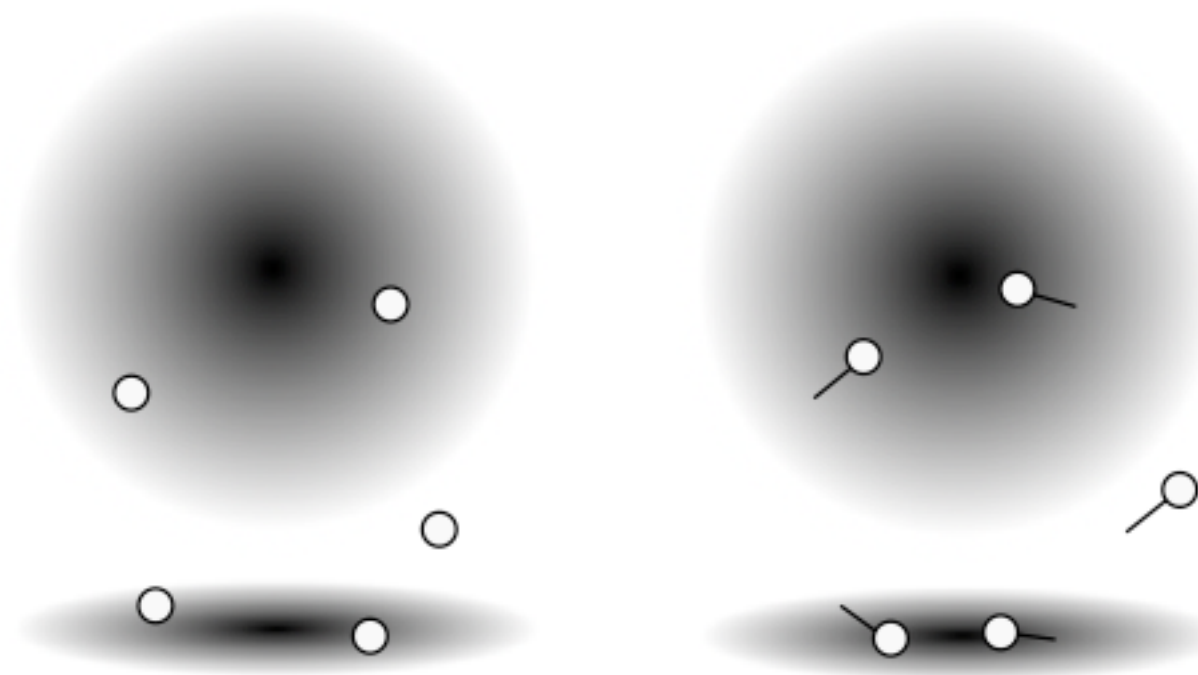
Kör! Spara till `states` om det är bättre än någon gammal

Kräver start med ett antal tillstånd med stark variation eller slumpvisa "mutationer"





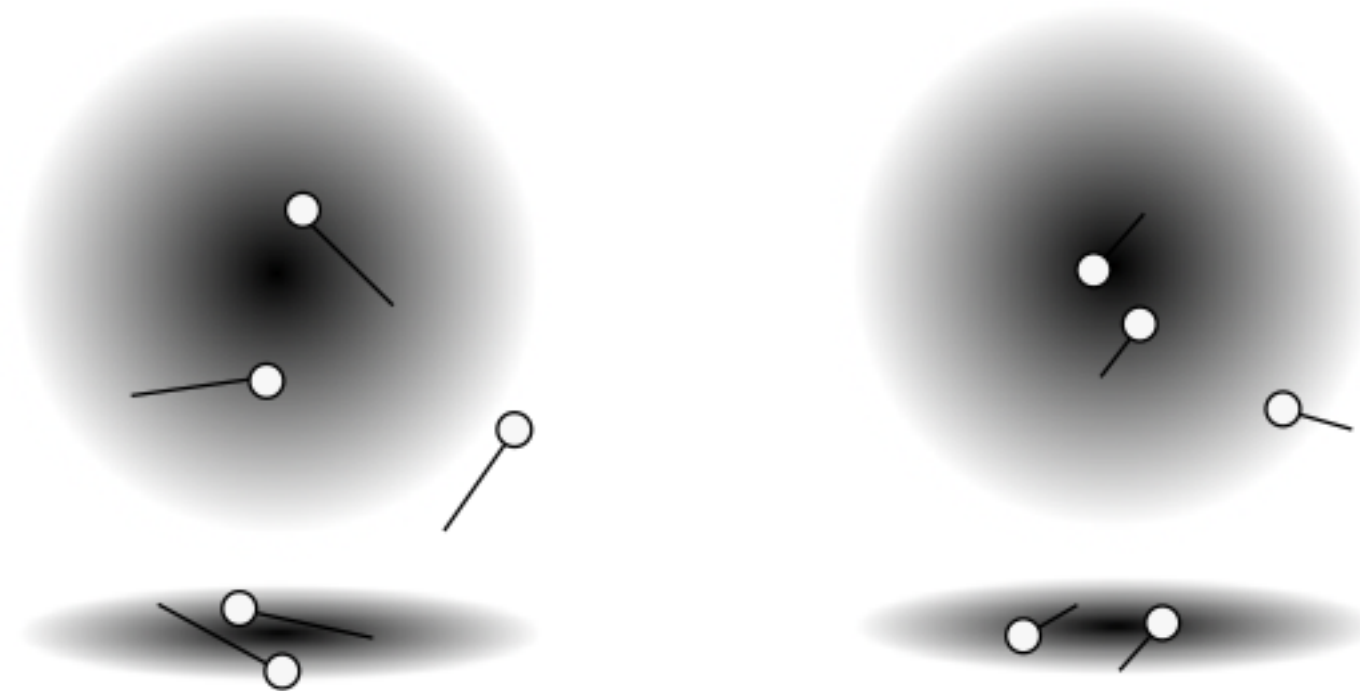
Hill climbing



Sök närliggande bättre platser tills vi hittar lokalt minimum



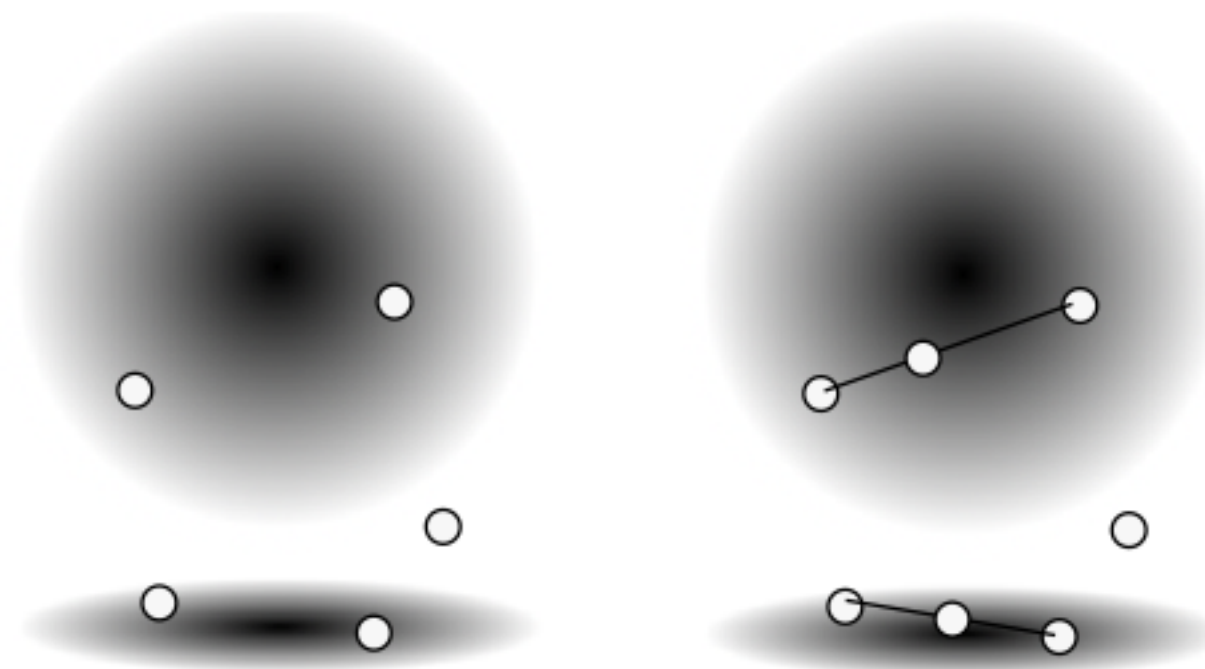
Simulated annealing



Tag mindre och mindre slumpmässiga steg



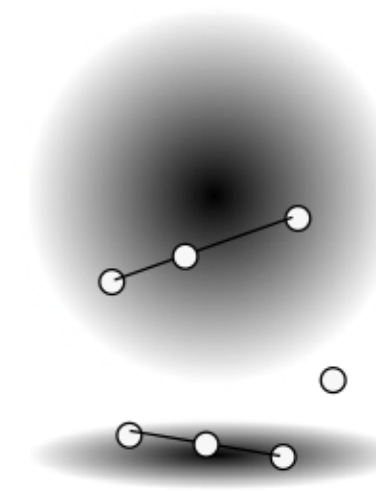
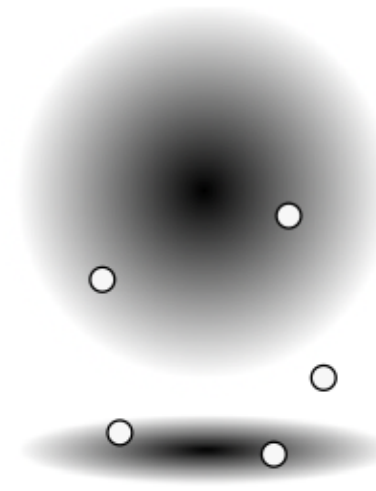
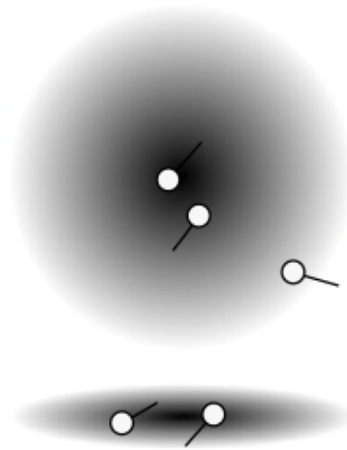
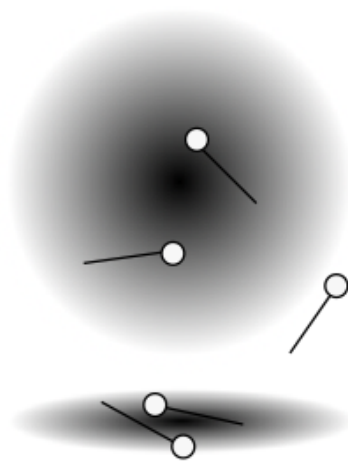
Genetiska algoritmer



Kombinerar tidigare lägen



Simulated annealing eller genetiska algoritmer?



Simulated annealing: Mer frihet att ta steg utanför tidigare sampel. Fastnar inte i felaktiga antaganden.

Genetic algorithms: Kan söka bättre längs "smala" områden, samlar sig bättre kring optimum i sådana fall.



Neurala nät, Neural networks

En representation av systemet, av den önskade funktionen.

Kan med tidigare nämnda metoder modellera komplexa funktioner.

Mycket hett ämne idag, "deep learning", tack vare att vi nu kan hantera stora nät.



Information Coding / Computer Graphics, ISY, LiTH

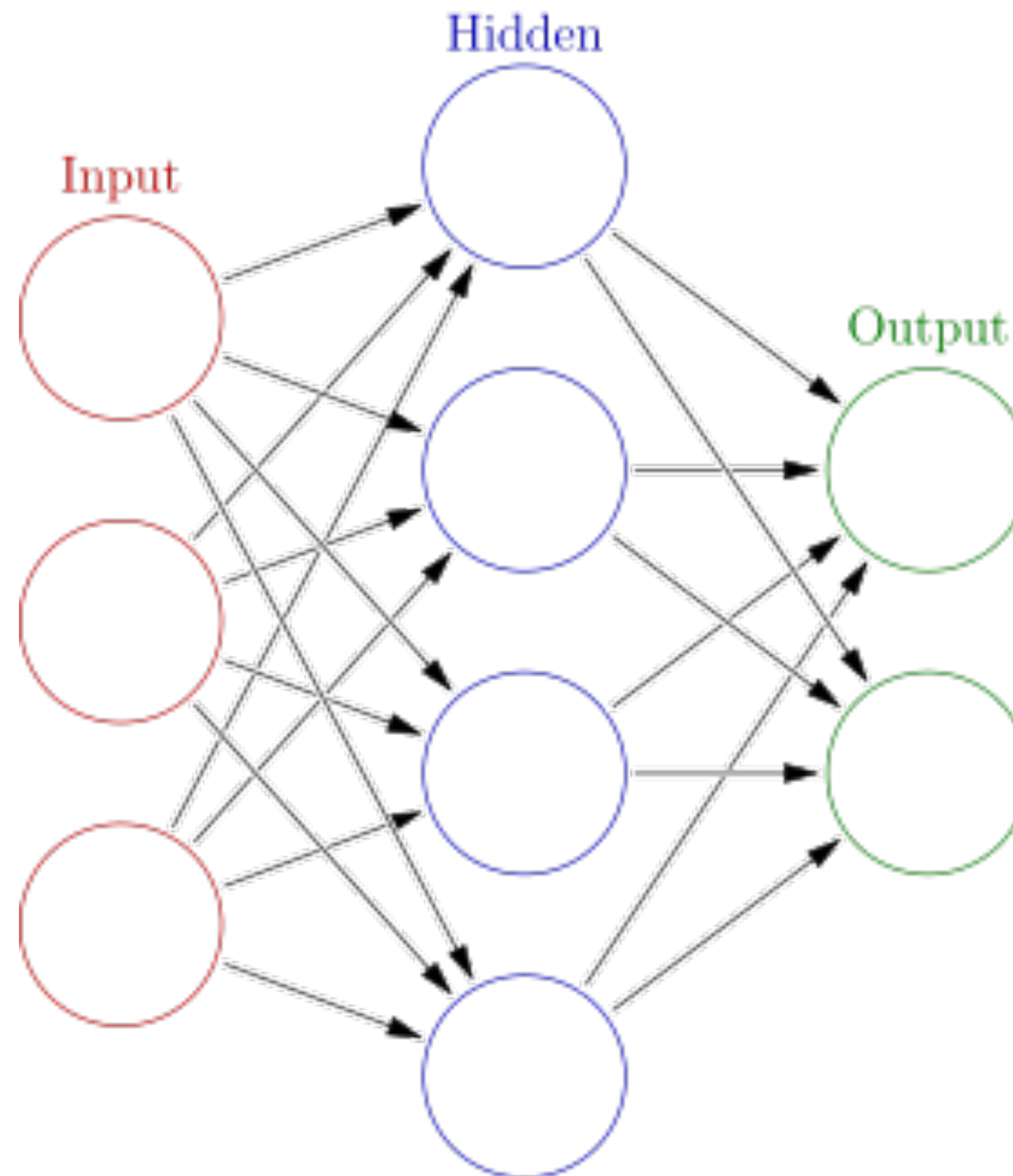


Bild från Wikipedia



Information Coding / Computer Graphics, ISY, LiTH

Tensor cores

Accelerates matrix multiply and accumulate



4x4 matrix multiplication

Matrix multiplication in low precision

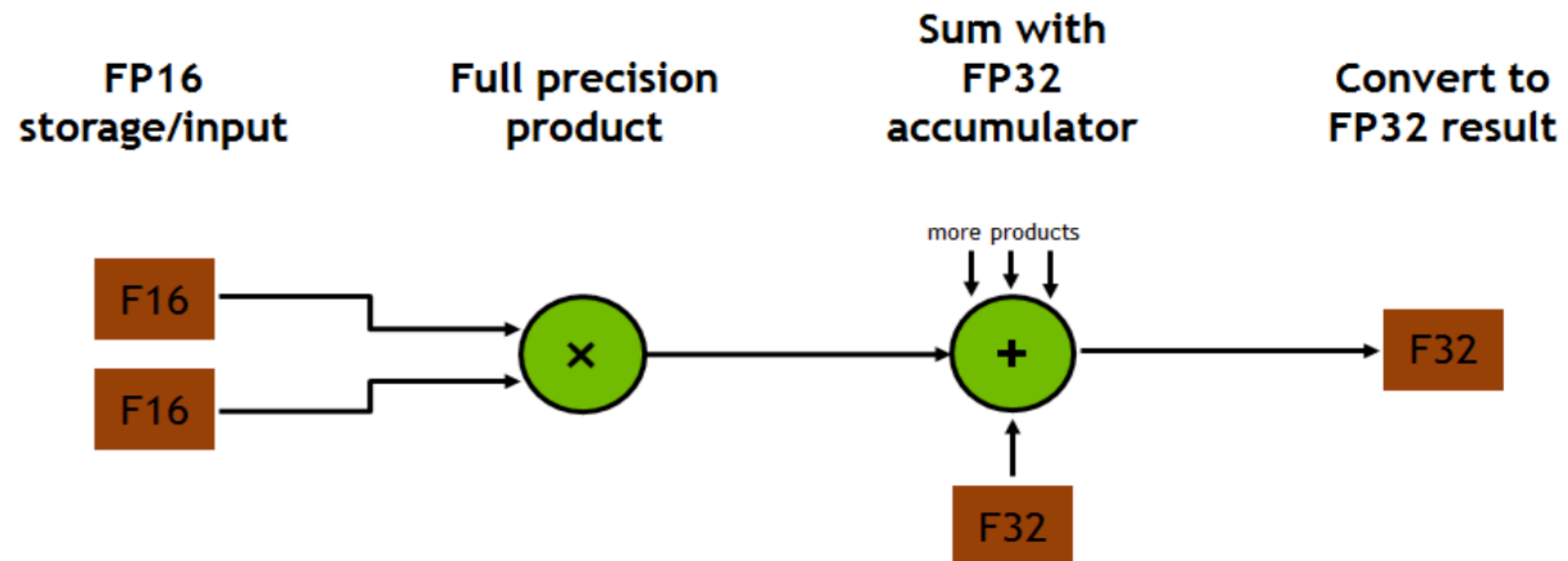
$$\mathbf{D} = \begin{matrix} \text{FP16 or FP32} & \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} & \begin{matrix} \text{FP16} \\ \text{FP16} \end{matrix} & + & \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix} & \text{FP16 or FP32} \end{matrix}$$

Build bigger multiplications with 4x4 as building block.



Low precision for faster calculation

FP16 in, FP32 out





Funktionsapproximation är inte alls tråkigt, men...

Mitt dilemma:

Är detta relevant för spel?

Nej! Inlärning har en ytterst liten roll i spel!

Heuristiska metoder troligen intressantast, för begränsad inlärning hos agenter.

Nervnät mm har framgångsrikt använts på bl.a. bildanalys, men knappast på spel.



Information Coding / Computer Graphics, ISY, LiTH

Spel med inläring eller liknande inslag:

SimLife

Viss simulering av genetik, men

- urval av population är inte (exakt) samma sak som inläring och upplevs inte som det
- Spelet var inte särskilt underhållande och simuleringen var dålig

Black&White

Inläring av beteende på “varelsen”

Intressant idé, mindre lyckat resultat



Men är AI viktigt för spel?

Javisst!

- Beteenden och vägplanering
 - Världsrepresentation
- Analys av speldata i denna representation
- Komplex agentrepresentation för styrning av beteenden

Däremot är lärande mer marginellt (men bevisa gärna att jag har fel)



Information Coding / Computer Graphics, ISY, LiTH

AI-projekt?

Många möjligheter.

De bra baserar sig ofta på influence maps och annan analys av världen.

A* räcker inte som huvudpoäng.

Våga gå lite bortom grunderna.