KI_Praktikum_2

Güray Ercan, Elias Schrüfer, Gruppe 3

May 2025

Inhaltsverzeichnis

1 Einführung

Die Suche nach einem Weg durch ein Labyrinth ist ein klassisches Problem der Informatik. In dieser Arbeit werden zwei heuristische Ansätze zur Lösung untersucht: Simulated Annealing und Ant Colony Optimization (ACO) (auch Ameisenalgorithmus).

Beide Algorithmen verfolgen unterschiedliche Strategien, um einen möglichst kurzen Pfad vom Start zum Ziel zu finden. Ziel der Arbeit ist es, die beiden Verfahren in Bezug auf Lösungsqualität und Effizienz zu vergleichen und herauszufinden, welcher Ansatz sich besser für die Pfadsuche im Labyrinth eignet.

2 Forschungsfrage

Ziel dieser Arbeit ist es, die beiden Verfahren hinsichtlich ihrer Effizienz und Lösungsqualität bei der Pfadsuche in Labyrinthen zu vergleichen.

Dabei soll untersucht werden, unter welchen Bedingungen eines Labyrinthproblems sich signifikante Unterschiede zwischen ACO und SA zeigen, wie sich Parameter wie Abkühlrate bzw. Alpha und Beta bei ACO auf die Ergebnisqualität auswirken und welche Methode in Bezug auf Laufzeit und Erfolgsrate vorteilhafter ist.

Die zugrunde liegende Forschungsfrage lautet: „Welche Unterschiede zeigen sich zwischen ACO und Simulated Annealing in Bezug auf Effizienz und Lösungsqualität bei der Pfadsuche in Labyrinthen?“

Zur Beantwortung dieser Frage werden beide Algorithmen implementiert, auf identischen Testumgebungen ausgeführt und anhand definierter Metriken miteinander verglichen.

3 Theorie

3.1 Ant Colony Optimization

Die Ant Colony Optimization (ACO) ist ein von biologischen Ameisen inspiriertes Verfahren, das auf kollektiver Intelligenz basiert. Die Grundidee: Ameisen hinterlassen auf ihrem Weg zur Nahrung Pheromone.

Andere Ameisen folgen mit höherer Wahrscheinlichkeit stark markierten Wegen. In der Informatik werden Pfade ähnlich aufgebaut: Jede virtuelle Ameise legt einen Pfad im Graphen (z. B. Maze) zurück, wobei die Auswahl probabilistisch auf der Basis von Pheromon und Heuristik erfolgt. Gute Lösungen erhalten durch Pheromonverstärkung ein positives Feedback.

Als Parameter erhält der Algorithmus :

- Alpha: Einfluss des Pheromons
- Beta: Einfluss der Heuristik (Distanz zum Ziel)
- Evaporation: Verdunstung des Pheromons

3.2 Simulated Annealing

Simulated Annealing (SA) ist ein probabilistischer Optimierungsalgorithmus, der von der Metallurgie inspiriert ist – konkret vom Abschreckprozess (engl. annealing), bei dem ein Material durch kontrolliertes Abkühlen in einen energetisch günstigen Zustand überführt wird.

Das Ziel des Algorithmus ist es eine (möglichst optimale) gute Lösung in einer großen oder komplexen Suchlandschaft zu finden, auch wenn sie lokale Minima enthält.

Der Algorithmus startet so mit einer anfänglich hohen Temperatur. Bei jeder Iteration wird eine neue Nachbarlösung erzeugt. Diese wird:

1. immer akzeptiert, wenn sie besser ist,

2. manchmal akzeptiert, wenn sie schlechter ist — abhängig von der Temperatur.

Dadurch kann der Algorithmus lokale Minima verlassen und hat eine Chance, das globale Optimum zu finden.

Die Gefahr von diesem Algorithmus ist, dass kein globales Optimum garantiert wird. Wenn die Abkühlung zu schnell erfolgt (z.B. Alpha zu niedrig). Auch andersherum birgt dies eine Gefahr : Sinkt die Temperatur zu langsam, so werden schlechtere Lösungen tendenziell weiterhin akzeptiert; suboptimale Pfade können in der Folge als Lösung akzeptiert werden.

3.3 Anwendung der Algorithmen auf das Lösen von Labyrinthen

In dem Labyrinth wird eine Lösung durch einen Pfad vom Start zum Ziel repräsentiert. Ein Schritt wird immer als Weg von einer Zelle zu einer benachbarten Zelle angesehen.

Der Pfad kann (und wird in dieser Implementierung auch) als Liste von Koordinaten dargestellt werden:

$$[(0,0),(0,1),(1,1),...,(Ziel)]$$

Der Algorithmus muss in einer Art und Weise die Fitness der Nachbarschaftszellen evaluieren. Dies geschieht anhand einer Heuristik; in einem Labyrinth bieten sich hier die euklidische oder auch die Manhattan-Distanz an (je näher eine Zelle am Ziel ist, desto besser). In dieser Implementierung wurde sich für die euklidische Distanz entschieden.

Lokale Maxima (für Simulated Annealing) werden als Sackgassen in einem Labyrinth betrachtet, die den Weg zum Ziel verschließen.

4 Versuchsbeschreibung

4.1 Simulated Annealing

Die Art und Weise der Implementierung des Algorithmus wird anhand des folgenden Ablaufdiagramms erklärt bzw. erläutert.

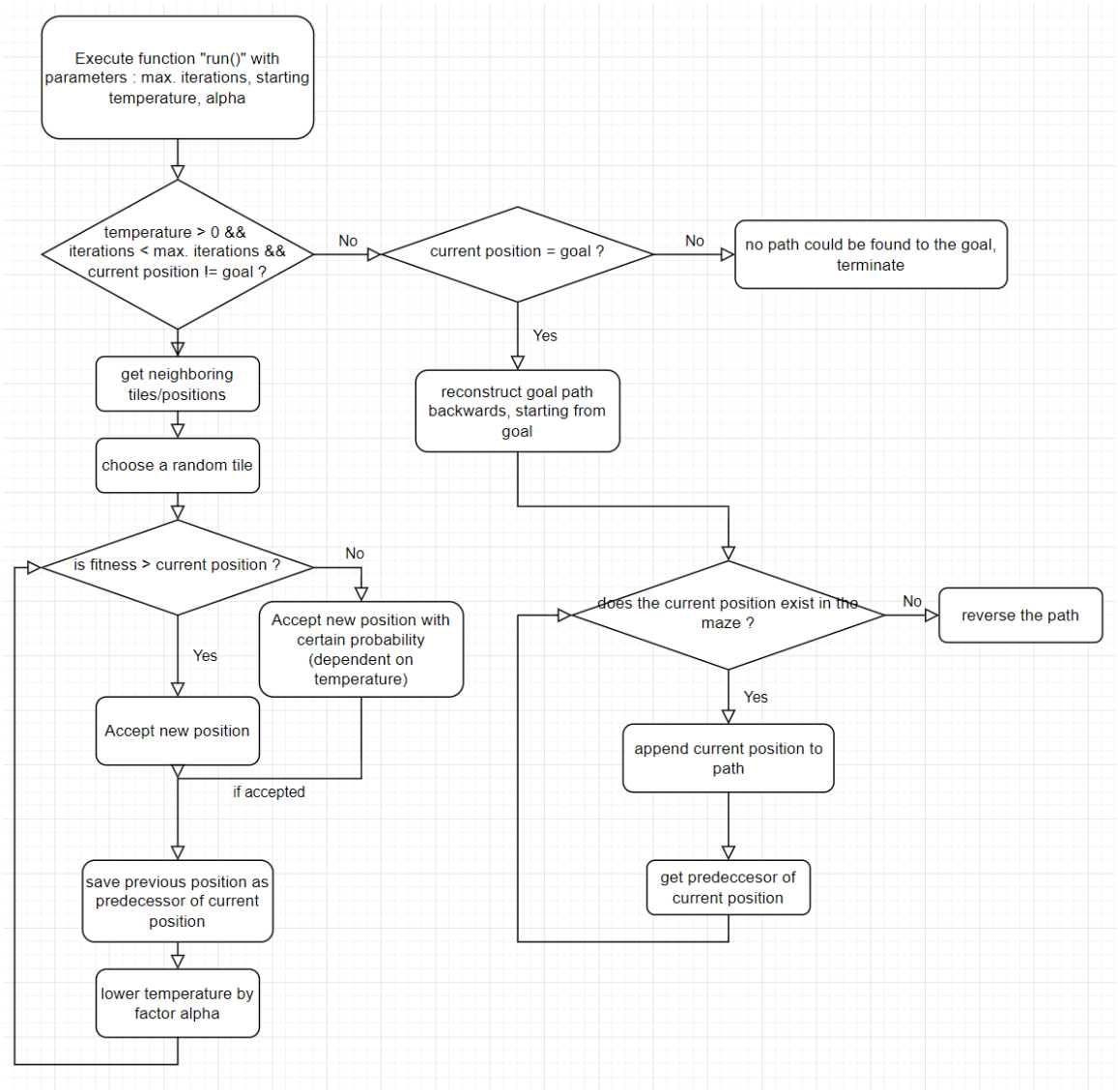

Abbildung 1: Ablaufdiagramm für Simulated Annealing

4.2 Ameisenalgorithmus (ACO)

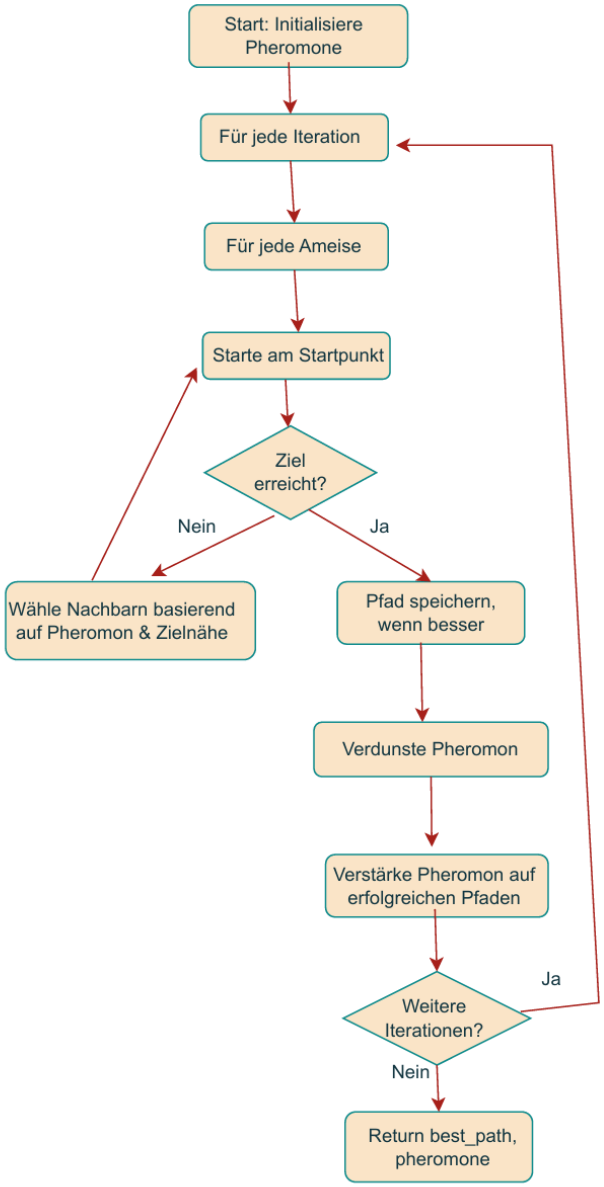

Abbildung 2: Ablaufdiagramm für ACO

5 Ergebnis

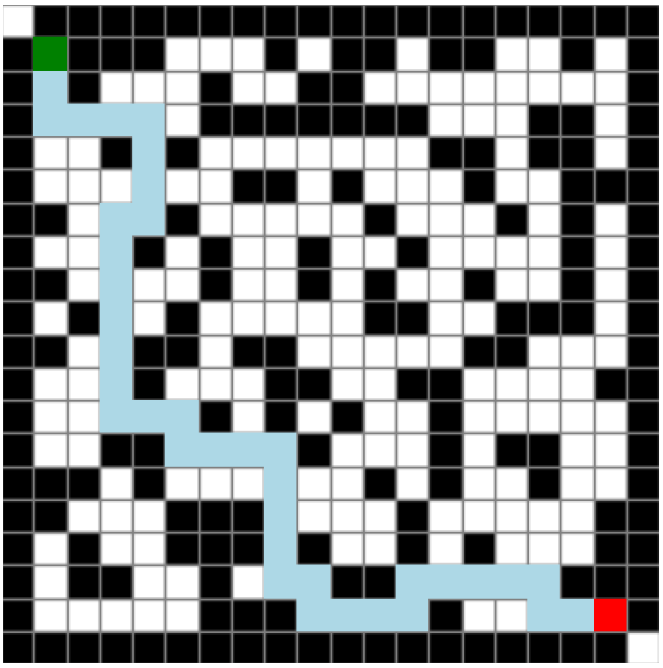

Abbildung 3: 20x20 Labyrinth mit Beispielslösung

ITERATIONS	ALPHA	BETA	SUCCESS_RATE	TIME PER RUN (S)	AVG PATH LENGTH
20	1	2	79%	0.18953	40.92
20	1	5	95%	0.20998	39.82
20	5	1	52%	0.23769	42.77
50	1	2	96%	0.55515	39.8125
50	3	1	94%	0.61821	42.43
50	1	5	100%	0.61230	39.42
50	5	1	89%	0.55200	41.16

Abbildung 4: ACO für 20x20

TEMPERATURE	ALPHA	SUCCESS_RATE	TIME PER RUN	AVG PATH LENGTH
10000	0.95	6%	0.00072	41.33
100000000	0.95	8%	0.00115	44.0
10000	0.99	23%	0.00232	45.14
100000000	0.99	32%	0.00375	45.89

Abbildung 5: Simulated Annealing für 20x20

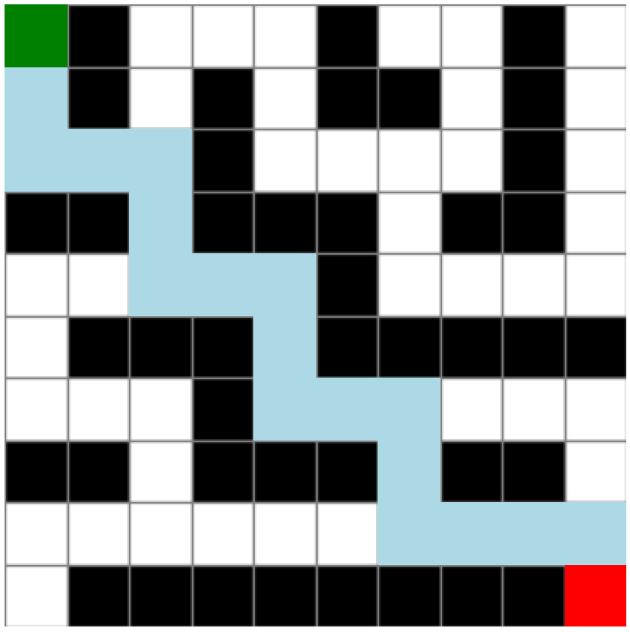

Abbildung 6: 10x10 Labyrinth mit Beispielslösung

ITERATIONS	ALPHA	BETA	SUCCESS_RATE	TIME PER RUN (S)	AVG PATH LENGTH
20	1	2	100%	0.12246	19.0
20	1	5	100%	0.12816	19.0
20	5	1	100%	0.12521	19.0
50	1	2	100%	0.30986	19.04
50	5	1	100%	0.30924	19.0

Abbildung 7: ACO für 10x10

TEMPERATURE	ALPHA	SUCCESS_RATE	TIME PER RUN	AVG PATH LENGTH
10000	0.95	73%	0.00049	20.42
100000000	0.95	70%	0.00078	20.76
10000	0.99	84%	0.00125	20.71
100000000	0.99	95%	0.00158	20.70

Abbildung 8: Simulated Annealing für 10x10