

Wolf and Hare

Nils Kohl, Thomas Stadelmayer, David Uhl

Friedrich-Alexander Universität Erlangen-Nürnberg

03.07.2015

1 Wolf and Hare

- Spielregeln
- Visualisierung

2 Implementierung

- Seriell
- Parallel

- 2D Spielfeld mit zwei Wölfen und einem Hasen
- Zufällige Startposition auf Spielfeld
- Pro Zug: Wölfe jeweils einen Schritt, Hase einen Schritt
- Spielende: Wolf fängt Hasen oder Hase erreicht Ziel

Aufgabe

- Spiel parallelisieren
- Jede Maschine auf Cluster testet verschiedene Routen

Visualisierung (1)

x-Position	y-Position
0	0
0	1
0	2
0	3
0	4
1	4
2	4
3	4
4	4

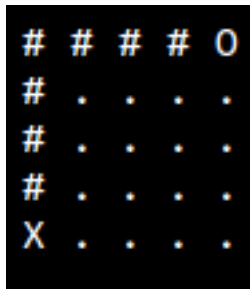
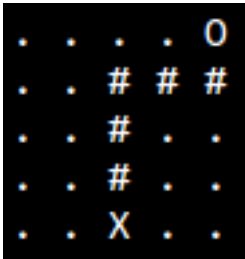
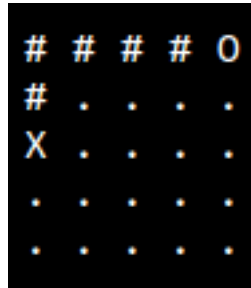


Table: Wolf1

Visualisierung (2)



RouteX Wolf1



Route Hase

Visualisierung (3)

x-Position	y-Position
0	0
1	0
2	0
3	0
4	0
4	1
4	2
4	3
4	4

Table: RouteX von Wolf1

x-Position	y-Position
0	2
0	3
0	4
1	4
2	4
3	4
4	4
4	4
4	4

Table: RouteX von Hase

- Vergleiche Route1 von Wolf1 einer Routen vom Hasen
- In neue Liste Anzahl der Schritte bis Hase gefangen wird (-1 wenn Hase vorher im Ziel)
- Alle Routen von Wolf1
- Analog Wolf2
- Vergleiche Liste von Wolf1 und Liste von Wolf2
- Ausgabe Beste Routen

```
for all wolf1 routes do
|
|   for all hare routes do
|   |
|   |   compare route_i of wolf1 with route_j of hare;
|   |       ▷ compares each element;
|   |       ▷ counter++;
|   |
|   |   if capture then
|   |   |   push counter to list;
|   |   else
|   |   |   add -1 in list;
|   |   end
|   end
| end
end
```

Algorithm 1: How to get best routes

- Tupel von einer Route Wolf1 und Wolf2
- Vergleiche Tupel mit allen Routen von Hase
- Schreibe in Liste Anzahl der Schritte und Wahrscheinlichkeit als Indikatoren
- Wenn fuer neues Tupel bessere Loesung gibt, dann loesche alten Wert und schreibe neuen rein
- Tupel auf Prozessoren aufteilen
- Nach Berechnung vergleiche die Listen

```
<r1w1,r2w2> = choose tupel of routes wolf1 and wolf2;  
for all hare routes do  
    | compare tupel with route_i of hare;  
        | ▶ compares each element;  
        | ▶ in case of success probability-local++;  
end
```

.
. .
.

```
if probability-local < probability-global then  
    | probability-global = probability-local;  
    | delete all elements in list;  
    | add <r1w1, r2w2> to list;  
end  
if p-local == p-global AND counter-local < counter-global then  
    | counter-global = counter-local;  
    | delete all elements in list;  
    | add <r1w1, r2w2> to list;  
end  
if p-local == p-global AND counter-local == counter-global then  
    | add <r1w1, r2w2> to list;  
end
```

Algorithm 2: How to get best routes

- Sammle alle p-global und counter-global (MPI-Allgather)
- Root: entscheide welcher Rank die besten Werte hat
- Teile jedem Rank mit ob er die besten hat (1) oder nicht (0) (MPI-Broadcast)
- Jeder Rank, der eine 1 erhalten hat, sendet seine ermittelten Routen zu root (MPI-Send)