Algorithmen und Datenstrukturen 2

Sommersemester 2007
13. Vorlesung

Peter F. Stadler

Universität Leipzig
Institut für Informatik
studla@bioinf.uni-leipzig.de

Genetisches Programmieren

Rückblick: Bisher wurden Parameter optimiert.

Nichts widerspricht der Optimierung von ***Programmen*** im Gegensatz zu ***Daten.***

Daten	-	Programm
Fitnessfunktion	-	Qualität
Vermehrung	-	??
Mutation	-	??

Darstellung von Programmen

Für Programme kann man eine **LISP**- artige Notation wählen. Während in **LISP**-Files wie üblich alle Programmschritte linear auf die Zeilen geschrieben werden, kann man sie ebenfalls als Baum auffassen.

Dazu spezifiziert man die Menge der Funktionen und Terminalen, z. B.

$\boldsymbol{F}$ = { AND, OR, NOT, XOR },

$\boldsymbol{T}$ = { D0, D1, D2, . . . }.

ACHTUNG ! Die Menge $\boldsymbol{F}$ muss abgeschlossen sein, d. h. alle Ein- und Ausgänge haben denselben Typ (arithmetisch, boolsch, . . .). Außerdem sollte das Problem prinzipiell mit ($\boldsymbol{F}$ **,** $\boldsymbol{T}$) lösbar sein**.**

Die Ausgangspopulation

Zuerst initialisiert man eine Population von zufällig generierten Programmbäumen.

1. Wahl des zu ergänzenden Knotens,

2. Wahl der Funktion / Terminale,

3. Einfügen.

Damit dabei nicht zu komplizierte Programme entstehen, kann man die Entwicklung durch Restriktionen steuern. Zudem entfernt man doppelte Individuen.

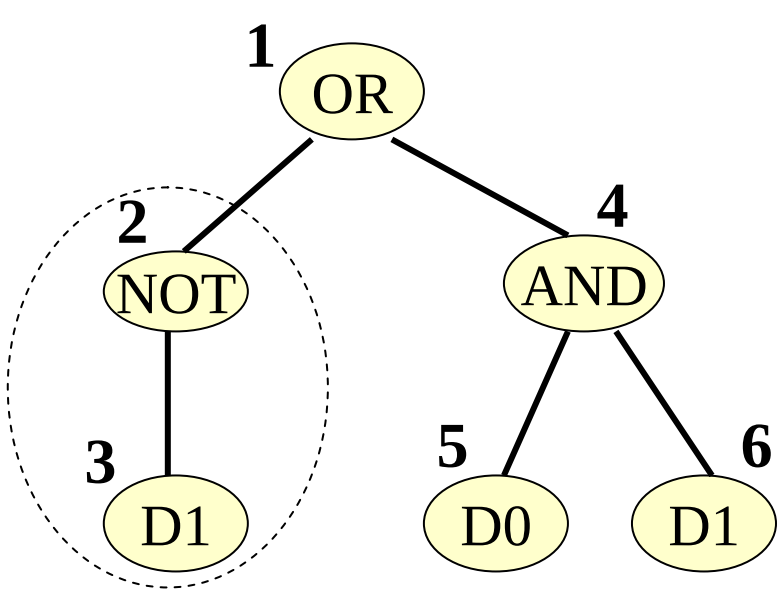
Die Programme sind also dem Problem in Struktur und Tiefe in keiner Weise angepasst.

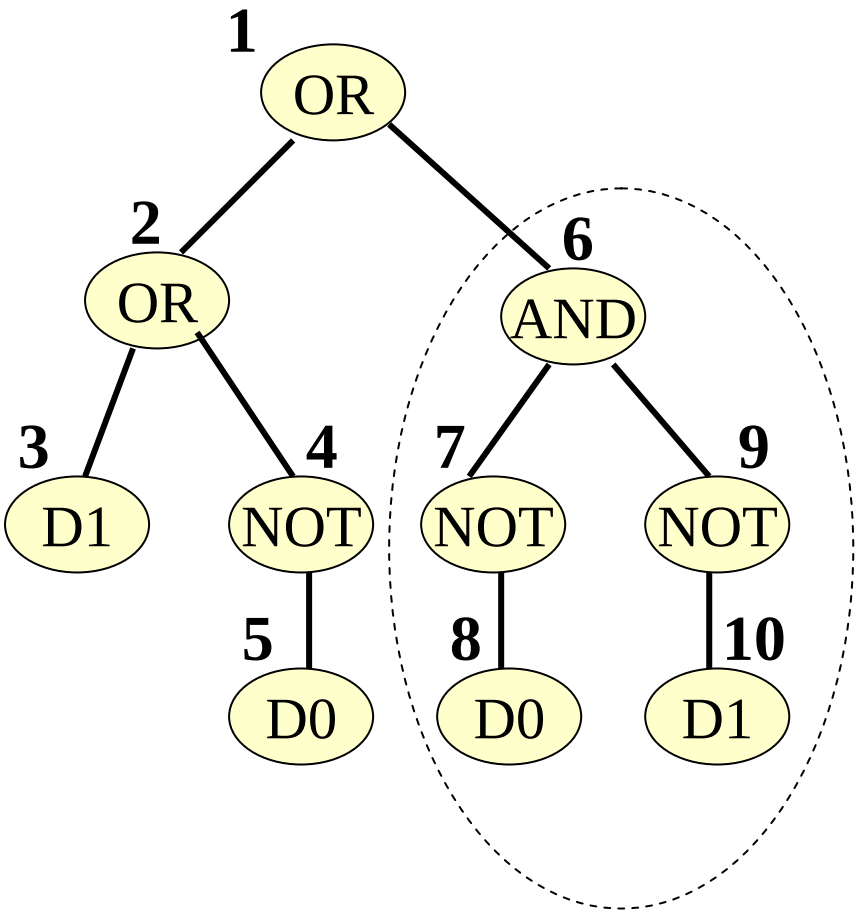
Genetische Operationen: Crossover

Crossover ist komplizierter als gewöhnlich und nutzt die Abgeschlossenheit der Funktionenmenge aus. Das Programm wird als Baum dargestellt. Dann werden mit Copy und Paste Teilbäume vertauscht.

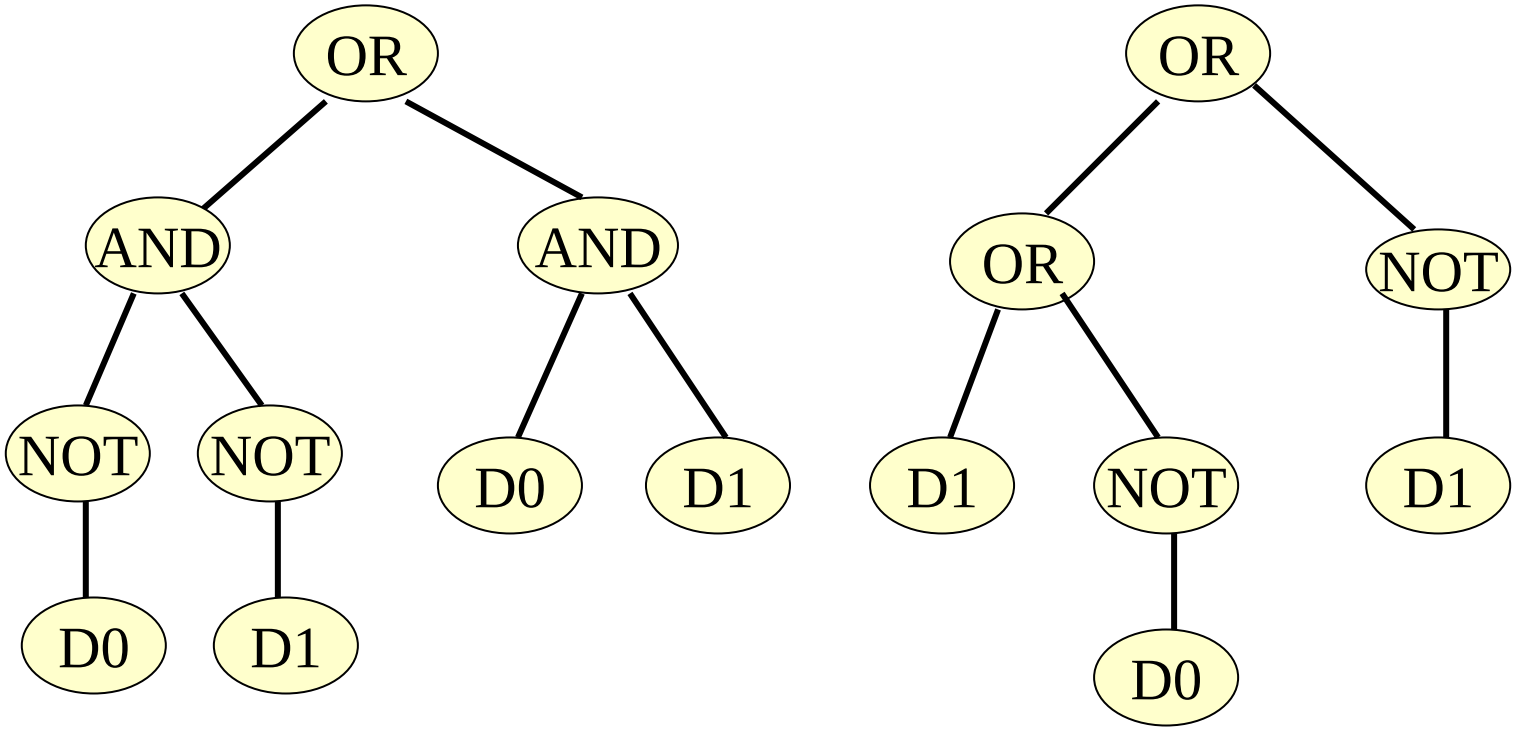
Da die Baumeigenschaft erhalten bleibt, liegen auch automatisch syntaktisch korrekte Programme vor.

Programmbäume





Austausch der markierten Teilbereiche



Programmierung von Ameisen

Ameisen sind abstrakte Lebewesen,

- die auf einem quadratischen Gitter leben,
- sich schrittweise darauf in eine Richtung ihrer Wahl (oder auch zufällig) bewegen können und
- über einfache Sensoren verfügen, z.B. Nahrung erkennen.

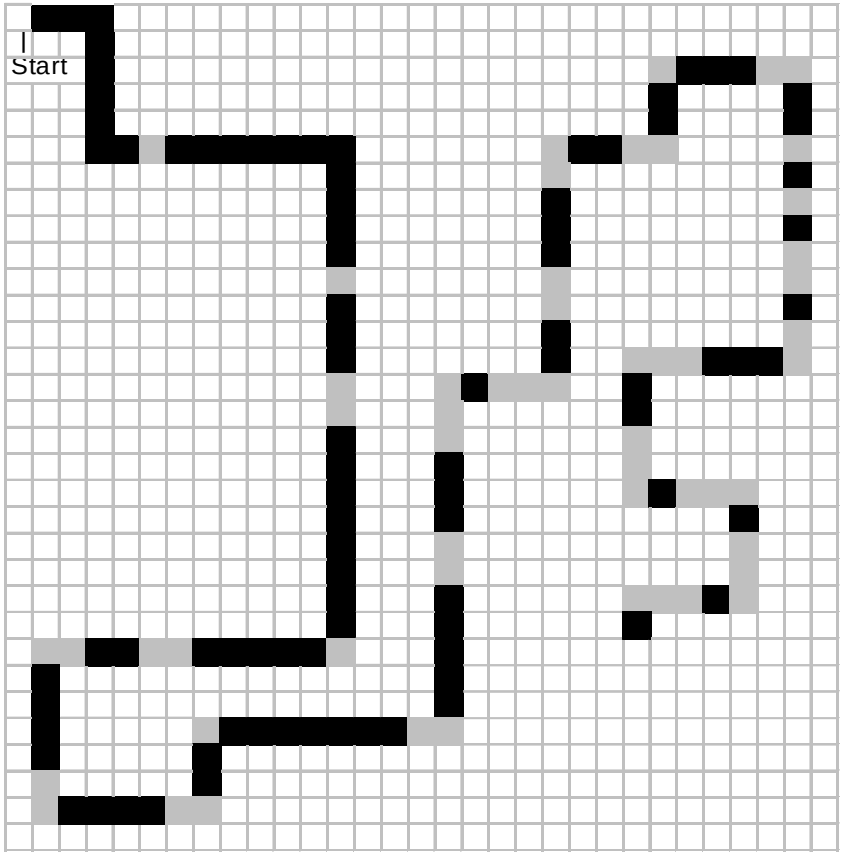
Mit solchen Funktionen ausgestattet soll die Ameise Aufgabenstellungen lösen. D.h. es soll mittels genetischer Programmierung ein Programm entstehen, welches die Aufgabenstellung löst.

Literatur für unsere Ameisen:

Soucek, B. and the IRIS Group: Dynamic, Genetic, and Chaotic Programming, Wiley 1992, Signatur: K 62 12

Aufgabe: Weg verfolgen

Auf einem 32x32-Feld (zum Torus gefaltet) liegt Nahrung (schwarz) auf einem Weg aus. Die Folge von Nahrungsstücken hat Unterbrechungen (grau) der Länge 1, 2 oder 3, auch an Ecken. Die Ameise soll den Weg verfolgen und möglichst viele Nahrungsstücke aufnehmen. Gesamtzeit: 400 Programmschritte.



Funktionen der Ameisen

Eine Ameise ist charakterisiert durch ihre Position und die Richtung ihres Kopfes.

- RIGHT: Die Ameise dreht sich auf der Stelle nach rechts
- LEFT: Die Ameise dreht sich auf der Stelle nach links
- MOVE: Die Ameise läuft einen Schritt vorwärts. Betritt sie ein Feld mit Nahrung, wird diese automatisch gefressen.
- IF-FOOD-HERE: Steuerung für eine bedingte Anweisung. Sie wird ausgeführt, falls das Feld in Blickrichtung unmittelbar vor der Ameise Nahrung enthält.
- Zusätzlich: PROGN begrenzt Blöcke von Anweisungen beliebiger Länge (vgl. LISP-Funktion PROGN).

Entwicklung

Initialisierung: 500 zufällige Programme

Fitness: Anzahl der gefressenen Nahrungsstücke binnen 400 Programmschritten

Es entstehen Programme mit typischem Verhalten und (zunächst) geringer Fitness:

(PROGN (MOVE) (MOVE)) läuft immer geradeaus. Fitness 3 von 90.

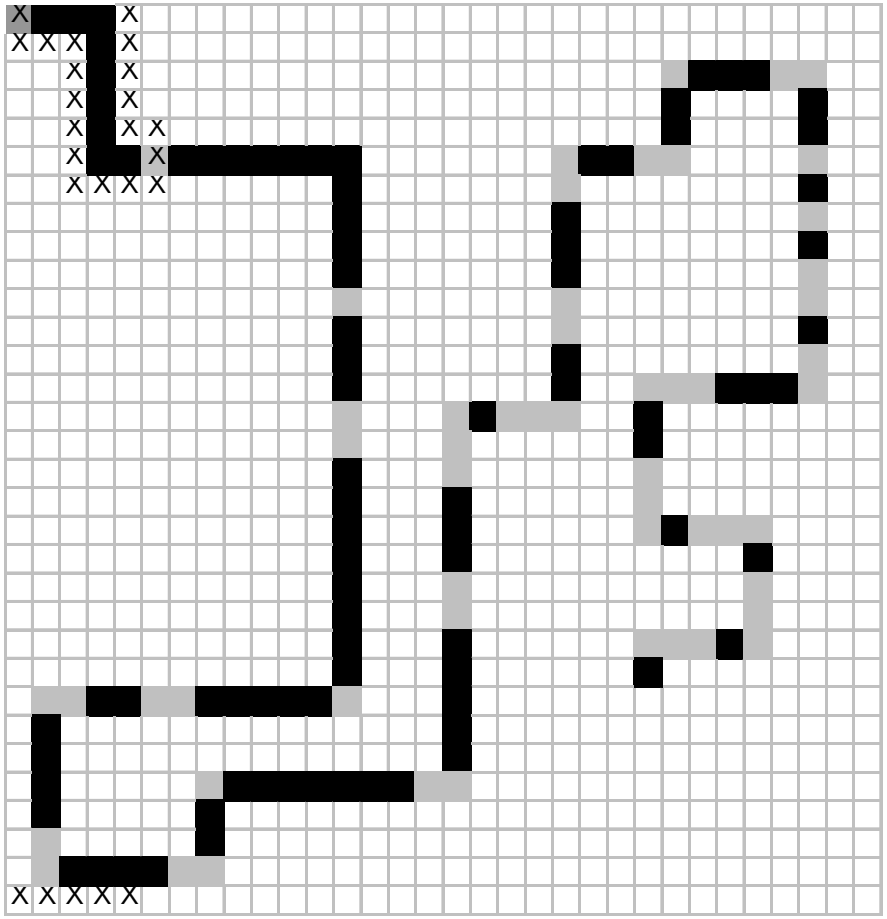
Übungsaufgabe: Wie bewegt sich der folgende „Quilter“?

```
(PROGN (RIGHT)
        (PROGN (MOVE) (MOVE) (MOVE))
        (PROGN (LEFT) (MOVE)))
```

Der „Avoider“

Der Avoider beobachtet die Nahrungsbrocken genau. Er läuft neben dem Weg entlang und nutzt die erste Lücke, um hindurchzuschlüpfen und heimzulaufen.

```
(IF-FOOD-HERE (RIGHT)
    (IF-FOOD-HERE (RIGHT)
        (PROGN (MOVE)
               (LEFT))))
```

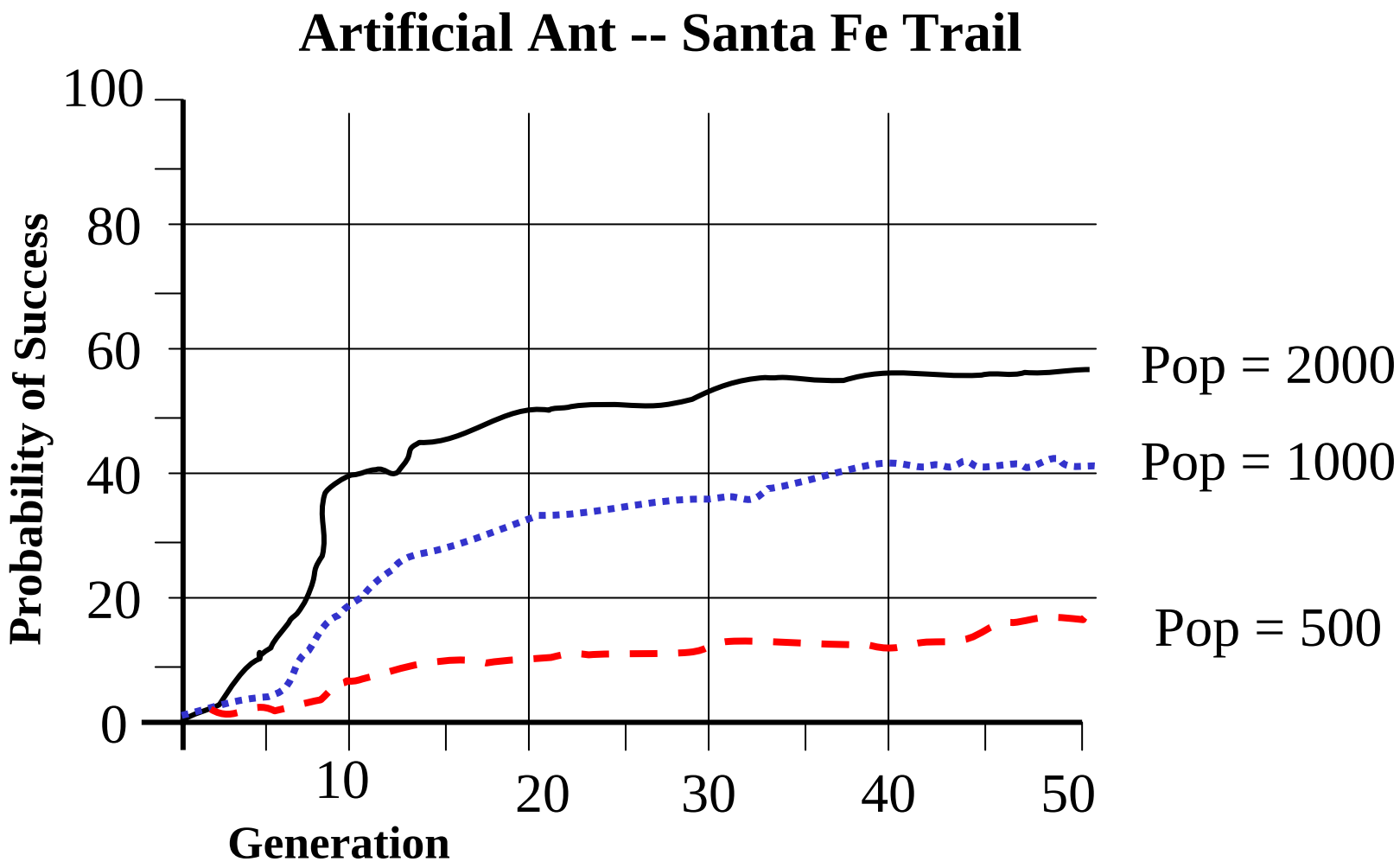


Erfolgreiches Programm

Das folgende Programm leert erfolgreich den ganzen Weg:

```
(IF-FOOD-HERE (MOVE)
              (PROGN (LEFT)
                     (PROGN (IF-FOOD-HERE (MOVE)
                                          (RIGHT))
                     (PROGN (RIGHT)
                            (PROGN (LEFT)
                                   (RIGHT))))
                     (PROGN (IF-FOOD-HERE (MOVE)
                                          (LEFT))
                            (MOVE))))
```

Abhängigkeit von der Populationsgröße



Emergentes Verhalten

Emergentes Verhalten eines Systems ist das scheinbar sinnvolle Zusammenwirken von Komponenten (hier: Individuen), so daß die Fähigkeiten des Systems weit über die Fähigkeiten der Komponenten hinausgehen.

Dieses Zusammenwirken wird dabei nicht durch ein hierarchisches System organisiert, sondern entsteht „von selbst“ durch die Interaktion der Komponenten.

Emergentes Verhalten eines Ameisenvolkes

Aufgabenstellung: Das Ameisenvolk soll Futter zu seinem Bau transportieren. Da möglicherweise viel Futter an einer Stelle zu finden ist, soll anderen Ameisen die Orientierung erleichtert werden, indem Duftmarken (Pheromone) gesetzt werden.

Die Welt besteht aus

- 32x32 - Feld
- 144 Portionen Futter in zwei Blöcken 3x3 Grundfläche, Höhe 8
- 20 Ameisen, beschrieben durch Position und Blickrichtung (8 Möglichkeiten)

Funktionen der Ameisen I

- MOVE-RANDOM: Zunächst Blickrichtung zufällig ändern, dann 2 Schritte in diese Richtung
- MOVE-TO-NEST: Ein Schritt in Richtung Bau.
- PICK-FOOD: hebt Nahrung (falls vorhanden) auf, falls nicht schon welche transportiert wird
- DROP-PHEROMONE: Setzt eine Duftmarke, falls Nahrung transportiert wird. Die Duftmarke ist als Wolke im Bereich der Größe 3x3 wahrnehmbar. Sie verschwindet nach einiger Zeit.
- IF-FOOD-HERE
- IF-CARRYING-FOOD

Funktionen der Ameisen II

- MOVE-TO-ADJACENT-FOOD-ELSE: Falls Nahrung in einer oder mehreren der Nachbarzellen, so bewegt sich die Ameise zu demjenigen Feld mit Nahrung, für das die geringste Richtungsänderung nötig ist. Sonst ELSE-Zweig verfolgen.
- MOVE-TO-ADJACENT-PHEROMONE-ELSE: analog
- PROGN: wie immer

Leben des Ameisenvolkes

Alle Ameisen verfügen über die gleiche Funktion. In jedem Zeitschritt wird diese für jede Ameise ausgeführt.

Zu Beginn befinden sich alle Ameisen im Nest mit zufälliger Blickrichtung.

Der Versuch läuft über eine bestimmte, hinreichend große Zahl von Schritten.

Fitness: Wieviel der 144 Futterstücke werden nach Hause gebracht?

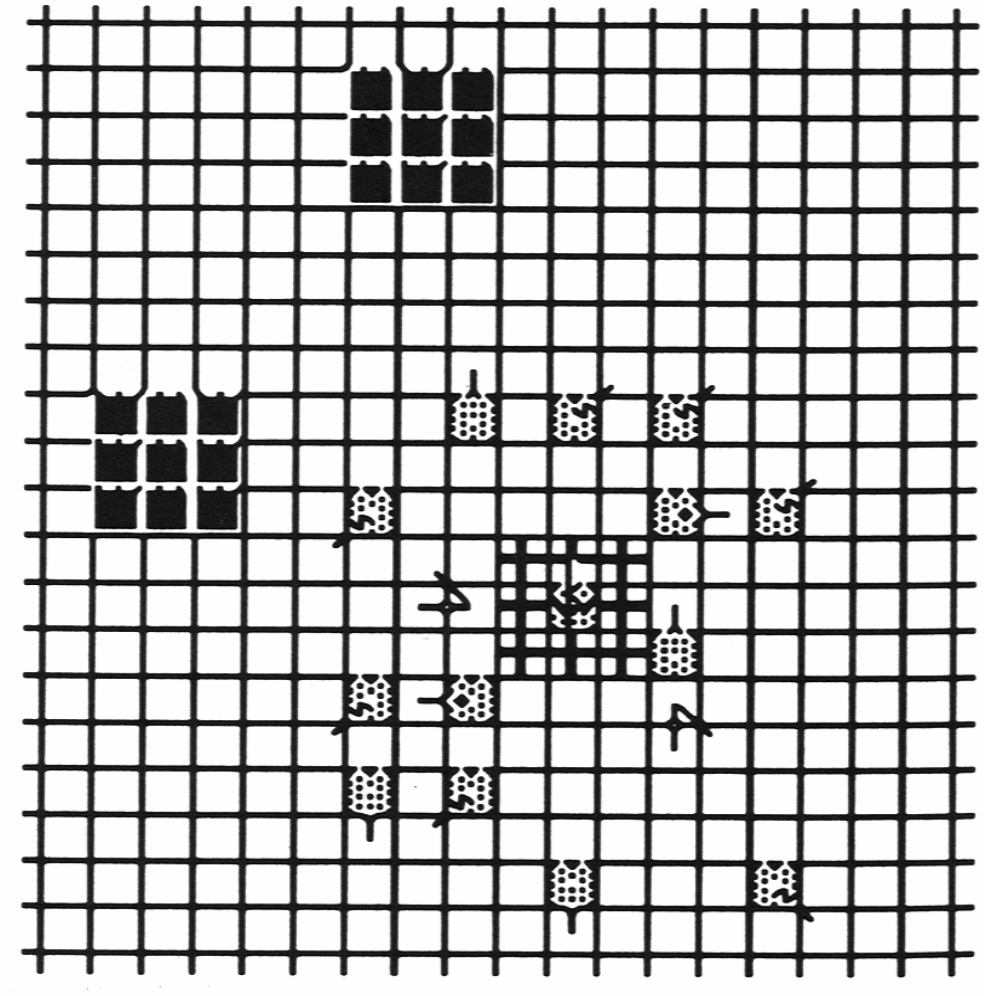
Zufällig erzeugte Programme: 93% haben Fitness 0, weitere 3% haben Fitness 1 (von 144)

Das erfolgreiche Programm

```
(PROGN (PICK-UP)
    (IF_CARRYING_FOOD
        (PROGN (MOVE-TO-ADJACENT-PHEROMONE-ELSE
                 (MOVE-TO-ADJACENT-FOOD-ELSE (PICK-UP)))
               (MOVE-TO-ADJACENT-FOOD-ELSE (PICK-UP))
               (MOVE-TO-NEST)
               (DROP-PHEROMONE)
               (MOVE-TO-NEST)
               (DROP-PHEROMONE))
        (MOVE-TO-ADJACENT-FOOD-ELSE
               (IF-FOOD-HERE
                 (PICK-UP)
                 (MOVE-TO-ADJACENT-PHEROMONE-ELSE
                    (MOVE-RANDOM)))))))
```

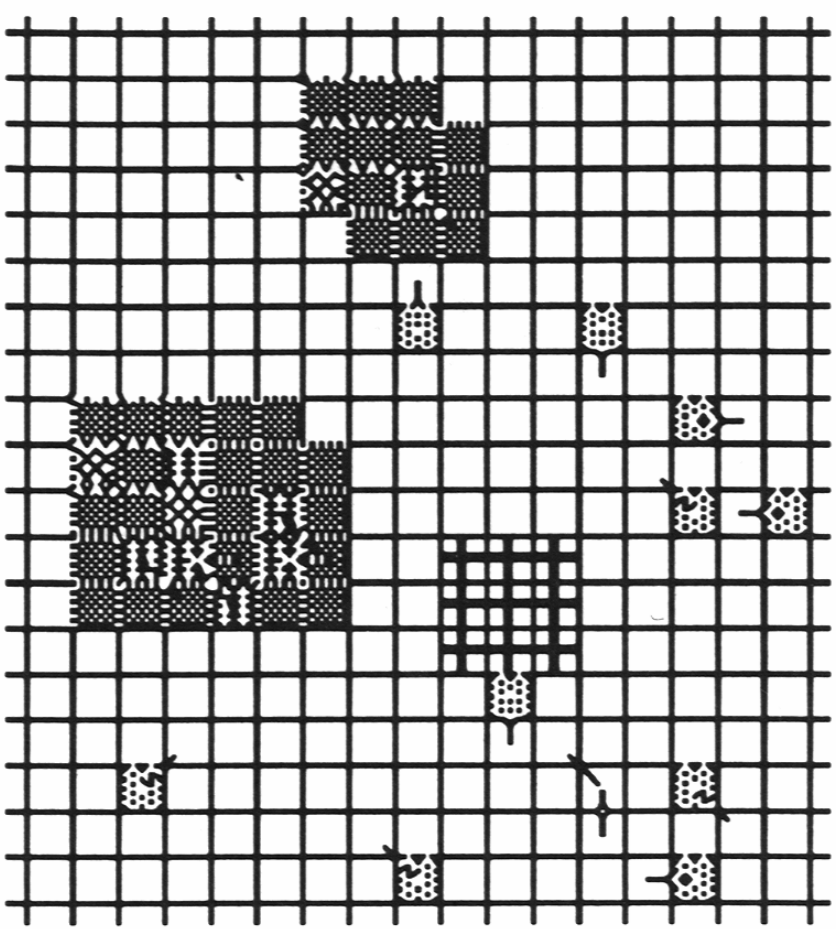
Das Leben zur Zeit 3

Die Ameisen haben den Bau verlassen und irren herum.



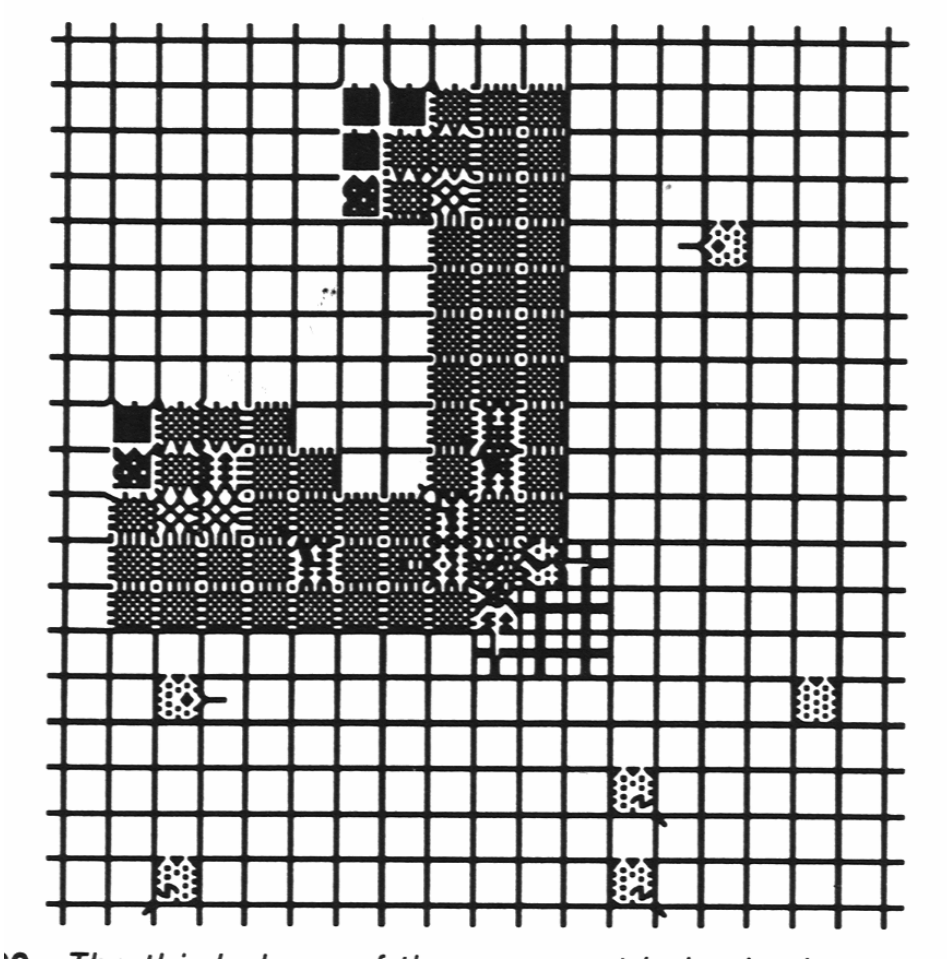
Das Leben zur Zeit 12

Einige Ameisen haben Futter gefunden und setzen Duftmarken um die Fundstelle.



Das Leben zur Zeit 15

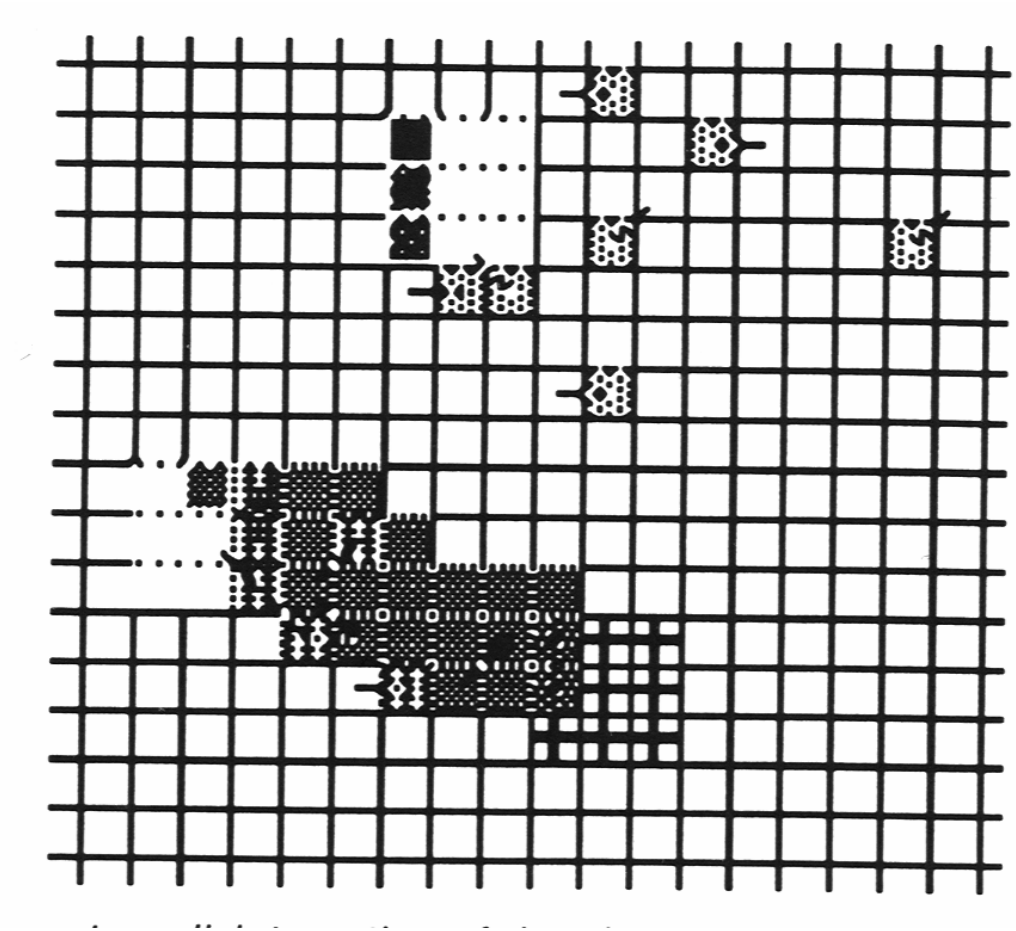
Die ersten zwei Portionen Futter sind heim gebracht, es gibt eine Duftspur, die den Weg anzeigt.



Das Leben zur Zeit 129

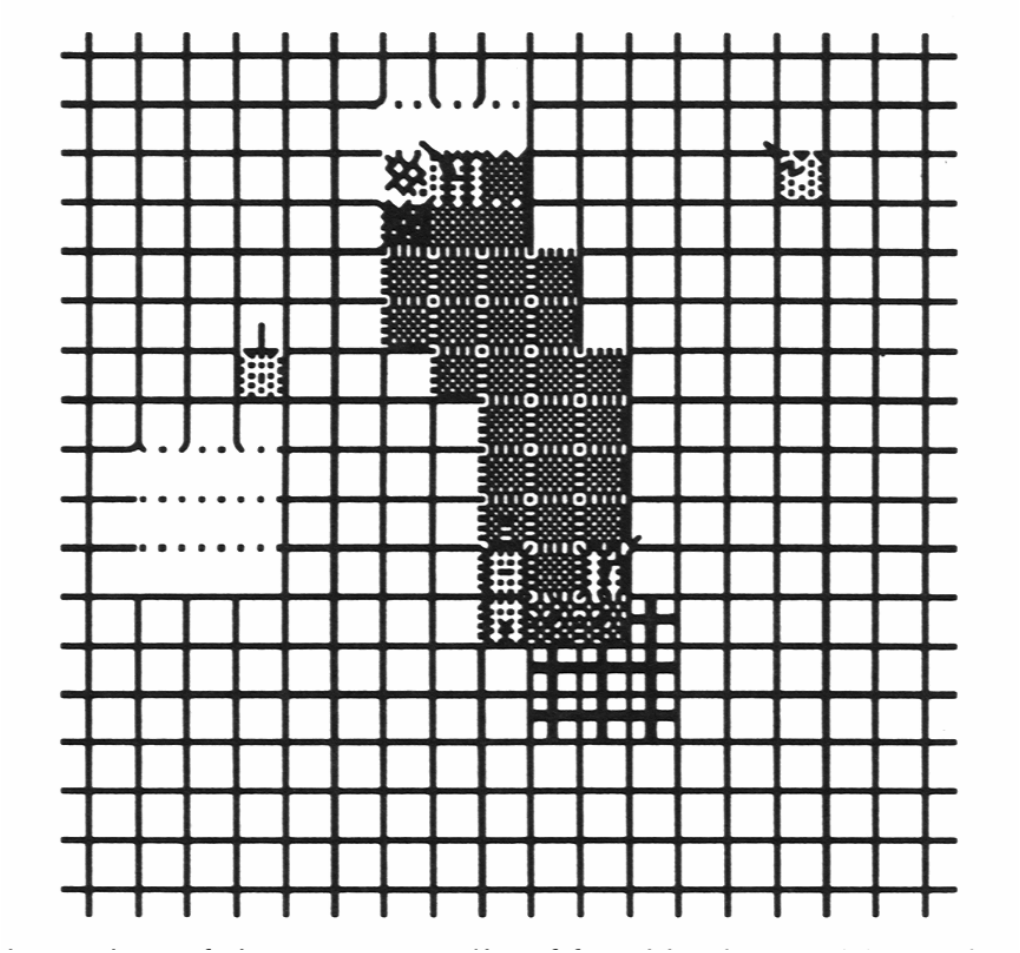
118 der 144 Portionen Futter sind heim gebracht, der obere Duftweg zerfällt, obwohl dort noch Futter ist.

Der linke Duftweg bleibt intakt.



Das Leben zur Zeit 152

Der linke Stapel ist abgeräumt, der Duftweg zum oberen Stapel wieder aufgebaut. Die verbliebenen Portionen werden sicher nach Hause gebracht.



Food Collection Genetic Program Parameters I

Parameter	Default Value
POPULATION_SIZE	1000

The number of ant programs in the population. This is the number of random solutions that will be creared when the GP starts.

TOURNAMENT_SIZE	4

The number of ant programs from the population that are selected for the tournament for crossover. The best two solutions from the tournament are used in the crossover, and replace the worst two solutions in the tournament.

FOOD_LEFT_WEIGHT	10000000

How much each piece of food not returned to the nest affects the fitness of the ant program.

FOOD_UNFOUND_WEIGHT	100000

How much each piece of food not found affects the fitness of the ant program.

NODE_WEIGHT	10000

How much the size of the ant program affects the fitness of the ant program.

Food Collection Genetic Program Parameters II

NODE_GROUP 50

Used to group together ant programs with similar sizes, so that NODE_WEIGHT has a less affect on fitness. Thus, all other factors being equal, a solution with 1 node and a solution with 49 nodes will have the same fitness, while a solution with 50 nodes will be worse.

TIME_WEIGHT 1

How much the time taken to move all of the food to the nest affects the fitness of the ant program.

LEAF_ODDS 9

The chance that the next node will be a leaf node, when creating the initial population. The chance the node will be a leaf node is LEAF_ODDS / (LEAF_ODDS + NON_LEAF_ODDS)

NON_LEAF_ODDS 6

The chance that the next node will be a non-leaf node, when creating the initial population. The chance the node will be a non-leaf node is NON_LEAF_ODDS / (LEAF_ODDS + NON_LEAF_ODDS)

Food Collection Genetic Program Parameters III

MUTATE_ODDS 100

The odds that the new ant programs created by crossover will be mutated.

PERCENT_GREEDY_MUTATE 80

The odds that the mutation will be greedy (fitness must increase with mutation).

MAX_TURNS 1000

The maximum number of turns the ants are given to find all of the food. Each move forward counts as a turn.

NUM_ANTS 15

The number of ants that execute the ant program in the environment.

GOAL_FITNESS 0

The fitness that must be achieved for the genetic program to end.

WAIT_ODDS 0

The chance that an ant waiting for help to pick up food will leave instead.

AUTO_FOOD_DROP 1

Boolean for automatically dropping food when the ant is over the nest.

Food Collection Genetic Program Parameters IV

PHEROMONE_STRENGTH 300

The amount of pheromone released on a RELEASE PHEROMONE command.

PHEROMONE_SPREAD 50

The amount of pheromone that is spread to adjacent squares each iteration.

POP_SEED 0

The random number generator seed when creating the initial population.

INTERP_SEED 0

The random number generator seed when executing the ant program.

0 = no new seed.

Auf Wiedersehen

zur Prüfungsklausur für ADS

am 17. Juli 2007