Forschungsprojekt

Swarm Intelligence

Dokumentation
Professur Virtuelle Realität
Bauhaus-Universität Weimar

Andrea Lahn,
Nicolai Marquardt,
Christian Nitschke

Projektbetreuung:
Hendrik Wendler,
Prof. Bernd Fröhlich

Sommersemester 2004

'Das Ganze ist mehr als die Summe seiner Teile.'
Aristoteles

Fakultät Medien, Professur Virtuelle Realität
Projekt-Leitung: Hendrik Wendler und Prof. Bernd Fröhlich
Teilnehmer: Andrea Lahn, Nicolai Marquardt, Christian Nitschke

Kontakt:
bernd.froehlich@medien.uni-weimar.de
andrea.lahn@medien.uni-weimar.de
nicolai.marquardt@medien.uni-weimar.de
christian.nitschke@medien.uni-weimar.de

Autoren:
Kapitel 1 und Abstract: Andrea Lahn, Christian Nitschke, Nicolai Marquardt
Kapitel 2: Christian Nitschke
Kapitel 3: Christian Nitschke
Kapitel 4: Andrea Lahn
Kapitel 5: Christian Nitschke
Kapitel 6: Nicolai Marquardt
Kapitel 7: Nicolai Marquardt

Abstract:
Swarm intelligence

The basic principle of the swarm intelligence algorithms is to divide complex calculations between multiple, simple executive agents. In these cases we use the emergent collective intelligence of groups of these simple agents (Bonabeau). These computer science algorithms are inspired by observations of real ant swarms, since they solve complex tasks by simple local behaviour and activities. The swarm is capable to solve complex problems, while at the same time each individual ant has no overall view of the situation.
With our swarm simulation implementations, various SI algorithms can be used to cluster data sets and display the virtual swarm environment with a two or three dimensional visualization. The applied algorithms are founded on the research of V. Ramos, J. Handl, M. Dorigo, E. Bonabeau, E. D. Lumer, B. Faieta and some further researchers of the swarm intelligence science community. Beyond this literature research in the SI domain we have implemented simulation systems inspired by the algorithms of the current research projects of Ramos, Dorigo et al. Furthermore we have developed various extensions to the original systems, such as dynamic pheromone evaporation, object switching while carrying another object, dynamic picking and dropping probabilities and varying border conditions as well as changes of the world dimensions in two and three dimensions as well as elegant solutions for some problems concerning orientation and boundary conditions. Some of these different extensions to the initial simulation systems are leading to measurable changes of the swarm behaviour, while other extensions have not changed the behaviour significantly. All extensions have been evaluated and are described in the chapters three, four and five of this project documentation.
For the evaluation of the developed systems input data is essential. One could simply generate object-features that are randomly distributed over their particular dimensions. However this technique brings less benefit. What's necessary here are some artificial generated object clusters. We therefore implemented a functionality for the flexible generation of gaussian distributed objects in n-dimensional feature-space. The configuration is done via self explaining config-files, where to specify the parameters of the distribution for the particular features of the objects. The algorithm can be used as a function included with the developed swarm systems as well as a stand-alone programm creating a text-file containing the object-data for further usage.
The results and the states of a swarm-clustering have to be measured and interpreted. So the need for robust entropy-measures arises. Therefore we have developed a set of measures namely sorting and connectivity, that can be applied in every step of the clustering-process. It provides us with information about the current state of the information infrastructure, that is the world of the swarm system. The gathered information is necessary for logging the performance in progression of the algorithms.

Here Sorting is deals with the feature-space similarity of neighboring objects on the information infrastructure of the swarm. Connectivity measures the density distribution of the objects as well.
For the visualization of the environment of the swarms we used two dimensional worlds, as well as varying three dimensional worlds (e.g. cube, dish and tube). We have analyzed the swarms sorting behaviour in the environments, and also evaluated the main swarm features: flexibility, robustness, decentralized organization and self-organization of the swarm. The self-organization of the swarm is based on the feedback that each agent can transmit with the change of the swarm's environment (e.g. picking or dropping) or the deposition of pheromones.
With the applied evaluation and the variety of simulations we obtained a profound insight into the paradigms of swarm intelligence behaviour. We have observed advantages as well as disadvantages of these systems. With this in mind we see some very interesting further development areas in this research field. Future work includes the application of swarm intelligence algorithms for data mining, further research with three dimensional swarm worlds and also the implementation of fast swarm simulation systems with up to date GPU and CPU combinations.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung und Grundlagen

Auch wenn die Algorithmen und Methoden zur *Swarm Intelligence* erst in den letzten Jahren entwickelt wurden, so sind die zu Grunde liegenden Ideen doch schon lange bekannt. Schon immer war die Natur Inspiration für die Entwicklungen in der Technik und dort haben die Swarm- bzw. Multi-Agenten-Systeme auch ihren Ursprung. Aus Beobachtungen der Biologie ließen sich nun Mathematiker und Informatiker inspirieren, um durch die nachgebildeten Systeme das Verhalten und die Aktionen von Schwärmen des Tierreichs nachzuahmen sowie die über Generationen in der evolutionären Geschichte gewonnenen Erfahrungen von natürlichen Schwärmen zu extrahieren und nutzbar zu machen. Ein Ameisenschwarm beispielsweise besteht aus sehr vielen autonom handelnden aber kaum spezialisieren Individuen, die keine konkrete Vorstellung von ihrer vollständigen Umgebung besitzen. Auch zu lösende Probleme mit denen der gesamte Schwarm konfrontiert wird, sind von den einzelnen Ameisen in der Regel nicht zu überschauen. Allein das Arbeiten und Wirken nach einer Anzahl vorgegebenen Regeln führt zur impliziten globalen Problemlösung. Wie solche Systeme funktionieren, wie man sie auf Computern abbilden kann und wo die Anwendungen dafür liegen, darauf werden wir in dieser Dokumentation umfassend eingehen.
Die wichtigen Prinzipien zu Swarm-Systemen folgen in den Abschnitten dieses Kapitels. Es folgt ein umfassender Literaturüberblick zu den Schwärmen und der *Swarm Intelligence* im Kapitel 2. Im dritten Kapitel beschreiben wir Details zu allgemeinen Modulen zur Generierung von Clustering-Daten und Algorithmen zur Bestimmung der aktuellen Sortierung der Objekte, auf die wir in den Simulations-Programmen zurückgreifen. In den darauf folgenden drei Kapiteln werden wir die Prinzipien und Technologien der drei von uns entworfenen Simulations-Umgebungen sowie deren Vorteile, Nachteile und Anwendungsmöglichkeiten beschreiben. Dies sind das 2D-System von Andrea Lahn in Kapitel 4, die Simulations-Umgebung nach dem Algorithmus von J. Handel [Handl et al. 2003a] von Christian Nitschke in Kapitel 5 sowie im sechsten Kapitel das 2D/3D System nach dem Lumer&Faieta- sowie Ramos-Algorithmus [Lumer & Faieta 1994] [Ramos et al. 2002] von Nicolai Marquardt. Ausführlich wird in diesen drei Kapiteln auch auf die aufgetretenen Probleme und Einschränkungen der Systeme sowie die von uns implementierten Erweiterungen bestehender Algorithmen eingegangen. Desweiteren erläutern wir die durchgeführten Testserien der Schwarm-Simulationen und deren Ergebnisse. Schließlich folgt im letzten Kapitel eine kurze Zusammenfassung unserer Forschungen sowie ein Ausblick für die weitere Entwicklung der Systeme.

1.1 Systeme

Zu einem Swarm-System gehörten prinzipiell nur zwei Bestandteile: Die künstlichen Individuen (Agenten, Ameisen), die nach einer bestimmten Menge von Regeln arbeiten und die künstliche Welt in der sie agieren.
Dabei sind die Agenten nur mit sehr begrenzten Fähigkeiten ausgestattet und meist nicht spezialisiert. Sie nehmen im Allgemeinen nur das lokale Umfeld und nicht die gesamte Welt wahr und können dementsprechend nur in diesem beschränkten Raum handeln bzw. Entscheidungen treffen. In manchen Systemen können sie ein Kuzzeitgedächtnis besitzen, um sich bestimmte Handlungen und Zustände zu merken.
Ein großer Vorteil von Swarm-Systemen liegt in der Abwesenheit direkter Kommunikation. Zwar sind aus der Natur einige Beispiele bekannt, wie Insekten direkt miteinander kommunizieren, doch kommen diese bei Swarm Intelligence nicht zur Anwendung. Indirekte Kommunikation über die Umgebung wird auch als Stigmergy bezeichnet [Grassé 1959]. Dabei ergibt sich die Kommunikation durch Veränderung der Umgebung und entsprechende Reaktionen.
Weiterhin gibt es in Swarm-Systemen keine globale Kontrollinstanz. Die Agenten handeln völlig autark.

Swarm-Systeme können zur Bearbeitung ganz unterschiedlicher Probleme eingesetzt werden. Dadurch ergeben sich verschiedene Strukturen für den Aufbau der künstlichen Welt. Sie kann beispielsweise wie bei ant-based optimization und ant-based routing durch einen Graphen repräsentiert sein oder wie bei ant-based clustering und sorting durch eine Matrix von Positionen.

1.2 Prinzipien

Aus den Prinzipien in Swarm-Systeme ergeben sich wichtige Vorteile gegenüber konventionellen Systemen und Algorithmen.

1. **Dezentralität**
   Es gibt keine zentrale Kontrolle. Die Agenten handeln völlig autonom. Die globale Instanz übernimmt lediglich die Funktion eines Schedulers und weist den Schwarmmitgliedern Zeit zur Durchführung ihrer Aktionen zu.

2. **Selbstorganisation**
   Bei der Bearbeitung eines Problems gibt es keinen vordefinierten Lösungsweg oder Endzustand. Die Agenten agieren stets nach ihren Regeln und finden so implizit einen optimalen Zustand.

3. **Flexibilität**
   Da die Agenten kein Modell von ihrer Umgebung haben, bleiben sie flexibel bei Änderungen in ihrer Welt. So ist es möglich auch während das System läuft, beispielsweise zusätzliche oder neue Objekte hinzuzufügen.

4. **Robustheit**
   Da die Agenten wenig bis gar nicht spezialisiert sind, gefährdet ein Ausfall eines oder mehrerer Individuen nicht die Lösung des Problems.

Das Zusammenwirken der vier Prinzipien führt zur gewünschten Problemlösefähigkeit von Swarm-Systemen. Wenn die Selbstorganisation bzgl. der künstlichen Welt so abstrahiert werden kann, daß sich ein globaler Problemlösecharakter abzeichnet, spricht man von *Emergenz*. Dieses Eigenschaft läßt sich treffend mit dem bekannten Zitat von Aristoteles zusammenfassen: „Das Ganze ist mehr als die Summe seiner Teile“.

1.3 Motivation und Ziele

Im Rahmen dieses Forschungsprojektes haben wir uns aus dem umfangreichen Forschungsgebiet der *Swarm Intelligence* Technologien insbesondere die folgenden Ziele und Aufgaben vorgenommen.

- **Funktionsweise der Systeme**
  Basierend auf den Forschungen von Dorigo, Bonabeau, Ramos, Théraulaz, Handl und anderen möchten wir die zahlreichen unterschiedlichen Verfahren eingehender untersuchen. Insbesondere die Verfahren zum *Clustering* möchten wir implementieren, um deren Möglichkeiten und Leistungen beurteilen zu können.

- **System-Erweiterungen**
  Mit welchen Erweiterungen lassen sich die Swarm-Systeme ergänzen, um bessere Ergebnisse zu erhalten? Mit dieser Frage werden wir uns beschäftigen und in die zu entwickelnden Simulations-Systeme solche Erweiterungen integrieren und untersuchen. In den drei Kapiteln zu den Simulations-Systemen (4-6) werden wir näher darauf eingehen.

- **Visualisierung**
  Anwendungsgebiete von Swarm-Intelligence-Systemen wurden im vorhergehenden Abschnitt bereits aufgeführt, wobei uns insbesondere interessiert, wie die Systeme zur Visualisierung von Daten angewendet werden können. Vor allem sehr große Datensätze mit hoher Parameter-Anzahl machen es notwendig, geeignete Verfahren zur Visualisierung anzuwenden, die den hochdimensionalen Datenraum auf zwei

oder drei Dimensionen abbilden können.
Wie diese Abbildungen der Daten in den SI Systemen möglich ist, möchten wir mit unterschiedlich aufgebauten Schwarm-Umgebungen im zwei- und dreidimensionalen Raum untersuchen. Eine Frage dabei ist sicher, mit welchem Aufbau der Umgebung sich die besten Clustering Ergebnisse erreichen lassen und gleichzeitig eine geeignete Visualisierung gewählt werden kann.

- **Abbildungen, insbesondere Vergleich zwischen 2D und 3D**
  Bisher werden zur Visualisierung der Daten beim *swarm-based Clustering* vorallem zweidimensionale Abbildungen eingesetzt. Wir möchten neben 2D-Abbildungen auch Abbildungen in drei Dimensionen anwenden, und ihre Möglichkeiten und Einschränkungen herausfinden.

- **Interaktion**
  Hier möchten wir der Frage nachgehen, in welchem Maße die Swarm-Systeme eine Nutzer-Interaktion während der Laufzeit ermöglichen sollten und dies eine Nutzung des Systems erleichtert oder erschwert. Auch sind Interaktionen während der Initialisierungs-Phase der Systeme notwendig, wo zahlreiche Parameter erfragt werden. Einige Parameter können von dem Simulations-System selbst bestimmt werden, für andere wiederum ist dies nicht möglich. Zusätzlich möchten wir die Auswirkungen von Parameteränderungen während einer laufenden Simulation untersuchen, sowie Einschränkungen für sinnvolle Parameter finden.

- **Evaluierung der Systeme**
  Um Erweiterungen der Systeme bewerten zu können wird es notwendig, Meß- und Evaluierungsverfahren zu implementieren. Diese ermöglichen sowohl die Bewertung der Erweiterungen als auch den Vergleich der Verfahren untereinander. Dazu werden sowohl Algorithmen zur Messung der Sortierung und der dichten Anordnung von Objekten entwickelt als auch geeignete Graphen zur Visualisierungen, die eine Kontrolle der Werte zur Laufzeit ermöglichen.

2 Anwendungen

Swarm Societies sind vor allem sehr interessant wegen ihrer effizienten natürlich etablierten Problemlösestrategien. Franks[Franks 1989] führte dazu Experimente mit der Ameisenart Lasius Niger durch und entdeckte einige nutzbringende Abläufe, wie beispielsweise:

- das Finden des kürzesten Weges zwischen Nest und potientiellen Nahrungsquellen,
- der Bevorzugung von Nahrungsquellen anhand ihrer qualitativen und quantitativen Güte,
- dem Bau und Schutz des Nests,
- der Sortierung von Brut, Nahrung, Körpern und Dingen,
- der Formation mehrerer Individuen, z.B. zum gemeinsamen Tragen größerer Teile bzw. Erstellen von „lebenden Brücken“ sowie
- der Emigration der Kolonie im Notfall.

Aus Beobachtungen von Forschern wie Franks entwickelten sich Theorien über Prinzipien und Anwendungen von Swarm Systemen[Bonabeau, Dorigo and Theraulaz 1999]. In diesem Zusammenhang wurden künstliche Swarms für verschiedenste Anwendungsgebiete wie Optimierung, Steuerung, Dezentralisierung, Sortierung/Clusterung oder Robotik entwickelt und untersucht. Im Folgenden werden jeweils Erkenntnisse aus der Natur und daraus resultierende technische Anwendungen vorgestellt.

2.1 Ant-Colony Optimization (ACO)

Unter Ant-Colony Optimization versteht man Strategien und Algorithmen zum Lösen von Problemen der kombinatorischen Optimierung basierend auf Erkenntnissen aus Ameisenkolonien. Vor allem die Suche nach Nahrungs- und Baumaterialquellen sowie das Aufbauen von Pfaden sind hier von Bedeutung.

Einer der Vorreiter auf dem Gebiet von ACO ist Marco Dorigo[Dorigo 2004]. Er entwickelte 1991 den ersten ACO-Algorithmus, das Ant-System (AS)[Dorigo, Maniezzo and Colorni 1991]. Seit dieser Zeit folgen viele weitere verbesserte und effizientere ACO-Algorithmen. Sehr gute Resultate lassen sich auch in Kombination mit lokalen Optimierungsverfahren (sog. hybride Verfahren) erzielen. Zu Problemen, die effizient mit ACO-Verfahren gelöst werden können, zählen das Traveling Salesmen Problem (TSP), das Quadratic Assignment Problem (QAP), das Graph Coloring Problem, das Job Scheduling Problem (JSP), das Sequential Ordering Problem (SOP) oder das Vehicle Routing Problem (VRP) aus der Tourenplanung[White 2000].

Alle gennanten Probleme sind statisch. ACO läßt sich aber auch in dynamischen, z.B. zeitvarianten Kontexten anwenden. Das zeigt der effiziente Algorithmus Ant-Net zur Verteilung von Netzwerk-Traffic[Schoonderwoerd, Holland and Bruten 1997]. Er wird zum adaptiven Aufbau von Routing-Tabellen in Daten-Netzwerken eingesetzt und übertrifft dabei andere aktuelle Verfahren in der Performance.

2.1.1 Natur

Viele Ameisenarten legen Pfade an, auf denen sich die Artgenossen bewegen. Wenn Ameisen in einem neuen Gebiet ansiedeln und auf Nahrungs- oder Materialsuche gehen, wissen sie anfänglich nicht, wo qualitative Quellen zu finden sind. Sie strömen in zufällige Richtungen aus und gehen auf Suche. Dabei hinterlassen sie einen chemischen Botenstoff, das Pheromon. Andere Ameisen orientieren sich, indem sie Pheromonspuren folgen.

Um die Entwicklung solcher Pheromonpfade zu untersuchen, führten Deneubourg et al.[Deneubourg, Goss, Franks and Pasteels 1990] ein interessantes Experiment (Binary-Bridge Experiment) durch. Dabei wurde eine Nahrungsquelle über zwei gleichlange Pfade mit dem Ameisennest verbunden. Die Ameisen müssen sich auf dem Weg an einer Gabelung zwischen Pfad A und Pfad B entscheiden. Nach einiger Zeit wird einer der beiden Pfade durch eine Erhöhung der Pheromonkonzentration etabliert. Die höhere Pheromonkonzentration veranlaßt weitere Ameisen, sich für diesen Weg zu entscheiden, was wiederum eine Erhöhung der Konzentration nach sich zieht, usw. Die Wahrscheinlichkeiten P_A und P_B, daß sich Ameisen für Weg A oder B entscheiden hängt also von der Anzahl der Ameisen A_i und B_i ab, die sich bis zu diesem Zeitpunkt für einen der Wege entschieden haben:

$$P_A = \frac{(k + A_i)^n}{(k + A_i)^n + (k + B_i)^n} = 1 - P_B$$

Dabei steht n für den Grad der Nichtlinearität der Auswahlfunktion und k für den Grad der Attraktivität eines nicht-markierten Zweigs.

Das Experiment kann leicht modifiziert werden, indem ein Pfad verlängert wird. Hierbei kann gezeigt werden, daß sich kürzere Pfade über die Zeit hin durchsetzen (Abbildung

2.1):

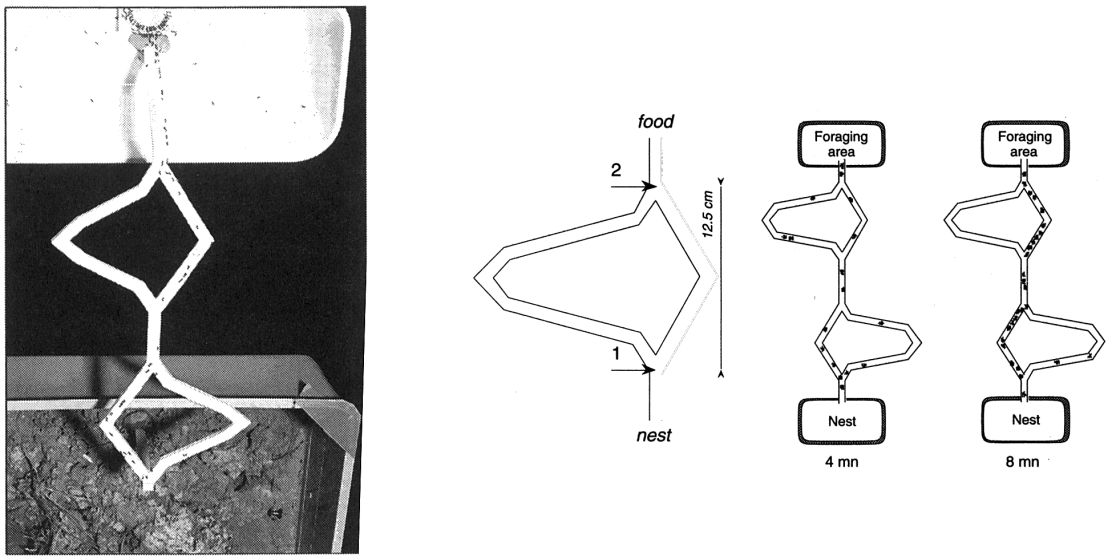

Abbildung 2.1: Experiment zur Selektion kürzester Pfade bei Ameisen / links: reales Experiment mit Nest unten und Futterquelle oben / rechts: schematisches Setup und Verlauf nach 4, 8 min, [Bonabeau, Dorigo and Theraulaz 1999]

1. Zwei Ameisen verlassen zur selben Zeit das Nest.

2. Die Ameise auf dem kürzeren Weg kehrt am schnellsten zurück.

3. Ein kürzerer Weg bedeutet damit eine höhere Pheromonkonzentration für diesen Pfad am Nest.

4. Ameisen auf Nahrungssuche entscheiden sich nun aufgrund der höheren Pheromonkonzentration für den kürzeren Weg.

→ Kürzeste Wege werden über eine höhere Pheromonkonzentration etabliert.

Einen anderen nützlichen Versuch im Zusammenhang mit Inter-Nest Traffic führten Aron et al.[Aron, Deneubourg, Goss and Pasteels 1990] durch. Dabei wurde untersucht, wie sich Pheromonpade bei mehreren unterschiedlich weit entfernten Nahrungsquellen über die Zeit entwickeln (Abbildung 2.2):

1. Ameisen strömen vom Nest aus.

2. Nahe gelegene Quellen werden zuerst entdeckt und genutzt.

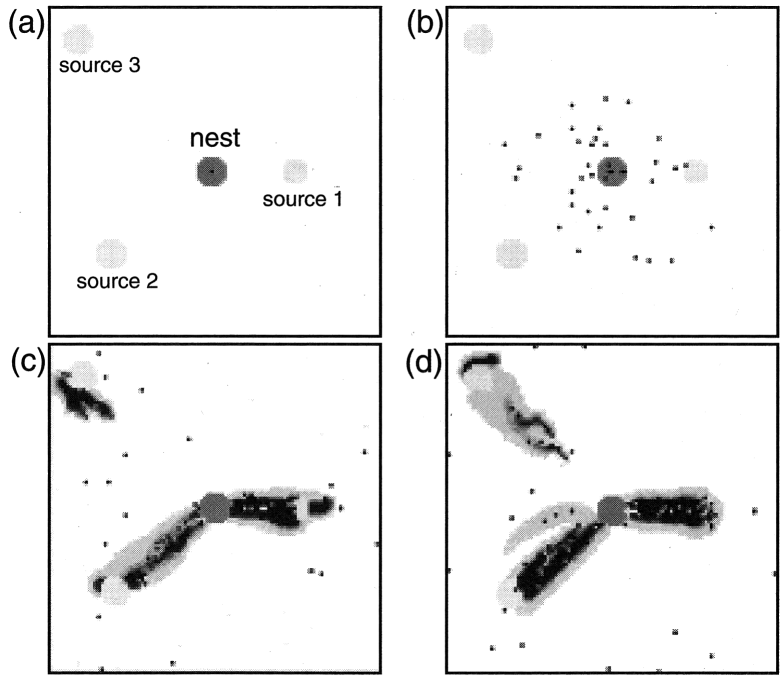

Abbildung 2.2: Simulation der Ausbeutung unterschiedlich weit entfernter Nahrungsquellen durch Ameisen und der Dynamik von Pheromonpfaden / (a): Anfangszustand t_0, 3 gleichwertige Nahrungsquellen sind unterschiedlich weit vom Nest entfernt / (b): t_1, Ameisen strömen aus, laufen zufällig / (c): t_2, Pheromonpfade zu den nahegelegenen Quellen bilden sich aus / (d): t_3, Pheromonpfade zur weiter entfernten Quelle bildet sich aus[Bonabeau, Dorigo and Theraulaz 1999]

3. Wenn der Nahrungsvorrat in den nahen Quellen versiegt, nimmt auch die Pheromonkonzentration auf den entsprechenden Pfaden ab. Ameisen, die eine solche Quelle erreichen kommen nicht zurück zum Nest, sondern suchen weiter nach anderen Quellen.

4. Dadurch werden weitere Quellen gefunden und die zugehörigen Pfade etabliert.

→ Nahrungsquellen werden abhängig von ihrer Entfernung und der Qualität/Quantität der Nahrung abgearbeitet.

Dies zeigt, daß die Ameisen das Minimal Spanning Tree Problem lösen können. Dies ist zwar kein kombinatorisch schwer zu lösendes Problem, aber einige seiner Varianten, wie z.B. das Steiner-Problem.

2.1.2 Technik

ACO-Algorithmen wurden anfänglich auf das NP-harte TSP angewendet. Der erste ACO-Algorithmus, das Ant-System (AS) schnitt dabei im Vergleich zu speziellen TSP-Heuristiken, v.a. bei Problemen größerer Dimension weniger erfolgreich ab. Trotzdem zeigte das AS eine höhere Performace als generelle Heuristiken [Dorigo, Maniezzo and Colorni 1996a].
Eine Weiterentwicklung des AS mit verbesserter Performance stellt das Ant Colony System (ACS) dar. ASC wurde von Dorigo und Gambardella entwickelt, um auch für größere TSP Problem Sets gute Lösungen zu erhalten [Dorigo and Gambardella 1997a] [Dorigo and Gambardella 1997b].
Basierend auf AS und ACS wurden weitere ACO-Algorithmen vorgestellt, wie das MAX-MIN AS (MMAS) von Stützle und Hoos [Stuetzle and Hoos 1997] [Stuetzle and Hoos 1997] und der AS-Rank-Algorithmus von Bullnheimer et al.[Bullnheimer, Hartl and Strauss 1997].
Neuere hybride Verfahren, die ACO mit lokaler Optimiererung kombinieren sind sehr erfolgsversprechend. Ihre Performance ist vergleichbar mit der guter Heuristiken wie dem Iterated Lin-Kernighan-Algorithmus von Johnson und McGeoch [Johnson and McGeoch 1997] oder dem Genetischen Algorithmus (GA) von Freisleben und Merz [Freisleben and Merz 1996]. Desweiteren stellen Gambardella et al. in Zusammenhang mit der Lösung des Quadratic Assignment Problem (QAP) das Hybrid Ant System (HAS) vor, welches sich sehr von den anderen AS-Algorithmen abgrenzt. Dieses System nutzt auch lokale Optimierung [Gambardella, Taillard and Dorigo 1997].

Die grundlegende Idee aller ACO-Algorithmen bildet der positive feedback Mechanismus, welcher Aufbau, Erhaltung und Attraktion von Pfaden die zu einer guten Lösung bzw. Teillösung führen durch virtuelles Pheromon beschreibt [Dorigo, Maniezzo and Colorni 1991b] [Dorigo 1992].
Eine weitere wichtige Basis bildet das Konzept des kooperativen Verhaltens, was die Fähigkeit von ACO-Algorithmen beschreibt, den Lösungsraum parallel mit vielen gleichen künstlichen Individuen zu explorieren.

ACO-Algorithmen führen für t_{max} Iterationen folgende zwei grundlegenden Prozeduren aus [Bonabeau, Dorigo and Theraulaz 1999]:

1. Konstruktion/Modifikation einer Lösung des Problems parallel durch m Ants

2. Regel zum Aktualisieren der Pheromonkonzentrationen auf den Kanten des Problemgraphen

Schritt 1 basiert dabei auf Wahrscheinlicheiten, die von der heuristischen Erwünschtheit η sowie der Pheromonkonzentration auf dem Lösungspfad abhängig sind. Schritt 2

stellt eine Funktion der Pheromon-Verdampfungsrate τ und der Qualität der produzierten Lösungen dar. ACO-Algorithmen können auf jedes Problem der kombinatorischen Optimierung angewendet werden, sofern folgende Anforderungen erfüllt sind:

- Eine geeignete Problemrepräsentaion, die die Konstruktion und Modifikation von Lösungen mittels Übergangsregeln basierend auf Wahrscheinlichkeiten in Zusammenhang mit Pheromonen und lokalen Heuristiken erlaubt.
- Die heuristische Erwünschtheit η von Kanten des Graphen muß definiert sein.
- Eine Methode, die für jede konstruierte Lösung sicherstellt, daß die Bedingungen für eine valide Lösung erfüllt sind.
- Eine Aktualisierungsregel für die Pheromonkonzentration τ der Kanten des Problemrepräsentationsgraphen.
- Eine Übergangsregel basierend auf Wahrscheinlichkeiten als Funktion von η und τ.

Algorithmus: High-level Beschreibung eines Basic ACO-Algorithmus [Bonabeau, Dorigo and Theraulaz 1999]

```
1   /* Initialization */
2   For every edge (i,j) do
3     tau_ij(0) = tau_0
4   End For
5   For k=1 to m do
6     Place ant k on a randomly chosen city
7   End For
8
9   /* Main Loop */
10  For t=1 to t_max do
11    For k=1 to m do
12      Build a solution S^k(t) by applying n-1 times a probabilistic
            construction/modification rule where choices are a function
            of a pheromone trail tau and of a heuristic desirability eta
13    End For
14    For k=1 to m do
15      Compute the cost C^k^(t) of the solution S^k(t) built by ant k
16    End For
17    If an improved solution is found then
18      Update best solution found
19    End If
20    For every edge (i,j) do
21      Update pheromone trails by applying a pheromone trail update rule
22    End For
23  End For
```

```
24    Print the best solution
```

Alle vorgestellten ACO-Algorithmen arbeiten auf einem diskreten Lösungsraum. Erweiterungen für die Optimierung in kontinuierlichen Lösungsräumen wurden von Bilchev und Parmee[Bilchev and Parmee 1995] vorgestellt. Wodrich[Wodrich 1996] schlägt dazu Erweiterungen vor.

Bisher haben wir nur ACO-Algorithmen betrachtet, bei denen das Optimierungsproblem rein statisch war. ACO läßt sich aber auch besonders gut auf dynamische (z.B. zeitvariante) Probleme anwenden. Als Anwendungsfelder sind hier beispielsweise Routing und Lastenverteilung in Telekommunikationsnetzwerken oder Lastenverteilung bei Industrieprozessen zu nennen. Da Swarm-Systeme eingesetzt werden, um Abläufe optimal zu steuern, werden die dynamischen Verfahren auch als Ant Based Control (ABC) bezeichnet. Dabei bewegen sich künstliche Ants auf einem Modell der Prozesskette bzw. des Netzwerks und vergeben Pheromon für gegangene Wege basierend auf der freien Kapazität, der Länge und der benötigten Zeit.

Schoonderwoerd et al. [Schoonderwoerd, Holland, Bruten and Rothkrantz 1996] stellen einen adaptiven Ant-Routing-Algorithmus vor, bei dem die Ants die Routing-Policy für jeden Knoten eines Telephonnetzwerkes durch Abgabe von Pheromon in die entsprechenden Routing-Tabellen aktualisieren.

Di Caro und Dorigo präsentieren das AntNet, einen adaptiven Algorithmus für das Routing in Datenkommunikations-Netzwerken. AntNet basiert auf ähnlichen Prozipien wie der Algorithmus von Schoonderwoerd et al. ist aber komplexer. In Tests brillierte AntNet gegenüber anderen Verfahren wie Open Shortest Path First (OSPF), dem asynchronen Bellman-Ford Algorithmus (BF), Shortest Path First (SPF) mit dynamischer Kostenfunktion sowie dem Q-routing Algorithmus und dessen Erweiterung mit Prediction [Di Caro and Dorigo 1998] [Di Caro and Dorigo 1997a] [Di Caro and Dorigo 1997b].

White et al.[White, Pagurek and Oppacher 1998] entwickelten einen Ant-Routing Algorithmus auf der Basis des Ant System, welches im Gegensatz zu den beiden letzten Verfahren nicht in Packet-Switched Networks, sondern in Circuit-Switched Networks zum Einsatz kommt.

Das Routing-Verhalten in Telekommunikationsnetzwerken ist mathematisch nur sehr schwierig zu analysieren. Daher sollten auf Selbstorganisation basierende Algorithmen folgende Bedingungen erfüllen:

- Asymptotische Konvergenz gegen eine nahezu optimale Lösung für stationäre Netzwerklast.

- Schnelle Adaption in dynamischen Situationen.

- Kein oszillierendes Verhalten entwickeln.

Einen etwas anderen Ansatz für Ant-Based Algorithmen in dynamischen Situationen stellt Holden [Holden 1997] vor. Dabei geht es nicht um eine Routing-Problematik sondern um effizientes Information-Retrieval in großen Datennetzwerken mittels Digital Information Pheromones (DIP). Die Idee ist, die Erfahrungen aus früheren Suchanfragen zu nutzen und in Form von DIP abzulegen.

2.2 Arbeitsteilung und Zuweisung von Aufgaben

Viele Insektenarten besitzen eine Form von Arbeitsteilung. Die Flexibilität der Gesellschaft im Falle von Störungen ist dabei auf die Flexibilität im Handeln einzelner Individuen zurückzuführen. Ob und wie einzelne Insekten auf bestimmte Tasks ansprechen, kann mit dem Modell der Reaktionsschwellen beschrieben werden. Ist die Reaktionsschwelle für einen bestimmten Task niedrig, ist es sehr wahrscheinlich, daß das Individuum schon auf geringe Stimuli im Bezug auf den Task reagiert und ihn ausführt.
Das Modell der statischen Reaktionsschwellen kann um eine Lernkomponente erweitert werden: Wenn ein Individuum einen Task ausführt, steigt die Schwelle an, wenn es einen Stimulus bekommt und den Task aufgrund einer zu hohen Schwelle nicht ausführt, sinkt die Schwelle.
Das statische Modell findet beispielsweise Anwendung in Multi-Agenten-Systemen (MAS), bei denen ein Markt entsteht, also Angebote, wie Produkte und Leistungen auf Nachfrage treffen. Die Reaktionsschwelle entscheidet dann, ob ein Angebot zu einem bestimmten Preis akzeptiert wird oder nicht. Das adaptive Modell führt zu emergenter Arbeitteilung und sorgt für Robustheit und Flexibilität eines Systems gegenüber unvorhersehbaren Störungen. Eine Anwendung ist z.B. die Verteilung von Post.

2.2.1 Natur

Oft werden in Insektenkolonien verschiedene Tasks parallel durch spezialisierte Arbeiter ausgeführt. Dieses Verhalten wird als Arbeitsteilung bezeichnet. Es wird angewendet, da es scheinbar effizienter ist, als das sequentielle Ausführen verschiedener Tasks durch nicht-spezialisierte Arbeiter [Oster and Wilson 1978][Robinson 1992].

Es existieren 3 Formen von Arbeitsteilung:

Temporal polyethism Individuen werden aufgrund ihres Alters in gesellschaftliche Schichten eingeteilt. Jede gesellschaftliche Schicht führt andere Aufgaben durch.

Worker polymorphism Individuen werden morphologisch aufgrund ihres Körperbaus in verschiedene gesellschaftliche Schichten eingeteilt. Hier wird auch zwischen normalen Arbeitern und den stärkeren Soldaten unterschieden.

Individual variability In einer gesellschaftlichen Schicht können trotzdem Unterschiede zwischen Individuen bzgl. der Reaktion auf bestimmte Tasks bestehen. Man spricht hier auch von verhaltensbedingten gesellschaftlichen Schichten.

Die Strukturen sind aber keineswegs fest. Im Falle von Ungleichmäßigkeiten im System, z.B. in Zusammenhang mit der Verfügbarkeit von Nahrung, Klimaveränderungen, dem aktuellen Stadium der Kolonie, den Jahreszeiten sowie der Struktur der Gesellschaft können sich die Zuweisung von Individuen an Tasks oder Abläufe ändern. Dadurch kann die Kolonie immer flexibel und robust reagieren.
Wilson[Wilson 1984] führte wichtige Experimente bzgl. Arbeitsteilung und Anpassung von Individuen durch, indem er die gesellschaftliche Struktur von Ameisenkolonien veränderte und die Auswirkungen untersuchte.

2.2.2 Technik

Um Wilson's Experimente zu untermauern, entwickelten Bonabeau et al.[Bonabeau, Theraulaz and Deneubourg 1996] ein einfaches Modell basierend auf Reaktionsschwellen. Dabei besitzen Individuen einer Kolonie feste Schwellen bzgl. bestimmter Aufgaben. Steigt der Stimulus für eine Aufgabe über den Schwellwert, führt das Individuum die Aufgabe aus. Dadurch sinkt der Stimulus wieder und andere Individuuen reagieren nicht, da er unter ihrem Schwellwert liegt, usw. Dieses Verhalten wurde z.B. beim Füttern von Larven oder dem Abtransport von toten Individuen bestätigt. Man kann hier gut die indirekte Kommunikation (Stigmergy), also die Kommunikation über Umgebungszustände und -modifikationen in Insektenkolonien beobachten[Grassé 1959][Grassé 1984].

Formal kann das Verhalten für eine Reaktionsschwelle ϑ, ausgedrückt in Stimulus-Einheiten über eine Antwortfunktion $T_\vartheta(s)$ beschrieben werden. Dabei steht s für die Intensität eines Stimulus für eine bestimmte Tätigkeit. Die Wahrscheinlichkeit für das Ausführen einer Aufgabe ist nierdrig für $s \ll \vartheta$ und hoch für $s \gg \vartheta$. Eine Familie von Antwortfunktionen $T_\vartheta(s)$ (Abbildung 2.3) ist gegeben durch

$$T_\vartheta(s) = \frac{s^n}{s^n + \vartheta^n}$$

Dabei steht $n > 1$ für den Anstieg der Kurve. Meist wird ein Wert von $n = 2$ verwendet.

Eine andere Familie von (exponentiellen) Antwortfunktionen $T_\vartheta(s)$ (Abbildung 2.4) ist

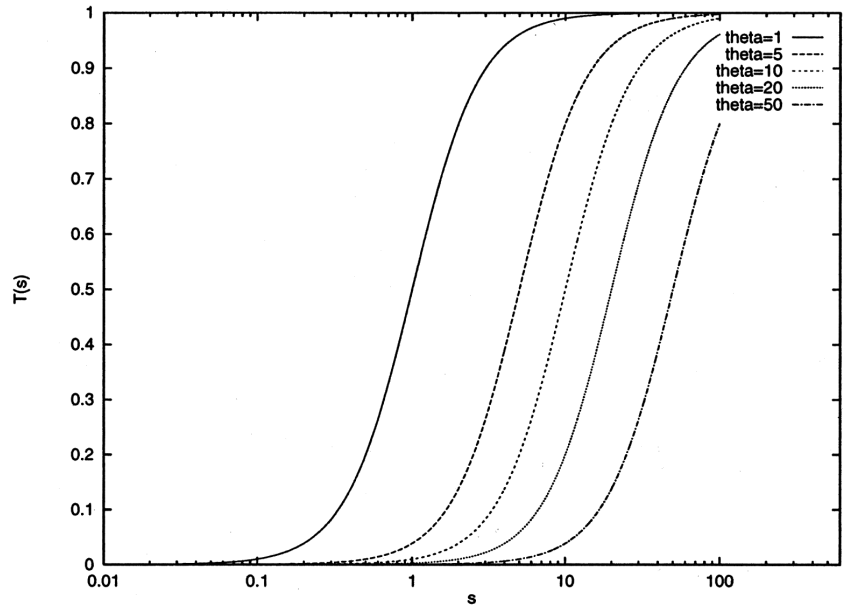

Abbildung 2.3: Semi-log Plot von Antwortfunktionen ($n = 2$) mit verschieden hohen Reaktionsschwellen ($\theta = 1, 5, 10, 20, 50$), [Bonabeau, Dorigo and Theraulaz 1999]

definiert als

$$T_\vartheta(s) = 1 - e^{-s/\vartheta}$$

Obwohl exponentielle Antwortfunktionen die natürlichen Sachverhalte am genausten beschreiben, führen alle Arten von Antwortfunktionen zu ähnlichen Ergebnissen.

Man kann untersuchen, wie sich der Stimulus für einen bestimmten Task über die Zeit verhält. Dies ist davon abhängig wie effizient und regelmäßig der Task ausgeführt wurde. Die Gleichung für das zeitdiskrete Verhalten eines Stimulus s im Modell mit einem Task ist gegeben durch

$$s(t+1) = s(t) + \delta - \frac{\alpha N_{act}}{N}$$

wobei N_{act} für die Anzahl der aktiv am Task beteiligten Individuen steht. N gibt die Anzahl der potentiell aktiven Individuen der gesamten Kolonie an. δ ist das Intensitätsinkrement des Stimulus pro Zeiteinheit und α ein Skalierungsfaktor für das Messen der Effizienz bei der Taskabarbeitung.

In einer Kolonie kann es verschieden Typen von Individuen geben. Sie unterscheiden sich durch unterschiedlich hohe Reizschwellen. Ein Modell, welches mehrere Typen von Individuen und mehrere Tasks einbezieht, kann durch ein System von deterministischen

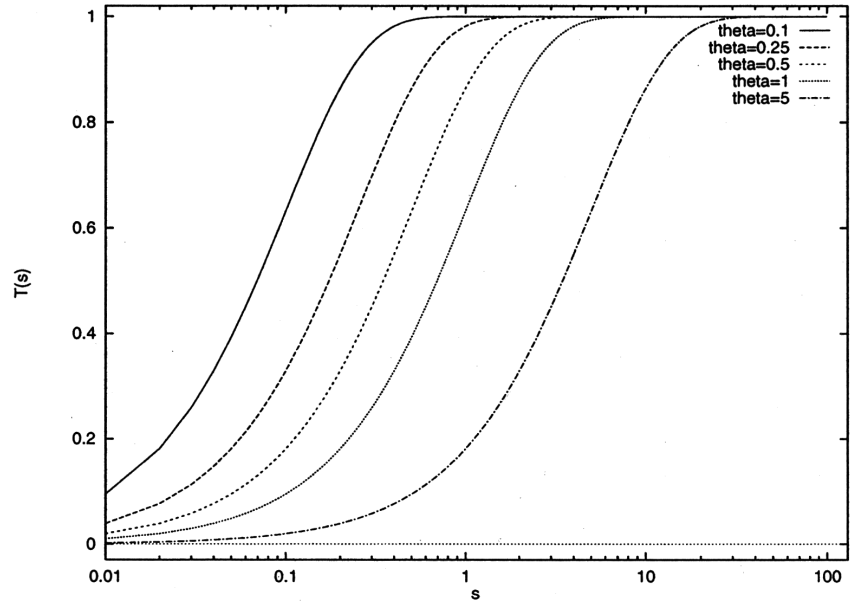

Abbildung 2.4: Semi-log Plot von exponentiellen Antwortfunktionen mit verschieden hohen Reaktionsschwellen ($\theta = 0.1, 0.25, 0.5, 1, 5$), [Bonabeau, Dorigo and Theraulaz 1999]

Differentialgleichungen beschrieben werden:

$$\partial_t x_{ij} = \frac{s_j^2}{s_j^2 + \vartheta_{ij}^2} \left(1 - \sum_{k=1}^{m} x_{ik} \right) - p x_{ij}$$

wobei s_j für die Intensität des Stimulus für Task j steht. Die Anzahl von verschiedenen Tasks wird mit m bezeichnet. Legt man N_{ij} als Anzahl der Individuen vom Typ i fest, die gerade mit Task j beschäftigt sind, so ist mit $x_{ij} = N_{ij}/n_i$ das Verhältnis der N_{ij} zu n_i, der Anzahl der Individuen vom Typ i in der Kolonie definiert. ϑ_{ij} bezeichnet die zugehörige Reaktionsschwelle.

Modelle basierend auf statischen Reaktionsschwellen [Robinson, Page and Huang 1994] [Calderone and Page 1996] haben folgende Nachteile:

- Die Entstehung von Task-Zuweisungen wie auch das Verändern von Aufgabengebieten während des Lebens von Insekten können nicht erklärt werden, da angenommen wird, daß Individuen fest zugewiesene Rollen besitzen.

- Die Task-Spezialisierung in physischen oder altersabhängigen Gesellschaftsschichten kann nicht erklärt werden.

- Die Modelle sind nur für kurze Zeitintervalle gültig, solange Schwellwerte als kon-

stant angenommen werden können.

- Experimente bei Bienen haben ergeben, daß das Altern bzw. Lernen zu einer Veränderung des Gehirns und der Funktion im Arbeitsprozess führt. Dies kann nicht erklärt werden.

Die aufgezeigten Nachteile können durch Modelle mit zeitvarianten Reaktionsschwellen umgangen werden. Théraulaz et al. [Theraulaz, Goss, Gervet and Deneubourg 1991] [Theraulaz, Bonabeau and Deneubourg 1998] stellen ein solches Modell vor. Dabei sinkt die Schwelle, wenn ein bestimmter Task ausgeführt wird und steigt ansonsten.

Die Dynamik des an Task j gekoppelten Stimulus s_j kann durch die Gleichung

$$\partial_t s_j = \delta - \frac{\alpha_j}{N}\left(\sum_{i=1}^{N} x_{ij}\right), (j = 1, 2)$$

beschrieben werden. Dabei steht δ für die Erhöhung der Stimulusintensität pro vergangener Zeiteinheit, α_j für einen Skalierungsfaktor für die Effizienz der Abarbeitung von Task j, N für die Anzahl der Individuen und x_{ij} für die Anzahl der Individuen vom Typ i, die am Task j beteiligt sind.

Derartige Modelle haben einen weiteren Vorteil. Sie sind flexibler. Werden beispielsweise Individuen mit geringer Schwelle entnommen, so werden bei den anderen Individuen die Schwellen gesenkt, was dazu führt, daß diese schneller auf einen Stimulus reagieren. Mittels solcher flexiblen Modelle läßt sich auch das Phänomen der adaptiven Task-Zuweisung erklären. Bonabeau et al. [Bonabeau, Sobkowski, Theraulaz and Deneubourg 1997] beschreiben den zone allocation Algorithmus anhand eines Postunternehmens. Dabei sollen diejenigen Postbeamten eher auf einen Stimulus (Abholauftrag) reagieren, je näher sie sich befinden. Insgesamt soll die globale Nachfrage niedrig gehalten werden.

Algorithmen basierend auf Strategien der Arbeitsteilung von Insektenkolonien haben ähnliche Eigenschaften wie auch Marktwirtschafts-Algorithmen, bei denen Ressourcen für die Ausführung eines Tasks „ersteigert“ werden können. Der Agent mit dem besten Gebot kann letztendlich den Task ausführen. Waldspurger et al.[Waldspurger, Hogg, Huberman and Kephart 1992] haben mit „Spawn“ ein System entwickelt, bei dem verschiedene Tasks als Agenten mit Eigenwohl agieren. Sie haben je nach Priorität einen bestimmten Geldbetrag frei und können auf Ressourcen bieten. Sind viele Kapazitäten frei, liegen die Preise niedrig und Tasks können beispielsweise auch gleiche Ressourcen parallel allokieren.
Bezieht man sich auf das Reaktionsschwellen-Modell, so kann man sagen, daß Agenten mit einem niedrigen Schwellwert bereit sind, die höchsten Preise zu bezahlen. Morley und Ekberg[Morley 1996][Morley and Ekberg 1998] beschreiben ein solches System in Zusammenhang mit der Lackierung von Trucks in einer Fabrik.

Es existieren mehrere Lackiervorrichtungen. Die zu verwendenden Farben hängen von speziellen Kundenwünschen ab. Da der Wechsel der Farbe an einem Roboter sehr viel Zeit und Material verschwendet, gilt es dies zu minimieren. Kommen jedoch dringende Aufträge mit gerade nicht verfügbaren Farben, wird entschieden, ob gewechselt werden soll. Dabei fließen auch die Warteschlangen und Zustände der anderen Lackierroboter mit ein. Jeder Roboter ist dabei ein Agent, der nach folgenden vier Regeln vorgeht:

1. Versuche einen Truck mit der aktuellen Farbe zu bekommen.

2. Nimm eiligere Jobs zuerst.

3. Nimm jeden Job an, um nicht in einen Leerlauf zu geraten.

4. Nimm keinen neuen Job an, wenn keine Farbe mehr vohanden ist oder die Warteschlange zu lang ist.

Die optimalen Parameter wurden mithilfe von Techniken der Evolutionären Programmierung bestimmt. Insgesamt konnte gegenüber der konventionellen Steuerung eine hohe Effizienzsteigerung erreicht werden.

2.3 Analyse, Organisation, Sortierung und Clusterung von Objekten

Viele Ameisenarten clustern ihre toten Artgenossen in einer Art „Friedhof“ außerhalb des Nests (Abbildung 2.5) oder ordnen ihre Larven in bestimmten Strukturen an. Dieses Verhalten ist noch nicht vollständig verstanden. Trotzdem läßt sich ein einfaches Modell auf der Basis lokaler Informationen ableiten, welches sehr nützlich sein kann. Dabei bewegen sich Agenten zufällig und nehmen nach bestimmten Regeln Objekte auf, tragen Objekte und legen Objekte ab. Das Modell läßt sich u.a. auf Verfahren zur Clusterung, Sortierung, Analyse von Daten sowie auf die Partitionierung von Graphen anwenden.

2.3.1 Natur

In der Natur wurde oft berichtet, daß Ameisen Ansammlungen von ähnlichen Objekten anlegen. Chrétien[Chrétien 1996] führte Untersuchungen zum Aufbau von Friedhöfen bei der Ameisenart Lasius niger durch. Deneubourg et al.[Deneubourg et al. 1991] berichtet von ähnlichen Experimenten bei der Art Pheidole pallidula. Dabei geht man davon aus, daß einerseits tote Körper im Nest, andererseits Ansammlungen außerhalb des Nestes attraktiv auf Ameisen, die mit dem Clustering-Task betraut sind wirken.

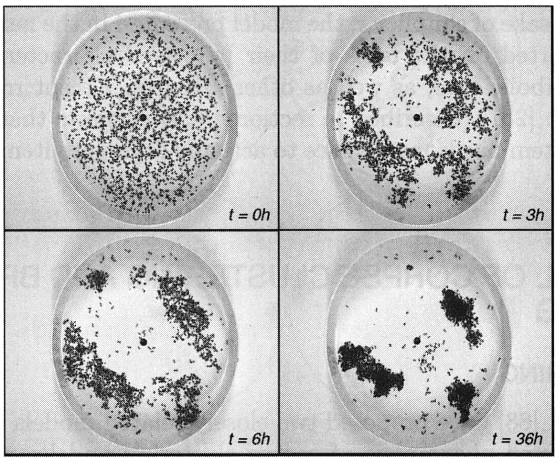

Abbildung 2.5: Experiment zur Sortierung toter Körper bei Ameisen: 1500 Kadaver zufällig in runder Schale ($d = 25cm$) angeordnet, 4 Zeitschritte mit jeweiliger Clusterung, [Bonabeau, Dorigo and Theraulaz 1999]

Die Sortierung von Brut wurde für die Ameisenart Leptothorax unifasciatus weitestgehend untersucht[Franks and Sendova-Franks 1992]. Dabei wird die Brut nicht nur geclustert, sondern konzentrisch nach einem festen Schema organisiert. Die Organisation hat nicht nur mit der Größe der Larven zu tun, sondern auch mit ihren Stoffwechselanforderungen. Müssen sie gefüttert werden, liegen sei weiter außen, damit ein Zugang besteht. Man geht davon aus, daß die nahe Positionierung ähnlicher Objekte auch der Effizienz spezieller Tasks dient.

2.3.2 Technik

Deneubourg et al.[Deneubourg et al. 1991] stellen Modelle für Clustering und Sorting von Objekten vor. [1]

Die zugrunde liegende Idee ist, daß vereinzelte Objekte aufgenommen werden und an Ansammlungen von Objekten wieder abgeladen werden (Abbildung 2.6). Geht man von nur einem Typ von Objekten aus, so ist die Wahrscheinlichkeit p_p, daß eine unbeladene Ameise ein Objekt aufnimmt gegeben durch

$$p_p = \left(\frac{k_1}{k_1 + f}\right)^2$$

f steht dabei für das wahrgenommene Verhältnis von Objekten in der Nachbarschaft,

[1] In der Literatur wird im Zusammenhang mit der Organisation von Objekten zwischen Clustering und Sorting unterschieden. Dabei wird unter Clustering die einfache Ansammlung von gleichen Objekten verstanden. Als Sorting wird die Gruppierung von Objekten unterschiedlicher Typen bezeichnet.

k_1 für einen konstanten Schwellwert. Für $f \ll k_1$ geht p_p gegen 1. Anschaulich bedeutet das, wenn sich wenige Objekte in der Nachbarschaft befinden, ist es sehr wahrscheinlich, daß ein untersuchtes Objekt aufgenommen wird. Für den Fall, daß $f \gg k_1$, geht p_p gegen 0.
Die Wahrscheinlichkeit für das Ablegen eines Objektes p_d ist beschrieben durch

$$p_d = \left(\frac{f}{k_2 + f}\right)^2$$

wobei k_2 eine andere Schwellenkonstante darstellt. Hier gilt, wenn $f \ll k_2$, geht p_d gegen 0, was bedeutet, daß die Ablage von Objekten an Orten geringer Objektdichte sehr unwahrscheinlich ist.

Deneubourg et al.[Deneubourg et al. 1991] arbeiteten mit einem Roboter-Szenario. Die Objektdichtefunktion f wurde als Verhältnis der Anzahl der Objekte N der letzten Zeitschritte zur Gesamtzahl der möglichen Objekte in diesem Zeitintervall T bestimmt ($f = N/T$).

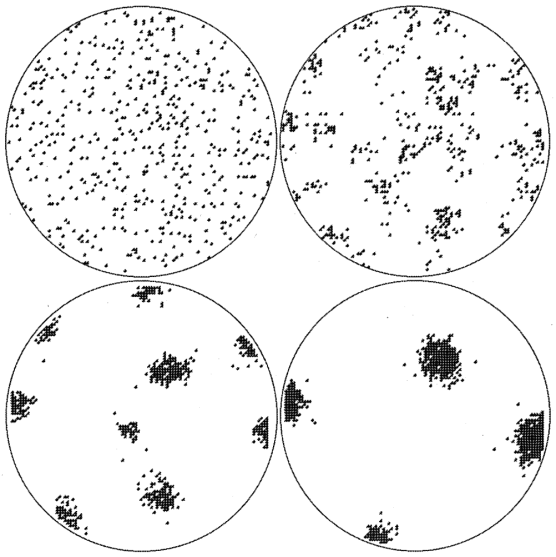

Abbildung 2.6: Simulation eines Clustering-Modells: 4 Zeitschritte ($t = 0, 50000, 1000000, 5000000$) mit jeweiliger Clusterung / 5000 Objekte, 10 Agenten, 31416 Positionen, $T = 50$, $k_1 = 0.1$, $k_2 = 0.3$ / $\rightarrow$ relativ schnell enstehen kleinere Cluster, welche dann nach längerer Zeit zur größeren Clustern verschmelzen. [Bonabeau, Dorigo and Theraulaz 1999]

Das Clustering-Modell kann zum Sorting-Modell erweitert werden, indem man für verschiedene Typen von Objekten, z.B. Typ A und Typ B die jeweiligen Dichtefunktionen f_A und f_B verwendet (Abbildung 2.7).

Die Sortierung von Brut nach speziellen Patterns[Franks and Sendova-Franks 1992] kann

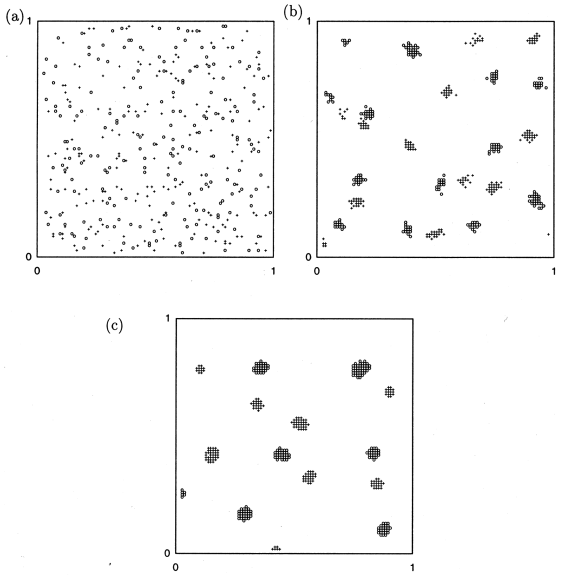

Abbildung 2.7: Simulation eines Sorting-Modells: 3 Zeitschritte ($t = 0, 500000, 5000000$) mit jeweiliger Clusterung / 400 Objekte zu 2 Typen (bezeichnet durch „o“ und „+“), 10 Agenten, 10000 Positionen, $T = 50$, $k_1 = 0.1$, $k_2 = 0.3$, [Bonabeau, Dorigo and Theraulaz 1999]

mit den vorgestellten Modellen nicht beschrieben werden. Hier können die Modelle um attraktive chemische Substanzen, sog. *Pheromone* erweitert werden.

Lumer und Faieta[Lumer & Faieta 1994] haben das Basismodell von Deneubourg et al.[Deneubourg et al. 1991] auf die Datenanalyse mit Rechnern erweitert. Die Idee von mehreren diskreten Objekttypen wird hier ins Kontinuierliche verallgemeinert: Angenommen, alle zu analysierenden Datenobjekte haben unterschiedliche Ausprägungen von n Eigenschaften. Diese können für jedes Datum als Vektor der Dimension n aufgefaßt werden, der seine Lage als Punkt im n-dimensionalen Attributraum beschreibt. Über diesem Raum ist ein Distanzmaß $d(o_i, o_j)$ zweier Objekte o_i und o_j, welches als Maß für die Verschiedenheit gesehen werden kann definiert. $d(o_i, o_j)$ kann z.B. als l-Norm implementiert werden:

$$\|o_i - o_j\|_l = \sum_{k=0}^{n-1} \sqrt[l]{(w_k (o_{ik} - o_{jk}))^l}$$

w bezeichnet dabei einen speziellen Gewichtungsfaktor für jede Attributdimension. Normalerweise hat dieser Faktor den Wert 1. Falls verschiedene Attribute unterschiedliche Prioritäten haben, kann man die jeweiligen Faktoren anpassen. Oft wird hier die euklidische Norm mit $l = 2$ verwendet.

Der Algorithmus von Lumer und Faieta führt eine intelligente Projektion von $\mathbb{R}^n$ auf $\mathbb{Z}^2$

durch. Dabei wird ein 2-dimensionales Grid als Untermenge des $\mathbb{Z}^2$ angenommen, auf denen die Objekte angeordnet werden sollen. Agenten bewegen sich über die Felder des Grids und können ihre Nachbarschaftsregion $Neigh_{(s\times s)}$, d.h. die $s^2 - 1$ an die aktuelle Position r angrenzenden Felder wahrnehmen. Angenommen, zu einem bestimmten Zeitpunkt t befindet sich ein unbeladener Agent auf einem mit Objekt o_i belegten Feld r, kann die lokale Objektdichte der Nachbarschaft von r über die Funktion $f(o_i)$, die auch ein Maß für die Ähnlichkeit von o_i mit den Objekten in der Nachbarschaft darstellt ausgewertet werden:

$$f(o_i) = \begin{cases} \frac{1}{s^2} \sum\limits_{o_j \in Neigh_{(s\times s)}(r)} \left[1 - \frac{d(o_i,o_j)}{\alpha}\right] & f > 0 \\ 0 & sonst \end{cases}$$

α steht dabei für einen Skalierungsfaktor für die Verschiedenheit. Er ist sehr wichtig für das Ergebnis der Clusterung: Ist α zu groß, gibt es kaum Unterscheidung zwischen den verschiedenen Objekten, was zu großen Clustern unterschiedlicher Objekte führt. Im anderen Fall ergeben sich viele kleinere Cluster die alle ähnliche Objekte enthalten und somit zu einem Cluster gehören würden. Angelehnt an das Basismodell von Deneubourg et al.[Deneubourg et al. 1991] ergeben sich folgende Wahrscheinlichkeiten für die Aufnahme und die Ablage von Objekten:

$$p_p(o_i) = \left(\frac{k_1}{k_1 + f(o_i)}\right)^2$$
$$p_d(o_i) = \begin{cases} 2f(o_i) & f(o_i) < k_2 \\ 1 & f(o_i) \geqslant k_2 \end{cases}$$

Algorithmus: High-level Beschreibung des Algorithmus von Lumer und Faieta [Bonabeau, Dorigo and Theraulaz 1999]

```
1   /* Initialization */
2   For every item o_i do
3     place o_i randomly on grid
4   End For
5   For all agents do
6     place agent at randomly selected site
7   End For
8
9   /* Main Loop */
10  For t=1 to t_max do
11    For all agents do
12      If ((agent unladen) and (site occupied by item o_i)) then
13        Compute f(o_i) and p_p(o_i)
14        Draw random real number R between 0 and 1
15        If (R <= p_p(o_i)) then
16          Pick up item o_i
17        End If
18      Else If ((agent carrying item o_i) and (site empty)) then
```

```
19          Compute f(o_i) and p_d(o_i)
20          Draw random real number R between 0 and 1
21          If (R <= p_d(o_i) then
22            Drop item
23          End If
24        End If
25        Move to randomly selected neighboring site not occupied by other
              agent
26      End For
27    End For
28    Print locations of items
29
30    /* Values of parameters used in experiments */
31    k_1=0.1, k_2=0.15, alpha=0.5, s^2=9, t_max=10^6 steps
```

LF demonstrieren ihren Algorithmus an einem Beispiel von 800 Datenobjekten (x, y) aus dem Attributraum $\mathbb{R}^2$. Je 200 Objekte werden durch eine $N(\mu, \sigma)$-Normaverteilung ihrer Attributwerte auf vier Cluster erzeugt (Abbildung 2.8).

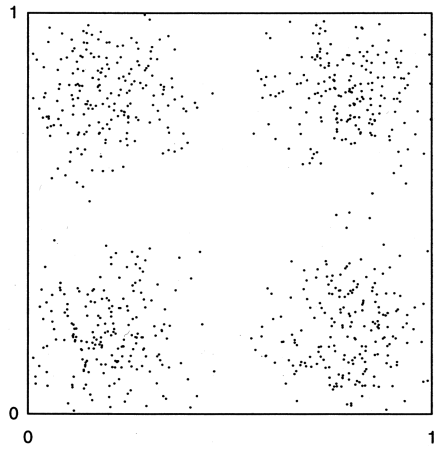

Abbildung 2.8: Verteilung der Objekte als Punkte (x, y) in 2D-Attributraum, 4 $N(\mu, \sigma)$-normalverteilte Cluster: $[x \propto N(0.2, 0.1), y \propto N(0.2, 0.1)]$, $[x \propto N(0.8, 0.1), y \propto N(0.2, 0.1)]$, $[x \propto N(0.8, 0.1), y \propto N(0.8, 0.1)]$, $[x \propto N(0.2, 0.1), y \propto N(0.2, 0.1)]$, [Bonabeau, Dorigo and Theraulaz 1999]

Die Objekte werden nun auf einem 100×100 $\mathbb{R}^2$-Grid verteilt[2]. Das Clustering-Experiment wird mit 10 Agenten ausgeführt.

Normalerweise ergeben sich mehr Cluster, als in der eigentlichen Initialverteilung vorhanden. LF schlagen vor, dieses Problem durch 3 mögliche Erweiterungen zu ihrem

[2]Dieses Grid wird auch Informations-Infrastruktur genannt. Das ist die Welt, in der die künstlichen Agenten die Datenobjekte zu Clustern gruppieren, nicht zu verwechseln mit dem Attributraum der Objekte.

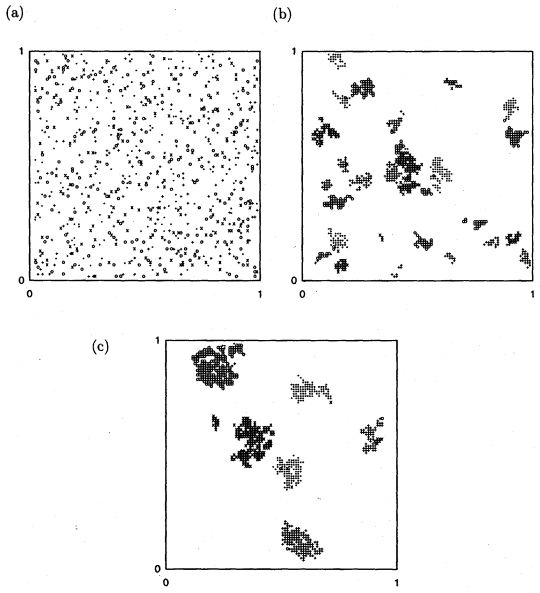

Abbildung 2.9: Sortierung mit Algorithmus von Lumer&Faieta: 3 Zeitschritte ($t = 0, 500000, 1000000$) mit jeweiliger Clusterung / 800 normalverteilte Objekte in 4 Clustern (siehe Abbildung 2.8) (bezeichnet durch „o“, „+“, „*“ und „×“), 10000 Positionen, $k_1 = 0.1$, $k_2 = 0.15$, $\alpha = 0.5$, $s^2 = 9$, [Bonabeau, Dorigo and Theraulaz 1999]

Algorithmus zu beheben:

1. *Agenten besitzen unterschiedliche Geschwindigkeiten* Dabei werden den Agenten unterschiedliche Geschwindigkeiten v aus $[1, v_{max}]$ zugewiesen. v steht dabei für die Anzahl von Grideinheiten, die pro Zeiteinheit gegangen werden. Tendenziell werden schneller bewegende Agenten nicht so genau mit der Bewertung der Ähnlichkeit von Objekten in der Umgebung sein und daher grob sortierte Cluster auf großer Fläche formen. Langsamere Agenten bilden kleinere akkuratere Cluster aus. Die lokale Dichtefunktion $f(o_i)$ ist nun zusätzlich von der Geschwindigkeit der Agenten abhängig:

$$f(o_i) = \begin{cases} \frac{1}{s^2} \sum\limits_{o_j \in Neigh_{(s \times s)}(r)} \left[1 - \frac{d(o_i, o_j)}{\alpha\left(1 + \frac{v-1}{v_{max}}\right)} \right] & f > 0 \\ 0 & sonst \end{cases}$$

2. *Agenten besitzen ein Kurzzeitgedächtnis* Die Agenten merken sich dabei die letzten m erfolgreich abgelegten Objekte mit den zugehörigen Positionen. Nehmen sie ein neues Objekt auf, so überprüfen sie die Ähnlichkeit mit den m gespeicherten Objekten und springen zur Position mit dem ähnlichsten Objekt. Dies verhindert

neben einer zeitaufwendigen Suche durch zufälliges Bewegen auch das Ausbilden mehrerer Cluster ähnlicher Objekte.

3. *Angenten können ihr Verhalten anpassen* Wenn Agenten für eine gewisse Zeit keine Aktion getätigt haben, beginnen sie Cluster zu zerstören. Dies verhindert ein Verfangen in lokal optimalen Ergebnissen.

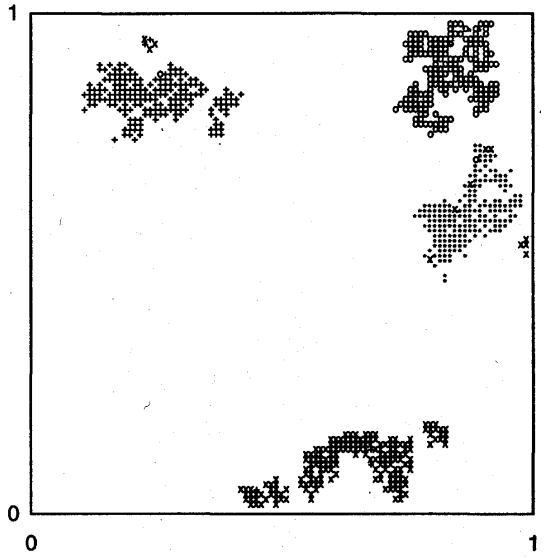

Abbildung 2.10: Sortierung mit Algorithmus von Lumer&Faieta und zusätzlichen Features: 800 normalverteilte Objekte in 4 Clustern (siehe Abbildung 2.8) (bezeichnet durch „o“, „+“, „*“ und „×“), 10000 Positionen, $k_1 = 0.1$, $k_2 = 0.15$, $\alpha = 0.5$, $s^2 = 9$, Kurzzeitgedächtnis $m = 8$, $v_{max} = 6$, [Bonabeau, Dorigo and Theraulaz 1999]

LF beschränkten sich nicht nur auf Testdaten, sondern wenden ihren Algorithmus auch auf eine Datenbank mit den Profilen von 1650 Bankkunden an[Lumer and Faieta 1995]. Für die verschiedenen Attribute der Objekte „Bankkunde“, wie Alter, Geschlecht, Status etc. mussten passende Distanzmaße gefunden und in einem globalen Maß für die Ähnlichkeit kombiniert werden. Obwohl der Algorithmus interessante Resultate liefert, ist er bezüglich der Rechenzeit nicht sehr effizient. Trotzdem bietet er einen einfachen Weg, multidimensional scaling zu realisieren.

LF bezeichnen ihren Algorithmus als eine Kombination von Clustering und Multidimensional Scaling, da Elemente in unterschiedlichen Konzentrationsgebieten zu unterschiedlichen Clustern gehören und zusätzlich eine Intra-Cluster Ähnlichkeitsstruktur konstruiert wird.

Das Clustering Modell wurde desweiteren auch im Bezug auf Swarm-Based-Robotics [Beckers, Holland and Deneubourg 1994] [Gaussier and Zrehen 1994]

[Gaussier and Zrehen 1994] [Martinoli, Ijspeert and Mondada 1999] [Sugawara and Sano 1997] implementiert. Alle robotischen Systeme basieren auf lokaler Wahrnehmung und Handlung. Es ist darauf zu achten, daß sich die einzelnen Roboter nicht behindern, wenn ihre Anzahl zu hoch ist. Bisher vorgestellte Arbeiten auf diesem Gebiet beschränken sich auf eher simple Tasks.

Kuntz und Layzell[Kuntz and Layzell 1995] sowie Kuntz, Layzell und Snyers[Kuntz, Layzell and Snyers 1997] (KLS) stellen eine Anwendung von LF auf das Graph Partitioning Problem vor. Sie transformieren das Problem von einer Graph-Repräsentation in eine geometrische Repräsentation im Euklidischen Raum. KLS kann auch aus Preprozessing eines Graph-Drawing-Algorithmus verwendet werden[Tamassia, Battista and Battini]. KLS ist besonders nützlich, wenn der Graph natürliche Cluster enthält. Es kann beispielsweise als Werkzeug zum Extrahieren von Komponenten genutzt werden.

Handl, Knowles und Dorigo (HKD)[Handl et al. 2003b] stellen einen performanten robusten Ant-based-Clustering Algorithmus basierend auf den Erkenntnissen von Deneubourg und LF vor. Sie legen dabei v.a. Wert auf eine räumliche Abgrenzung der Cluster, um den späteren Analysevorgang zu vereinfachen sowie eine hohe Genauigkeit der resultierenden Klassifikation.

Basic Ant Algorithmus von Handl, Knowles und Dorigo[Handl et al. 2003b]

```
1   /* Initialization */
2   Randomly scatter data items on the toroidal grid
3   for each j in 1 to #agents do
4     i = random_select(remaining_items)
5     pick_up(agent(j), i)
6     g = random_select(remaining_empty_grid_locations)
7     place_agent(agent(j), g)
8   end for
9
10  /* Main Loop */
11  for each it_ctr in 1 to #iterations do
12    j = random_select(all_agents)
13    step(agent(j), stepsize)
14    i = carried_item(agent(j))
15    drop = drop_item?(f*(i)) // see equations for dropping
16    if drop = TRUE then
17      while pick = FALSE do
18        i = random_select(free_data_items)
19        pick = pick_item?(f*(i)) // see equations for picking
20      end while
21    end if
22  end for
```

Der Basic-Ant-Algorithmus unterscheidet sich vom LF-Algorithmus darin, daß unbeladene Ameisen nur zwischen Positionen mit Objekten springen und keine unnützen Wege laufen. Die Größe *stepsize* spiegelt die Bewegungsgeschwindigkeit der Ameisen wieder. Für die Wahrscheinlichkeiten zur Aufnahme und Ablage von Objekten werden folgende Schwellenfunktionen eingesetzt:

$$p^*_{pick}(i) = \begin{cases} 1.0 & f^*(i) \leqslant 1.0 \\ \frac{1}{f^*(i)^2} & sonst \end{cases}$$

$$p^*_{drop}(i) = \begin{cases} 1.0 & f^*(i) \geqslant 1.0 \\ f^*(i)^4 & sonst \end{cases}$$

Dabei ist $f^*(i)$ eine modifizierte Version der Nachbarschaftsfunktion von LF:

$$f^*(i) = \begin{cases} \max\left(0, \frac{1}{\sigma^2} \sum_j \left(1 - \frac{d(i,j)}{\alpha}\right)\right) & \forall j \left(1 - \frac{d(i,j)}{\alpha} > 0\right) \\ 0 & sonst \end{cases}$$

Die Funktion weißt folgende wichtige Eigenschaften auf: Die Division durch die Größe der Nachbarschaft $\frac{1}{\sigma^2}$ führt zur Bevorzugung einer dichten lückenlosen Clusterung. Desweiteren führt die Bedingung $\forall j \left(1 - \frac{d(i,j)}{\alpha} > 0\right)$ zu einer sehr schlechten Gewichtung hoher Unähnlichkeiten und damit zu einer räumlichen Trennung der Cluster.
Der Basic-Ant-Algorithmus wird um die folgenden Komponenten erweitert.

Kurzzeitgedächtnis HKD verwenden eine modifizierte Variante des von LF eingeführten Kurzzeitgedächtnisses. Die Ameisen speichern bei LF nur die letzen n erfolgreich abgelegten Datenobjkete mit den zugehörigen Positionen. Nimmt eine Ameise ein neues Objekt auf, berechnet sie die Ähnlichkeiten zu den gespeicherten Objekten und springt an die Position mit der niedrigsten Distanz. Dieses Vogehen ist nicht unproblematisch. Seit dem Ablagevorgang des gespeicherten Objektes kann sich die Umgebung verändert haben, so daß ein neues Objekt nicht abgelegt werden kann, obwohl es eine gewisse Ähnlichkeit zum gespeicherten Objekt besitzt. Daher speichern HKD nur die letzten n Ablagepositionen. Hat eine Ameise ein Objekt aufgenommen, evaluiert sie alle gespeicherten Positionen mithilfe der Nachbarschaftsfunktion $f^*(i)$. Sie versucht nun direkt zu derjenigen gemerkten Position zu springen, bei der $f^*(i)$ den höchsten Wert liefert. Der Sprung wird mit der Wahrscheinlichkeit $p^*_{drop}(i)$ durchgeführt. Fällt die Entscheidung negativ aus, wird das Gedächtnis für das getragene Objekt deaktiviert.

Erhöhung des Wahrnehmungsradius Der Wahrnehmungsradius bestimmt die Größe der wahrgenommenen Nachbarschaft um die aktuelle Position einer Ameise. Ein kleiner Radius begrenzt die Information während des Sortiervorgangs. Ein größer Radius erfordert nicht nur mehr Rechenzeit, da mehr Positionen evaluiert werden müssen (quadratischer Anstieg), sondern begünstigt auch die Formierung großer Cluster zu Beginn

der Laufzeit, was nicht gewünscht ist. HKD erhöhen den Wahrnehmungsradius während der Laufzeit von 1 auf 5, was gleichzietig zu einer Erhöhung der erreichbaren Werte für $f^*(i)$ von $[0,1]$ zu $[0,15]$ führt. Zu Beginn ist die Aufnahme vollständig deterministisch, die Ablage bevorzugt nur dichte und ähnliche Regionen. Mit zunehmender Laufzeit entsteht eine Neigung zur Aufnahme falsch abgelegter Objekte. Der Einfluß der Dichte für die Aufnahme von Objekten erhöht sich, im Gegenzug verringert er sich für die Ablage von Objekten. Dies führt zu einer räumlichen Abgrenzung der Cluster zueinander.

Räumliche Abgrenzung der Cluster Räumliche Abgrenzung der Cluster ist wichtig für die eindeutige Zuordnung von Objekten zu den Clustern während der Analyse der Ergebnisse. Zusammenliegende Cluster entstehen zu Beginn der Laufzeit, wenn an beliebigen Positionen dicht mit ähnlichen Objekten abgelegt wird. Um dies zu verhindern, führen HKD nach einer anfänglichen Clusterphase für die Zeit $[t_{start}, t_{end}]$ eine veränderte Nachbarschaftfunktion $f^*(i)$ ein. Der Skalierungsparameter $1/\sigma^2$ wird durch $1/N_{occ}$, mit N_{occ} als Anzahl der belegten Positionen in der wahrgenommenen Nachbarschaft ersetzt. Nun wird nur die Ähnlichkeit und nicht die Dichte der Objekte untersucht. Dies führt zu einem Aufbrechen der Cluster und einer Verteilung auf bestimmte Bereiche des Grids in einer sortierten Art und Weise. Wird die Nachbarschaftsfunktion bei t_{end} wieder zurückgesetzt, ziehen sich die Cluster wieder zusammen. Die Mittelpunkte formieren sich nun mit großer Wahrscheinlichkeit im Zentrum der belegten Bereiche, was insgesamt zu einer besseren Verteilung und damit räumlicher Abgrenzung der Cluster führt.

Parameter Wichtig ist die Unterteilung der Parameter bzgl. ihrer Abhängigkeit von den Daten. Datenunabhängige Parameter sind die Anzahl der Agenten, die Größe des Kurzzeitgedächtnisses sowie der Zeitraum für den Einsatz der modifizierten Nachbarschaftsfunktion $[t_{start}, t_{end}]$. Zu den datenabhängigen Parametern zählen die Größe des Grids, die Schrittweite für eine Bewegung und die Anzahl der Iterationen. Eine konkrete Diskussion der Parameterwerte ist unter [meine Sektion Handl-Algorithmus] zu finden.

Der Parameter α zur Skalierung der Distanzen zwischen Objekten wird dynamisch, basierend auf der Aktivität der Agenten angepaßt. Die Aktivität ist bestimmt durch die Anzahl erfolgreicher Aufnahmen und Ablagen von Objekten. Jede Ameise besitzt ihr individuelles α, welches zufällig aus $[0,1]$ initialisiert wird. Zu bestimmten Zeitpunkten berechnet sie das Verhältnis $r_{fail} = N_{fail}/N_{active}$, wobei N_{active} für die Anzahl der Bewegungen (hier auf den Wert 100 festgelegt) und N_{fail} für die Anzahl der erfolglosen Ablageversuche steht. Der Parameter α wird durch folgende Regel angepaßt:

$$\alpha \leftarrow \begin{cases} \alpha + 0.01 & r_{fail} > 0.99 \\ \alpha - 0.01 & r_{fail} \leqslant 0.99 \end{cases}$$

HKD evaluieren ihren Ant-Clustering-Algorithmus durch einen Vergleich mit den Al-

gorithmen k-means, average link agglomerative clustering und one-dimensional self-organising maps (1D-SOM). Zusätzlich vergleichen sie die Bestimmung der Anzahl der Cluster in den Daten mit der gap statistic. Für die Evaluation wird ein Set aus repräsentativien synthetischen Testdaten verwendet. Eine umfangreiche Vorverarbeitung der Daten wird eingesetzt, um einen fairen Vergleich der Algorithmen zu ermöglichen. Die implizite Repräsentation des Clusterings nach dem Durchlauf des Ant-based Verfahrens wird in eine explizite Graphendarstellung konvertiert. Die Stärke des Ant-Algorithmus liegt in der impliziten Erkennung der Anzahl der Cluster. Weiterhin ist das Verfahren ziemlich robust bei unterschiedlich großen Clustern. Problematisch wird es, wenn die Distanz der Clusterzentren abnimmt. Hier wird die Performanz des Ant-Algorithmus signifikant schlechter als die von k-means und 1D-SOM.

Ramos und Merelo (RM)[Ramos and Merelo 2002] umgehen die Vorschläge zur Spezialisierung der Agenten und des Algorithmus bei LF und dem damit verbundenen komplexen Parameter-Tuning sowie dem Bedarf an zusätzlicher Rechenleistung. Sie stellen einen Algorithmus ACLUSTER vor, der sich sehr stark am natürlichen Verhalten in Ameisenkolonien orientiert.
Dabei wird ein Modell zur Bewegung mit unterschiedlichen Wahrscheinlichkeiten für verschiedene Zellübergänge verwendet, welches auf künstlichem Pheromon basiert. Pheromone werden von Ameisen auf jeder Position abgegeben. Wird eine Position von vielen Ameisen besucht, bildet sich eine hohe Pheromonkonzentration, die sich attraktiv auf die Entscheidung anderer Ameisen auswirkt. Im Verlauf des Algorithmus entsteht so ein Netz aus Pheromonpfaden. Bei der Bewegung ist die Wahrscheinlichkeit hoch, einer hohen Pheromonkonzentration und damit einem Pfad zu folgen. Man geht davon aus, dass die Ameisen schneller zwischen den verschiedenen Clustern zirkulieren und keine Zeit durch zufällige Bewegung in unerwünschten Regionen verloren geht. Das Pheromon in der Umgebung verdampft mit einer bestimmten Rate. Verschwinden Cluster, werden bestimmte Pfade von den Ameisen nicht mehr besucht. Die Verdampfungsrate wird nun höher als die Abgaberate durch Ameisen und die Pfade verschwinden. Dies führt zu einem gewünschten adaptiven Verhalten. Durch die Verwendung von künstlichem Pheromon wird eine zweite stigmergetische Komponente neben der Positionierung von Objekten im System etabliert.

RM orientieren sich bei der Entwicklung des vorgestellten Verhaltens an einem Modell von Chialvo und Millonas (CM)[Chialvo & Millonas 1995]. Dabei wird die Bewegung von einer Position zu einer Position in der Nachbarschaft beschrieben, die nur auf lokaler Wahrnehmung und Interaktion beruht. Ameisen befinden sich auf einer Position r mit einer Orientierung θ. Gesucht ist die Wahrscheinlichkeit für den Übergang zu einer Nachbarzelle $(r, \theta) \rightarrow (r^*, \theta^*)$. Es wird eine Gewichtungsfunktion für Pheromonkonzentrationen eingeführt:

$$W(\sigma) = \left(1 + \frac{\sigma}{1 + \gamma\sigma}\right)^{\beta}$$

Die Gleichung gibt die relative Wahrscheinlichkeit für die Bewegung zu einer Position r

mit der Pheromonkonzentration $\sigma(r)$ an. Der Parameter β gibt die osmotropotaxische Sensitivität an. Sie steuert den Grad der Unbestimmtheit beim Folgen von Pheromongradienten. $1/\gamma$ steht für die sensorische Kapazität einer Ameise. Sie gibt an, inwieweit feine Unterschiede bei hohen Konzentrationen noch wahrgenommen werden. Desweiteren sei $w(\Delta\theta)$ der Gewichtungsfaktor für die Orientierungsänderung $\Delta\theta$ beim Verlassen von r. Ein größerer Winkel zwischen den Orientierungen wird dabei geringer gewichtet. Die normalisierten Wahrscheinlichkeiten P_{ik} für die die Übergänge von Position k zu Position i auf dem Grid sind gegeben durch:

$$P_{ik} = \frac{W(\sigma_i)\, w(\Delta_i)}{\sum_{j/k} W(\sigma_j)\, w(\Delta_j)}$$

Dabei steht j/k für die Positionen j in der lokalen Nachbarschaft von k. Weiterhin gibt jedes Individuum zu jedem Zeitpunkt t einen konstanten Betrag η an Pheromon ab. Das Pheromon verdampft dabei mit einer Rate von k, was zu einer Verminderung der Konzentration pro Zeitschritt von $\tau \cong 1/k$ führt. RM erweitern den konstanten Pheromonbetrag η um eine dynamische Komponente Δ_h, die von der Art und Anzahl der Objekte in der Nachbarschaft abhängig ist. Die resultierende Pheromonabgabe wird durch $T = \eta + p\Delta_h$ bestimmt, wobei p eine Konstante ist. Für die Bildung der Wahrscheinlichkeiten für die Aufnahme und Ablage von Objekten benutzen RM ein Modell für die Arbeitsteilung bei Insekten (siehe Sektion 2.2), wobei für jeden Task ein entsprechender Stimulus existiert. Je höher der wahrgenommene Stimulus, desto wahrscheinlicher führt ein Individuum den zugehörigen Task aus. RM machen die Tasks Aufnahme und Ablage von Objekten von zwei Stimuli, jeweils für die Anzahl und Ähnlichkeit der Objekte in der Nachbarschaft abhängig. Sie berechnen die Schwellwertfunktionen χ (Anzahl der Objekte), δ (Ablage von Objekten) und ε (Aufnahme von Objekten) wie folgt:

$$\chi = \frac{n^2}{n^2 + 5^2}$$

$$\delta = \left(\frac{k_1}{k_1 + d}\right)^2$$

$$\varepsilon = \left(\frac{d}{k_2 + d}\right)^2$$

Dabei steht n für die Anzahl der Objekte in der wahrgenommenen lokalen Nachbarschaft und d für die normalisierte euklidische Distanz. Durch Kombination der Funktionen erstellen RM verschiedene Berechnungsvorschriften für Aufnahme- und Abgabewahrscheilichkeiten.

Algorithmus ACLUSTER von Ramos & Merelo[Ramos and Merelo 2002]

```
/* Initialization */
  For every item o_i do
    Place o_i randomly on grid
  End For
  For all agents do
    Place agent at randomly selected site
  End For

  /* Main loop */
  For t=1 to t_max do
    For all agents do
      sum=0
      Count the number of items n around site r
      If ((agent unladen) and (site r occupied by item o_i)) then
        For all sites around r with items present do
          Compute d, chi, epsilon and P_p
          Draw random real number R between 0 and 1
          If (R<=P_p) then
            sum=sum+1
          End If
        End For
        If ((sum>=n/2) or (n=0) then
          Pick up item o_i
        End If
      Else If ((agent carrying item o_i) and (site r empty)) then
        For all sites around r with items present do
          Compute d, chi, delta and P_d
          Draw random real number R between 0 and 1
          If (R<=P_d) then
            sum=sum+1
          End If
        End For
        If (sum>=n/2) then
          Drop item o_i
        End If
      End If
      Compute W(sigma) and P_ik
      Move to a selected neighboring site not occupied by other agent
      Count the number of items n around that new site r
      Increase pheromone at site r according to n, that is:
      P_r  = P_r  + [eta+(n/alpha)]
    End For
    Evaporate pheromone by K, at all grid sites
  End For
  Print location of items

  /* Values of parameters used in experiments */
  k_1 = 0.1, k_2 = 0.3, K = 0.015, eta = 0.07, alpha = 400, beta = 3.5,
       gamma = 0.2, t_max = 10^6 steps
```

Dies führt zum Algorithmus ACLUSTER. Wie auch LF testen RM ihren Algorithmus an einem Clusteringproblem mit vier Klassen zu je 200 Gauß-verteilten Punkten.

Sie verwenden ein nicht-parametrisches Grid ohne Grenzen von 57x57 Positionen. Es werden 80 Ameisen eingesetzt. Dies wurde durch folgende empirisch ermittelte Regeln berechnet: $A = 4 \cdot n_o$, $n_a = A/40$ und $n_a/n_o = 0.1$. n_o gibt dabei die Anzahl der Objekte, n_a die Anzahl der zu verwenden Ameisen sowie A die Anzahl der Gridzellen an.

Zusammenfassend werden Objekte gleicher Datencluster wieder gleichen Clustern zugeordnet. Es kann eine schnellere Verringerung der Entropie mit fortschreitendem Clustering erreicht werden als bei LF.

Als Anwendung ihres Verfahrens stellen RM die Clusterung von Wörtern nach deren Bedeutung vor. Für die Feature-Extraktion wird das LSA-Verfahren (Latent Semantic Analysis) eingesetzt.

Bei den bisher vorgestellten Cluster- und Sortierverfahren entstehen Anhäufungen von Objekten an beliebigen Positionen. Der Algorithmus von LF ist nicht-parametrisch. D.h. die Anzahl der Cluster sowie deren Ort müssen nicht spezifiziert werden. Trotzdem existieren parametrische Erweiterungen für LF. Er stellt eine Mischform zwischen hierarchischen [3] und partitionierenden [4] Clustering-Algorithmen dar.

2.4 Selbstorganisation und Templates in Zusammenhang mit Organisation, Sortierung und Clusterung von Objekten

Im vorherigen Abschnitt (siehe Sektion 2.3) wurde die Analyse, Sortierung und Organisation von Objekten vorgestellt mit dem natürlichen Vorbild der Ansammlung von Objekten sowie der Anordnung von Brut bei verschiedenen Ameisenarten. Dabei richteten sich die Entscheidungen für das Aufnehmen und Ablegen von Objekten nur nach Art und Anzahl der Objekte in der Nachbarschaft. Die Cluster und Strukturen der Selbstorganisation entstanden an beliebigen Positionen. Die Verfahren werden daher auch nicht parametrisch genannt. Nun sollen Templates (Prepatterns) zur Bevorzugung von Clustern an vordefinierten Positionen eingeführt werden. Dabei handelt es sich um Informationen, die von den Agenten zusätzlich zu Objektinformationen aus der Umgebung aufgenommen werden können. Templates geben eine Auskunft darüber, ob ein Ort für die Formation eines Clusters mehr oder weniger geeignet ist. Sie bestimmen somit die Organisation der Strukturen durch die Individuen. Es existieren auf Templates basierende parametrische Verfahren für Clustering und Graph Partitioning.

[3] Initial ist jedes Objekt ein Cluster. Iterativ findet Merging statt.
[4] Initiale Clusterung der Objekte wird iterativ verfeinert, indem Objekte zu anderen Clustern wandern.

2.4.1 Natur

In der Natur können Templates bsp. durch natürliche Gradienten, Felder oder heterogene Strukturen entstehen. In verschiedenen Ameisenarten spielen auch Temperatur und Feuchtigkeit eine Rolle für die Organisation des Nests sowie der Anordnung der Brut [Brian 1983][Ceusters 1980][Ceusters 1986]. Bei der Ameisenart Leptothorax albipennis wurde der Bau von Mauern um das Nest herum beobachtet[Franks, Wilby, Silverman and Tofts 1992][Franks and Deneubourg 1997]. Dabei werden in einem bestimmten Abstand zum Nest Sandkörner, Steine oder Erdkügelchen abgelegt. Mit zunehmendem Bau kommt auch Selbstorganisation mit ins Spiel: Je höher die Mauer, desto attraktiver wird sie für die Ablage neuer Bausubstanz.

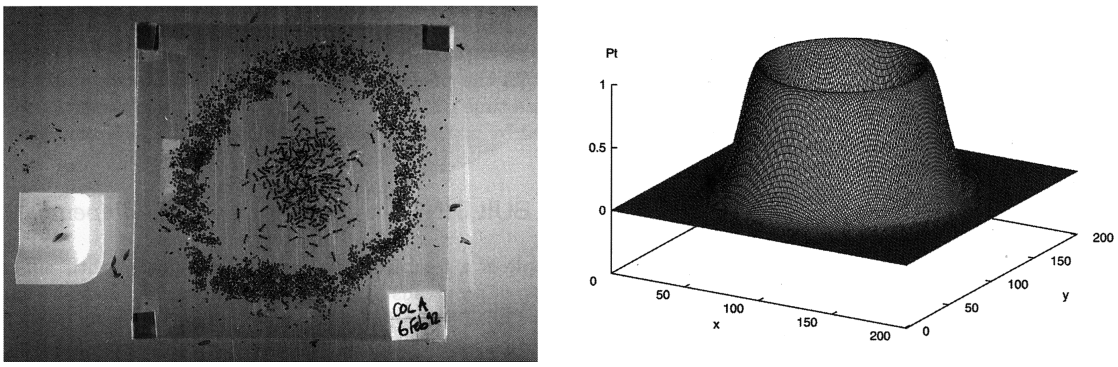

Abbildung 2.11: Experiment bei Ameisen zum Bau einer Mauer um das Nest mit der Königin in der Mitte (links). Template für die Simulation (rechts). [Bonabeau, Dorigo and Theraulaz 1999]

Termiten bauen ein sog. Royal Chamber um ihre Königin[Grassé 1984]. Dabei handelt es sich um eine Art Verpuppung. Die Königin gibt ein spezielles Pheromon ab, dessen Konzentration mit zunehmendem Abstand geringer wird. Das Pheromon signalisiert den Individuen, daß gebaut werden soll und agiert daher als Template.

2.4.2 Technik

Selbstorganisation basierend auf Templates stellt einen parametrischen Ansatz für das Problem der Sortierung und Organisation von Datenobjekten dar. Anwendung finden solche constraint-basierten Verfahren, wenn die Anzahl der Cluster oder Klassen vorgegeben ist. Um ein Template zu definieren wird eine Template-Wahrscheinlichkeit $P_t\,(x, y)$ eingeführt [Bonabeau, Dorigo and Theraulaz 1999], wobei x und y für die planaren Koordinaten eines Agenten auf dem $\mathbb{R}^2$-Grid stehen. Nimmt man o.B.d.A. an, daß das Grid mit $[0, 1] \times [0, 1]$ beschränkt ist, und darin vier getrennte Clusterpositionen angelegt

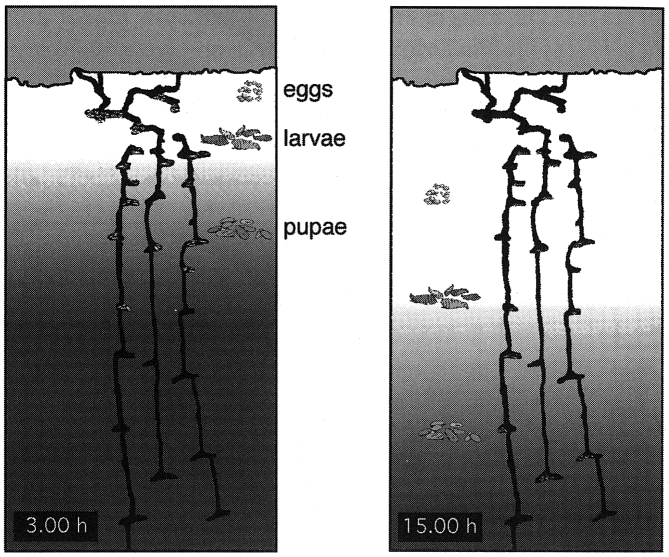

Abbildung 2.12: Die räumliche Anordnung von Eiern, Larven und Puppen bei einer speziellen Ameisenart (links: 3.00 Uhr, rechts: 15.00 Uhr). Der Temperaturgradient hat entlang der Tiefenachse bestimmt den Ort des jeweiligen Clusters. [Bonabeau, Dorigo and Theraulaz 1999]

werden sollen, kann man die Wahrscheinlichkeit $P_t(x, y)$ wie folgt konkret definieren:

$$P_t(x,y) = a\left[e^{-\frac{x^2+y^2}{\sigma^2}} + e^{-\frac{(x-1)^2+y^2}{\sigma^2}} + e^{-\frac{(x-1)^2+(y-1)^2}{\sigma^2}} + e^{-\frac{x^2+(y-1)^2}{\sigma^2}} - b\right]$$

Dabei gibt σ^2 den Gradienten, also die Steilheit des Templates an. Die Konstanten a und b repräsentieren Parameter, die für σ^2 angepaßt werden müssen, damit $0 \leqslant P_t(x, y) \leqslant 1$ gilt. In den vier Ecken des Grids ist die Wahrscheinlichkeit hoch, was zu einer Favorisierung der Positionen für die Bildung von Clustern betrachtet werden kann.

Im Folgenden sollen parametrische Erweiterungen der nicht-parametrischen Algorithmen (siehe Sektion 2.3) bzgl. Clustering (LF) und Graph Partitioning (KLS) vorgestellt werden.

Für das parametrische Clustering kann der LF-Algorithmus ziemlich einfach angepaßt werden. Dabei ist $r = (x_i, y_i)$ die Position eines Objektes o_i. Die Wahrscheinlichkeiten für Aufnahme $p_p(o_i)$ und Ablage $p_d(o_i)$ von Objekten sind wie folgt definiert:

$$p_p(o_i) = 0.7\left(\frac{k_1}{k_1 + f(o_i)}\right)^2 + 0.3\,(1 - P_t(r_i))$$

$$p_d(o_i) = \begin{cases} 2P_t(r_i)\,f(o_i) & fuer \quad f(o_i) < k_2 \\ P_t(r_i) & fuer \quad f(o_i) \geqslant k_2 \end{cases}$$

Das Aufnahmeverhalten wird für $P_t(r_i) \rightarrow 1$, das Ablageverhalten für $P_t(r_i) \rightarrow 0$ begünstigt.

Gleichermaßen läßt sich der KLS-Agorithmus für das Lösen des Graph Partitioning Problems in eine parametrische Form überführen. Die Wahrscheinlichkeiten für Aufnahme $p_p(v_i)$ und Ablage $p_d(v_i)$ von Vertices v_i und Positionen w_i sind wie folgt definiert:

$$p_p(v_i) = 0.7\left(\frac{k_1}{k_1 + f(v_i)}\right)^2 + 0.3\left(1 - P_t(w_i)\right)$$

$$p_d(v_i) = P_t(w_i)\left(\frac{f(v_i)}{k_2 + f(v_i)}\right)^2$$

Problematisch wird der Fall, in denen ein Graph mehr oder weniger Cluster c enthält, als spezifiziert. Hier kann ein Regulierungs-Mechanismus [Bonabeau, Dorigo and Theraulaz 1999] eingesetzt werden.

2.5 Nestbau und Konstruktion von Architekturen

Soziale Insekten weisen ziemlich komplexe Nestarchitekturen auf. Hier soll ein Agentenbasiertes Modell vorgestellt werden, um dieses Verhalten zu simulieren, zu untersuchen und zu erklären.

Soziale Insekten kommunizieren bei vielen Tasks indirekt durch Modifikation und Reaktion auf die Umgebung. Diese Form der Kommunikation wird Stigmergy genannt. Man unterscheidet zwei Typen von Stigmergy: *quantitative Stigmergy (kontinuierlich)* und *qualitative Stigmergy (diskret)*. Quantitative Stigmergy liegt vor, wenn der Stimulus für einem Task verschieden hoch sein kann. Sie findet beispielsweise Anwendung bei der Konstruktion von Termitenbauten[Grassé 1959]. Bei qualitativer Stigmergy unterscheidet man verschieden Klassen von Stimuli, die jeweils ein anderes Verhalten auslösen. Sie erklärt auch das Nestbauverhalten einiger Wespenarten.

Das vorgestellte Modell basiert auf diskreter Stigmergy. Dabei gibt es unterschiedliche Bauteile, die zusammen komplexe Strukturen ergeben. Die Anbringung eines Bauteils eines bestimten Typs richtet sich nach der Konfiguration der Bauteile in der Nachbarschaft des Agenten. Die entstehenden Muster können mehr oder weniger strukturiert sein. Sie werden begutachtet und bewertet. Diese Bewertung kann über die Fitnessfunktion eines Genetischen Algorithmus erfolgen, wobei sich die jeweilige Fitness aus der Qualität der Struktur ergibt.
Über eine solche künstliche Evolution kann der Raum an möglichen Architekturen und Mustern effizient exploriert werden.

2.5.1 Natur

Die Nestarchitekturen sozialer Insekten sind oftmals sehr komplex und erfüllen mehrere Funktionen, wie z.B. Schutz vor Feinden, Fortpflanzung oder Wärmeregulierung. Sie bilden damit die Infrastruktur des sozialen Lebens. Der Vorgang des Nestbauens ist einer der spektakulärsten Tasks, da sich hier der größte Unterschied zwischen Einzelaktionen und dem globalen enstehenden Ganzen auf Kolonielevel ergibt. Anfänglich ging man davon aus, daß Insekten ein globales Modell ihres Nests verinnerlichen. Entscheidungen werden auf Basis des Modells getroffen. Je komplexer also das Verhalten der Insekten, desto komplexer wird die enstehende Nestarchitektur. Heute weicht man von dieser Vorstellung ab. Man geht davon aus, daß Insekten einfach strukturierte Lebewesen sind, die nie in der Lage wären, eine komplexe Repräsentation des Nests zu speichern und zu verarbeiten. Die Komplexität beruht auf dem Prinzip von Stigmergy. Dabei nehmen Insekten Reize in ihrer lokalen Umgebung auf und reagieren mit individuellem adaptivem Verhalten. Die Komplexität des Nests hängt u.a. von der Diversität und Größe der Kolonie sowie der Struktur der bisherigen Architektur ab. Je mehr verschiedene Individuen es in einer Kolonie gibt, desto verschiedener sind die erzeugten Stimuli. Je komplexer die Struktur einer Architektur ist, desto vielseitiger werden die Reaktionen darauf ausfallen und damit noch interessantere Strukturen hervorbringen[Grassé 1984]. Zwei Mechanismen spielen eine Hauptrolle bei Bauaktivitäten: *Selbstorganisation*[Deneubourg and Goss 1989][Bonabeau, Theraulaz and Deneubourg 1997] und *diskrete Stigmergy*[Theraulaz & Bonabeau 1995a][Theraulaz & Bonabeau 1995b].

Selbstorganisation ist eine Folge einer Menge von Interkationen. Solche Interaktionen können auf direkter oder indirekter Kommunikation (Stigmergy) basieren. Die Intensität der Interaktionen ist abhängig von der Intensität der Reize. Es ergibt sich eine Art „Schneeballeffekt“. Man nennt diese Kombination von Stigmergy und Selbstorganisation auch quantitative Stigmergy. Hier entscheidet die Höhe eines bestimmten Stimulus (z.B. Gradienten, Pheromonfelder). Diskrete Stigmergy bedeutet, daß es verschiedene Typen von Stimuli gibt, auf die jeweils verschiedene Reaktionen folgen. In der Natur existieren beide Arten von Stigmergy parallel, so daß es für unterschiedliche Arten von Stimuli auch ein quantitatives Unterscheidungsmerkmal gibt. Am Beispiel diskreter Stigmergy kann man ein Konstruktionsmodell basierend auf verschiedenen Typen von Bausteinen und Reizen durch Strukturen in der Nachbarschaft erklären [Theraulaz & Bonabeau 1995a] [Theraulaz & Bonabeau 1995b][Bonabeau, Théraulaz, Arpin and Sardet 1994].

2.5.2 Technik

Um diskrete Stigmergy näher zu untersuchen, haben Théraulaz, Bonabeau, Arpin und Sardet [Theraulaz & Bonabeau 1995a] [Theraulaz & Bonabeau 1995b] [Bonabeau, Théraulaz, Arpin and Sardet 1994] eine Sammlung einfacher Algorithmen

basierend auf dem Verhalten einiger Wespenarten beim Nestbau entwickelt. Dabei bewegen sich Agenten in einem 3D-Raum fort und tragen einen Baustein eines bestimmten Typs. Sie können Konstruktionen in der lokalen Nachbarschaft von 27 Feldern wahrnehmen (Abbildung 2.13) und danach ihre Entscheidungen ausrichten. Wenn eine stimulierende Konfiguration angetroffen wird, ist die Wahrscheinlichkeit für die Ablage des Bausteins gleich 1. Die Entscheidung ist also nicht deterministisch. Es dürfen keine Bausteine entnommen werden. Begonnen wird die Simulation mit einem zufällig platzierten Baustein. Die Entscheidungen basieren auf einer Menge von kombinierbaren Reiz-Reaktions-Regeln (siehe Konstruktions-Algorithmus, Sektion 2.5.2).

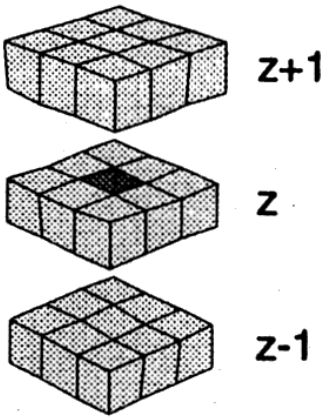

Abbildung 2.13: Schematische Repräsentation der Nachbarschaft, die eine Ameise wahrnehmen auf einem kubischen Grid wahrnehmen kann. [Bonabeau, Dorigo and Theraulaz 1999]

Konstruktions-Algorithmus[Bonabeau, Dorigo and Theraulaz 1999]

```
1  /* Initialization */
2  Construct looup table /* identical for all agents */
3  Put one initial brick at predefined site /* top of grid */
4  For k=1 to m do
5    assign agent k a random unoccupied site /* distribute the m agents */
6  End For
7
8  /* Main Loop */
9  For t=1 to t_max do
10   For k=1 to m do
11     Sense local configuration
12     If(local configuration is in lookup table) then
13       Deposit brick specified by lookup table
14       Draw new brick
15     Else
16       Do not deposit brick
17     End If
```

```
18      Move to randomly selected, unoccupied neighboring site
19    End For
20  End For
21
22  /* Values of parameters used in simulations */
23  m=10
```

Théraulaz und Bonabeau fanden heraus, daß sich strukturierte Formen nur mit constraint-basierten koordinierten Algorithmen erreichen lassen. Dabei dürfen z.B. verschiedene stimulierende Konfigurationen nicht räumlich und zeitlich überlappen. Dies führt in verschiedenen zeitlichen Bauabschnitten zu einer konsistenten Entwicklung von stimulierenden Strukturen.

Um den Raum von Architekturen zu explorieren nutzen Bonabeau et al.[Bonabeau, Guérin, Snyers, Kuntz, Théraulaz and Cogne 1998] einen Genetischen Algorithmus (GA). Jede Lösung, kodiert als Genom, kann durch eine Fitnessfunktion evaluiert werden. Aufgrund der Fitnesswerte wird eine Selektion vorgenommen, was zu einem abstrahierten Evolutionsprozess führt. Ziel ist es, eine Population von Genomen mit hohen Fitnesswerten, also eine Menge von guten Lösungen abzuleiten. Die Fitnessfunktion wurde aufgrund von Aussagen zu Strukturen durch 17 menschliche Beobachter erstellt. Es wurde herausgefunden, daß Struktur u.a. folgende Vorraussetzungen hat:

- Die Algorithmen besitzen viele Regeln.
- Die Regeln stehen miteinander in Zusammenhang und können komplementär oder korelliert sein.
- Struktur wird charakterisiert durch die Wiederholung atomarer Strukturen. Komplexität wird durch große atomare Strukturen erreicht. Daher sollten Wiederholungen von großen atomaren Strukturen entstehen.

Nicht nur Struktur und Komplexität einer Architektur können als Basis für eine Fitnessfunktion dienen. Aus Sicht der Biologie kann die Fitness durch die Funktionalität bestimmt werden, was zu einer Art Kosten-Nutzen-Rechnung führt[Krink and Vollrath 1997].

Die Idee der Konstruktion mittels Agenten durch abgelegte Bausteine kann weiter abstrahiert werden, um ein Modell zu bilden, bei dem Bausteine eigene Agenten, Roboter oder Maschinenteile sind, die sich mit anderen oder bereits entstandenen Strukturen verbinden [Bowden, Terfort, Carbeck and Whitesides 1997] [Hosokawa, Shimoyama and Miura 1995] [Hosokawa, Shimoyama and Miura 1996] [Hosokawa, Shimoyama and Miura 1997]. Zusammen mit evolutionären Algorithmen können selbstkonstruierende oder selbstrekonfigurierende Modelle exploriert werden, um funktionale Systeme zu generieren.

Anwendungsgebiete für selbständige Konstruktion sind Mikroelektronik, Optik, Displays und Mikro-Elektromechanische Systeme, aber auch Architektur[Broughton, Tan and Coates 1997][Coates, Healy, Lamb and Voon 1996] und Kunst[Sims 1991].

2.6 Kooperativer Transport bei Insekten und Robotern

Bisher wurden sequentielle Arbeitsabläufe betrachtet. Dabei führte ein Individuum einen Task zu einem Zeitpunkt t aus. Zum Zeitpunkt $t+1$ konnte ein anderes Individuum die durch die Ausführung des Tasks veränderte Umgebung wahrnehmen (Stigmergy) und darauf aufbauend einen weiteren Task ausführen. Hier soll nun die parallele kooperative Aufteilung von Tasks zu einem Zeitpunkt betrachtet werden. Interessant ist hier das Verhalten von Ameisen beim Transport von Beute, die viel schwerer und größer ist, als mehrere Individuen zusammen. Dabei können zwei Stufen eingeteilt werden:

1. Eine einzelne Ameise findet Beute und versucht, sie zum Nest zu tragen. Funktioniert das nicht, versucht sie weitere Artgenossen direkt oder indirekt zu rekrutieren.

2. Sind genug Individuen am Ort, um die Beute zu transportieren, findet eine kooperative Arbeitsaufteilung statt. Die Beute kann nun zum Nest gebracht werden. Dieses Verhalten ist noch nicht ganz verstanden.

Aus den empirischen Untersuchungen zum kooperativen Transport bei Ameisen wurde ein weitverbreitetes Modell für die robotische Anwendung konstruiert.

2.6.1 Natur

Das Aufsuchen und kooperative Transportieren von Beute wurde für viele Ameisenarten beobachtet[Sudd 1963] [Robson and Traniello 1998] [Traniello and Beshers 1991]. Dies kann sehr impressive Formen annehmen: Moffett berichtet in [Moffet 1988] von einer Gruppe von etwa 100 Ameisen, die einen Regenwurm von 10cm Länge tragen. Bei einem Gewicht von 1.92g bleiben 19.2mg verteilt auf jede Ameise. Bei einem Gewicht eines Arbeiters von etwa 0.3-0.4mg ergibt sich etwa das 55-fache des Körpergewichts zum Tragen. Einzelne Arbeiter können hingegen „nur“ etwa das 5-fache Gewicht transportieren. Dies ist möglich, da beim gemeinsamen Transport das Problem des Ausbalancierens für eine einzelne Ameise entfällt.

Wichtige Fragen der Forscher in diesem Zusammenhang betreffen Vorteile des kooperativen Transports gegenüber dem Einzeltransport, Rekrutierung anderer Individuen,

Koordination beim Transport und Vermeidung von Deadlocks sowie Entscheidungen über eigene Fähigkeiten und Feststellung, ob genug Individuen rekrutiert sind, um den Transport durchzuführen.

Eine Ameise versucht zuerst immer, den Task allein zu bewältigen. Dabei bewegt sie das Objekt und sich selbst, um eine geeignete Position und Technik zu finden. Scheitert dies, werden andere Ameisen rekrutiert. Dabei ist nach Sudd[Sudd 1965] die Zeit, die für das Probieren aufgebracht wird abhängig vom Gewicht der Beute. Mit zunehmendem Gewicht, sinkt die Zeit.
Das Rekrutieren weiterer Individuen kann nach Hölldobler [Hölldobler and Wilson 1978] [Hölldobler 1990] in zwei Kategorien erfolgen: short-range recruitment (SRR) und long-range recruitment (LRR). Beim SRR gibt eine Ameise kurz nach Aufspüren der Beute ein Pheromon über Drüsen in die Luft ab. Diese Form der Rekrutierung wirkt in einem Umkreis von bis zu 2m. Kann die Beute durch die rekrutierten Ameisen nicht transportiert werden, wird LRR eingesetzt. Dabei legen Ameisen Pheromon-Pfade vom Nest zur Beute.

Für die Koordination des Transports ist Stigmergy verantwortlich. Dabei zieht eine Ameise in eine bestimmte Richtung. Benachbarte beteiligte Ameisen nehmen den Reiz wahr und ziehen in dieselbe Richtung. Alle beteligten Ameisen werden nach kurzer Zeit in die selbe Richtung ziehen und das Objekt dadurch fortbewegen. Ergibt sich eine Stagnation oder ein Deadlock, versuchen die Ameisen, sich neu zu positionieren [Sudd 1960][Sudd 1965].

2.6.2 Technik

Das vorgestellte Prinzip des kooperativen Transports von Beute bei Ameisen wird als Grundlage für dezentralisierte robotische Implementationen ohne direkte Kommunikation verwendet. Es existieren sehr viele Arbeiten zum Task, wobei Roboter gemeinsam Objekt-Boxen an bestimmte Orte schieben (u.a. [Bay 1995] [Caloud, Choi, Latombe, Pape and Yim 1990] [Donald, Jennings and Ris 1994] [Doty and van Aken 1993]). Kube und Zhang stellen mit [Kube and Zhang 1994] [Kube and Zhang 1995] [Kube and Zhang 1997] einen konsistenten Ansatz für Ant-Based-Transport in der Robotik vor. Das System ist kostengünstig, robust und flexibel. Um das Verhalten der Roboter abzubilden verwenden sie Adaptive Logic Networks (ALNs), die eine vereinfachte Form von Artificial Neural Networks (ANNs) darstellen. Dabei wird die Funktionalität durch einen Binärbaum von logischen UND-/ODER-Knoten dargestellt. Das Training funktioniert ähnlich wie bei ANNs mit Testdaten, bestehend aus Eingangs- und zugehörigen gewünschten Ausgangsdaten in Form von Vektoren.

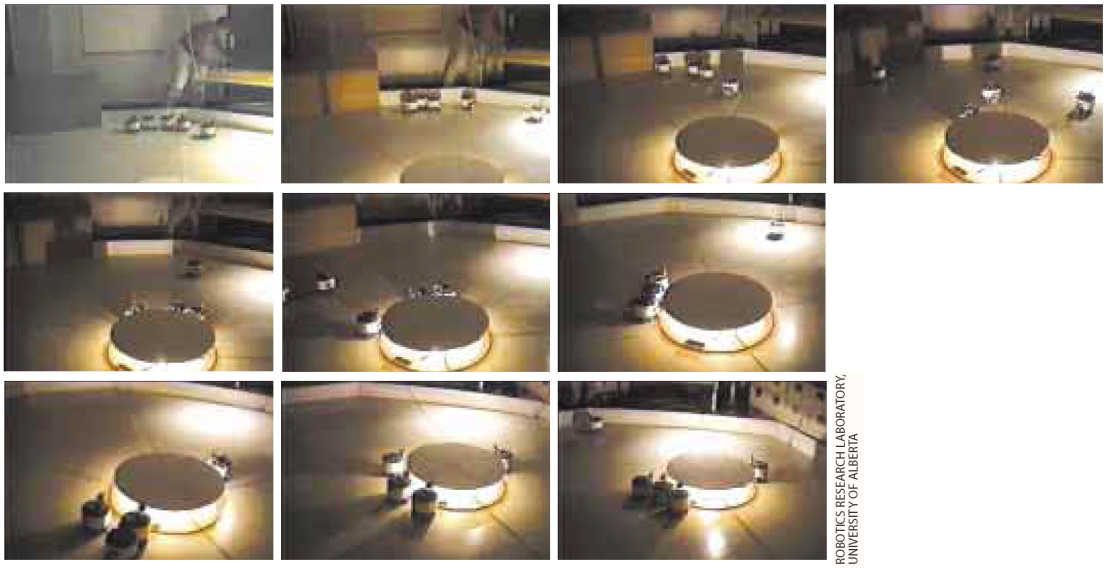

Abbildung 2.14: Mehrere autonome Roboter arbeiten zusammen. [Bonabeau and Théraulaz 2000]

2.7 Zusammenfassung

Soziale Insekten und andere natürliche Systeme können dazu beitragen, Algorithmen und künstliche Problemlösesysteme zu entwickeln. Künstliche Swarm Intelligence Systeme weisen wichtige Vorteile natürlicher Kolonien auf: Flexibilität, Robustheit, Dezentralisierung und Selbst-Organisation. SI-basierte Ansätze werden immer vielversprechender, da die Aufgaben und Probleme immer komplexer, dynamischer und größer werden.

Trotzdem sollten einige Punkte bei der Implementation von SI-Systemen beachtet werden:

1. Es wäre sehr nützlich, wenn sich eine Methode zum Parametrisieren/Spezialisieren/Trainieren von SI-Systemen entwickeln ließe, wie beispielsweise für das Trainieren eines Neural Networks. Der Raum aller möglichen Parameterkonstellationen von SI-Systemen kann u.a. mit einer natürlich inspirierten evolutionären Technik wie Genetischen Algorithmen (GA) exploriert werden[Goldberg 1989][Holland 1975].

2. Wichtig sind Fragestellungen zur Definition von SI-Systemen, beispielsweise zur Komplexität und Individualität der einzelnen Agenten, zu ihrer Lernfähigkeit sowie zur Art der Kommunikation. Meist sind diese Fragestellungen sehr problemspezifisch. Eine Möglichkeit wäre, für einen Task mit simplen Agenten zu starten und

falls notwendig die Komplexität zu Erhöhen. Die Beobachtung, ob Notwendigkeit vorliegt, kann auch automatisch erfolgen.

3. SI-Systeme weisen wie alle adaptiven Problemlösesysteme zwei Hauptprobleme auf:

   a) Sie sind nicht absolut verläßlich. Die Dynamik in nicht-deterministischen Systeme ist ziemlich schwierig vorherzusagen. Weiterhin kann die Leistungsfähigkeit während der Laufzeit unter unterschiedlichen Bedingungen ziemlich stark schwanken. Dies bedeutet ein hohes Risiko.

   b) Es sind keine standardisierten Benchmarks verfügbar, um die Performace solcher Systeme zu messen.

   Die genannten Probleme treten auf, da der vollständige theoretische Hintergrund solcher Systeme entweder noch nicht verstanden oder beschrieben werden kann.

Auch wenn SI-Syteme einen gewinnbringenden und interessanten Ansatz für Problemlösestrategien darstellen, wird noch viel Arbeit benötigt, ihr enormes Potential zu erforschen und zu kontrollieren. Einige wichtige Antworten zu Fragestellungen bzgl. der Erforschung von SI-Systemen im Zusammenhang mit der Analyse und Clusterung von Datenobjekten (siehe Abschnitt 2.3) werden mit dieser Arbeit vorgestellt.

3 Entwickelte Module für die Verwendung in Swarm-Systemen

3.1 Generierung normalverteilter Objektcluster

Für die Anwendung der entwickelten Swarm-Systeme werden Testdatensätze benötigt. Ein solcher Datensatz S besteht aus m Objekten mit jeweils n konkreten Eigenschaftswerten, $S = \{x_i = [x_{i1}, \ldots, x_{in}]\}, i = 1 \ldots m$. Die Objekte können geometrisch als Punkte im $\mathbb{R}^n$ interpretiert werden.
Die Generierung der Punkte kann durch Analyse/Messung realer Objekte oder durch künstliche Berechnung erfolgen. Die künstliche Erzeugung von Datensätzen kann durch eine spezielle Berechnungsvorschrift oder zufällig vorgenommen werden. Eine solche deterministische Berechnungsvorschrift ist beispielsweise die Generierung einer Punktwolke, die eine spezielle geometrische Form repräsentiert. Wir haben Verfahren entwickelt, beispielsweise kugel- und quaderförmige Anordnungen von Punkten zu erzeugen.
Unter nicht-deterministischen Erzeugungsverfahren versteht man die Erzeugung mit Hilfe eines Zufallszahlengenerators. Die einfachste Form stellt die gleichverteilte Erzeugung von Punkten dar. Die enstehenden Datensätze sind aber nur von geringem Wert für die Erprobung der Systeme, da sie keine natürlichen Cluster enthalten, die durch die Algorithmen aufgefunden werden können. Datensätze mit Clustern bzw. Punktwolken lassen sich durch eine $\mathcal{N}(\mu, \sigma)$-Normalverteilung der jeweiligen Attributwerte erzeugen.

Im Folgenden wird die entwickelte Funktionalität zur Erzeugung normalverteilter Punkte im $\mathbb{R}^n$ als Funktion sowie als eigenständige Anwendung vorgestellt.

3.1.1 Theorie

Die Erzeugung der Attributwerte x_i wird mithilfe einer Normalverteilung $\mathcal{N}(\mu, \sigma)$ mit der Wahrscheinlichkeitsfunktion $P(x_i)$ unter Angabe des Erwartungswertes μ und der Standardabweichung σ durchgeführt:

$$P(x_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}, -\infty < x < +\infty$$

Ist für eine Dimension nur eine Verteilung gewünscht, kann die effiziente Polar-Methode von Marsaglia [Wikipedia, Normalverteilung] zur Generierung der Attributwerte verwendet werden. Der Algorithmus funktioniert wie folgt:

1. Generiere zwei gleichverteilte Zufallsvariablen $u_1, u_2 = U(0,1)$
2. Berechne $v = (2u_1 - 1)^2 + (2u_2 - 1)^2$. Falls $v \geqslant 1$ wiederhole 1.
3. $x = (2u_1 - 1)\left(\frac{-2\log v}{v}\right)^{1/2}$

Die Methode simuliert eine Standard-Normalverteilung mit $\mathcal{N}(0,1)$. Durch lineare Transformation der generierten x_i läßt sich eine Normalverteilung mit $\mathcal{N}(\mu,\sigma)$ nachbilden:

$$x_i \sim \mathcal{N}(\mu,\sigma) = x_i \sim \mathcal{N}(0,1)\sqrt{\sigma} + \mu$$

Werden mehrere Verteilungen für ein Attribut benötigt, wird eine Überlagerung mehrerer Normalverteilungsfunktionen vorgenommen. Die Überlagerung muß so transformiert werden, daß das Integral im definierten Intervall $[lowerbound, upperbound]$ einen Wert von 1 besitzt. Zur Berechnung einer Verteilungsfunktion wird hier die effiziente Approximation durch Abramovitz und Stegun [Abramowitz 1965] verwendet. Im Folgenden wird dies kurz begründet.

Durch Integration der Wahrscheinlichkeitsfunktion $P(x_i)$ ergibt sich die Verteilungsfunktion $F(x_i)$:

$$F(x_i) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x_i} e^{-\frac{(\zeta-\mu)^2}{2\sigma^2}} d\zeta$$

Das Integral kann nicht durch eine endliche Kombination von elementaren Funktionen dargestellt werden. Trotzdem läßt es sich als unendliche Reihe formulieren:

$$F(x) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{+\infty} \frac{(-1)^n z^{2n+1}}{n!2^n(2n+1)}, z = \frac{x-\mu}{\sigma}, z \geqslant 0$$

Die Gleichung ist zwar mathematisch korrekt, aber sehr ineffizient zu berechnen, da sie nur sehr langsam konvergiert. Eine effiziente rationale Approximation der Verteilungsfunktion wurde durch Abramowitz und Stegun entwickelt:

$$F(x) = 1 - \frac{1}{2}\left[\frac{1}{1 + c_1 z + c_2 z^2 + c_3 z^3 + c_4 z^4}\right]^4 + R(z), z = \frac{x-\mu}{\sigma}, z \geqslant 0$$

mit den Koeffizienten

$$c_1 = 0.196854 \qquad c_2 = 0.115194$$
$$c_3 = 0.000344 \qquad c_4 = 0.019527$$

Der absolute Wert des Approximationfehlers $R(z)$ ist immer kleiner oder gleich 0.00025. Für $z \leqslant 0$ wird die Beziehung

$$F(-x) = 1 - F(x)$$

angewendet.

Wird für ein Attribut keine Normalverteilung angegeben, so erhalten alle Punkte den Wert der Untergrenze des definierten Intervalls *lowerbound*.

3.1.2 Implementierung

Die Funktion zur Erzeugung eines normalverteilten Datensatzes S wird über die Klasse `CRandom` bereitgestellt. Der zugehörigen Funktion `vector<float*> CRandom::GenerateData(const char *paramfile, const char *outfile)` wird ein Parameter-File sowie ein Output-File für die erzeugten Punkte übergeben. Die Punkte werden zeilenweise mit kommagetrennten Attributwerten in das File geschrieben. Desweiteren gibt die Funktion einen vector mit den erzeugten Punkten zurück.

Pseudo-Code der Funktion zur Generierung normalverteilter Objektcluster im $\mathbb{R}^n$

```
1  /*! Generates a vector of gaussian distributed feature space points.
2   *
3   * @param paramfile Pointer to parameter-file.
4   * @param outfile Pointer to output-file.
5   * @return Vector with points.
6   */
7  vector<float*> CRandom::GenerateData(const char *paramfile, const char
       *outfile)
8    {
9    read paramfile
10   initialize return_vector
11   for feature_space_dimension=0 to n-1
12     for point=0 to number_of_points-1
13       if(number_of_distributions == 0)
14         point_value = lowerbound
15       else if(number_of_distributions == 1)
16         point_value = polar_method()
17       else if(number_of_distributions >= 1)
18         generate overlay_of_distributions
19         scale overlay_of_distributions
20         point_value = overlay_of_distributions() //with approximization
               from Abramowitz and Stegun
21     end for
```

```
22    end for
23    write outfile
24    return points_vector
25    }
```

3.1.3 Anwendung

Die Funktionalität zur Erzeugung normalverteilter Punkte im $\mathbb{R}^n$ kann innerhalb einer Software sowie durch das mitgelieferte Programm `gaussian.exe` verwendet werden. Im ersten Fall bekommt die Funktion einen Pointer auf ein geöffnetes Parameter-File sowie einen Pointer auf ein Output-File übergeben. Als Rückgabewert wird ein Vektor mit Punkten erzeugt.
Im zweiten Fall erstellt `gaussian.exe` ein OutputFile `points.txt` im gleichen Verzeichnis (siehe Abbildung 3.1). Als Parameter-File wird eine Datei `params.txt` im gleichen Verzeichnis erwartet. Ein solches Parameter-File ist folgendermaßen aufgebaut:

```
1  1000  % number of feature space points to generate
2
3  % feel free to edit this file
4  %
5  % the configuration in this demo file creates the discussed feature-
       space clusters
6  % (something like 4 3D-cluster-balls)
7
8  [FEATURE 0]
9  2     % number of gaussian distributions
10 0.0     % lowerbound
11 1000.0  % upperbound
12 1.0     % stepsize
13 250.0 % mu1
14 75.0    % sigma1
15 750.0   % mu2
16 75.0    % sigma2
17
18 [FEATURE 1]
19 2     % number of gaussian distributions
20 0.0     % lowerbound
21 1000.0  % upperbound
22 1.0     % stepsize
23 250.0 % mu1
24 75.0  % sigma1
25 750.0 % mu2
26 75.0  % sigma2
27
28 [FEATURE 2]
29 1     % number of gaussian distributions
30 0.0     % lowerbound
31 1000.0  % upperbound
32 1.0     % stepsize
```

```
33  500.0 % mu1
34  75.0  % sigma1
```

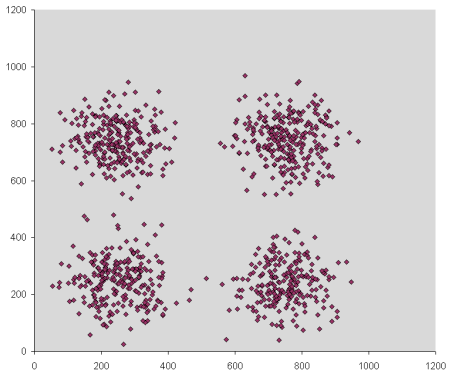

Abbildung 3.1: 2D-Plot (FEATURE 0 und FEATURE 1) der 1000 erzeugten Punkte (siehe Parameter-File)

Nach dem Schema können weitere Attribute hinzugefügt werden. Für jedes Attribut ist die Anzahl der Verteilungen und das Intervall, auf dem die Funktion definiert ist anzugeben sowie die Schrittweite (Auflösung) auf dem Intervall. Es folgen Erwartungswert μ und Standardabweichung σ für jede Verteilung.

3.2 Funktion zur Gewichtung der Orientierungsänderung bei der Bewegung von Agenten

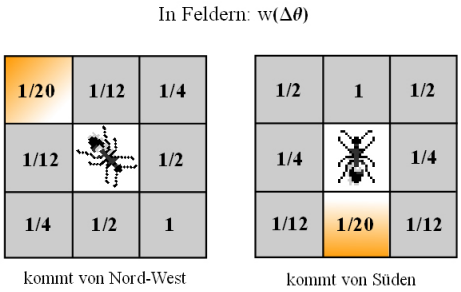

Abbildung 3.2: Gewichtungsfunktion für die Orientierunsänderung $w(\Delta\theta)$ [Marquardt 2003]

Nach [Chialvo & Millonas 1995] (CM) sollten die Ameisen bei stetiger inkrementeller Bewegung (keine Sprünge) dazu neigen, ihre Orientierung θ beim Übergang $(r, \theta) \rightarrow (r^*, \theta^*)$ von Position r zu Position r^* beizubehalten. Die Wahrscheinlichkeit für größere Winkel bei der Orientierungsänderung sollte exponentiell sinken. CM stellen ein Modell für eine diskrete 2D-Informationsinfrastruktur vor. Die Nachbarschaft einer Ameise besteht

dabei aus den acht angrenzenden Grid-Positionen. Je nach Winkel $\Delta\theta$ der Orientierungsänderung beim Übergang zu einer bestimmten Position kann dieser Position ein Gewicht w_i zugeordnet werden (siehe Abbildung 3.2): $0° := w_0 = 1$, $45° := w_1 = 1/2$, $90° := w_2 = 1/4$, $135° := w_3 = 1/12$, $180° := w_4 = 1/20$.
Da die alleinige Bereitstellung von Werten für bestimmte Winkel nicht immer ausreichend ist, wird im Folgenden ein Ansatz für die Erweiterung dieses Modells vorgestellt.

3.2.1 Theorie

Probleme ergeben sich, wenn die Informationsinfrastruktur Teilmenge des $\mathbb{Z}^3$ ist oder Schrittweiten größer 1 erlaubt werden. Dann werden Gewichte für zusätzliche Winkel benötigt. Als Lösung wurde eine konfigurierbare $e-$Funktion entwickelt. Sie ist stetig und kontinuierlich und liefert daher für jeden beliebigen Winkel einen Wert. Für die in [Chialvo & Millonas 1995] definierten Winkel ergeben sich ähnliche Werte.
Die Funktion (siehe Abbildung 3.3) ist definiert als

$$w\left(\Delta\theta\right) = e^{k\Delta\theta}$$

Dabei stellt k einen Koeffizienten für die Steilheit der Funktion dar. Je größer k, desto schneller sinken die Wahrscheinlichkeiten für größer werdende Orientierungsänderungen $\Delta\theta$. Die Winkel für die Orientierungsänderung sind als Winkel zwischen beiden Orientierungsvektoren zu berechnen.

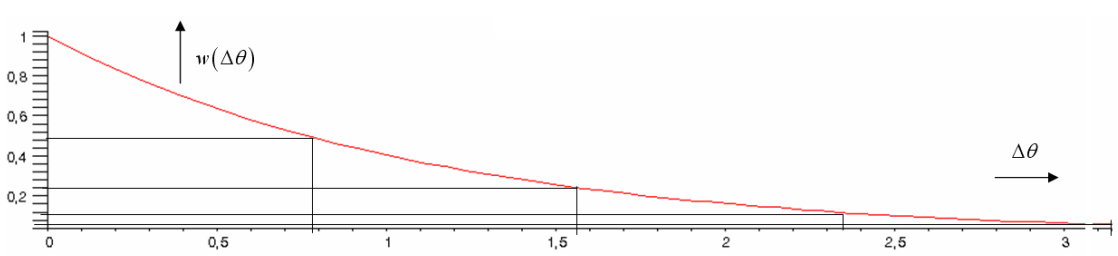

Abbildung 3.3: Gewichtungsfunktion für die Orientierunsänderung $w\left(\Delta\theta\right) = e^{k\Delta\theta}$

3.2.2 Implementation

Die Funktion zur Gewichtung der Orientierungsänderung bei der Bewegung von Agenten `double CAnt::WeightingOrientation(int pos[])` wird über die Klasse `CAnt` des Swarm Systems von Christian Nitschke bereitgestellt.

Pseudo-Code der Funktion zur Gewichtung der Orientierungsänderung bei der Bewegung von Agenten

```
1  /*! Computes the relative probability of moving to the site at pos,
       dependent on difference in orientation.
2   *
3   * @param pos Coordinates of the neighboring position to be computed.
4   * @return The weight for the given position.
5   */
6  double CAnt::WeightingOrientation(int pos[])
7    {
8    compute the scalar product between ant orientation and orientation to
         the given position
9    compute the product of the norms of the ant orientation and the
       orientation to the given position
10   compute the arccosine of the angle for the difference in orientation
11   return exp(-k * Delta_theta)
12   }
```

Für den Koeffizienten k wird hier per default ein Wert von 0.9 verwendet, um auch die Stützstellen im Modell von CM $w_0 \dots w_4$ bestmöglich zu approximieren.

3.2.3 Anwendung

Die Anwendung erfolgt durch einfache Verwendung der Funktion im jeweiligen System. Dazu sollte jeweils ein Vektor `int pos[]` mit den diskreten Koordinaten der zu untersuchenden Zelle in der Nachbarschaft übergeben werden. Die Funktion sollte Zugriff auf die intern gespeicherte aktuelle Position der Ameise haben.

3.3 Funktion zur Messung der Entropie auf der Informationsinfrastruktur

Möchte man ein Ant-based Clustering-System evaluieren oder mit einem weiteren System vergleichen, sind Maße für die Güte des Clusterings unumgänglich. Sie werden verwendet, um zu einem bestimmten Zeitpunkt t, eine Aussage über die globale Lage der Objekte zueinander zu treffen. Unter dem Zeitpunkt t versteht man beispielsweise die Anzahl von verstrichenen Iterationen oder die benötigte Rechenzeit. Hier ist darauf zu achten, daß die Größen bei unterschiedlichen Systemen vergleichbar sind.

Ramos und Merelo (RM) schlagen in [Ramos and Merelo 2002] eine einfache Entropiedefinition vor, wobei sie davon ausgehen, daß alle Objekte diskreten Objekttypen zugeordnet werden können. Die Entropie für die Anordnung der n Objekte eines Typs A auf

der Informationsinfrastruktur kann mit folgender Formel bestimmt werden:

$$E_A = \frac{\sum_{i=1}^{n} e_i}{n \cdot e_{max}}, E_A \in [0, 1]$$

Dabei gibt $e_{max} = 8$ für ein 3x3-Grid im $\mathbb{Z}^2$ die maximale Anzahl von leeren Zellen oder Zellen mit Objekten eines anderen Typs an, die sich in der Nachbarschaft eines Typ-A-Objektes befinden können. e_i ist die aktuelle Anzahl solcher Zellen für ein bestimmtes Typ-A-Objekt i. Die Gesamtentropie eines Systems ist bestimmt durch die Summe der Entropien für die beteiligten Objekttypen:

$$E_{gesamt} = \sum_{j=1}^{Anzahl\_Objekttypen} E_j$$

Befinden sich keine Objekte gleichen Typs in der Nachbarschaft, wird die maximale Entropie von $E_{gesamt} = Anzahl\_Objekttypen$ erreicht, was dem Zustand maximaler Unbestimmtheit und damit dem schlechtesten Clustering entspricht.

Im Rahmen des Projekts wurde ein neues Entropiemaß entwickelt, welches die Einschränkung durch die Zuordnung von Objekten zu diskreten Typen aufhebt. Dieses Maß wird nachfolgend erläutert.

3.3.1 Theorie

Geht man nicht von einer diskreten Zuodnung von Objekten zu Typen aus, sondern erlaubt einen kontinuierlichen Wertebereich für die Attribute der Objekte, kann man nicht mehr einfach Entropiewerte für bestimmte Typen bestimmen. Wir schlagen daher ein Entropiemaß, bestehend aus zwei Maßen vor: *Sortierung* und *Zusammenhang*.

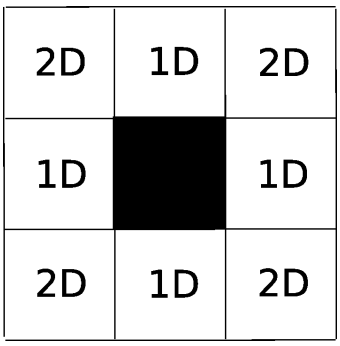

Abbildung 3.4: Nachbarschaften im $\mathbb{Z}^2$ als Basis für den Gewichtungsfaktor w_{ij}

Die Sortierung wird bzgl. des Attributraums $\mathbb{R}^l$ von Objekten angewendet. Sie funktioniert ähnlich dem Entropiemaß von RM. Bei der Sortierung werden für alle Objekte i nur die belegten Zellen j in der Nachbarschaft betrachtet. Für jedes Paar von Objekten (i, j) ist für die Sortierung die normalisierte Distanz zwischen beiden Objekten $\frac{d_{ij}}{d_{\max}}$ sowie ein Gewichtungsfaktor für die Wertigkeit der Nachbarschaft w_{ij} von Bedeutung.
Für die Distanz d_{ij} wird die euklidische Distanz im Attributraum verwendet. d_{max} steht für die maximal mögliche Distanz zweier Objekte. Sie ist definiert als:

$$d_{max} = \sqrt{\sum_{i=1}^{l} (upperbound_i - lowerbound_i)^2}$$

Nachbarschaften werden bzgl. der Informationsinfrastruktur unter Einfluß der Manhattan-Distanz unterschiedlich stark gewichtet. Je geringer die Distanz, desto stärker wird die Nachbarschaft interpretiert. Dabei wird für den $\mathbb{Z}^l$ von 1D bis lD Nachbarschaften unterschieden. Anschaulich kann man im $\mathbb{Z}^2$ von Kantennachbarschaften (1D) und Eckennachbarschaften (2D) sprechen (siehe Abbildung 3.4). Der Gewichtungsfaktor w_{ij} ist definiert als:

$$w_{ij} = \frac{1}{\sqrt{\sum_{k=0}^{l-1} |i_k - j_k|}}$$

Für die Sortierung ergibt sich folgende Formel:

$$S = 1 - \left[\frac{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \frac{d_{ij} \cdot w_{ij}}{d_{\max}}}{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} w_{ij}} \right]$$

Zu beachten ist, daß es sich hier um ein inverses Entropiemaß handelt. D.h. anders als bei RM wird hier der Grad der Bestimmtheit betrachtet. Ist der Wert für die Sortierung hoch, bedeutet dies, daß die Entropie gering ist und ähnliche Objekte nahe beeinander positioniert sind.

Da die leeren Zellen bei der Sortierung per Definition nicht mit betrachtet werden, kann sie nicht als alleiniges Maß bestehen. Sie macht zwar eine Aussage über die Ähnlichkeit benachbarter Objekte, gibt aber keine Information über die Dichte der Nachbarschaften. D.h., nur benachbarte Objekte fließen in das Maß für die Sortierung ein. Sind die Objekte ziemlich verstreut, ist das Maß nicht aussagekräftig. Um die Aussagekraft der Sortierung zu bewerten, wurde das Maß für den Zusammenhang bzgl. der Informationsinfrastruktur $\mathbb{Z}^l$ entwickelt. Je größer der Wert, desto dichter liegen die Objekte beeinander und desto höher ist die Güte der Aussage für die Sortierung.

Der Zusammenhang wird folgendermaßen bestimmt:

$$C = 1 - \left\lfloor \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{k-1} w_j \left(act_j - min_j\right)}{\sum_{i=0}^{n-1} \sum_{j=0}^{k-1} w_j \left(max_j - min_j\right)} \right\rfloor$$

Dabei steht w_i mit

$$w_j = \frac{1}{\sqrt{j}}$$

ähnlich wie bei der Sortierung für einen Gewichtungsfaktor der Stärke der Nachbarschaft von Objekten. Für jedes Objekt i der Informationsinfrastruktur wird die aktuelle Anzahl act_j an j-Nachbarn, die *kein* Objekt enthalten bestimmt. Zur Skalierung des Maßes für den Zusammenhang sind die Werte min_j und max_j von Bedeutung. Sie stehen jeweils für die minimal und maximal mögliche Anzahl an unbelegten j-Zellen in der Nachbarschaft von i.
max_j ist bestimmt durch die Anzahl der j-Zellen in der Nachbarschaft. Der Wert ist abhängig von der Dimension l der Informationsinfrastruktur. Für $\mathbb{Z}^2$ ergibt sich: $max_0 = 4$ (Anzahl Kanten), $max_1 = 4$ (Anzahl Ecken), für $\mathbb{Z}^3$: $max_0 = 6$ (Anzahl Flächen), $max_1 = 12$ (Anzahl Kanten), $max_2 = 8$ (Anzahl Ecken).
Weitaus komplexer ist die Bestimmung der Werte min_j. Die Idee ist hier, daß man von der dichtesten Packung von n Objekten auf der Informationsinfrastruktur ausgeht. Das bedeutet, n Objekte sind im $\mathbb{Z}^l$ so angeordnet, daß sich global für alle Objekte eine minimale Anzahl an unbelegten Nachbarzellen ergibt. min_j ist abhängig von der Dimension l der Informationsinfrastruktur sowie der aktuellen Anzahl an Objekten im System.

3.3.2 Implementierung

Die Implementierung der Maße Sortierung und Zusammenhang wurde für 2D und 3D Informationsinfrastrukturen durchgeführt. Dadurch können die Maße für alle im Laufe des Projekts entwickelten Systeme eingesetzt werden.
Der gut kommentierte Quellcode der Funktionen wird mitgeliefert (entropy_ functions2D, entropy_ functions3D).

Für die Berechnung des Zusammenhangs wird eine Datei `neighbors(k)d.txt` erwartet. Sie enthält die Werte min_j, die die Anzahl freier Nachbarzellen für die dichteste Packung von Objekten angibt. Die Anzahl der Objekte ist durch die Zeilennummer in der txt-Datei kodiert. Eine Zeile enthält die Werte min_j, kommagetrennt für aufsteigende j.
Die entsprechenden Dateien mit den Daten für maximal 500000 zu clusternde Objekte wurden bereits erzeugt und werden mitgeliefert.

Die Algorithmen sind als Stand-alone-Programme implementiert. Hier kann die maximale Anzahl an Objekten für die spätere Datei übergeben werden. Nach dem Start wird intern ein Supercluster als dichte Packung von Objekten generiert. Nach Hinzufügen eines jeden Objektes wird die durchschnittliche Anzahl freier j-Nachbarn bestimmt und in die Datei eingetragen. Das Supercluster wird so aufgebaut, daß jedes neu hinzugefügte Objekt auf die nächste freie Zelle mit der kleinsten Distanz zum Clustermittelpunkt gesetzt wird.
Der gut kommentierte Quellcode der Funktionen wird mitgeliefert (entropy2D, entropy3D).

3.3.3 Anwendung

Die Funktionen für Sortierung und Zusammenhang können mit geringen Anpassungen an die aktuelle Architektur direkt ins jeweilige Swarm-System integriert werden. Die benötigte Datei `neighbors(k)d.txt` kann über die bereitgestellte Funktion `vector<int*> CSettings::BuildMaxConnectivityVector(const char *maxConnFile)` in einen Vector eingelesen werden. Zur Laufzeit müssen nur Lookups auf diesen Vektor durchgeführt werden.

Die Generierung entsprechender Dateien erfolgt durch die jeweiligen Stand-alone-Programme. Hier wird die maximale Anzahl von Objekten als Parameter beim Programmstart übergeben.

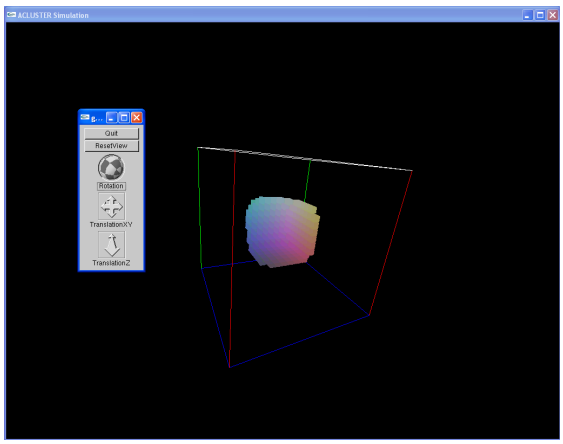

Abbildung 3.5: Anwendung zur Visualisierung des Superclusters zur dichtesten Packung von Objekten im $\mathbb{Z}^2$

Desweiteren gibt es eine `OpenGL`-Demoanwendung (entropy3D_ OpenGL), die den Aufbau des Superclusters für eine 3D-Informationsinfrastruktur visualisiert (siehe Abbildung 3.5).

4 Swarm-System in 2D nach Prinzipien von Deneubourg

4.1 Einleitung

Bei diesem Swarm-System handelt es sich um einen Algorithmus auf der Basis der Systeme von Deneubourg et al. [Deneubourg et al. 1991]. Da sich die Formeln in dieser Arbeit jedoch nur auf gleiche Objekte beziehen, wurde eine Schwelle eingeführt, die festlegt, ab wann Objekte gleich oder ungleich sind. Diese „Gleichheitsschwelle“ wird in Prozent angegeben. Die größte euklidische Distanz bezüglich der Eigenschaften der Objekte sind dabei 100 Prozent. Würde man nun 100 Prozent als Schwelle nehmen, wären alle Objekte gleich. Liegt die Schwelle bei 5 Prozent, bedeutet das, dass Objekte gleich sind, wenn die euklidische Distanz in Bezug auf ihre Eigenschaften nicht größer als 5 Prozent vom maximalen Abstand abweicht. Diese Schwelle wurde im Programm 'degree of equality' genannt und kann in der Simulation zu jeder Zeit geändert werden.

4.2 Überblick über das System

Bei einem Swarm-System sind die Agenten bzw. Ameisen für ihr tun selbst verantwortlich. Ihnen wird lediglich von einer Instanz eine Zeitscheibe zugewiesen, in der sie dann agieren und nach den vorgegebenen Regeln handeln.
Eine Ameise bewegt sich über eine 2D-Welt, die aus regelmäßigen Quadraten besteht. Jede Ameise bewegt sich jeweils nur mit der Schrittweite eins, d.h. die Ameisen gehen bei ihrem Schritt nur auf eines der 8 benachbarten Felder. Diese Welt ist in Schleifen aufgebaut. Wie bereits erwähnt bedeutet dies, dass eine Ameise, die beispielsweise nach rechts hinaus läuft, auf der linken Seite der Welt wieder herein kommt.
Auf einem Feld des Gitters kann immer nur ein Objekt liegen. Die Ameisen können auf einem Feld oder auf einem Objekt stehen. Zwei Ameisen dürfen nicht auf ein und dem selben Feld stehen. Jede Ameise kann nur ein Objekt tragen.

Der grobe Ablauf des Algorithmuses

- Jede Ameise
    1. Bewegt sich auf eine neues Feld
    2. Prüft, ob auf diesem Feld ein Objekt liegt
        - Kein Objekt
        - Ein Objekt
            * Ameise unbeladen - evtl. Aufnahme
            * Ameise beladen - evtl. Ablage

Eine Ameise macht einen Schritt und schaut, ob auf der neuen Position ein Objekt liegt. Ist das Feld leer, ist die nächste Ameise dran. Wenn nicht, prüft eine unbeladene Ameise, ob sie das Objekt aufnimmt und eine beladene, ob sie ihr mitgebrachtes Objekt dazu legt.

4.3 Der grundlegende Algorithmus

Als erstes ändert die Ameise ihre Position, d.h. sie bestimmt eine neue Richtung und bewegt sich dann um einen Schritt dorthin. Dabei ist die ursprüngliche Richtung entscheidend für die neue. Da Ameisen tendenziell eher geradeaus laufen als zum Beispiel in die entgegengesetzte Richtung. Ist das Feld schon durch eine andere Ameise besetzt, bleibt die Ameise stehen und beendet ihre Aktivitäten.
Um die neue Laufrichtung zu bestimmen besitzt jede Ameise eine Variable, die jeweils die zuletzt gegangene Richtung speichert. Da sich die Ameisen hier in einer 2D-Umgebung bewegen und die Ameisen sich auf einem regelmäßigen Gitternetz befinden, sind die Richtungen Integerwerte zwischen 0 und 7 (siehe Abb. 4.1. Dabei steht eine 0 für die Bewegung nach oben, also entlang der y-Achse. Die restlichen Felder wurden dann nach dem Uhrzeigersinn durchnummeriert.

Befindet sich die Ameise nun auf ihrer neuen Position, prüft sie, ob an dieser Stelle ein Objekt liegt. Somit gibt es zwei mögliche Abläufe:

1. Es liegt kein Objekt auf dem Feld. So hat die Ameise keine weiteren Regeln zu befolgen und die nächste Ameise ist an der Reihe.

2. Es liegt ein Objekt auf dem Feld. In diesem Fall ist das weitere Vorgehen von dem Zustand der Ameise abhängig.

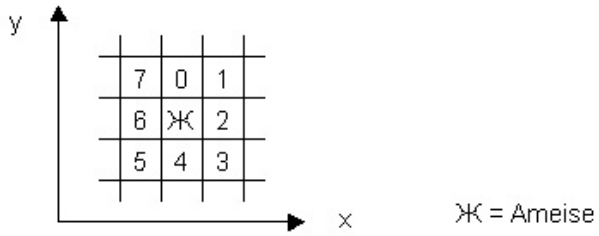

Abbildung 4.1: Die Richtungen

Die Ameise kann zwei Zustände haben. Entweder sie ist beladen, trägt also ein Objekt mit sich herum, oder sie ist unbeladen. Da eine Ameise immer nur ein Objekt auf einmal tragen kann, ist nur eine leere Ameise fähig, Objekte aufzunehmen. Ebenso kann nur eine Ameise, die beladen ist etwas ablegen.
Für den Fall, dass eine Ameise unbeladen ist, prüft die Ameise, ob sie das Objekt aufnehmen soll. Dazu wird das direkte Umfeld, d.h. die angrenzenden acht Felder, untersucht. Hierbei ist ausschlaggebend, wieviele gleiche Objekte dort herumliegen. Hierbei ist die 'Gleichheitsschwelle' entscheidend. So sind zwei Objekte sind gleich, wenn deren euklidische Distanz kleiner oder gleich dieser festgelegten Schwelle ist.
Mit der Anzahl gleicher Nachbarn wird über die folgende Formel von Deneubourg et al. [Deneubourg et al. 1991] die Wahrscheinlichkeit für das Aufnehmen bestimmt:

$$\mathit{Aufnahmewahrscheinlichkeit} = \left(\frac{k_1}{k_1 + n}\right)^2 \tag{4.1}$$

Dabei ist k_1 die Reaktionsschwelle für das Aufnehmen und n die Anzahl gleicher Objekte in direkter Nachbarschaft.

Eine Zufallszahl zwischen 0 und 1 entscheidet dann entsprechend der berechneten Wahrscheinlichkeit die Aufnahme. Ist diese kleiner oder gleich der Aufnahmewahrscheinlichkeit, wird das Objekt aufgenommen, sonst bleibt es liegen.
Ist eine Ameise beladen und steht auf einem Objekt, wird mit der euklidischen Distanz die Gleichheit des liegenden Objektes mit dem mitgebrachten Objekt bestimmt. Nur wenn die zwei Objekte gleich sind, wird die Ablagewahrscheinlichkeit wiederum nach Deneubourg et al. [Deneubourg et al. 1991] bestimmt:

$$\mathit{Ablegewahrscheinlichkeit} = \left(\frac{n}{k_2 + n}\right)^2 \tag{4.2}$$

Dabei ist k_2 die Reaktionsschwelle für das Ablegen und n die Anzahl gleicher Objekte

in direkter Nachbarschaft.
Wieder entscheidet eine Zufallszahl dann über die eigentliche Aktion. Ist die Zufallszahl kleiner oder gleich der Wahrscheinlichkeit, wird das Objekt abgelegt.
Die Ameise wählt zufällig ein Nachbarfeld und prüft ob dieses frei ist. Wenn ja wird das Objekt an diese Stelle abgelegt, sonst wird jeweils das im Uhrzeigersinn nächste Nachbarfeld überprüft. Findet sie so ein freies Feld, wird das mitgebrachte Objekt auf diesem abgelegt. Sollten jedoch alle Nachbarfelder belegt sein, bricht die Ameise ihre Suche ab und behält das Objekt. Sie trägt das Objekt also so lang mit, bis sie es auf einem freien Feld neben einem gleichen Objekt ablegen kann.

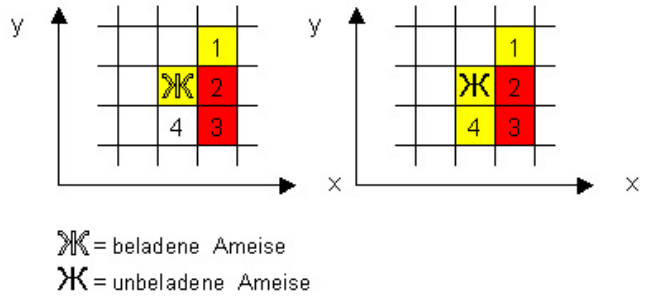

Abbildung 4.2: Ablegen eines Objektes

Die Ameise in Abb. 4.2, die mit einem gelben Objekt[1] beladen ist, hat sich entschlossen, das von ihr mitgbrachte Objekt abzulegen. Sie muss nun ein Nachbarfeld finden, dass frei ist. Die Ameise beginnt mit einem zufällig bestimmten Feld (hier: Feld 1). Ist dieses belegt, prüft sie jeweils auf dem im Uhrzeigersinn nächsten Feld. In diesem Fall würde die Ameise noch die Felder 2 und 3 überprüfen und schließlich ihr Objekt auf dem Feld 4 ablegen.

4.4 Die Erweiterungen

4.4.1 Densepacking

Wenn man eine 2D-Welt mit Objekten betrachtet, so erscheinen Objekte mit einer gemeinsamen Kante eher zusammen gehörig, als Objekte, die nur eine gemeinsame Ecke haben. Bisher legen die Ameisen ihre Objekte an einem beliebigen Nachbarfeld ab. Es

[1] Bei der Implementierung wurden Objekte mit drei verschiedenen Eigenschaften erzeugt und diese wurden entsprechend den drei Farbwerten des RGB-Farbschemas zugeordnet. Die Farben stehen also für die jeweiligen Eigenschaften des Objektes und werden hier, wie auch in der Implementierung zu optischen Darstellung und besseren Verständlichkeit genutzt. Wird also von einem blauen Objekt gesprochen, so bedeutet das nur, dass die Eigenschaften zusammen auf die Farbe projektiert blau ergeben.

wäre aber besser, sie würden die mitgebrachten Objekte bevorzugt auf den Feldern ablegen, die eine gemeinsame Kante haben. Dies wurde beim Densepacking umgesetzt.

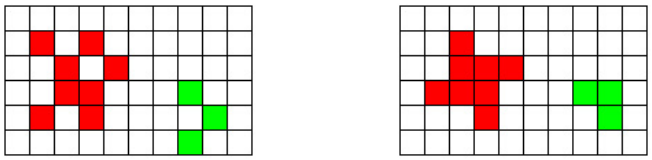

Abbildung 4.3: Lockeres versus dichter Packen

Die Ameisen fangen nicht mehr auf einem beliebigen Feld an auf bereits liegende Objekte zu testen, sondern schauen zuerst auf den Feldern mit gemeinsamen Kanten nach (in Abb. mit rot gekennzeichneten Flächen an). So werden die Objekte erst an den Seiten abgelegt und nur wenn diese schon belegt sind, auf den Ecken. Daher sollten schneller dichtgepackte Cluster entstehen.

Abbildung 4.4: Alte und neue Ablegereihenfolge

4.4.2 Tauschen

Normalerweise bewegt sich eine beladene Ameise so lang über ihre Welt, bis sie auf einem Objekt steht, dass dem getragenen gleich ist. Trägt eine Ameise nun ein blaues Objekt mit sich und steht auf einem roten, läuft die Ameise einfach weiter. Wenn nun aber dieses rote Objekt neben einem blauen läge, würde es die Ameise nicht mitbekommen. Es wäre hier aber sinnvoll, wenn die Ameise auf einem Objekt steht, überprüft, ob das mitgebrachte Objekt vielleicht besser an diese Stelle passt, als das was da schon liegt. D.h. würden die Ameisen die Objekte austauschen, könnte schneller ein gutes Clustering entstehen, da Objekte die nicht so gut ṗassen; gegen besser passende ausgetauscht werden.
Außerdem würden zwei Schritte auf einen Fallen, der dass eine unbeladene Ameise, dass unpassende Objekt mitnimmt und eine beladene Ameise ein passendes ablegt. Die Ameisen haben an drei verschiedenen Stellen, die Möglichkeit des Tauschens. Sie wurden jeweils an Stellen hinzugefügt, an denen sonst die beladene Ameise mit der Regelabarbeitung fertig gewesen wäre.
Der Ablauf beim Tauschen (Abbildung 4.5): Die Ameise steht auf einem gelben Objekt und trägt ein rotes bei sich. Die Anzahl der gleichen Nachbarn ist in diesem Fall für das

liegende Objekt zwei und für das mitgebrachte drei. Da die Anzahl für das getragene Objekt größer ist, werden die zwei Objekte getauscht und die Ameise trägt nun das gelbe Objekt weiter.

Abbildung 4.5: Das Tauschen

An die erste Stelle gelangt die beladene Ameise, wenn sie auf einem Objekt steht, welches ihrem mitgebrachten nicht gleicht. In diesem Fall wird die Anzahl gleicher Nachbarn des liegenden mit der des getragenen Objektes verglichen. Ist die Anzahl gleicher Nachbarn des getragenen Objektes größer, so tauscht die Ameise die zwei Objekte aus (das ist sicher der Fall, wo das Tauschen am häufigsten genutzt werden wird). Steht die Ameise auf einem Objekt, welches dem mitgebrachten gleich ist, so entscheidet eine Zufallszahl, ob das Objekt abgelegt wird oder nicht. Sollte die Zufallszahl bestimmen, dass nicht abgelegt wird, kann die Ameise wiederum über die Anzahl gleicher Nachbarn entscheiden, ob sie die Objekte austauscht.
Der Ablauf in diesem Fall (Abbildung 4.6): Die Ameise steht auf einem roten Objekt und trägt ein rotes bei sich. Die Ameise möchte ihr Objekt in der direkten Nachbarschaft ablegen und beginnt bei dem Feld eins zu prüfen, ob frei ist. Da alle Felder mit Objekten beleget sind, kann die Ameise ihr Objekt nicht ablegen. Bei der Überprüfung stellte die Ameise jedoch fest, dass es in der Nachbarschaft ungleiche Objekte (hier gelbe) liegen. Da diese Anzahl kleiner als 5 ist, wird die Ameise tauschen. Dazu geht sie auf das gelbe Objekt, welches sie zuletzt gesehen hat und tauscht die Objekte aus.

Abbildung 4.6: Weiterer Tauschvorgang

Als letztes kann eine Ameise Objekte tauschen, wenn sie ein Objekt ablegen möchte, aber ringsum keine freie Position ist. Sind auf den umliegenden Feldern nicht passende Objekte, rückt die Ameise auf das zuletzt als ungleich erkannte und tauscht.

4.4.3 Springen

Wenn einige Iterationen vergangen sind und das Clustering schon gut fortgeschritten ist, laufen die Ameisen sehr lang leer über die Welt bis sie zu einem Objekt kommen, welches sie aufnehmen können. Es wäre also günstig, sich die Schritte über einen leeren Raum zusparen und die Ameisen, wenn sie unbeladen sind nur von Objekt zu Objekt springen zu lassen. Bei dieser Implementierung können, wie eben beschrieben nur die leeren Ameisen springen. Es wäre aber auch denkbar, die beladenen Ameisen so zu ihrer neuen Position kommen zu lassen. Um dies zu realisieren, mussten die Objekte ihr Position in der Welt ebenso mitspeichern, wie auch einen Vermerk, ob sie grad getragen werden. Die Ameise springt also nur auf Objekte, die liegen und auf denen noch keine andere Ameise steht. Pro Iteration hüpft eine Ameise nur einmal und sie springt, bis sie auf einem Objekt ladet, welches sie aufnimmt.

4.4.4 Verkleinern der Welt

Im Laufe der Simlation können viele kleinere Cluster (Haufen gleicher Farbe bestehend aus mindestends 6 Objekten) entstehen, die selten wieder aufgelöst werden. Durch das verkleinern soll nun die Möglichkeit gegeben werden, kleine Cluster, die am Rand liegen aufzulösen. Dabei kann die Welt oben und an der rechten Seite abgeschnitten werden. Jedoch ist diese Verkleinerung nur möglich, wenn die Anzahl der Objekte auf dem abzuschneidenden Bereicht kleiner als die Anzahl leerer Felder in der verbleibenden Welt. Denn Objekte dürfen nicht aufeinander liegen. Bei der Weltverkleinerung werden alle Objekte und Ameisen, die sich auf dem verschwindenen Teil befinden, entfernt und zufällig auf die restliche Welt verteilt.

4.5 Technische Umsetzung

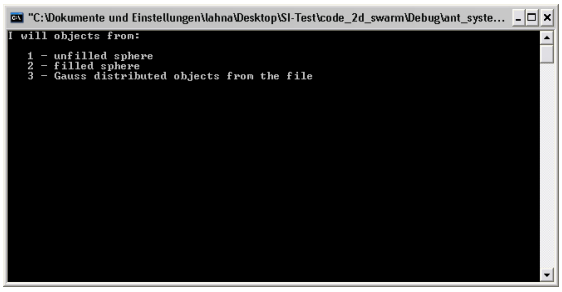

Abbildung 4.7: Startfenster

Das Programm ist in C programmiert und besteht aus nur einer Datei. Dabei wurden die einzelnen Funktionalitäten (wie beispielsweise die Objektnenerierung, die Objektplazierung, die Entscheidungsfindung für das Aufnehmen bzw. Ablegen von Objekten und die Wegwahl) in seperate Methoden implementiert. Zur Visualsierung des Sortierungsfensters und des Scatterplots wurde OpenGL benutzt. Das Steuerungsfenster wurde mit GraphApp programmiert.
Da die grafische Ausgabe das Programm verlangsamt, kann man das Pogramm in verschiedenen Geschwindigkeitsstufen laufen lassen. So besteht neben der Möglichkeit sich jede Iteration anzeigen zulassen auch die Variante sich nur jede zehnte, hundertste, tausendste bzw. zehntausendste Iteration graphisch darstellen zu lassen.
So benötigt beispielsweise ein Athlon 2000+ mit 512 MB RAM und einer ATI Radion 9000 pro bei der Anzeige jeder zehntausendsten Iteration für 10.000.000 Iterationen im Durchschnitt nur 13 Minuten. Wird jede Iteration angezeigt können nur ca. 500 Iterationen in der Sekunde berechnet werden.

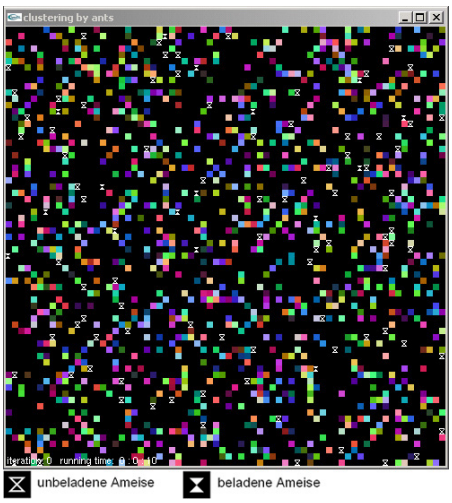

Abbildung 4.8: Sortierungsfenster Visualisierung: OpenGL. Die Welt, in der die Ameisen agieren. Die farbigen Vierecke stellen dabei die Objekte dar. Unten links werden die jeweilige Iteration und die Programmlaufzeit angezeigt.

Zur Evaluation des Programmes wurden die zwei Methoden zur Berechnung der Sortierung und der Verbundenheit (Kapitel 3) von Christain Nitschke übernommen. Die letztere Methode benötigen die Datei „neighbors2D.txt“ um die minimale Verbundenheit der Objekte zu bestimmen. Die Sortierung, die Verbundenheit und weitere Werte (z.B. die Anzahl der beladenen Ameisen und Anzahl der Aufnahmen) können in der Wiederholungsrate der Anzeige auch in eine Datei geschrieben werden. Standartmäßig ist dies jedoch ausgeschalten.
Das Programm bietet die Möglichkeit Objekte auf drei verschiedene Arten zu erzeugen. So können die Objekte Volumenpixel einer hohlen oder gefüllten Kugel sein oder die Objekte sind gaussverteilt in einem dreidmensionalen Raum. Im letzteren Fall werden

die Objekte aus der Punktedatei „points.txt“ eingelesen, die im gleichen Verzeichnis wie das Programm liegen muß.
Die erzeugten Objekte haben jeweils drei Eigenschaften, die ihrer ursprünglichen Position im Raum bei der Erzeugung entsprechen. Diese drei Werte werden dann jeweils den rgb-Farbwerten zugeordnet. Somit haben Objekte mit ähnlichen Eigenschaften auch ähnliche Farben. Das ermöglicht dem Betrachter die Güte der Sortierung leicht zu erkennen.

4.6 Aufbau des Programms

Das Programm besteht aus vier Fenstern:

- **Startfenster:** Konsolenfenster in dem man die Objekterzeugungsart festlegt (Abb. 4.7)
- **Sortierungsfenster:** Die Welt in der die Ameisen agieren (Abb. 4.8)
- **Scatterplot:** Anordnung der Objekte bezüglich ihrer Objekteigenschaften
- **Steuerungsfenster:** Zum Steuern der Simulation (Abb. 4.9)

Insbesondere kann man im Sinnes des ”Brushing and Linkingßu jeder Zeit im Laufe der Simulation, im Sortierungsfenster Objekte markieren, welche dann im Sortierungsfenster und im Scatterplot hervorgehoben werden (Abb. 4.10). Ebenso kann man Objekte im Scatterplot markieren, die ebenfalls in den beiden Fenstern hervorgehoben werden. In dem Scatterplot erscheinen die markierten Objekte weiss und im Sortierungsfenster werden unmarkierte Objekte transparent. Dies bietet die Möglichkeit die Güte der Sortierung zu prüfen, d.h. sind die Objekte, die nebeneinander sortiert wurden wirklich ähnlich oder aber zu sehen, wo Objekte mit bestimmten Eigenschaften liegen.

4.7 Ablauf

Beim starten des Programms wird zuerst über eine Konsole abgefragt, welche Art von Objekten sortiert werden soll (Abb. 4.7).
Hierbei gibt es drei Möglichkeiten:

- 1 - Eine hohle Kugel
- 2 - Eine gefüllte Kugel

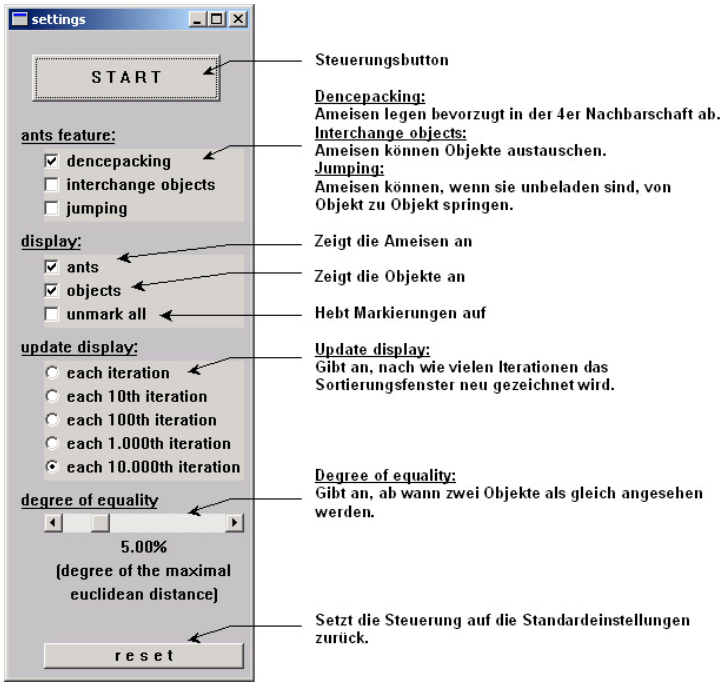

Abbildung 4.9: Steuerungsfenster Visualisierung: GraphApp

- 3 - Gaußverteilte Objekte

Hier wird die gewünschte Zahl eingegeben und mit Enter bestätigt.
Anschließend sieht man die drei Fenster des Hauptteils des Programms: das Sortierungsfenster (Abb. 4.8), den Scatterplot und das Steuerungsfenster (Abb. 4.9). Im Sortierungsfenster werden alle Objekte in ihrem Eigenschaftsraum (Abb. 4.11). So hat man eine Vorstellung in welchem Rahmen sich die Eigenschaften bewegen.

Die Objekte werden mit klick auf den „MIX“-Button im Steuerungsfenster in die 2d-Welt eingeworfen. Nun kann man die Einstellungen für die Simulation verändern. So ist es möglich zu bestimmen:

- Welche Fähigkeiten die Ameisen haben sollen.
- Ob die Ameisen und die Objekte angezeigt werden sollen.
- Wie oft das Sortierungsfenster neu gezeichnet wird.
- Ab wann zwei Objekte gleich sind in bezug auf die Objekteigenschaften (Gleichheitsschwelle, siehe 4.1).

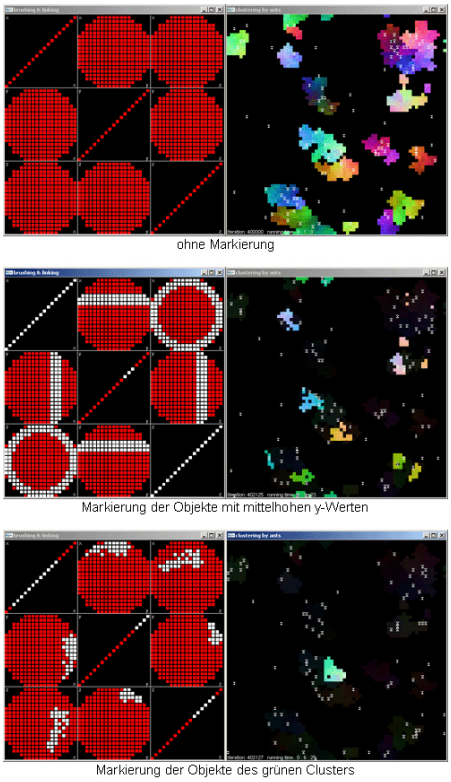

Abbildung 4.10: Brushing and Linking

Diese Einstellungen kann man im Laufe der Simulation jederzeit wieder ändern. Zum starten der Simulation klickt man nun auf den „START“-Button, dieser ändert seine Beschriftung in „STOP“. Zum Stoppen der Simulation klickt man nochmals den Button. Ebenso kann man die Simulation starten und stoppen indem man mit der rechten Maustaste in das Sortierungsfenster klickt[2].

4.7.1 Weitere Funktionen des Programms

„Brushing and Linking“
Zum Markieren von Objekten klickt man einfach die entsprechenden Objekte mit der linken Maustaste in dem Scatterplott oder dem Sortierungsfenster an. Es ist auch möglich mehrere Objekte durch gedrückthalten der Taste zu markieren. Die ausgewählten Objekte werden sofort hervorgehoben. Zum demarkieren der Objekte kann man zum einen mit der rechten Maustaste in das Fenster vom Scatterplott klicken, dann werden sofort alle Objekte wieder demarkiert. Man kann aber auch im Steuerungsfenster die Option

[2]Sollte der „START“-Button mal nicht den gewünschten Effekt haben, kann man durch 2 maliges Klicken mit der rechten Maustaste im Sortierungsfenster abhilfe schaffen.

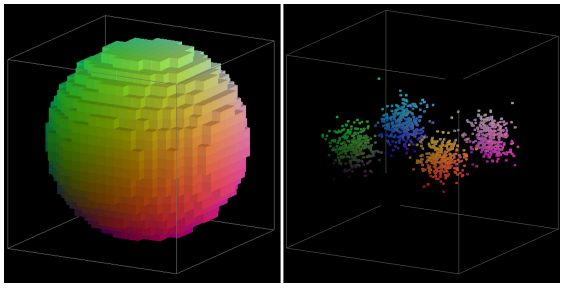

Abbildung 4.11: Brushing and Linking an Objekten einer hohlen Kugel

„unmark all“ auswählen und über den „START“-Button die Objekte wieder demarkieren.

Welt verkleinern
Das verkeinern der Welt bedeutet, dass ein Teil der Welt abgeschnitten wird und alles was sich in dem Bereich, der wegfällt, befindet, wird in die restliche Welt eingeworfen. Das Abschneiden ist jedoch nur am oberen und am rechten Rand möglich. Dazu muss mit der mittleren Maustaste in eine kurze Linie gezeichnet werden. Hierzu reichen wenige Millimeter aus. Der markierte Bereich wird grau hinterlegt (Abb. 4.12). Möchte man diesen markierten Bereich löschen und die Objekte neu einwerfen, so drückt man die Taste „E“ (engl. erase). Zum Rückgängigmachen der Auswahl drückt man die Taste „L“(engl.leave) oder man wählt im Steuerungsfenster „unmark all“ aus und klickt dann auf „START“.
Ist die verbleibende Welt zu klein für die neueinzuwerfenden Objekte so, wird nicht verkleinert.

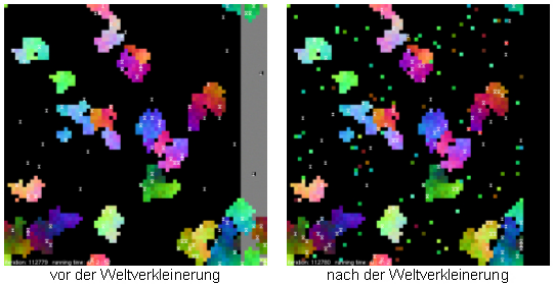

Abbildung 4.12: Brushing and Linking

4.8 Erkenntnisse und Auswertungen

Im folgenden werden die Ergebnisse der Tests bezüglich der Erweiterungen des Programmes ausgewertet.

4.8.1 Gleichheitsschwelle

Die Tests haben gezeigt, dass die Gleichheitsschwelle ein entscheidender Parameter für das Gelingen der Sortierung ist. Wählt man diese Schwelle zu klein, kommen kaum Cluster zustande, da die Ameisen sehr lang brauchen, um gleiche Objekte zu finden. Aber bereits eine Änderung um 0.25 Prozent kann entscheident sein wie Abb. 4.13 zeigt. Hierbei wurden Objekte der hohlen Kugel eingeworfen. Im linken Bild betrug die Gleichheitschwelle 7,00 Prozent und im rechten 7,25 Prozent. Links sind auch nach 10.000.000 Iterationen noch fast alle Objekte unsortiert und die Ameisen sind alle beladen. Rechts kann man bereits nach ca. 50.000 Iterationen sehen, dass alle Objekte zu Clustern hinzusortiert wurden.

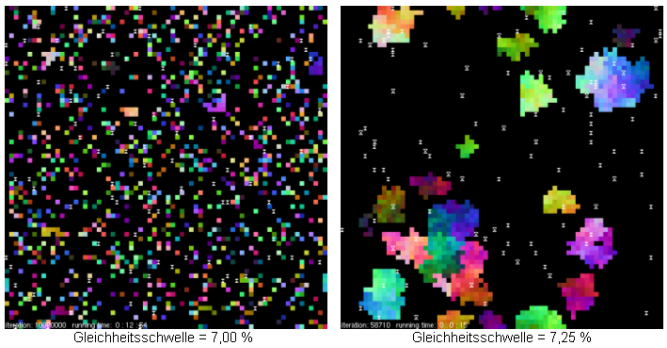

Abbildung 4.13: Vergleich Gleichheitsschwelle links: 7,00 - rechts: 7,25 Prozent

Ist diese Schwelle zu hoch, sind auch Objekte mit großen Unterschieden in den Eigenschaften als gleich angesehen und zusammensortiert. Für den Betrachter ist dies gut sichbar, da kaum eine farbliche Sortierung stattfindet.

4.8.2 Densepacking

Um schnell dichte zusammenhängene Cluster zu erhalten wurde die Fähigkeit des Densepacking eingeführt. Wie die Tests gezeigt haben, brachte diese Erweiterung durchaus eine Verbesserung. Wie jedoch die Abb. 4.14 zeigt, ist diese Verbesserung nur in der Anfangsphase gegeben. Die Güte der Clusterung wird mit dem Densepacking schnell besser, doch nach ca. 65.000 Iterationen ist der Unterschied wieder ausgeglichen. Somit bringt das Clustering eine Beschleunigung, aber die ist nicht von Dauer.

Der Grund dafür ist, dass die Ameisen ohne Densepacking zwar ihre Objekte zufällig ablegen, aber dennoch werden die Objekte abgelegt, bis alle umliegenden Felder belegt sind. Ab einer gewissen Größe des Clusters ist auch ein Objekt, das an einer Kante abgelegt wird, an der Kante eines anderen Objektes dieses Clusters.

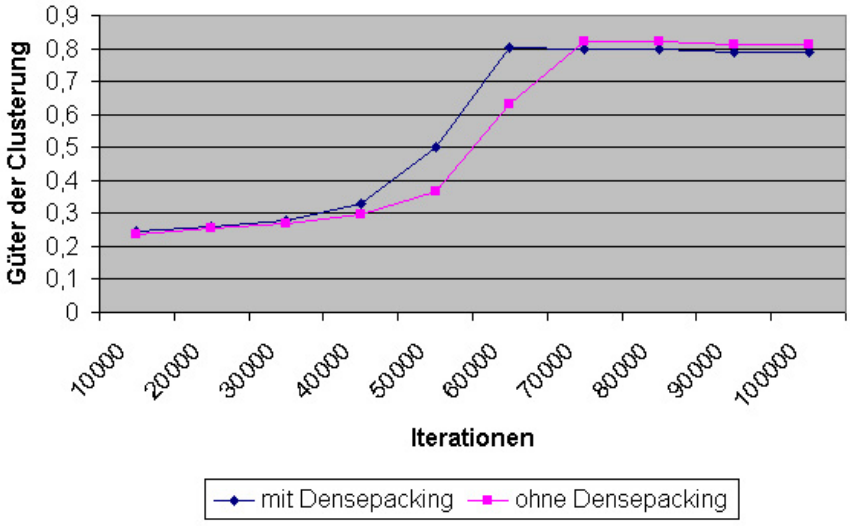

Abbildung 4.14: Vergleich von Clustering mit und ohne Densepacking

4.8.3 Tauschen

Um die Ablage von Objekten zu beschleunigen wurde das Tauschen eingeführt. Dies brachte tatsächlich eine Beschleunigung. Können Ameisen tauschen sind bereits alle Objekte einer hohlen Kugel mit einer Gleichheitsschwelle von 7,25 Prozent, wie Abb. 4.15 zeigt, schon vor der 9.000 Iteration einem Cluster inzugefügt.

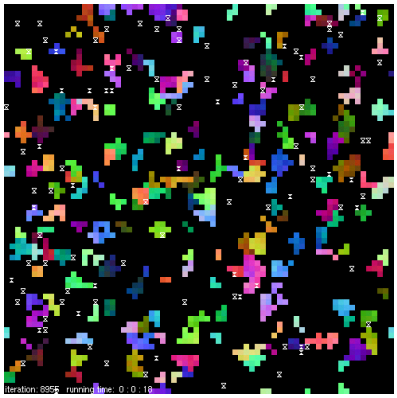

Abbildung 4.15: Clusterung mit Tauschen

Grund dafür ist der erweiterte „Sichtradius“ der Ameise. Wenn die Ameise auf einem blauen Objekt steht, aber selbst ein gelbes trägt, kann sie das gelbe zwar nicht ablegen. Aber sie testet, ob ihr mitgebrachtes nicht besser an diese Stelle passt. Liegen dann in der Nachbarschaft mehr gelbe als blaue Objekte, wird das liegende mit dem mitgebrachten

Objekt getauscht. Somit ist ein Cluster der Größe zwei entstanden. Dies führt auch dazu, dass Objekte schneller angelegt werden. Eine Ameise berechnet, wenn sie zwei gleiche Objekte gefunden hat, eine Ablagewahrscheinlichkeit und diese ist umso höher je größer der Cluster ist.
Dies führt auch dazu, dass die Gleichheitsschwelle verringert werden kann. Während man in Abb. 4.13 links mit einer Gleichheitsschwelle von 7,00 Prozent kaum Cluster gebildet haben, zeigt Abb. 4.16, dass nicht nur bei einer Gleichheitsschwelle von 7,00 Prozent Cluster entstehen sondern auch bei 6,00 Prozent. Es dauert dann nur länger. Um alle Objekte einem Cluster hinzuzufügen, benötigt man bei einer Gleichheitsschwelle von 7,25 Prozent ca. 9.000 Iterationen, bei 7,00 Prozent ca. 50.000 Iterationen und bei 6,00 Prozent ca. 250.000 Iterationen.

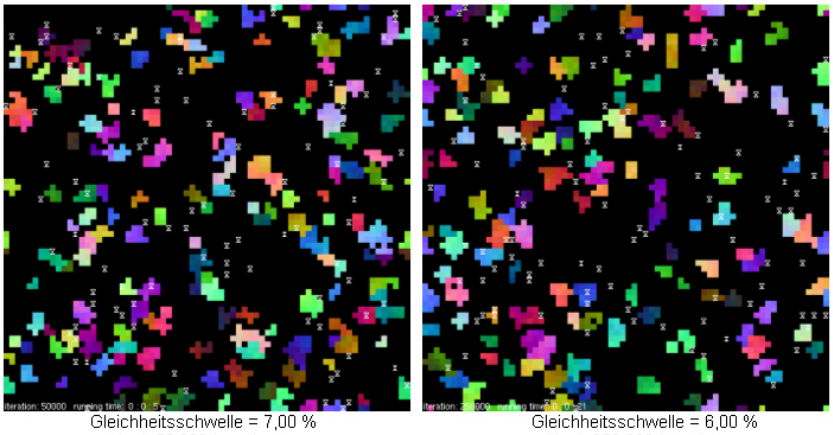

Abbildung 4.16: Clusterung mit Tauschen

Das Tauschen hat aber auch einen entscheidenden Nachteil. Eben weil schnell kleine Cluster entstehen und das Ziel der Sortierung eher eine geringe Cluster-Anzahl ist, müssen mehr Cluster wieder aufgelöst werden. Cluster die aus nicht mehr als 5 Objekten bestehen, werden stets wieder aufgelöst. Größere bleiben jedoch meist bestehen, wie man auch in Abb. 4.17 erkennen kann. Dieses Bild zeigt die gleiche Simulation wie Abb. 4.15 nur nach 10.000.000 Iterantionen.

Die Fähigkeit des Tauschens ermöglicht es die Gleichheitsschwelle zu senken und somit eine genauere Sortierung zu erreichen. Man hat aber gleichzeitig das Problem der vielen Cluster. Somit sollte abhängig von den Objekten entschieden werden, welche der Eigenschaften wichig ist, entweder weniger Cluster oder gut sortierte Cluster.

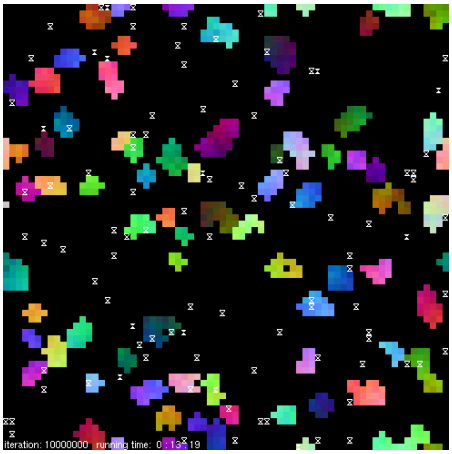

Abbildung 4.17: Clusterung mit Tauschen nach 10.000.000 Iterationen

4.8.4 Springen

Mit dem Springen sollte die Zeit eingespart werden, in der sich Ameisen unbeladen über die leere Welt bewegen. Da im normalen Algorithmus die Ameisen bei einem gewissen Grad der Sortierung meist leer sind, sollte damit später eine schneller eine bessere Sortierung erreicht werden. Ameisen, die sonst über mehrere hundert Iterationen leer bleiben, springen nun von Objekt zu Objekt und nehmen schon nach wenigeren Iterationen wieder Objekte auf.
Die Abb. 4.18 zeigt, die Clusterung einer hohlen Kugel mit einer Gleichheitsschwelle von 7,25 Prozent (aktivierte Fähigkeiten: Densepacking und Springen). Im Vergleich zur Abb. 4.13 rechts (gleiche Einstellungen nur ohne Springen), kann man erkennen, dass man deutlich mehr Iterationen benötigt bis alle Objekte einem Cluster hinzugefügt werden. In Abb. 4.13 rechts ist dieser Zstand schon nach ca. 60.000 Iterationen erreicht. In Abb. 4.18 sind auch nach 100.000 Iterationen noch Objekte nicht sortiert.

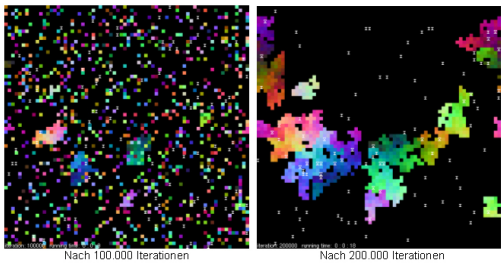

Abbildung 4.18: Clusterung mit Springen nach 100.000 bzw. 200.000 Iterationen

Clustert man eine hohle Kugel mit 7,25 Prozent Gleichheitsschwelle mit und ohne Springen (Abb. 4.19) wird deutlich, dass der Grad der Sortierung beim Springen, wie erwartet,

besser ist.

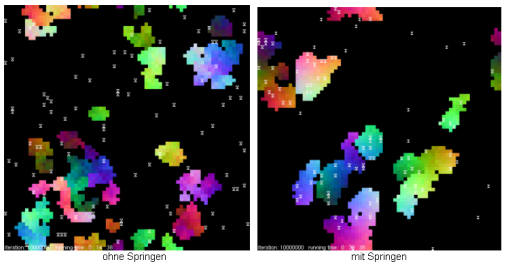

Abbildung 4.19: Clusterung mit und ohne Springen nach 10.000.000 Iterationen

Das Springen hat also das Sortieren nicht beschleunigt, aber der Grad der Sortierung ist besser.

4.8.5 Welt verkleinern

Die Erweiterung des Weltverkleinerns bringt für das System keine deutlichen Verbesserungen. Die Sortierung wird kaum beeinflußt. Mit dieser Funktion kann man lediglich eine Verdichtung der Cluster erlangen. Es bietet jedoch keine Lösung für den Fall, wenn zwei Cluster verschiedener Farben zu dicht aneinander liegen. Es sei denn, diese zwei bzw. eines der Cluster kann durch die Weltverkleinerung aufgelöst werden.

4.9 Zusammenfassung und Ausblick

Mit diesem Programm wurde eine Testumgebung für verschiedene Erweiterungen des Algorithmuses von V. Ramos und A. Abraham geschaffen, dass sich einfach bedienen lässt. Durch die freie Kombiniertbarkeit der einzelnen Fähigkeiten kann man viele Testbedingungen zusammenstellen. Dabei ist es günstig, dass man das Programm deutlich beschleunigen kann, indem man sich nicht nach jeder Iteration den aktuellen Zusand der Welt ausgeben läßt. Da man so die Wartezeit verkürzen kann. Gleichzeitig war die Wahl einer 2D-Welt für die Visualisierung sehr günstig. Da man schon auf dem ersten Blick den Zustand erkennen kann. So fallen bei Vergleichen Unterschiede sofort auf. Die Möglichkeit des „Brushing und Linking“ ist eine gute Ergänzung und hilft beim Verständnis. So kann man sich jederzeit über die Eigenschaften eines Clusters informieren bzw. die Lage vo Objekten mit bestimmten Eigenschaften überprüfen.

Im Laufe des Semsters wurde ein umfangreiches System mit vielen Erweierungen entwickelt. Besonders gegen Ende des Projektes haben sich Ideen angesammelt, die nicht mehr implementiert werden konnten. So könnte man die Objekte zeitlich versetzt einwerfen

oder die Schrittgeschwindigkeit der Ameisen variieren. Schließlich fehlt auch noch das automatische Einlesen von unbekannten Daten. Weitere Informationen in Kapitel 7.2.

5 Swarm-System in 2D nach Prinzipien von Handl, Knowles & Dorigo

5.1 Überblick

Im Folgenden wird das 2D-Swarm-System basierend auf Prinzipien von Handl, Knowles und Dorigo (HKD) [Handl et al. 2003a] beschrieben. Begonnen wird mit der Beschreibung der theoretischen Grundlagen. Dabei wird eine grobe Zusammenfassung wichtiger Forschungen auf dem Gebiet des Ant-Based-Clustering gegeben. Auf eine generellen Beschreibung der Komponenten von Ant-Based-Clustering folgt eine Analyse der Prinzipien von HKD.
Danach wird auf die Implementierung eingegangen. Speziell interessieren hier der Aufbau des Systems sowie die Abhängigkeiten im Laufzeitsystem, wo beschrieben wird, wie das System arbeitet.
Die Anwendung der Software mit Modifizierungsmöglichkeiten am Quellcode sowie Einstellungen des GUI werden im Folgenden betrachtet.
Es folgt die Evaluierung des Systems mit Simulationen zum Vergleich der Performanz der reinen Strategie von HKD mit Prinzipen basierend auf der Verwendung der entwickelten Entropiemaße. Die Ergebnisse werden anschließend diskutiert. Weiterhin werden Grenzen des Systems aufgezeigt.

5.2 Theorie

5.2.1 Überblick zur Forschung auf dem Gebiet des Ant-based-Clustering

Unter Ant-based-Clustering wird die Gruppierung von Objekten mit ähnlichen Eigenschaften durch ein dezentralisiert organisiertes Swarm-System verstanden. Dieses Gebiet ist seit langem ein Schwerpunkt in der Erforschung und Simulation von Swarm-Systemen. Die von Wissenschaftlern entwickelten Algorithmen basieren alle auf einem natur-inspirierten Modell der Anordnung von Objekten und Körpern in Ameisenkolonien, unterscheiden sich aber bzgl. ihrer Interpretation und Kombination von Prinzipien sowie in ihrem Anwendungsfeld.

Deneubourg et al. stellen in [Deneubourg et al. 1991] ein robotisches System vor. Die beteiligten Roboter bewegen sich zufällig und können Objekte aufnehmen, transportieren und ablegen. Die Wahrscheinlichkeit, eine Aktion auszuführen wird durch die Bewertung der lokalen Nachbarschaft bestimmt. Dabei ist die Wahrscheinlichkeit, ein Objekt aufzunehmen hoch, wenn sich in der Nachbarschaft des Objektes wenig Objekte oder Objekte mit anderen Eigenschaften befinden. Die Wahrscheinlichkeit, ein transportiertes Objekt abzulegen ist hoch, wenn sich viele Objekte mit ähnlichen Eigenschaften in der Nachbarschaft befinden.

Lumer und Faieta (LF) [Lumer & Faieta 1994] greifen die Forschungsergebnisse aus [Deneubourg et al. 1991] auf und entwickeln ein softwarebasiertes Swarm-System. Die Objekte werden als n-dimensionale *Datenobjekte* aufgefasst, deren Unterschied durch eine Distanzfunktion auf dem Attributraum bestimmt werden kann. Die künstlichen *Agenten* bewegen sich auf einer sog. *Informationsinfrastruktur*, einer künstlichen Welt (Teilmenge des $\mathbb{Z}^m$), die als Gitter von diskreten Positionen repräsentiert wird. Auf jeder Position evaluiert ein Agent die Güte seiner Nachbarschaft bzgl der Aufnahme und Ablage von Objekten mittels einer *Nachbarschaftsfunktion*. Um das Ergebnis des Clusterings zu verbessern, schlagen LF folgende rechenintensive Erweiterungen vor:

- Agenten mit unterschiedlichen Geschwindigkeiten,
- Agenten mit ein Kurzzeitgedächtnis,
- Agenten mit veränderbarem Verhalten.

Um der Komplexität des erweiterten Systems von LF zu entgehen, schlagen Ramos und Merelo (RM) [Ramos and Merelo 2002] einen Algorithmus vor, bei dem die Agenten sich nicht mehr zufällig bewegen. Ein Agent gibt auf jeder Position einen konstanten Betrag eines Botenstoffes, sog. *Pheromon* ab. Je mehr Agenten eine Position besuchen, desto höher wird die Pheromonkonzentration. Bei der Bewegung richten sich die Agenten nach der wahrgenommenen Pheromonkonzentration. Ist sie hoch, so ist auch die Wahrscheinlichkeit hoch, daß sich der Agent in diese Richtung bewegt. Man spricht hier von der Entstehung sog. *Pheromonpfade*, das sind Wege, auf denen sich die Agenten hauptsächlich bewegen. Sie sollen helfen, unnötige zufällige Bewegungen in uninteressante Gebiete zu vermeiden. Da das Pheromon mit der Zeit vom System abgebaut wird, entwickelt sich eine adaptive Struktur.

Handl, Knowles und Dorigo (HKD) [Handl et al. 2003a] entwickeln ein komplexes aber leistungsfähiges System. Unbeladene Agenten bewegen sich hier nicht zufällig, sondern *springen* zu einem freien Objekt. Jeder Agent besitzt ein *Kurzzeitgedächtnis* mit den letzten Ablagepositionen, die nach der Aufnahme eines neuen Objektes auf ihre Qualität untersucht werden. Mit der Zeit erweitern Agenten ihren *Wahrnehmungsradius*, so daß kleine Cluster schneller aufgelöst werden können. Für die Auswertung der Clus-

terung auf der Informationsinfrastruktur ist es wichtig, daß die Cluster räumlich voneinander separiert sind. Eine bessere Separation erreichen HKD durch die temporäre Verwendung einer *modifizierten Nachbarschaftsfunktion*. Sie erreichen weiterhin eine besondere Robustheit, da sie nicht von der Lage der Objekte im Attributraum abhängig sind. Sie führen einen Parameter α zur *dynamischen Skalierung der Distanzenfunktion* von Objekten ein. Dieser Parameter wird durch die Ablageaktivität eines jeden Agenten angepaßt.

Ein detaillierter Überblick zu den vorgestellten Verfahren im Rahmen dieser Arbeit ist unter Sektion Literaturüberblick zu finden.

5.2.2 Bestandteile von Ant-based-Clustering Systemen

Im Folgenden sollen kurz die wichtigsten Bestandteile von Ant-based-Clustering Systemen vorgestellt werden und ihr Zusammenwirken beschrieben werden.

Globale Kontrollinstanz
Zu den Prinzipien von Swarm Intelligence (SI) zählt die Dezentralisierung. Ein SI-Algorithmus basiert auf der Interaktion einfach strukturierter Agenten mit ihrer Umgebung. Es gibt keine globale Kontrollinstanz im bekannten Sinn, die ein Modell der Welt besitzt und Aktionen steuert. Vielmehr kann man hier die globale Instanz als Scheduler bezeichnen, der den Agenten Rechenzeit zur Verfügung stellt, um ihre selbstbestimmten Aktionen durchzuführen.

Informationsinfrastruktur
Unter der Informationsinfrastruktur ist die künstliche Welt zu verstehen, auf der alle Aktivitäten der Agenten stattfinden. Sie repräsentiert eine Gitterstruktur von diskreten Positionen und kann daher als Teilmenge des $\mathbb{Z}^m$ interpretiert werden. Normalerweise wird $m = 2$ verwendet (siehe 2, 4, 5), unter Sektion 6 wird ein System für $m = 3$ vorgestellt.
Meist wird sie ringförmig ohne Grenzen für jede Dimension angelegt, so daß keine Beeinflussungen entstehen. Versuchen Agenten, sich über den Rand zu bewegen, treten sie auf der gegenüberliegenden Seite wieder ins System ein.
Wichtig ist, je nach verwendetem Algorithmus auf die richtige Größe der Welt zu achten. Wählt man sie zu klein, tritt eine Behinderung durch eine zu hohe Objektdichte ein. Desweiteren können die Cluster später möglicherweise nicht analysiert werden, da sie zu dicht beeinander liegen. Wählt man die Welt zu groß, besteht die Gefahr, daß sich Agenten zu lange in uninteressanten Regionen aufhalten.

Objekte
Objekte stellen die zu clusternden Daten dar. Jedes Objekt hat n-Attributwerte. Seine

Lage bzgl. eines anderen Objektes im Attributraum kann durch eine definierte Distanzfunktion interpretiert werden. Der kontinuierliche Attributraum (Teilmenge des $\mathbb{R}^n$) ist dabei nicht mit der Informationsinfrastruktur zu verwechseln. Man möchte die Ähnlichkeiten von Objekten bzgl. des Attributraumes auf der Informationsinfrastruktur abstrahieren. Dieser Vorgang wird auch als *multidimensional scaling* bezeichnet.

Agenten
Unter Agenten versteht man künstliche Ameisen, die in ihrer Gesamtheit den Schwarm bilden. Sie repräsentieren den dezentralisiert arbeitenden impliziten Clustering-Algorithmus. Agenten können neben den Basiseigenschaften wie Aufnahme und Ablage von Objekten sowie der Wahrnehmung der Umgebung spezielle Fähigkeiten wie Adaption des Verhaltens, Wahrnehmung von Pheromon, verschiedene Bewegungsgeschwindigkeiten oder Sprungfähigkeiten besitzen.

Aufbau eines Swarm-Systems zur Clusterung
Der grundlegende Aufbau eines Swarm-Systems zur impliziten Clusterung von Datenobjekten mit Visualisierung der Informationsinfrastruktur kann in Abbildung 5.1 betrachtet werden.

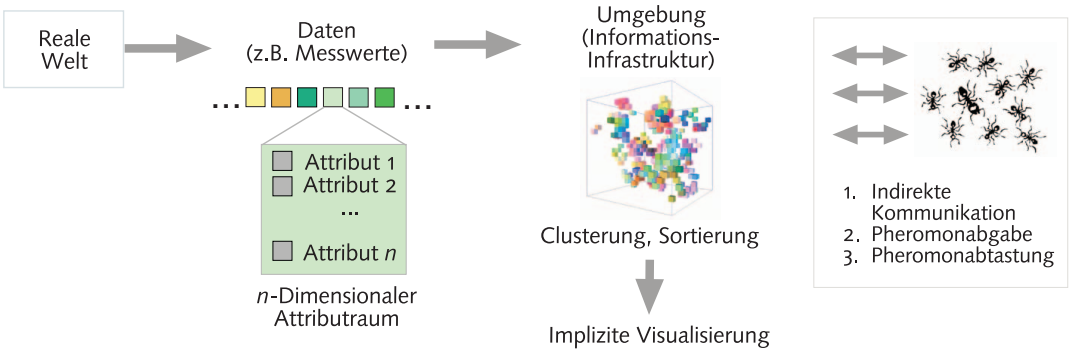

Abbildung 5.1: Aufbau eines Swarm-Systems zur Clusterung und Visualisierung.

HKD geben praktische Ratschläge zu Parametern bzgl. Informationsinfrastruktur, Anzahl von Objekten und Agenten. Darauf wird im Folgenden eingegangen.

5.2.3 Algorithmus nach Handl, Knowles & Dorigo

Das implementierte Swarm-System basiert auf Prinzipien zur Erhöhung der Robustheit und Performanz von Ant-based-Clustering nach Handl, Knowles und Dorigo (HKD) [Handl et al. 2003a]. Interessant an dem vorgestellten Verfahren sind seine komplexen adaptiven Erweiterungen, die Defiziten vorhandener Systeme entgegenwirken sollen. Neben einer robusten und performanten Clusterung wird Wert auf die räumliche Verteilung

der Cluster zur späteren Analyse des Clusterings sowie auf die Akkuratheit der Klassifikation gelegt. Hierauf wird im Folgenden näher eingegangen. Weitere Details zum Verfahren sind in Sektion Literaturüberlick zu finden.

Basic-Ant-Algorithmus nach HKD
HKD stellen das Grundgerüst eines robusten Ant-Algorithmus vor. Der Pseudocode ist in Sektion Literaturübersicht zu finden.
Es wird mit einer Initialisierugnsphase gestartet: Hier werden die Objekte zufällig auf der Informationsinfrastruktur verteilt. Jeder Agent nimmt ein Objekt auf und wird zufällig positioniert. Es folgt die eigentliche Clustering-Phase, die in einer Schleife ausgeführt wird: Jeder Agent bekommt Rechenzeit für seine eigene Ausführung zugewiesen. Er führt eine zufällige Bewegung mit einer gegebenen *Schrittweite* durch. An der Zielposition führt der Agent eine Bewertung mit Hilfe der Nachbarschaftsfunktion $f^*(i)$ durch. Fällt sie positiv aus, versucht der Agent das transportierte Objekt abzulegen. Ist die Position bereits mit einem anderen Objekt belegt, wird in der Umgebung abgelegt. Der unbeladene Agent springt jetzt zufällig an die Positionen freier Objekte und versucht über die Bewertung der Nachbarschaft das jeweilige Objekt aufzunehmen. Nun wird zufällig der nächste Agent ausgewählt, der dieselben Schritte ausführt.
Für einen Vergleich mit dem Verfahren von Ramos und Merelo [Ramos and Merelo 2002] siehe Tabelle 5.1.

	Ramos & Merelo	Handl, Knowles & Dorigo
Initialisierung	- Verteilen von Objekten - Verteilen von Ameisen	- Verteilen von Objekten - Verteilen und Beladen der Ameisen - Bewegung
Ablage	- Entscheidung nur auf leerem Feld - Erweiterung auf belegtem Feld durch Austausch	- Entscheidung auch auf belegtem Feld
Aufnahme	- Entscheidung auf belegtem Feld	- nach jeder Ablage werden freie Objekte versucht, aufzunehmen
Bewegung	- Abgabe und Aufnahme von Pheromon - Wahrscheinlichkeit für hohe Richtungsänderung gering - Schrittweite = 1	- nach Aufnahme eines Objektes, Sprungversuch zu gemerkter Position - ansonsten zufällige Bewegung (Schrittweite)

Tabelle 5.1: Gegenüberstellungen des Verfahres von Handl, Knowles und Dorigo [Handl et al. 2003a] mit dem von Ramos und Merelo [Ramos and Merelo 2002]

Nachbarschaftsfunktion und Wahrscheinlichkeiten für Aufnahme und Ablage von Objekten

Die Nachbarschaftsfunktion ist folgendermaßen definiert:

$$f^*(i) = \begin{cases} \max\left[0, \frac{1}{\sigma^2}\sum_j\left(1 - \frac{d_{ij}}{\alpha}\right)\right] & \forall j : 1 - \frac{d_{ij}}{\alpha} > 0 \\ 0 & sonst \end{cases}$$

Dabei ist $\frac{1}{\sigma^2}$ ein Term zur Normalisierung basierend auf der Größe der Nachbarschaft, d_{ij} die Distanzfunktion zweier Objekte i und j bzgl. des Attributraums und α ein adaptiver Skalierungsfaktor für die Gewichtung der Distanzfunktion. Für Anmerkungen zur Funktion siehe Abbildung 5.2.

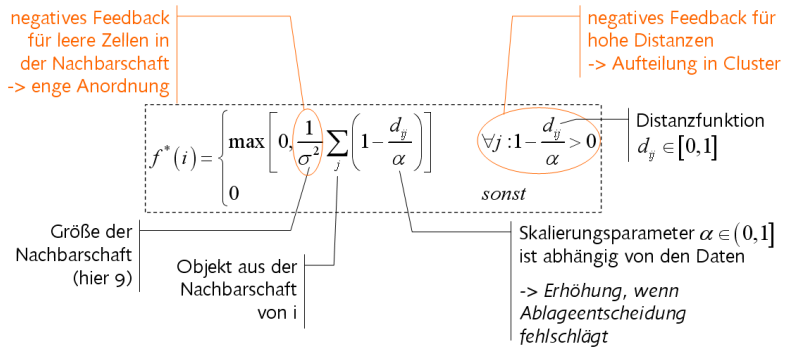

Abbildung 5.2: Die Nachbarschaftsfunktion $f^*(i)$.

Der Wert für $f^*(i)$ steigt, wenn

- die Anzahl der Objekte in der Nachbarschaft steigt, was zu einer höheren Intra-Cluster-Dichte führt (Zusammenhang mit Anzahl der Objekte) oder
- der Term $\frac{d_{ij}}{\alpha}$ für die Ähnlichkeit von Objekten klein ist (Zusammenhang mit Ähnlichkeit der Objekte). Dies ist der Fall, wenn entweder d_{ij} klein ist oder d_{ij} groß ist und der adaptive Skalierungsfaktor α bei nicht erfolgreicher Aktivität mit der Zeit ansteigt.

Die Nachbarschaftsfunktion wird für die Berechnung der Wahrscheinlichkeiten $p^*_{pick}(i)$ und $p^*_{drop}(i)$ für die Aufnahme und Ablage von Objekten verwendet.

$$p^*_{pick}(i) = \begin{cases} 1.0 & f^*(i) \leqslant 1.0 \\ \frac{1}{f^*(i)^2} & sonst \end{cases}$$

$$p^*_{drop}(i) = \begin{cases} 1.0 & f^*(i) \geqslant 1.0 \\ f^*(i)^4 & sonst \end{cases}$$

Wichtig ist, daß die Funktionen nicht mit dem reinen Basic-Ant-Algorithmus eingesetzt werden können. Dieser muß erst um die nachfolgend vorgestellten Features erweitert werden. $p^*_{pick}(i)$ und $p^*_{drop}(i)$ müssen daher in Zusammenhang mit einer Erhöhung des Wahrnehmungsradius der Agenten und den damit verbundenen möglichen höheren Werten für $f^*(i)$ gesehen werden (siehe Abbildung 5.3).
Ist der Term $\frac{d_{ij}}{\alpha}$ größer 1, d.h. die Objekte sind zu unterschiedlich, wird $f^*(i)$ gleich 0, was zu einer deterministischen Aufnahme von Objekten mit $p^*_{pick}(i) = 1$ führt. Eine deterministische Aufnahme von Objekten tritt weiterhin zu Beginn des Clustering-Prozeses auf, da die Werte für $f^*(i)$ immer kleiner oder gleich 1 sind. Steigt $f^*(i)$ über 1, wird durch $p^*_{pick}(i) = \frac{1}{f^*(i)^2}$ die Aufnahme falsch angeordneter Objekte begünstigt.
Zu Beginn favorisiert $p^*_{drop}(i)$ dichte Regionen sowie Regionen mit ähnlichen Objekten, da $f^*(i)$ kleiner gleich 1 ist. Je höher der Wert für $f^*(i)$ steigt, desto höher ist auch die Wahrscheinlichkeit für die Ablage von Objekten. Ist der Wert größer 1, ist die Ablageoperation mit $p^*_{drop}(i) = 1$ rein deterministisch.

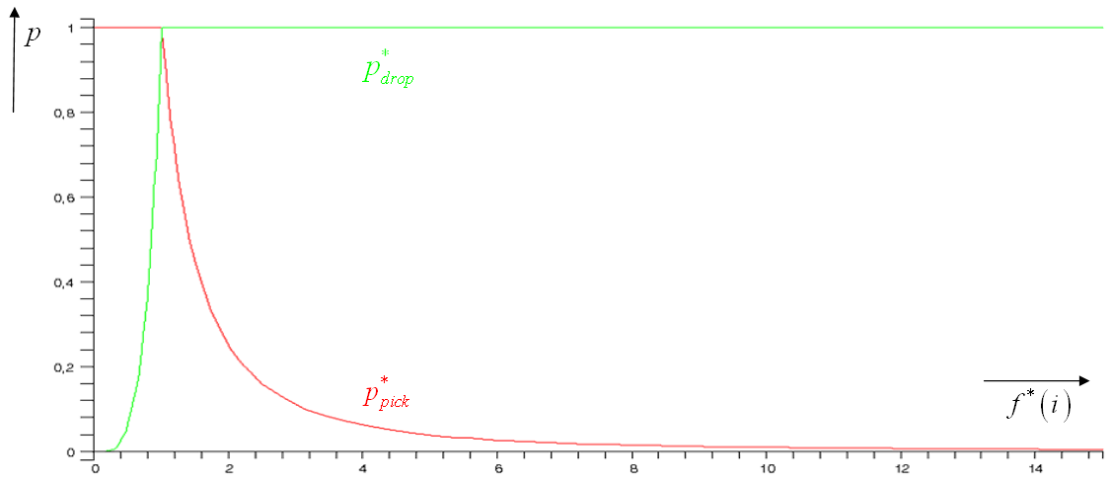

Abbildung 5.3: Die Wahrscheinlichkeitsfunktionen $p^*_{pick}(i)$ und $p^*_{drop}(i)$ in Abhängigkeit vom Wert der Nachbarschaftsfunktion $f^*(i)$.

Erweiterungen des Basic-Ant-Algorithmus

- **Kurzzeitgedächtnis**
  Jeder Agent besitzt ein Gedächtnis, um die letzten n Ablagepositionen zu speichern. Bei der Aufnahme eines neuen Objektes werden alle gespeicherten Positionen besucht und mittels der Nachbarschaftsfunktion $f^*(i)$ evaluiert. Für die Position mit dem höchsten Wert wird eine Ablageentscheidung mit $p^*_{drop}(i)$ durchgeführt. Fällt sie positiv aus, wird abgelegt, sonst wird das Gedächtnis für das aktuelle Objekt deaktiviert. Der Agent bewegt sich ab jetzt zufällig mit der definierten Schrittweite (siehe Basic-Ant-Algorithmus, Sektion 2.3.2).
  Dieses Vorgehen stellt eine Verbesserung des von Lumer und Faieta (LF)

[Lumer & Faieta 1994] vorgestellten Kurzzeitgedächtnisses dar, bei dem nur die Eigenschaften der abgelegten Objekte verglichen werden. LF berücksichtigen nicht, daß die Objekte durch andere Agenten wieder von den Ablagepositionen entfernt worden sein könnten.

- **Erhöhung des Wahrnehmungsradius**
  Eine Erhöhung des Wahrnehmungsradius von Agenten führt zu einer größeren Nachbarschaft. Während dies in der Endphase des Clustering-Prozesses gewünscht ist, um kleine Cluster schneller aufzulösen, sorgt es zu Beginn für Probleme, da die schnelle Formierung von Clustern begünstigt wird. Desweiteren steigt der Bedarf an Rechenzeit, da quadratisch mehr Positionen in der Nachbarschaft ausgewertet werden müssen.
  HKD schlagen eine inkrementelle Erhöhung des Radius von 1 auf 5 über die gesamte Laufzeit des Clustering-Prozesses vor (siehe Abbildung 5.4). $\sigma^2 = 9$ bleibt dabei konstant. Dies führt zu einer Erhöhung der möglichen Werte für Nachbarschaftsfunktion $f^*(i)$, da mehr Objekte wahrgenommen werden können. Zu Beginn ist $r = 1$, was zu einer rein deterministischen Aufnahmeoperation führt. Die Ablage wird hier bevorzugt in dichten und ähnlichen Nachbarschaften durchgeführt.

- **Dynamische α-Anpassung**
  Der Parameter α jedes Agenten dient zur dynamischen Skalierung des Distanzmaßes für zwei Objekte im Term d_{ij}/α der Nachbarschaftsfunktion. Für jeden Agenten wird $\alpha \in (0, 1]$ initialisiert. Nach der folgenden Vorschrift wird α angepaßt:

$$\alpha \leftarrow \begin{cases} \alpha + 0.01 & r_{fail} > 0.99 \\ \alpha - 0.01 & r_{fail} \leqslant 0.99 \end{cases}, r_{fail} = \frac{N_{fail}}{N_{active}}, N_{active} = 100$$

  Der Parameter N_{active} bezeichnet die Anzahl von Bewegungen, N_{fail} die Anzahl erfolgloser Ablageversuche.
  Führen die Agenten längere Zeit keine Ablageoperation durch, ist dies möglicherweise auf eine große Ausdehnung der Cluster bzgl. des Attributraums zurückzuführen. Durch eine dynamische Skalierung des Parameters α wird über die Zeit hin eine Skalierung der Distanz durchgeführt. Dies entspricht anschaulich gesehen einem Zusammenziehen der Cluster, was weitere Ablagevorgänge begünstigen sollte. Dadurch führt α zu einer Robustheit gegenüber den Form der natürlichen Cluster im Attributraum.

- **Räumliche Verteilung der Cluster**
  Eine Zielstellung ist die räumliche Verteilung der Cluster auf der Informationsinfrastruktur. Problembehaftet ist daher die anfängliche Clusterbildung in allen dichten Regionen von ähnlichen Objekten (sog. Minicluster). Hier ist die spätere Zusammenführung schwierig und zeitaufwendig. Zur Lösung kann bei etwa der halben Laufzeit des Algorithmus für ein kurzes Zeitintervall eine modifizierte Nachbarschaftsfunktion eingesetzt werden. σ^2 wird dabei ersetzt durch N_{occ}, der Anzahl

der belegten Positionen in der wahrgenommenen Nachbarschaft.
Nach einer anfänglichen Clusterbildung führt dieses Vorgehen zu einer alleinigen Betrachtung der Ähnlichkeit von Objekten in der Nachbarschaft und einer damit verbundenen Vernachlässigung der Dichte. Es hat zur Folge, daß sich Cluster über die gesamte Informationsinfrastruktur hin ausdehnen. Hier sollte auf eine ausreichende Größe des Grids geachtet werden, um die Expansion nicht zu behindern. Wird nach dem Zeitintervall wieder die normale Nachbarschaftsfunktion verwendet, ziehen sich die Cluster zusammen. Dabei werden sich mit hoher Wahrscheinlichkeit die Clustermittelpunkte voneinander entfernen. Dieses Vorgehen wird durch Abbildung 5.5 veranschaulicht.

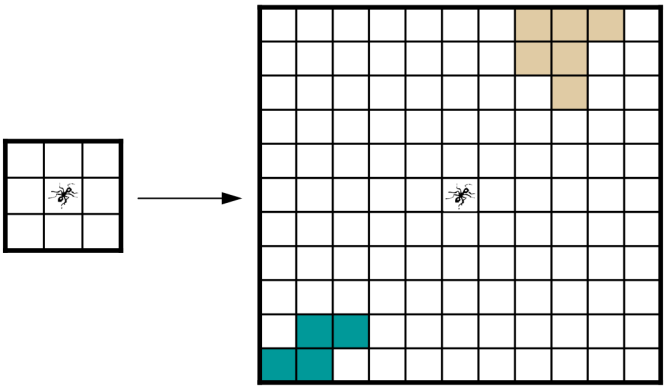

Abbildung 5.4: Durch eine Erhöhung des Wahrnehmungsradius der Agenten wird eine größere Nachbarschaft ausgewertet (hier $r = 1$ und $r = 5$).

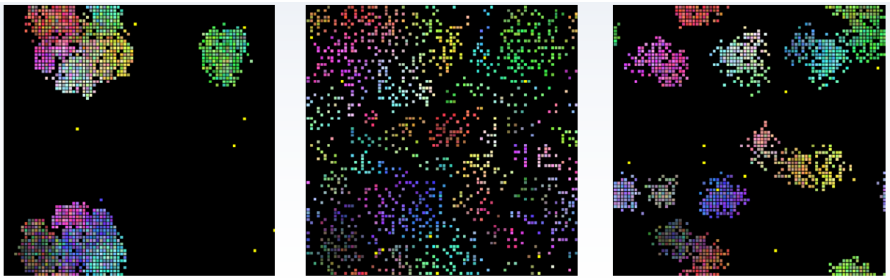

Abbildung 5.5: Räumliche Verteilung der Cluster durch den Einsatz einer modifizierten Nachbarschaftsfunktion (links: vorher, mittig: mod. Nachbarschaftsfunktion, rechts: nachher).

Parametereinstellungen
Experimentell bestimmte Parametereinstellungen sind in Tabelle 5.2 zusammengestellt.

	Parameter	Wert
Daten-unabhängig	N_{agents}	10
	Größe des Kurzzeitgedächtnisses	10
	t_{start} für mod. $f^*(i)$	$0.45 \cdot N_{iterations}$
	t_{end} für mod. $f^*(i)$	$0.55 \cdot N_{iterations}$
Daten-abhängig	$r_{occupied} = \frac{N_{items}}{N_{cells}}$	etwa $\frac{1}{10}$, konstant halten → quadratisches Grid mit Anzahl Zellen von $\sqrt{10 \cdot N_{items}} \times \sqrt{10 \cdot N_{items}}$
	Schrittweite bei Bewegung eines Agenten	$\sqrt{20 \cdot N_{items}}$ → jede Zelle sollte innerhalb einer Bewegung zu erreichen sein
	$N_{iterations}$	$2000 \cdot N_{items}$, min: 1000

Tabelle 5.2: Parametereinstellungen für das Verfahren von Handl, Knowles und Dorigo [Handl et al. 2003a]

5.3 Implementierung

Das System ist objektorientiert in C++ implementiert. Der Visual-Studio v7.0 Quellcode liegt bei. Für die Visualisierung wird `OpenGL` mit `GLUT` verwendet. Die Interaktion mit der Software geschieht über ein `GLUI`-Menü [GLUI 2004]. Im Folgenden werden die beteiligten Klassen mit ihren Aufgaben beschrieben.

5.3.1 Klassenstruktur

`CItem`
Durch diese Klasse wird ein n-dimensionales Datenobjekt repräsentiert. Sie stellt die Distanzfunktion bzgl. des Attributraums zur Verfügung.

`CAnt`
Diese Klasse repräsentiert einen Agenten mit seinen Eigenschaften und Funktionen. Ein Agent besitzt ein Kurzzeitgedächtnis, einen α-Parameter mit den zugehörigen Countern für die Anzahl der Bewegungen sowie der erfolglosen Ablageversuche, seine aktuelle Position und Orientierung sowie jeweils einen Pointer auf die Umgebung und das transportierte Objekt. `CAnt` stellt weiterhin folgende Funktionalitäten zur Verfügung: Durchführung einer nicht-deterministische Bewegung mit der definierten Schrittweite basierend auf einer Gewichtung der Orientierungsänderung und der Pheromonkonzen-

tration[1], Treffen von Aufnahme- und Ablageentscheidungen basierend auf den entsprechenden Wahrscheinlichkeitsfunktionen unter Zuhilfenahme der Nachbarschaftsfunktion, Explorierung des Kurzzeitgedächtnisses sowie eine Tauschfunktionalität für Objekte. Die genannten Funktionalitäten werden durch die Hauptfunktion `CAnt::Execute()` bzw. `CAnt::ExecuteSwapPhase()` in Anspruch genommen.

`CEnvironment`
Diese Klasse stellt die Umgebung oder Informationsinfrastruktur dar. Sie beinhaltet neben einer Matrix von `CGridpoint`-Pointern auch ein Array mit Pointern auf die `CAnt`-Objekte. Sie bietet viele Managementfunktionen an, wie z.B.: Hinzufügen, Verteilen und Entfernen von Agenten und Objekten, Helferfunktionalitäten für die Aufnahme und Ablage von Objekten, für Informationen über Koordinaten, Objekt- und Agentenbelegungen von Positionen in der Nachbarschaft einer Position sowie für die Bewegung von Agenten sowie Unterstützung von Pheromon. Desweiteren werden die im Projekt entwickelten Maße für Sortierung und Zusammenhang (siehe Kapitel 3) bereitgestellt. Die wichtigste Funktion ist der Scheduler `CEnvironment::Step()`, der Rechenzeit auf die Agenten verteilt.

`CGridpoint`
Diese Klasse stellt einen Position bzw. Zelle der diskreten Informationsinfrastruktur (`CEnvironment`) dar. Sie besitzt jeweils einen Pointer auf einen Agenten und ein Objekt. Es wird die Aufnahme von Pheromon unterstützt.

`CRandom`
Diese Klasse stellt die Funktionalität eines Zufallszahlengenerators zur Verfügung. Erzeugt werden können gleich- und normalverteilte Ganz- und Gleitkommazahlen auf einem gegebenen Intervall. Weiterhin ist hier die Funktionalität zur flexiblen Generierung normalverteilter n-dimensionaler Daten enthalten (siehe Kapitel 3). Dies wird beispielsweise zur Erzeugung von Clustern in Testdatensätzen verwendet.

`COpenGL`
Diese Klasse kapselt die gesamte Visualisierung- und Interaktionsfunktionalität. Die Klasse stellt über ihren `GLUT`-Display-Callback die Haupt Steuerungsroutine für das gesamte Framework dar. Von hier aus wird `CEnvironment::Step()` aufgerufen, die eine Iteration des Systems durchführt. Es ist nicht notwendig, daß die Visualisierung nach jeder Iteration geupdated wird.

`CSettings`
Die Klasse enthält alle Systemparameter. Gewünschte Modifikationen zur Compilezeit brauchen somit nur an einer globalen Stelle durchgeführt werden. Die statischen Klassenmember sind überall im Programm verfügbar.

[1] Pheromon wird hier aktuell nicht verwendet, aber unterstützt

5.3.2 Laufzeitsystem

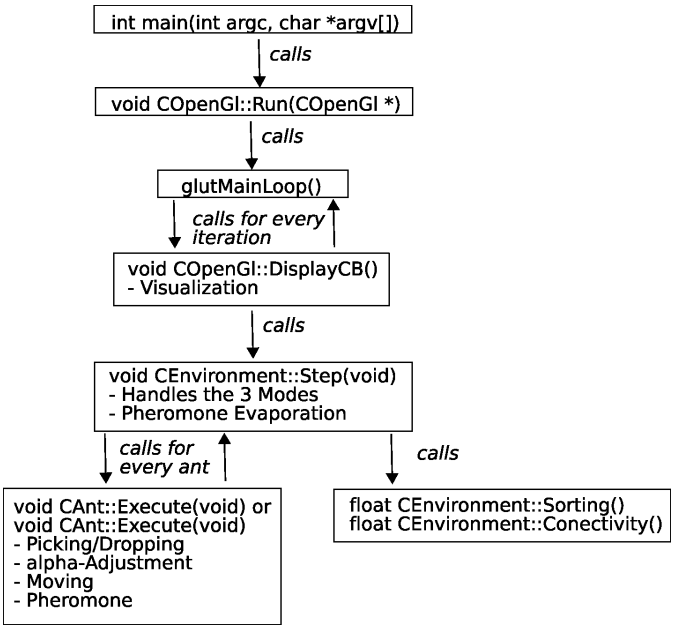

Abbildung 5.6: Einfacher Callgraph für die Ausführung des Swarm-Systems.

Die Abhängigkeiten und Aufführungsabläufe sind in Abbildung 5.6 aufgezeigt. Die Ablaufsteuerung liegt bei `::glutMainLoop()`. Hier wird der Glut-Display-Callback `COpenGL::DisplayCB()` aufgerufen, der immer eine Iteration durch Aufruf von `CEnvironment::Step()` veranlaßt. Die Visualisierung wird aus Perfomanzgründen nur alle n Iterationen geupdatet, wobei n über das GUI einstellbar ist.
`CEnvironment::Step()` teilt jedem Agenten Rechenzeit zu, um seine eigenen Aktionen auszuführen. Desweiteren implementiert diese Funktion folgende drei verschiedenen Ablaufmodi.

Modus: Handl
Dieser Modus realisiert das Verfahren nach Handl, Knowles und Dorigo. Es müssen die Anzahl von Interationen sowie t_{start} und t_{end} für die den Einsatz der modifizierten Nachbarschaftsfunktion angegeben werden. Die Anzahl der Iterationen sollte wie in Tabelle 5.2 aufgezeigt in Abhängigkeit der Anzahl von Objekten bestimmt werden.

Modus: Automatic
Der automatische Modus stellt eine Neuerung dar. Er basiert auf der Verwendung der Maße für Sortierung und Zusammenhang. Es sollen verschiedene Phasen nicht wie im Handl-Modus durch feste Zeitschranken kontrolliert werden, sondern adaptiv durch Messung von Zuständen. Dadurch kann einiges an Rechenzeit eingespart werden. Siehe dazu die Ergebnisse der Evaluation unter Sektion Evaluation.

In den einzelnen Phasen des automatischen Modus werden die Parameter Wahrnehmungsradius der Agenten, Einsatz modifizierter Nachbarschaft sowie Konvergenzverhältnis variiert. Das Konvergenzverhältnis spielt eine Rolle, wenn ein aktueller Messwert mit einem früheren Messwert ins Verhältnis gesetzt wird, um zu bestimmen ob Konvergenz vorliegt. Sind beide Werte gleich, so ist das Verhältnis 1. Werden Messwerte einander so ähnlich, daß sie das angegebene Konvergenzverhältnis überschreiten, wird Konvergenz angenommen. Für die Anzahl der Iterationen zwischen zwei Messwerten sind dabei folgende zwei Parameter von Bedeutung: `CSettings::m_numIterationsToCheck` gibt an, nach wievielen Iterationen eine Messung vorgenommen wird, `CSettings::m_numConvergence` die Anzahl an Messerten, zwischen früherem und aktuellem Wert für den Konvergenztest.
In Tabelle 5.3 wird eine Übersicht zu den Phasen des automatischen Modus gegeben.

	Phase 1	Phase 2	Phase 3	Phase 4
Wahrnehmungsradius	5	5	5	3
modifizierte Nachbarschaftsfunktion	✗	✓	✗	✗
Konvergenzverhältnis	0.998	0.98	0.98	>1.0
Entwicklung Sortierung	↑	↑	↑	↑
Entwicklung Zusammenhang	↑	↓	↑	↑
Anmerkung	Vor-sortierung	räumliche Trennung	Zusammen-ziehen	dichte Packung

Tabelle 5.3: Parametereinstellungen und Zustände für die Ausführung des Swarm-Systems.

Modus: Manual
Im manuellen Modus kann der Nutzer selber explorieren. Er hat dabei die Möglichkeit die modifizierte Nachbarschaftsfunktion sowie die Tauschphase ein- und auszuschalten. Während der Tauschphase benutzen die Agenten statt `CAnt::Execute()` die Ausführungsfunktion `CAnt::ExecuteSwapPhase()`. Hier haben sie zusätzlich die Möglichkeit, ein transportiertes Objekt auf einer belegte Position abzulegen. Dabei müssen sie das positionierte Objekt aufnehmen und weiter transportieren. Ein solcher Tausch findet statt, wenn das transportierte Objekt besser an die Position paßt als das dort befindliche Objekt.

Bei allen drei Modi hat der Nutzer die Möglichkeit, den Algorithmus anzuhalten und dabei Agenten hinzuzufügen oder zu entfernen sowie Objekte gleich- oder normalverteilt zu erzeugen oder zu entfernen. Desweiteren können andere Parameter verändert werden.

5.4 Anwendung

Nach dem Start der Software wird im Programmverzeichnis nach einer Parameterdatei `params.txt` für die Erzeugung von normalverteilten Clustern gesucht. Zum Aufbau der Datei siehe Kapitel 3.
Wird die Datei gefunden, wird die darin spezifizierte Anzahl von Objekten erzeugt und zufällig auf der Informationsinfrastruktur verteilt. Das Verhalten kann durch Setzen von `CSettings::_gaussian=0` in `CSettings.cpp`, Zeile 173 ausgeschaltet werden.

Die Anwendung besteht aus einem Visualisierungs-, einem Steuer- und einem Konsolenfenster, was sich rein auf die Ausgabe von Informationen beschränkt. Per default wird die Informationsinfrastruktur mit einer Größe von 100x100 initialisiert. Möchte man dies ändern, müssen die entsprechenden Werte in `CSettings.cpp`, Zeile 117,118 sowie in `CEnvironment.h`, Zeile 29 angepaßt werden. Alle Werte bzgl. des Handl-Algorithmus werden entsprechend automatisch gesetzt, können aber von Hand über das GUI modifiziert werden.

Per default werden Objekte aus einem 3-dimensionalen Attributraum mit einem Definitionsbereich von 1000 für jedes Attribut verwendet. Soll dies geändert werden, müssen die entsprechenden Werte in `CSettings.cpp`, Zeile 40,144-147 geändert werden. Bei den Zeilen 145-147, der Spezifikation der jeweiligen Definitionsbereiche, müssen Entfernungen bzw. Hinzufügungen vorgenommen werden.

Zu Beginn befindet sich das System im Pause-Modus, der über die entsprechende Checkbox verlassen werden kann. Vorher können über das Steuerfenster Parameter verändert werden. Nicht zu vergessen ist das Hinzufügen von Agenten durch Angabe des entsprechenden Parameterwerts. Eine Beschreibung aller Funktionen des GUI wird durch die Abbildungen 5.7 und 5.8 gegeben.

5.5 Evaluation des Systems

Um das System zu evaluieren wurden einige Simulationen durchgeführt. Diese sind nachfolgend beschrieben und erläutert. Anschließend wird das Gesamtergebnis diskutiert.

5.5.1 Simulationen

Das Ziel der Simulationen war es, den Algorithmus von Handl, Dorigo und Knowles mit dem Automatischen Modus unter Zuhilfenahme der Entropiemaße Sortierung und

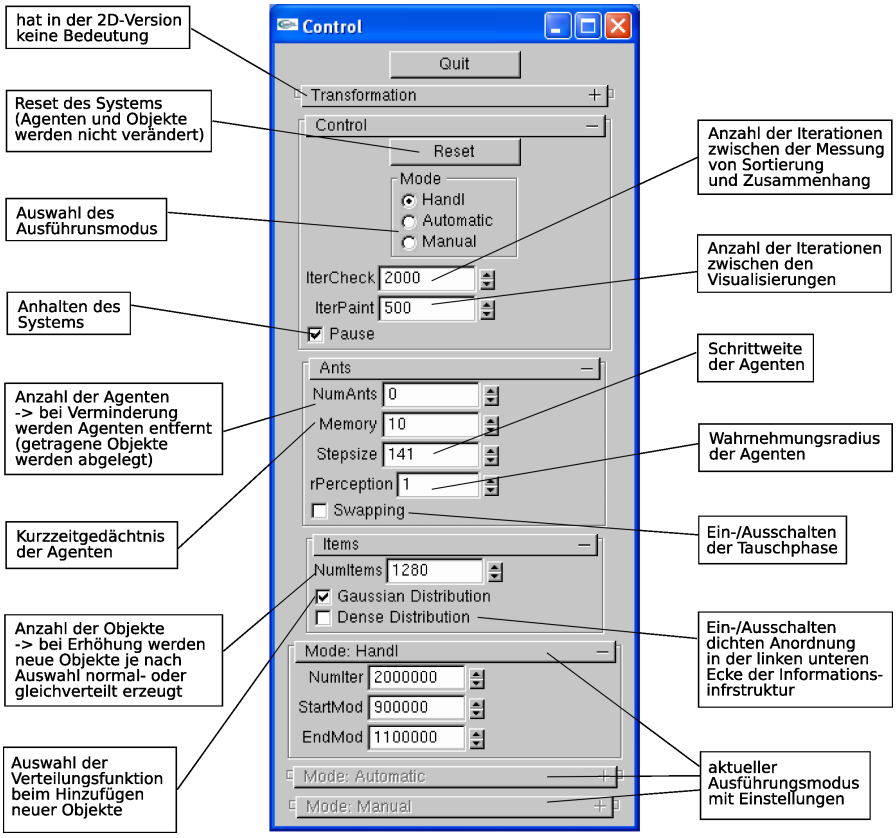

Abbildung 5.7: Erläuterungen zum GUI des Swarm-Systems.

Zusammenhang zu vergleichen. Es wurden dazu acht Simulationen mit unterschiedlichen Parametern durchgeführt. Eine Übersicht ist in Tabelle 5.4 zu finden.

Die Abbildungen 5.9-5.16 zeigen die Ergebnisse der Simulationen. Dabei wurden jeweils sechs Snapshots in den verschiedenen Phasen gemacht. Desweiteren gibt es für jeden Simulationslauf ein Diagramm mit den Kurven für die Entwicklung von Sortierung und Zusammenhang. Dabei kann die y-Achse als Maß für die Qualität der Clusterung gesehen werden. Die x-Achse bezeichnet die Anzahl von Meßwerten. Da für jede Simulation alle 1000 Iterationen eine Messung gemacht wurde, bildet sie ein relatives Maß für den Zeitverbrauch. Durch den Vergleich der jeweiligen Kurvenverläufe läßt sich die Effizienz der Simulation bestimmen.

Im Handl-Modus kann man in allen Diagrammen gut die verschiedenen, vom jeweiligen Wahrnehmungsradius abhängigen Phasen erkennen: In Phase 1 erhöht sich der Zusammenhang, wobei Sortierung ziemlich niedrig bleibt. Es entstehen unsortierte Cluster. Phase 2 läßt die Sortierung steigen, wobei der Zusammenhang erhalten bleibt. Für Clusterdaten (normalverteilt) steigt die Sortierung schneller als für gleichverteilte Daten, was

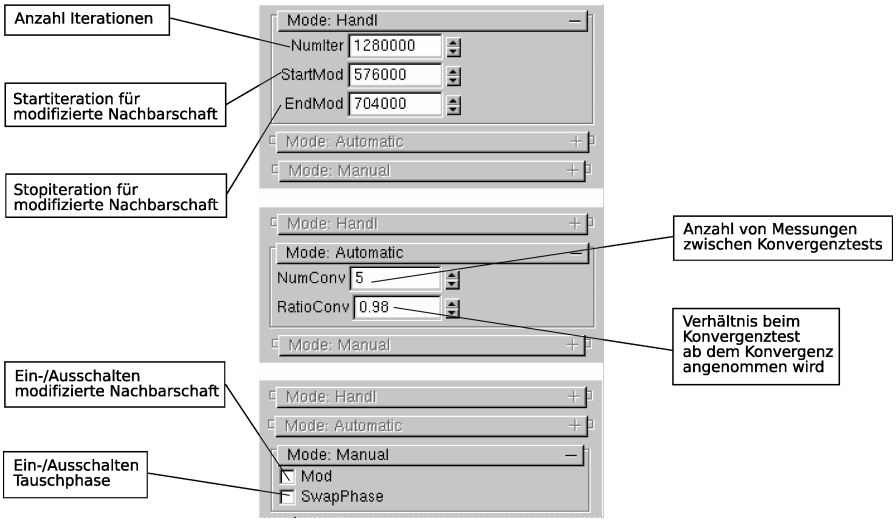

Abbildung 5.8: Erläuterungen zum GUI des Swarm-Systems für die drei verschiedenen Ausführungsmodi.

zum einen an einer niedrigeren maximal erreichbaren Sortierung, zum anderen an weniger festen Clustergrenzen liegt. In Phase 3 wird die modifizierte Nachbarschaftsfunktion eingesetzt. Dies ist deutlich durch die Absenkung des Zusammenhangs und die gute Sortierung zu erkennen. In Phase 4 bleibt die Sortierung auf ihrem erreichten hohen Stand, wobei der Zusammenhang wieder steigt. Hier bilden sich neue Clusterzentren aus. Die Höhe der Sortierung ist bei gleichverteilten Daten niedriger, da benachbarte Objekte im Allgemeinen größere Attributraumdistanzen aufweisen. In Phase 5 sinkt der Zusammenhang etwas. Dies sollte eigentlich zur besseren Separierung der Cluster eingesetzt werden. Trotzdem erweitern sich auch die Cluster selber, was unbeabsichtigt zu mehr freien internen Positionen führt.

Im Automatischen Modus sollen die starren Phasen des Handl-Modus durch die Verwendung der Entropiemaße durchbrochen werden. Auch hier ist ein ähnlicher Kurvenverlauf zu sehen. Man kann gut die vier Phasen (siehe Tabelle 5.3) erkennen. In Phase 1 steigen Sortierung und Zusammenhang leicht an. Bei Konvergenz wird die modifizierte Nachbarschaftsfunktion eingesetzt. Dies führt hier zu einem Maximum an Sortierung, da nur wirklich ähnliche Objekte benachbart sind. Nach dem Wechsel zur normalen Nachbarschaftsfunktion bleibt die Sortierung hoch. Zusätzlich steigt der Zusammenhang an, da sich wieder Cluster formieren. Im Unterschied zum Handl-Modus wird hier mit einem hohen Wahrnehmungsradius von 5 begonnen, der in der letzten Phase auf den Wert 3 gesenkt wird. Dadurch werden dichtere Cluster erreicht. Insgesamt kann hier schneller ein gleiches oder besseres Ergebnis erziehlt werden.

	Handl		Automatisch	
	normalverteilt	gleichverteilt	normalverteilt	gleichverteilt
1000 Objekte, 100x100 Grid, Schrittweite=141	2000000, 18	2000000, 21	154500, 2	335000, 4
5000 Objekte, 225x225 Grid, Schrittweite=318	10125000, 250	10125000, 225	572500, 13	600000, 13

Tabelle 5.4: Simulationen zur Evaluation des Swarm-Systems. In jeder Zelle befinden sich die Anzahl an Iterationen und die benötigte Rechenzeit in Minuten. Die Simulationen wurden auf einem AMD Athlon XP 1900+ Rechner mit 768MB RAM durchgeführt.

5.6 Zusamnmenfassung

Durch die Entwicklung der Entropiemaße Sortierung und Zusammenhang wurde ein neuer Automatischer Modus ermöglicht, der die starren Zeitstrukturen des Handl-Algorithmus aufbricht. Der Vergleich in Qualität und Zeitbedarf fällt zu Gunsten des Automatischen Modus aus.
Trotz ihrer Robustheit weisen beide Verfahren einige Schwächen auf. Es konnten beispielsweise bei der Verwendung vieler Objekte nicht alle Cluster zusammengeführt werden, die eigetlich zusammen gehören. Hier wäre eine Nutzerinteraktion vorstellbar. Desweiteren läßt sich die Informationsinfrastruktur verkleinern, was zu einer kleineren Schrittweite und damit weniger Rechenzeit führt (siehe Kapitel 4). Weiterhin sind die resultierenden Cluster nicht räumlich so gut voneinander getrennt, wie man es sich wünscht (gut verteilt über die Informationsinfrastruktur). Weitere konkrete Problemstellungen werden im Folgenden erläutert.

Probleme beim Algoritmus von HKD

- **Parallelisierung**
  Für die Implementierung eines Ant-based-Clustering-Systems auf aktueller Grafikhardware oder anderen parallelen Recheneinheiten, müssen die Aktionen der Agenten unabhängig voneinander ablaufen. Diese Unabhängigkeit ist nicht gegeben, da die Agenten eine gemeinsame Informationsinfrastruktur modifizieren. Es ist daher darauf zu achten, daß zwei Agenten nicht gleichzeitig auf eine Position bzw. Nachbarschaft verändernd zugreifen können. Dies ist ein eher allgemeines Problem aller Ant-based-Clustering-Algorithmen.

- **Hinzufügen von Objekten zur Laufzeit**
  Der großer Unterschied zu anderen vorgestellten Verfahren besteht in der zeitlichen

Adaption des Algorithmus, was u.a. durch die Erhöhung des Wahrnehmungsradius oder die Anpassung des Parameters α erreicht wird. Dieses Feature führt beim Hinzufügen von Objekten zur Laufzeit zu Problemen. Die Lösung wäre, nur so viele Objekte hinzuzufügen, daß der aktuelle Systemzustand nicht umkippt. Hier kann die Verwendung der im Projekt entwickelten Maße Sortierung und Zusammenhang (siehe Kapitel 3) helfen.

- **Veränderung der Gewichtung von Objektattributen**
  Eine Veränderung der Distanzfunktion führt zu einer Umstrukturierung der Cluster auf der Informationsinfrastruktur. Da der Algorithmus über die Zeit ein adaptives Verhalten aufweist (siehe vorherige Problemstellung), bauen verschiedene Phasen aufeinander auf. Daher kann hier oft nur ein Neustart des Systems helfen, was wiederum zu einer Erhöhung der Rechenzeit führt.

- **Rechenzeit**
  Die Zeitpunkte zur Inkrementierung des Wahrnehmungsradius und zum Einsatz der modifizierten Nachbarschaftsfunktion werden fest vorgegeben. Durch die Verwendung der Maße Sortierung und Zusammenhang, können Zustände erkannt werden, was eine adaptive Vorverlegung der o.g. Zeitpunkte ermöglicht und damit zu einer stark verringerten Laufzeit führt.

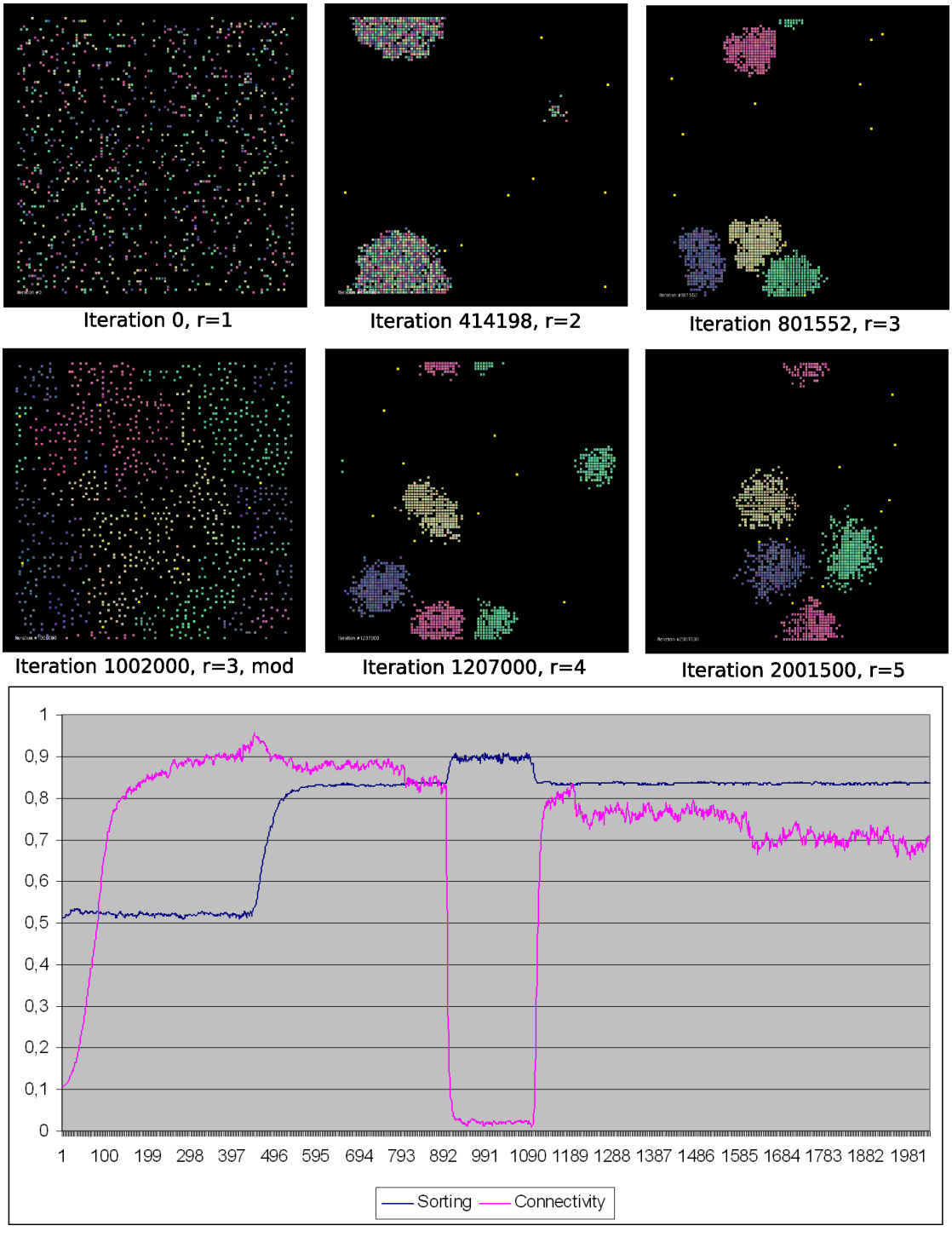

Abbildung 5.9: Simulation 1: Handl, 1000 Objekte normalverteilt, 2000000 Iterationen, 18 min.

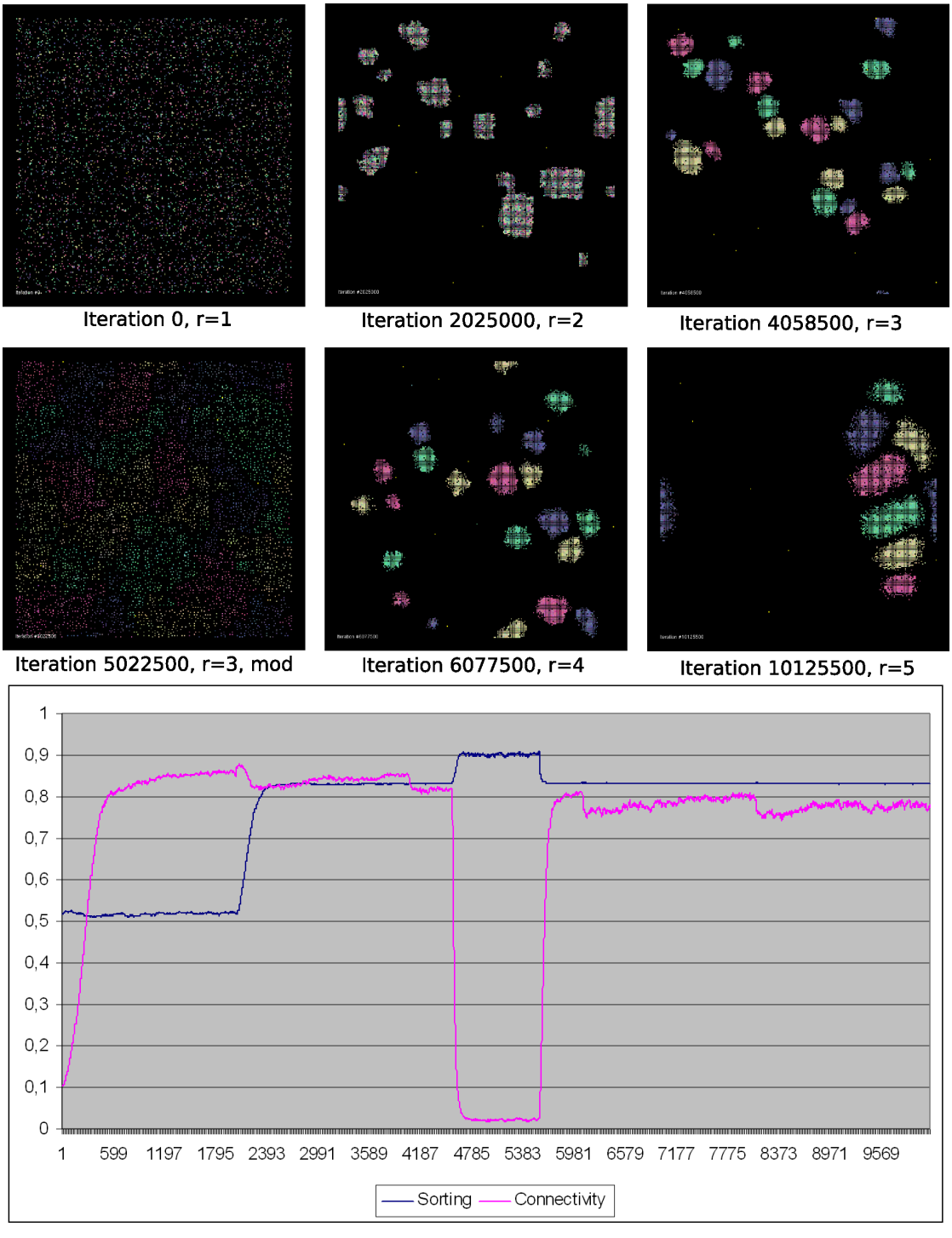

Abbildung 5.10: Simulation 2: Handl, 5000 Objekte normalverteilt, 10125000 Iterationen, 250 min.

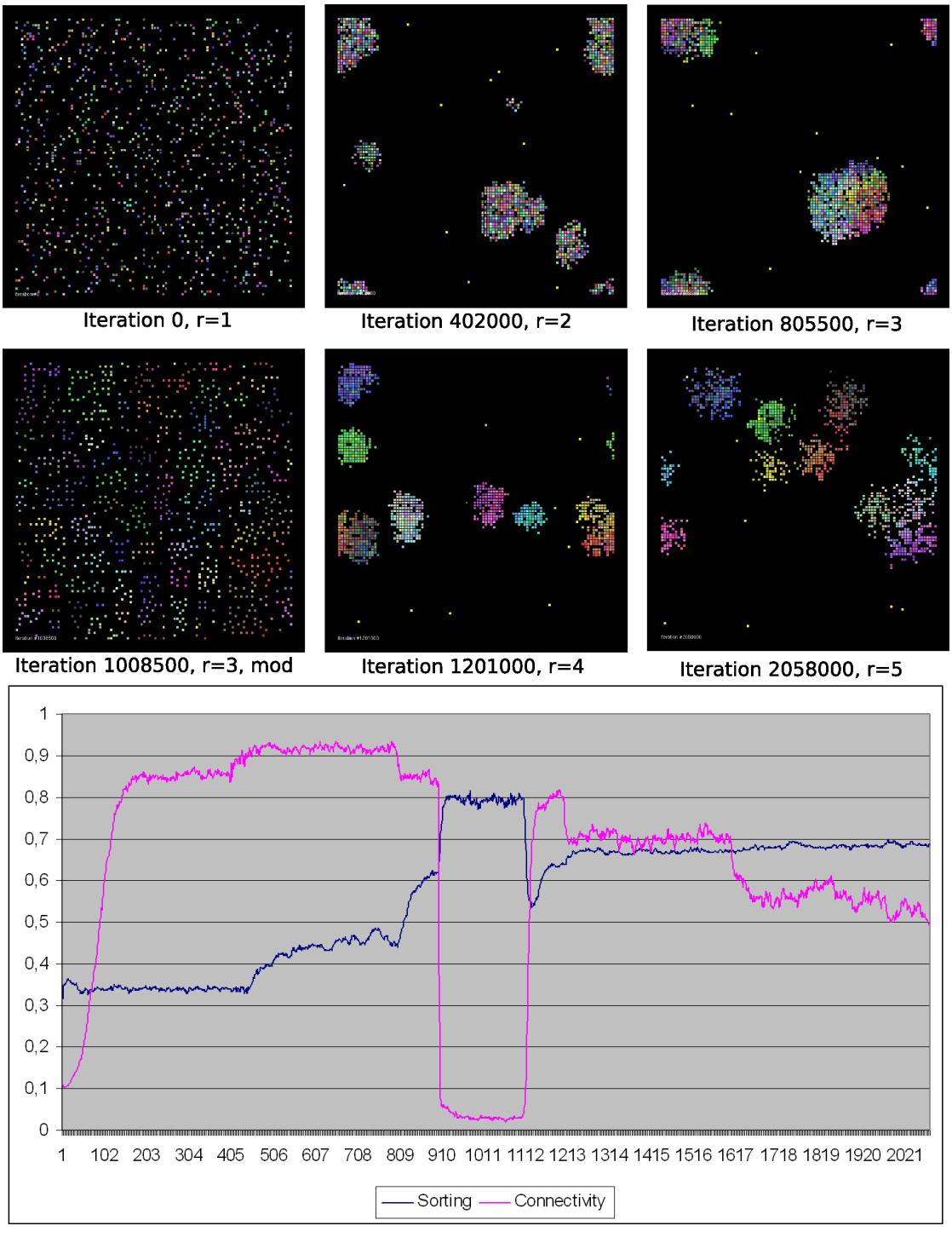

Abbildung 5.11: Simulation 3: Handl, 1000 Objekte gleichverteilt, 2000000 Iterationen, 21 min.

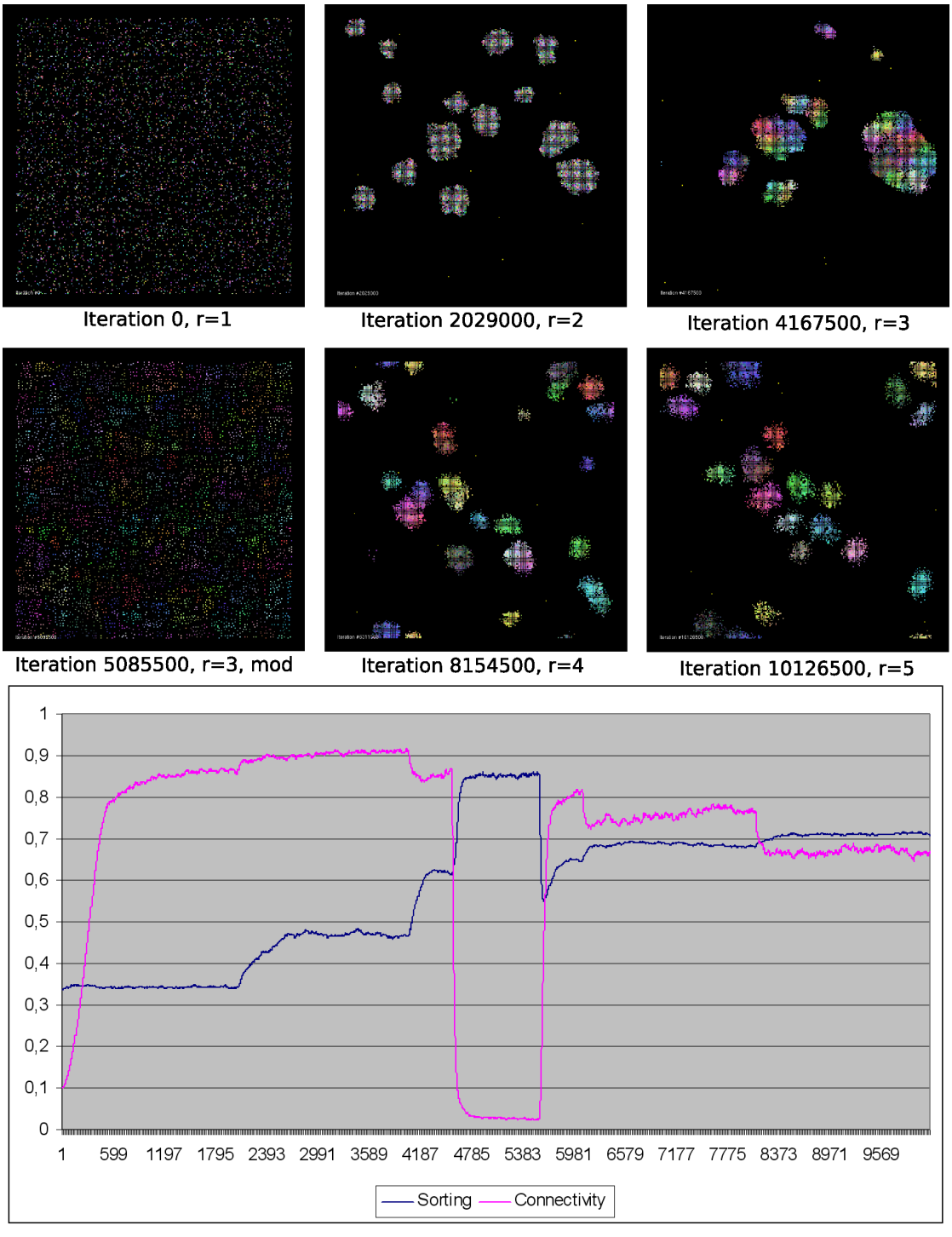

Abbildung 5.12: Simulation 4: Handl, 5000 Objekte gleichverteilt, 10125000 Iterationen, 225 min.

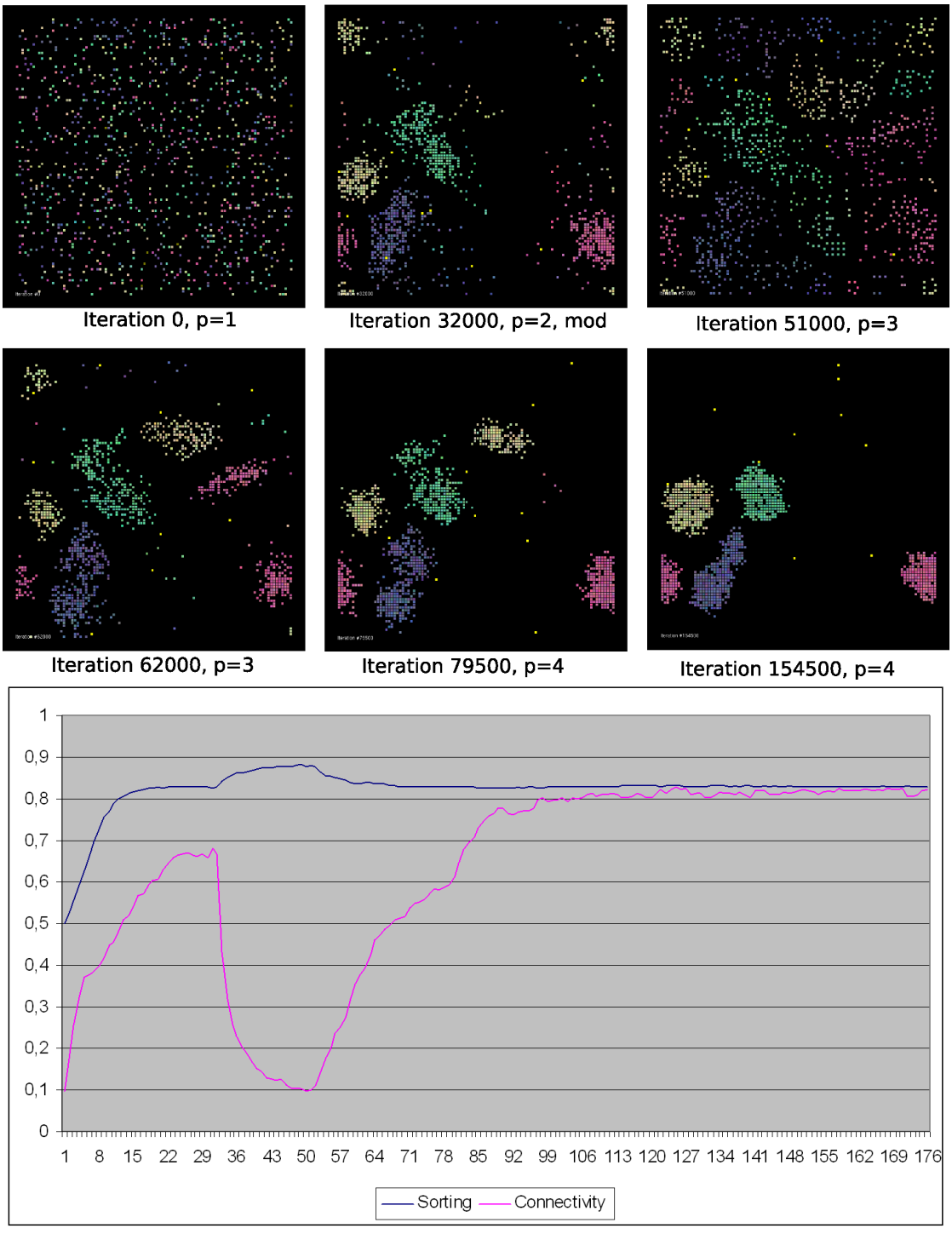

Abbildung 5.13: Simulation 5: Automatisch, 1000 Objekte normalverteilt, 154500 Iterationen, 2 min.

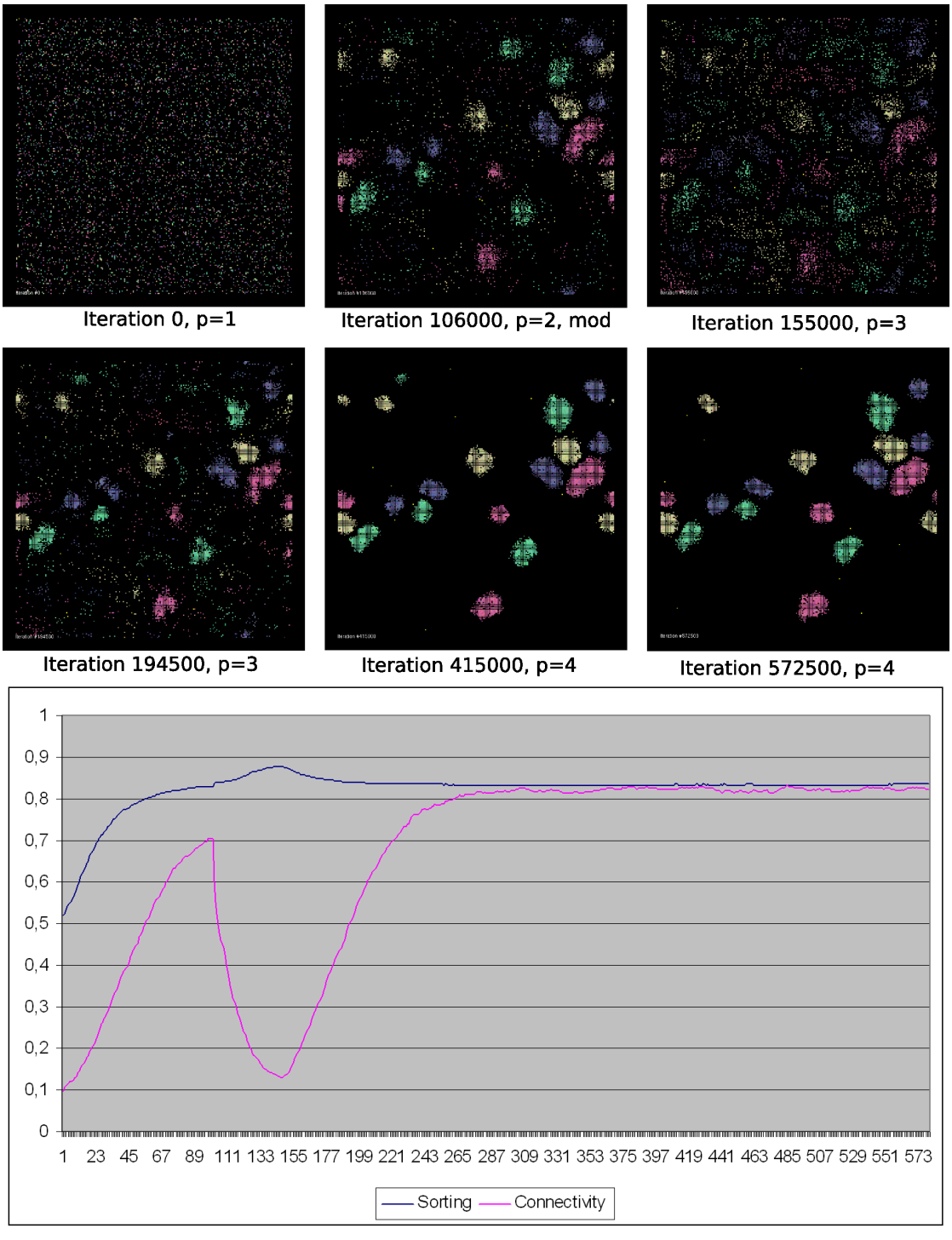

Abbildung 5.14: Simulation 6: Automatisch, 5000 Objekte normalverteilt, 572500 Iterationen, 13 min.

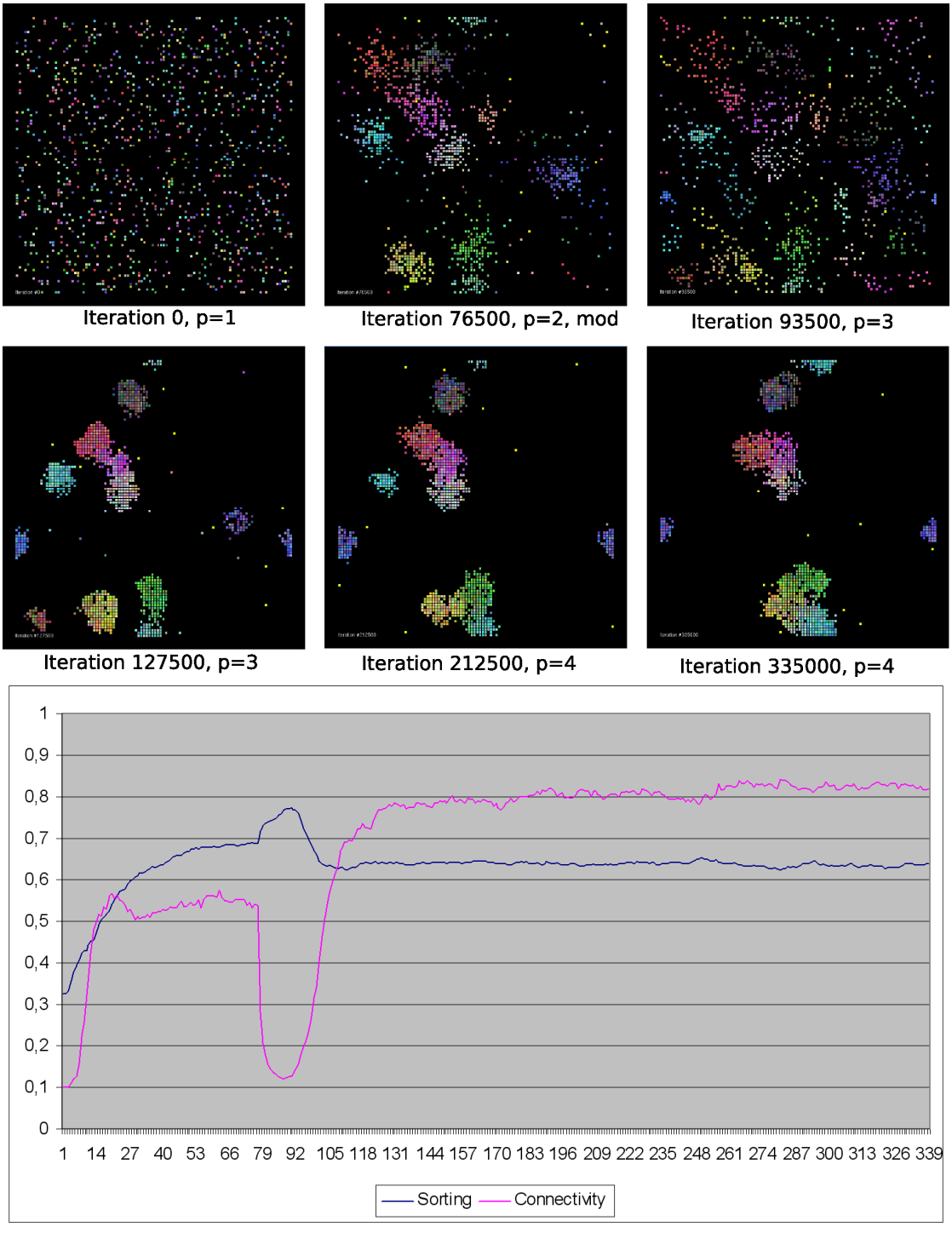

Abbildung 5.15: Simulation 7: Automatisch, 1000 Objekte gleichverteilt, 335000 Iterationen, 4 min.

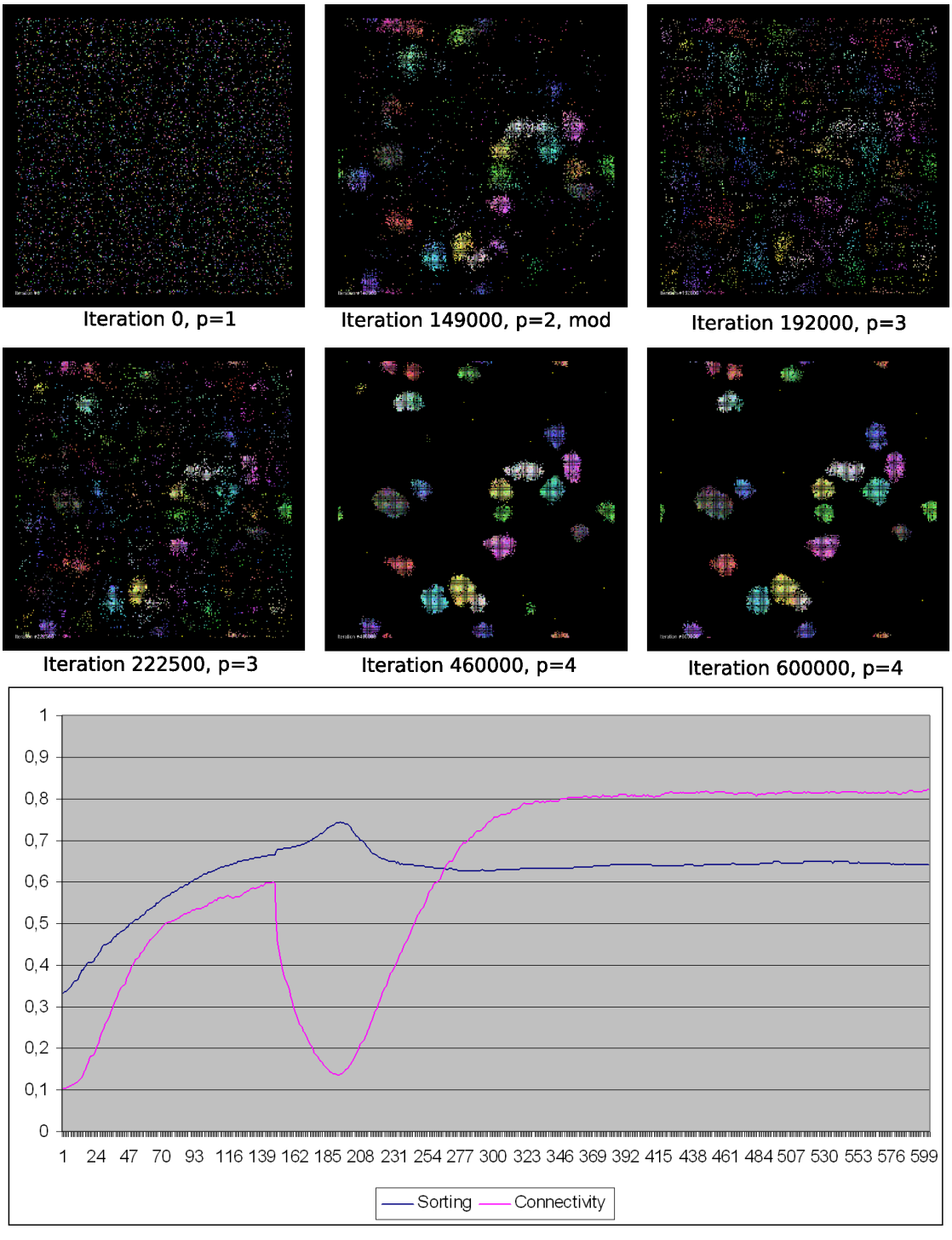

Abbildung 5.16: Simulation 8: Automatisch, 5000 Objekte gleichverteilt, 600000 Iterationen, 13 min.

6 Swarm-System in 3D und Pheromon-Bahnen

6.1 Einleitung

Im folgenden Abschnitt werde ich erst auf die Überlegungen zur Entwicklung des 3D-Simulations-Frameworks eingehen um danach die technische Implementierung und die einzelnen Komponenten zu erläutern. Schließlich folgen dann im letzten Abschnitt die Simulationen und Auswertungen sowie eine Zusammenfassung und der Ausblick für zukünftige Entwicklungen.

6.1.1 Schwerpunkte des Frameworks

Zahlreiche Aspekte zum Aufbau und der Funktionsweise der SI-Systeme wurden bereits in den beiden vorherigen Kapiteln näher betrachtet. Neben dem Untersuchen und Verstehen der grundlegenden Prinzipien bei SI-Systemen ist einer unserer weiteren Forschungsschwerpunkte gewesen, zu untersuchen wie sich die SI-basierten Systeme zur Visualisierung von Daten bei gleichzeitiger Interaktion durch den Nutzer einsetzen lassen. Auch sollten neben zweidimensionalen Visualisierungen der virtuellen Umgebung ebenfalls 3D-Darstellungen eingesetzt werden, und deren Vor- sowie Nachteile herausgefunden werden. In dem System sollten einige unterschiedliche Ansätze für die Sortierungen von Daten eingesetzt werden, und dies sind insbesondere die Algorithmen von Deneubourg [Deneubourg et al. 1991], Lumer, Faieta [Lumer & Faieta 1994], Ramos [Ramos & Abraham 2003], Dorigo [Dorigo 2004] und Bonabeau [Bonabeau et al. 1994]. Ziel war es weiterhin, durch eine neue Kombination der Verfahren und Erweiterungen der bestehenden Algorithmen weitere Erkenntnisse über den Ablauf von SI-Systemen zum *Clusterung* zu erhalten. Auch sollen unterschiedliche Formen der 3D Welten als Umgebung der Simulation eingesetzt und bewertet werden.

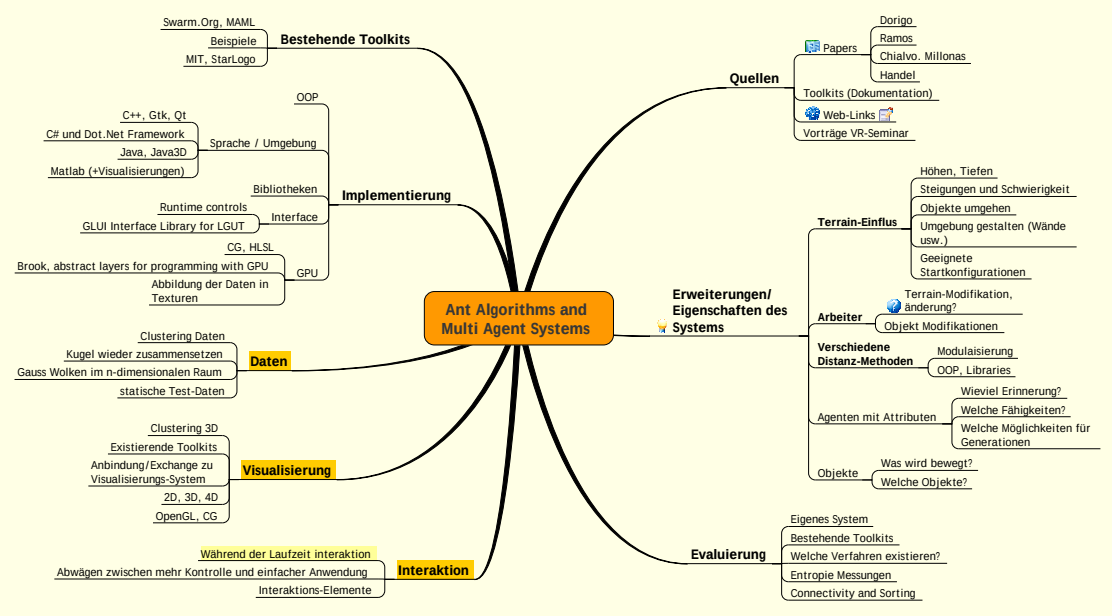

Abbildung 6.1: Mindmap des Projektes mit wichtigen Themenbereichen.

6.1.2 Projekt-Vorgehen

In den ersten Wochen nach dem Projektbeginn erfolgte die Einarbeitung in die aktuellen Veröffentlichungen zur *Swarm Intelligence*, aber auch in Arbeiten zur Gruppierung von Datenmengen und Visualisierung von Daten.
Nach der Erstellung von einigen Vorträgen zu den Themen „OpenGL“, „CG Programmierung“, „Einführung in SI“ und „Visualisierungen“ wurde auch der Rahmen des Projektes enger umfasst und Schwerpunkte definiert (siehe Mindmap in Abbildung 6.1). Anschließend erfolgten die ersten Implementierungen der Schwarm-Simulationen. Zu Beginn nur mit Textausgaben über die Systemkonsole, wurden in den nächsten Schritten Darstellungs- und Interaktionselemente integriert. Schrittweise wurde das umfangreiche 3D-Simulationsframework entwickelt. Es folgten Erweiterungen der Algorithmen als auch der Visualisierungen. Schließlich wurden zahlreiche Test-Simulationen durchgeführt, mit deren Hilfe es möglich wurde die Verfahren und Erweiterungen bewerten zu können.

6.1.3 Weitere Swarm-Intelligence Frameworks

In den letzten Jahren wurden einige Systeme entwickelt, mit denen Simulationen und Experimente zur *Swarm Intelligence* möglich sind. Zu den bekanntesten Systemen

gehört dabei sicher das Swarm.Org-Toolkit [Swarm.org 2004], für das zahlreiche Anwendungsbeispiele existieren (umfassend in Bezug auf den gesamten Bereich der Swarm-Intelligence und nicht nur für Clustering-Probleme). Drei der Systeme möchte ich exemplarisch vorstellen und erläutern, aus welchen Gründen ich mich für die Entwicklung eines eigenen Frameworks entschied.

- **Swarm.org**
  Das Schwarm-Simulations-System des Santa Fe Insitute gehört zu denen am häufigsten eingesetzten und erweiterten Anwendungen dieses Bereichs. Es wird von zahlreichen Entwicklern im Funktionsumfang kontinuierlich erweitert und ist unter der GPL Lizenz frei für Entwickler verfügbar[1].
  Das Toolkit ist vielseitig anwendbar für verschiedenste Simulationen zur *Swarm Intelligence*, jedoch ist das Toolkit nicht unmittelbar zur leichten Implementierung der unterschiedlichen *Clustering* Algorithmen geeignet. Eine Schwierigkeit wäre auch die dreidimensionale Visualisierung, denn über einfache 2D Visualisierungen gehen die Möglichkeiten dieses Toolkits nicht hinaus. Der Schwerpunkt für Simulationen des Swarm.org-Toolkits liegt nicht bei Clustering Verfahren, auch wenn ein einfaches Anwendungsbeispiel zum Gruppieren von Objekten implementiert werden kann, wie in Abbildung 6.2 zu sehen ist (diese Anwendung verfügt nur über einen simplen Algorithmus, und benötigte für die zu sehende Gruppierung der Elemente weit über 10 Millionen Iterationen).

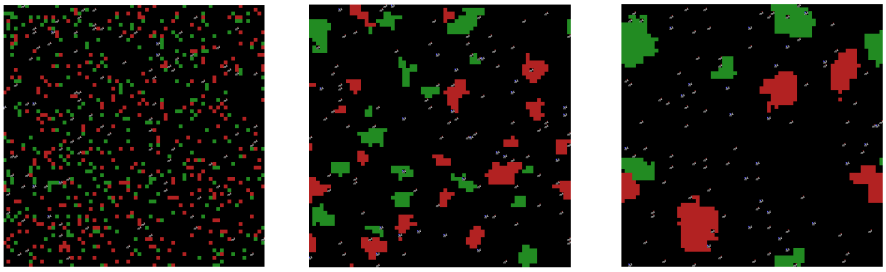

Abbildung 6.2: Clustering Simulation des Swarm.org Toolkits

- **RePast**
  Das RePast[2] System wurde an der University of Chicago entwickelt und ist ein in Java programmiertes Framework zur Modellierung von Agenten basierten Simulationen [RePast 2004]. Die Entwickler bezeichnen ihr System selbst als „sehr nahe zu Swarm.org“, betonen aber gleichzeitig die Unterschiede zwischen den beiden Systemen, wie u.a. die Erweiterugen zur direkten Interaktion während der

[1] Die ursprüngliche Website des Projektes ist unter http://www.swarm.org erreichbar; das derzeit aktualisierte Wiki unter http://wiki.swarm.org. Unter dieser Adresse ist auch der Download des Systems verfügbar, sowie eine ISO-CD Zusammenstellung unter `http://eco83.econ.unito.it/swarm/materiale/cd/` (Stand 07/2004)
[2] RePast bedeutet „REcursive Porous Agent Simulation Toolkit“

Simulations-Laufzeit durch GUI-Widgets[Collier 2003].
Das Framework bietet bereits zahlreiche der Funktionen, die in unseren Simulations-Systemen vorhanden sein sollen. Jedoch ist es nicht unmittelbar möglich, das System um eine dreidimensionale Visualisierung zu erweitern, da alle vorhandenen Module und Klassen auf den 2D Visualisierungen der Elemente und Agenten basieren. Der objektorientierte Ansatz und die Verbindungen der Module sind jedoch sehr überzeugend und bilden konzeptionell eine mögliche Grundlage für unsere eigenen Systeme.

- **Breve**
  Dieses System bietet 3D Visualisierungen in einer virtuellen Welt, in der physikalische Gesetze möglichst exakt von den modellierten Agenten eingehalten werden. Komplexe Berechnungen ermöglichen eine detailgetreue Modellierung, die jedoch nur den Einsatz einer geringen Anzahl von Agenten in dem System ermöglicht [Klein 2002]. Die Anwendungen des Systems erinnern an Experimente der Robotik (so die Anwendungen „Evolving Virtual Creatures“ und „Braitenberg Vehicles“), in der sich ebenfalls ausführlich mit Problemen der Schwärme und des Schwarm-Verhaltens beschäftigt wird. Das Framework hat seinen Schwerpunkt in der exakten virtuellen Umsetzung einer physikalischen Welt der Agenten, und eigent sich somit nicht unmittelbar für die Clustering Algorithmen (die mit einer hohen Anzahl von Agenten arbeiten; jedoch i.d.R. ohne exakte physikalische Berechnungen). Der Overhead für die in unseren Simulationen nicht notwendigen Physik-Berechnungen ist zu groß und es fehlen flexible Schnittstellen für Algorithmen oder ein Interface zur Interaktion.

Um alle für uns wichtige Aspekte der Swarm-Systeme untersuchen zu können, entschieden wir uns für die Entwicklung eigener Frameworks. Insbesondere sollten es die Systeme ermöglichen, neben 2D-Visualisierungen auch Darstellungen in drei Dimensionen darzustellen, wobei dieses Feature in den beschriebenen Swarm-Toolkits Swarm.org und Repast nicht unterstützt wird, und im Simulationssystem von Breve ein zu großer Overhead durch präzise physikalische Simulation erfolgt, die deses System zwar auf besondere Weise auszeichnet, für unsere Anwendung jedoch nicht notwendig ist.

6.2 System und Ablauf der Simulationen

6.2.1 Bestandteile des Systems

In diesem Abschnitt werden Zusammenfassend die wichtigsten Komponenten des Schwarm-Systems erläutert: Welt-Umgebung, Objekte, Agenten und der Schwarm.

Welt-Umgebung und Veränderungen
Alle Agenten gemeinsam agieren in der Welt-Umgebung, die einen in sich geschlossenen Raum darstellt (toroide Randbedingung). Während der Initialisierungsphase wird die Größe der Welt angegeben und anschließend als dreidimensionale Datenstruktur erstellt. Die Veränderungen welche die Agenten durch ihre Aktionen in der Welt ausführen (Objekte aufheben und ablegen, sowie Pheromon-Aussonderung) bilden das einzige indirekte Kommunikationsmedium der Agenten. Indirekt deshalb, weil jeder Agent die Veränderungen nur deshalb durchführt, um seine vorgegebene Tätigkeit zum bestmöglichen Ergebnis zu führen. Seine Veränderungen wiederum werden von allen anderen Agenten wahrgenommen, sollten sie sich in der Umgebung des anderen Agenten befinden oder später dorthin gelangen. Die stattgefundenen Veränderungen beeinflussen somit die Aktionen der weiteren Agenten, und somit bildet sich eine Kette zahlreicher aufeinander aufbauenden Elementar-Aktionen.

Agenten
Die Agenten sind die einzigen agierenden Komponenten in dem System und können Aktionen mit denen in der Welt vorhandenen Objekten durchführen. Die Agenten bilden Verhalten und Eigenschaften von Ameisen-Schwärmen nach, was durch Berechnungen auf den erfassten Objekten und der Umgebung sowie durch die Aussonderung von virtuellen Pheromon-Stoffen simuliert wird.
Zu den Berechnungen jedes einzelnen Agenten gehört zum einen die Bestimmung des Weges sowie auch die Berechnungen für das Aufheben und Ablegen von Objekten. Weiterhin können neben einer statischen Abgabe von Pheromonen in jedem Schritt zusätzlich auch dynamisch Pheromone abgegeben werden. Die genauen Algorithmen dazu werden in den nächsten beiden Abschnitten erläutert.

Schwarm und Spezialisierung
Zu einem Schwarm der Simulation gehört eine Anzahl von Agenten, die in einem STL-Vektor mit Pointern auf Instanzen der einzelnen Agenten abgelegt werden. Mit diesen Vektor wird im häufigsten Fall über alle vorhandenen Agenten iteriert, um die entsprechenden Methoden der Ant-Klasse aufzurufen.
Aber auch Iterationen über Teil-Bereiche des Schwarms sind möglich, um somit bei Veränderungen der Eigenschaften bei einem Teil der Agenten ausführen zu können. Damit werden Agenten des Schwarms spezialisiert, was in gewisser Weise den Grundprinzipien der Swarm-Intelligence entgegensteht. Denn prinzipiell sind die Eigenschaften der Agenten eines Systems gleich, woraus auch eine der größten Stärken der Systeme besteht: Robustheit gegenüber dem Ausfall einzelner Individuen. Denn ist ein einzelner Agent nicht mehr fähig seine Aufgabe durchzuführen, kann der restliche Schwarm die Aufgabe ohne große Schwierigkeiten zu Ende führen. Somit kann eine zu starke Spezialisierung der Agenten zu einem weniger robusten Gesamtsystem führen, und gleichzeitig steigert eine Spezialisierung auch die Komplexität des Systems und erschwert somit den Überblick über das Gesamtsystem. Je mehr Parameter bei den Agenten oder einigen

Individuen modifiziert werden, desto unklarer wird deren Einfluss auf den Verlauf des Gesamtsystems. Deshalb werden die Agenten eines Schwarms i.d.R. als Einheit betrachtet und auf Spezialisierung weitestgehend verzichtet.

Objekte
Die Objekte der virtuellen Welt stellen die zu gruppierenden Datenelemente dar. Alle Objekte werden während der Initialisierungsphase dynamisch erzeugt und anschließend an einer Position der Welt abgelegt, die durch einen Zufallsgenerator erzeugt wird. Auf die Verfahren zur Erstellung der Objekte werde ich später näher eingehen.
Objekte besitzen verschiedene Eigenschaftswerte, die in einem STL Vektor gespeichert werden. Über die entsprechenden Konstruktoren kann dieser Vektor bereits während der Erstellung mit Datenwerten besetzt werden, oder es werden weitere Eigenschaften dynamisch während der Laufzeit hinzugefügt. Im System gibt es eine definierte maximale Anzahl von Eigenschaften die bei allen Objekten berücksichtigt werden sollen; somit ist es nicht möglich, bei zwei Objekten eine unterschiedliche Anzahl von Eigenschaftswerten in die Berechnung aufzunehmen.

6.2.2 Berechnungen und Algorithmen

Eingeführt wurde die Methode des agenten-basierten Clusterings von Deneubourg et al. [Deneubourg et al. 1991] 1991, wobei sich die Forschungen dabei mit Simulationen durch Roboter beschäftigten. Diese konnten zufällig in ihrer Umgebung plazierte Objekte gruppieren indem sie zählten, wie viele Objekte sich in der unmittelbaren Nachbarschaft befinden. Die Roboter haben dazu ein Kurzzeitgedächtnis, in dem sie sich die letzten Positionen von Objekten merken und somit eine Entscheidung zum Aufheben und Ablegen von Objekten treffen können. Dazu wurden die folgenden Wahrscheinlichkeitsfunktionen in Formel 6.1 und Formel 6.2 für das Aufheben und Ablegen definiert [Deneubourg et al. 1991]:

$$p_{picking}\,(i) = \left(\frac{k_1}{k_1 + count_{similarNeighbours}((i)}\right)^2 \tag{6.1}$$

$$p_{dropping}\,(i) = \left(\frac{count_{similarNeighbours}((i)}{k_2 + count_{similarNeighbours}((i)}\right)^2 \tag{6.2}$$

Dabei sind k_1 und k_2 Schwellenwert-Parameter, welche die Wahrscheinlichkeit für das Ablegen und Aufheben von Objekten variieren (in den Algorithmen nach Deneubourg sind $k_1 = 0.1$ und $k_2 = 0.3$). $count_{similarNeighbours}((i)$ steht für die Anzahl der Nachbarn

die während den letzten Schritten des Roboters wahrgenommen wurden, geteilt durch die Gesamtzahl an verglichenen Feldern (also im zweidimensionalen 8 Felder). Bei der Picking-Probability gilt weiterhin: $f << k_1 \Rightarrow p_{picking} \approx 1$, d.h. die Wahrscheinlichkeit für das Aufheben des Objektes ist besonders hoch wenn nur wenige Objekte in der Nachbarschaft gefunden wurden. Analog gilt bei der Dropping-Probability: $f >> k_2 \Rightarrow p_{dropping} \approx 1$, d.h. die Wahrscheinlichkeit für das Ablegen des Objektes ist in diesem Fall besonders hoch, da sich viele Objekte in der Nachbarschaft befinden.
Eine Unterscheidung von unterschiedlichen Objekten wurde in diesen Simulationen vorerst nicht vorgenommen; es ging vorerst nur um die Entscheidung, ob ein Objekt sich bereits in einem Cluster befindet oder nicht und wo es ggf. an einer geeigneteren Stelle wieder abgelegt werden kann.
Mit der Durchführung der Simulation wurden mit den Robotern ganz ähnliche Ergebnisse erzielt wie sie auch bei der Beobachtung von Ameisen-Schwärmen zu sehen sind; so zum Beispiel bei der Sortierung von toten Ameisen-Kadavern durch noch einige lebende Ameisen[3]; zu sehen in Abbildung 6.3.

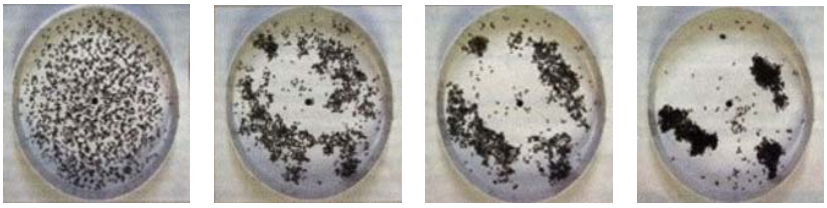

Abbildung 6.3: Die Untersuchungen zur Gruppierung von toten Ameisen [Abbildung von L. Chrétrien]

Auch wenn mit diesem System bereits Simulationen möglich sind, bezeichnet Ramos et al. [Ramos & Abraham 2003] die Ergebnisse als nicht zufriedenstellend. Es seien über fünf Millionen Iterationen notwendig um annehmbare Ergebnisse mit diesem Algorithmus zu erzielen (100x100 Gitter, 400 OBjekte, 10 Agenten).
Eine Erweiterung dieses Systems wurde 1994 von Lumer und Faieta [Lumer & Faieta 1994] entwickelt, mit der es nun möglich wurde, die SI-Algorithmen zur Datananalyse zu verwenden. Jedes Objekt besitzt eigene Attribute, die in einem Vektor der Dimension n abgelegt werden. Diese Attribut-Vektoren spannen nun einen Raum R^n auf, in dem jedes Objekt eindeutig positioniert ist und eine eindeutige Distanz zu allen anderen Objekten dieses Raumes aufweist. Zur Berechnung dieser Distanz wird die euklidische Distanz verwendet (wie in Formel 6.6), die normiert einen Wert im Intervall $[0, 1]$ darstellt.
Um ein Objekt mit seiner umliegenden Nachbarschaft[4] zu vergleichen, führen Lumer

[3] Studien nach L. Chrétrien: Organisation Spatiale du Matérial Provenant de l'excavation du nid chez Messor Barbarus et de Cadavres d'ouvrières chez Lasius niger, Ph.D. dissertation, Université libre des Bruxelles, 1996

[4] In 2D sind dies in der Regel 8 Nachbarn, in 3D insgesamt 26 Nachbarn, wenn der Radius der Wahrneh-

und Faieta die Formel für die Übereinstimmungs-Berechnung ein [Lumer & Faieta 1994]. In Formel 6.3 wird durch $f(o_i)$ der Übereinstimmungsgrad des Objektes o_i mit seinen Nachbarn berechnet.

$$f(o_i) = \max\left\{0, \frac{1}{s^2} \sum_{o_j \in Neighbour_{(sxs)}(r)} \left[1 - \frac{distance(o_i, o_j)}{\alpha}\right]\right\} \tag{6.3}$$

Dabei findet mit $\sum_{o_j \in Neighbour_{(sxs)}(r)}$ eine Iteration über alle Nachbarn o_j statt. Summiert wird der inverse Wert der Distanzberechnung, die vorher schon durch α dividiert wurde. Das α ist ein Schwellen-Wert, der den Einfluß der Distanzberechnung auf das Ergebnis der Nachbarschafts-Übereinstimmung regelt. Ist α zu hoch, so wird wenig zwischen den einzelnen Attributen der Objekte unterschieden (und somit werden die Objekte nahezu wahllos gruppiert). Ist α zu niedrig, so findet keine Cluster-Bildung statt, da keine gemeinsamen Merkmale gefunden werden.
Für die Wahrscheinlichkeit des Aufhebens und Ablegens von Objekten definierten Lumer und Faieta weiterhin zwei modifizierte Formeln 6.4 und 6.5:

$$P_{picking}(o_i) = \left(\frac{k_1}{k_1 + f((o_i)}\right)^2 \tag{6.4}$$

$$P_{dropping}(o_i) = \begin{cases} 2 \cdot f(o_i), & if\ f(o_i) < k_2 \\ 1, & if\ f(o_i) \geq k_2 \end{cases} \tag{6.5}$$

Dabei sind k_1 und k_2 wieder Schwellenwert-Parameter um Wahrscheinlichkeiten für das Ablegen und Aufheben von Objekten zu variieren, mit leicht von Deneubourg abweichenden Werten: $k_1 = 0.1$ und $k_2 = 0.15$.

$$distance_r(o_i, o_j) = \frac{1}{d_{\max}} \cdot \sqrt{\frac{1}{count_F} \cdot \sum_{n=1, count_F} \left[f_{o_i}(n) - f_{o_j}(n)\right]^2} \tag{6.6}$$

Weitere Einzelheiten zu den Algorithmen und den Grundlagen sind in den genannten Veröffentlichungen [Lumer & Faieta 1994] [Deneubourg et al. 1991] und der Seminarausarbeitung [Marquardt 2003] zu finden.

mung 1 Feld beträgt. Vereinzelt werden auch höhere Wahrnehmungsradien verwendet, wie in Kapitel 5 beim Verfahren nach J. Handel näher erläutert wurde.

6.2.3 Pheromone Grundlagen

Um die Anwendung von Pheromon-Spuren für die Simulationen zu erläutern, werden zuerst einige der Grundlagen zu der Verwendung von Pheromonen bei Ameisen-Schwärmen erläutert.
Ameisen besitzen viele Möglichkeiten sich in ihrer Umgebung zu orientieren und mit ihren Artgenossen zu kommunizieren. Die Orientierung findet in erster Linie durch die Wahrnehmung von Lichtquellen statt: Bewegung auf Lichtquellen zu (Phototaxis) und die Bewegung in einem bestimmten Winkel zu einer Lichtquelle (Menotaxis). Zusätzlich ist noch eine Orientierung anhand der Schwerkraft möglich, in dem die Ameise versucht, einen bestimmten Winkel ausgehend vom senkrechten Winkel nach unten einzuhalten. Dazu kommen noch zahlreiche Formen der Kommunikation, wie z.B. durch die Körpersprache (Abfolgen, Tänze), Fühlersprache und Akkustik. Die sicher interessanteste und komplexeste Form ist jedoch die indirekte Kommunikation und Orientierung durch Aussonderung von Pheromon-Stoffen.
Zahlreiche biologische Untersuchungen zum Einsatz der Pheromone bei Ameisenschwärmen, genannt Osmotropotaxis, wurden von Edward Wilson veröffentlicht. Sie beschreiben die Aussonderung von Pheromonen aus den Drüsen der Ameisen, die wiederum von zwei Fühlern der Ameisen erfasst werden können und ein Vergleich der Konzentrationen möglich wird. Die Konzentration der Pheromone wird nach ihrer Aussonderung durch Verdampfen stetig geringer bis sie schließlich (in den Untersuchungen der Feuer-Ameise) nach etwa 100 Sekunden unter die noch wahrnehmbare Schwelle sinken [Wilson 1990].
Der Einsatz dieser Pheromonmarkierungen dient den Ameisen insbesondere zur effizienten Koordinierung ihrer Futtersuche. Ameisenschwärme beginnen gleichzeitig vom Nest aus mit der Suche nach Nahrung und geben kontinuierlich Pheromone ab. Finden sie eine Futterquelle beginnen sie sofort mit dem Weg zurück zum Nest und treten den Weg von neuem an. Die Konzentration entlang dieses Pfades wird stetig höher, währen Konzentrationen bei anderen (nicht erfolgreichen und längeren) Pfaden schließlich verdampfen und nicht mehr wahrnehmbar werden. Dies ist ein gutes Beispiel für die sich ergebende Emergenz des Schwarmes, denn erst durch gemeinsame aktive und passive Tätigkeiten der einzelnen Ameisen wird eine schnelle und effektive Futtersuche des Schwarms ermöglicht [Wilson 1990]. Trotz einiger Erkenntnisse über die Pheromone sind viele Details zum Einsatz und der Methodik von Pheromonen bei den Ameisenschwärmen noch unbekannt.

6.2.4 Virtuelle Pheromone

Die Verwendung von Pheromonen bei den Simulations-System der SI einzusetzen geht auf Forschungen von Chialvo und Millonas [Chialvo & Millonas 1995] sowie weiteren Studien von Rauch [Rauch 1999] zurück. Der bei Ameisenschwärmen beobachtete Einsatz

von Pheromonen zum effizienteren Finden des Weges sollte somit auch in den virtuellen Simulationen nutzbar werden.
Die Simulationen nach Chialvo und Millonas [Chialvo & Millonas 1995] waren folgendermaßen aufgebaut: In der gesamten virtuellen Welt der Agenten (2D Fläche oder 3D Struktur) können auf jedem Feld Pheromone abgegeben werden und ebenfalls von den virtuellen Agenten erfasst werden. In der Regel beträgt der Radius der erfassten Felder 1, so dass bei einer zweidimensionalen Fläche acht Nachbarfelder und bei 3D-Stukturen insgesamt 26 Nachbarfelder erfasst werden[5]. In jedem Iterationsschritt erfasst der einzelne Agent die Konzentrationen dieser angrenzenden Felder und berechnet die Gewichtung für das Wählen der entsprechenden Richtung. Die Berechnung wird nach Formel 6.7 durchgeführt, die eine durch Chialvo und Millonas vereinfachte Version der Übergangs-/Reaktionsfunktion darstellt die ursprünglich gebildet wurde, um das Verhalten der Wegwahl von Ameisen möglichst exakt in einer Simulation nachzubilden.

$$W(\sigma) = \left(1 + \frac{\sigma}{1 + \gamma\sigma}\right)^{\beta} \tag{6.7}$$

In dieser Formel stellt die Variable β die osmotropotaxische Sensitivität dar (in späteren Publikationen auch als „inverse noise parameter“ oder „gain“ bezeichnet). Dies ist eine Gewichtung des Einflusses unterschiedlicher Pheromonkonzentrationen auf den Wert von $W(\sigma)$. Hat β einen hohen Wert so haben weit auseinander liegende Konzentrationen einen hohen Einfluss auf die Wegwahl, ist β dagegen klein so wirken sich diese stark unterschiedlichen Konzentrationen nicht so sehr darauf aus. Üblicher Wert von β in den Simulationen ist 3.5.
Die Kapazitätsschwelle des Sensors für Pheromone wird durch $\frac{1}{\gamma}$ angegeben und steht für die natürliche Sättigung des Sensors bei besonders hohen Konzentrationen, so dass ab dieser Schwelle Unterscheidungen kaum noch getroffen werden können.
Weitere Erklärungen und Diagramme zu Einflüssen der Wahl von β und γ stehen unter [Chialvo & Millonas 1995], [Rauch 1999] sowie in der VR-Seminar Ausarbeitung [Marquardt 2003].

Richtungswechsel
Weiterhin wird in einem Gewichtungsfaktor berücksichtigt, daß es am wahrscheinlichsten ist dass der Agent die Richtung wählt, die in einem möglichst geringen Winkel von seiner Ursprungsrichtung abweicht. Am unwahrscheinlichsten sind Drehungen um 180 Grad und bekommen damit den niedrigsten Wert für $w(\delta\theta)$. Im zweidimensionalen gibt es deshalb fünf Werte für die Belegung von $w(\delta\theta)$: von 1 für geradeaus bis $\frac{1}{20}$ für die 180 Grad Drehung (Abbildung 6.4).

[5] die Auswirkungen einer Erweiterung des Erfassungsradius wurde bereits in Kapitel 3 näher erläutert

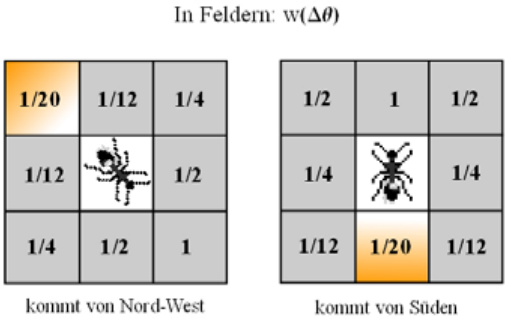

Abbildung 6.4: Verschiedene Gewichtungsfaktoren für Drehungen der Ameisen (nach [Ramos and Merelo 2002])

Für die Berechnung dieses Faktors im dreidimensionalen Raum könnten zum einen zwei Ebenen mit den entsprechenden Winkeln eingeführt werden, eine Belegungstabelle in 3D angelegt werden oder eine Gewichtungsberechnung mit Winkeldifferenzen und der e-Funktion ausgehend von der ursprünglichen Richtung des Agenten berechnet werden. Dieses letzte der möglichen Verfahren wurde bereits in Kapitel 3 erläutert, und wurde in dem folgenden entwickelten 3D-System ebenfalls verwendet.
Schließlich wird eine Funktion für den Wahrscheinlichkeit des Wechsels eines Agenten vom Feld k zu Feld i angegeben. Mit Hilfe von Formel 6.8 kann diese Wahrscheinlichkeit berechnet werden: Die Pheromongewichtungsfunktion multipliziert mit der Richtungsänderungs-Gewichtung geteilt durch die Summe aller Übergangswahrscheinlichkeiten der benachbarten Felder.

$$P_{ik} = \frac{W(\sigma_i) \cdot w(\Delta_i)}{\sum_{Neighbours\ j\ of\ k} W(\sigma_j) \cdot w(\Delta_j)} \tag{6.8}$$

Eine Zusammenfassung der Pheromon-Anwendung nach Chialvo et al. [Chialvo & Millonas 1995] [Rauch 1999] ist in Abbildung 6.5 zu sehen:

Lernprozess
Eine weitere Eigenschaft des Pheromonfeldes ist die Abnahme der Konzentration jedes Feldes in jedem Iterationsschritt, wobei damit das Verdampfen der Pheromone in der Natur nachgebildet wird. Diese Verdampfungsrate wird in dem Simulationssystem mit κ bezeichnet und stellt damit auch einen wesentlichen Faktor für die Schnelligkeit des Vergessens im gesamten System dar. Denn ein wesentlicher Anteil der indirekten Kommunikation zwischen allen Agenten wird durch die Spuren der Pheromone ermöglicht.
Auch die Pheromon-Abgabe wird in einer Rate angegeben, wobei die Konstante der in jedem Schritt abgegeben Pheromone mit η bezeichnet wird und es weiterhin zwei dynamische Abgaberaten von Pheromonen gibt. Zum einen ist dies eine erhöhte Abgabe von Pheromonen wenn besonders viele übereinstimmende Objekte innerhalb

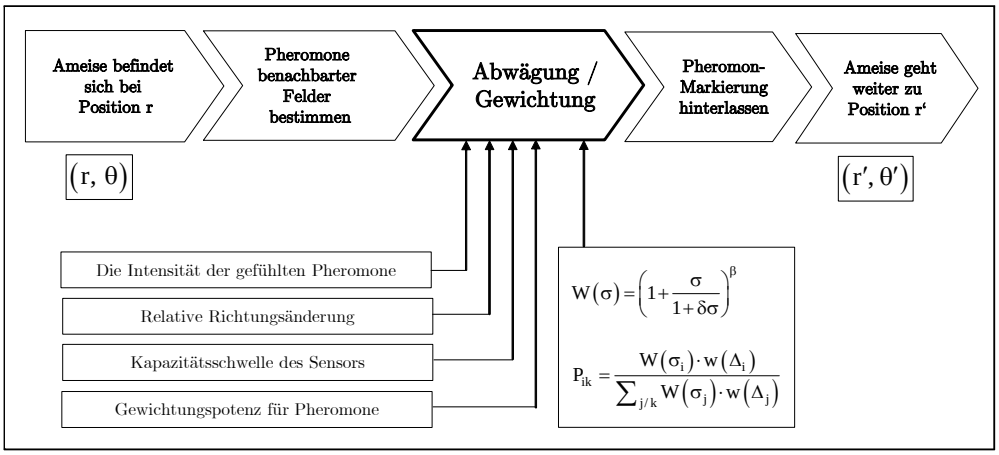

Abbildung 6.5: Übersicht der Weg-Entscheidungen nach Chialvo und Millonas [Marquardt 2003]

der Nachbarschaft gefunden werden. Zum anderen eine weitere dynamische Abgabe falls ein Objekt abgelegt wurde. Mit diesen dynamischen Abgaben der Pheromone wird es möglich, die Wahrscheinlichkeit für die Bildung von gewünschten Cluster-Verbindungen zu erhöhen.

Simulationen und Bildung von Pheromonpfaden

Die Ergebnisse der Experimente des Systems nach Chialvo und Millonas sind in 6.6 zu sehen: von einem inital zufällig verteilten Pheromonfeld führt das emergente Verhalten des gesamten Schwarms zu einer Bildung von Pfaden auf denen sich der größte Teil der Agenten für den zukünftigen Ablauf der Simulation bewegt [Chialvo & Millonas 1995]. *Anmerkung: In den Simulationen nach Chialvo und Millonas werden keine Objekte durch die Agenten manipuliert. Es werden ausschließlich die Bildung von Pheromonpfaden untersucht.*

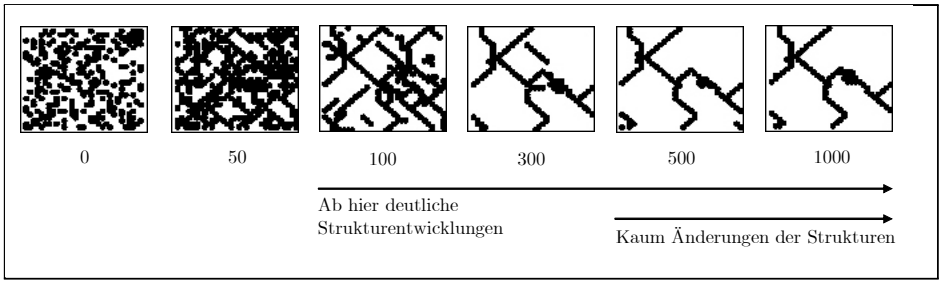

Abbildung 6.6: Entwicklung der Pheromonpfade nach Chialvo und Millonas (mit 32x32 Feldern und 307 Agenten[Chialvo & Millonas 1995]

Die Entwicklung dieser Pfade kann mit dem 3D-Simulationssystem aus diesem Forschungsprojekt ebenfalls beobachtet werden: nach der Initialisierung der Parameter (u.a. κ, η sowie die dynamischen Pheromon-Abgaberaten) für die Variablen der Algorithmen ist die schrittweise Entwicklung in der dreidimensionalen Struktur zu erkennen, wie in 6.7 zu sehen (es wurden hier noch keine Objekte in der Welt angelegt; es erfolgen ausschließlich Bewegungen der Agenten). Durch die transparente Darstellung des Pheromonfeldes wird die Übersicht über die Pheromonlandschaft erleichtert.

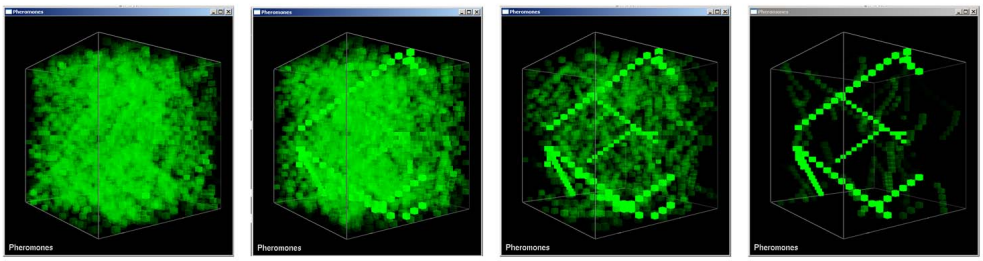

Abbildung 6.7: Entwicklung der Pheromonpfade

6.2.5 Technische Implementierung

Für die Umsetzung des 3D-Simulations-Systems wurde C++ verwendet, für die graphische Visualisierung OpenGL und die GLUT-Library. Weiterhin wurde für das Interface die GLUI-Library [GLUI 2004] verwendet, mit der alle wesentlichen Interaktionselemente angelegt werden konnten. Die einzelnen Bestandteile des Simulations-Systems sind in C++ objektorientiert implementiert, und ermöglichten so während der kontinuierlichen Weiterentwicklung des Programms eine einfache Erweiterung alle Komponenten und einen systematischen Aufbau der einzelnen Bestandteile des Simulations-Systems.
Nachfolgend werden die wichtigsten Komponenten zur Verwendung des Systems in wenigen Schritten erläutert sowie Anwendungshinweise gegeben. Dies erfolgt Anhand der Erstellung einer „minimalen“ Simulation:

A. Neue Simulation anlegen
In der folgenden `main()` Funktion wird eine neue Instanz des `GLUTMasters` angelegt, der sich um die Verwaltung aller OpenGL Fenster kümmert. Anschließend wird die Instanz des Simulations Singletons angefordert (mit dem `GLUTMaster` als Parameter, da zahlreiche GLUT Fenster angelegt werden). Nach dieser Initalisierung wird die GLUT `MainLoop` aufgerufen, in der alle OpenGL Ereignisse verarbeitet werden.

```
1  void main(void){
2    // Instance of the glutmaster for handling GLUT windows
3    myGlutMaster    = new GlutMaster();
```

```
4
5     // Instance of swarm simulation (the swarm simulation
6     // is implemented as singleton)
7     Simulation mySimulation  =
8       Simulation::GetInstance(myGlutMaster);
9
10    // Enter the GLUT main loop
11    myGlutMaster->CallGlutMainLoop();
12  }
```

B. OpenGL Ausgabefenster

Im Simulations-Objekt werden die OpenGL Fenster für die Anzeige der Welt und der Pheromone angelegt, sowie die drei Graphen mit jeweils drei Graphenlinien, mit deren Hilfe die Evaluierungsberechnungen als Graph ausgegeben werden können. Für jedes der Graph Fenster werden anschließend noch die drei Linien initialisiert. Schließlich folgt noch die Erstellung der GLUI-Controls: der Initialisierungs-Dialog (Weltgröße usw.) sowie die Laufzeit-Parameter-Einstellungen.

```
1     [...]
2     // Create the opengl windows for pheromones and the world
3     myPheromoneDisplay_    = new DisplayPheromones(glutMaster, this);
4     myWorldDisplay_        = new DisplayWorld(glutMaster, this);
5
6     // Create the graph painters
7     myGraph1_        = new GraphPainter(glutMaster);
8     myGraph2_        = new GraphPainter(glutMaster);
9     myGraph3_        = new GraphPainter(glutMaster);
10
11    // Initialize the graph lines of GraphPainter 1
12    myGraph1_->DefineLine(1,"Packed Objects",     1.0, 0.2, 0.0);
13    myGraph1_->DefineLine(2,"Average Pheromones", 1.0, 0.6, 0.0);
14    [...]
15
16    // create the glui controls
17    this->CreateInitDialog();
18    this->CreateControls();
```

C. Welt und Schwarm

Sobald der Nutzer den Initialisierungsdialog bestätigt hat, können mit den dort eingegebenen Werten sowohl die Welt (mit den geforderten Dimensionen) als auch der Schwarm angelegt werden. Der Schwarm erhält als Parameter einen Pointer auf die initialisierte Welt (dies ist somit die gültige Welt dieses Schwarms).

```
1   void Simulation::CreateAntWorld(){
2     // Hide the glui window with the world creation controls
3     gluiInitDialog_->hide();
4
5     // Create the world, set to the world-pointer of simulation
6     // use the dimensions from the interface
7     myWorld_ = new World(dimensionX_, dimensionY_, dimensionZ_);
```

```
8
9     // Create a swarm, and give the swarm the reference to
10    // the created world
11    myAntSwarm_ = new AntSwarm(myWorld_);
12    [...]
13  }
```

D. Iterationen

Von der Iteration-Methode der Simulation wird nun für jeden Iterationsschritt `myAntSwarm_->Iterate();` aufgerufen. In dieser Methode wird über den Vektor der Agenten eines Schwarms iteriert. Nach den Iterations-Schritten der Agenten werden die ausgesonderten Pheromone vermindert (Evaporation).

```
1  void AntSwarm::Iterate(void) {
2    // Iterator for swarm
3    std::vector<Ant *>::iterator swarmIterator;
4
5    // Iteration step for all registered agents
6    for (swarmIterator = swarm_.begin();
7          swarmIterator != swarm_.end();
8          ++swarmIterator) {
9      (*swarmIterator)->Iterate();
10   }
11
12   // Evaporate the pheromones in the world
13   swarmWorldRef_->EvaporatePheromones();
14 }
```

E. Iterations-Schritt eines Agenten

Betrachten wir nun den Iterations-Schritt der Agenten: Zuerst wird überprüft, ob an der aktuellen Position des Agenten eine Picking- oder Dropping-Decision überhaupt möglich ist. Dafür gibt es zwei Fälle: 1) Der Agent ist unbeladen und ein Objekt befindet sich an der aktuellen Position oder 2) Der Agent ist beladen und er befindet sich auf einem leeren Feld. Sind die jeweiligen Bedingungen erfüllt, werden entsprechend die Picking- oder Dropping-Method aufgerufen und bei einem *true* als Rückgabewert die Aktion zum Aufheben oder Abelgen des Objektes ausgeführt.

```
1  void Ant::Iterate() {
2    // Picking and dropping decisions only in two cases:
3    // ant is carrying object and the current position contains
4    // no object, or ant has no object and there is
5    // a object at the current position
6    if( ((antWorldRef_->GetObjectAt(position_) != NULL) &&
7         (carryObject_ == NULL)) ||
8        ((antWorldRef_->GetObjectAt(position_) == NULL) &&
9         (carryObject_ != NULL)) )
10   {
11     // Update the neighbourhood references
12     this->LookAroundAndUpdate();
13
```

```
14      // 1. case: there is a object at the current position
15      // and our agent carries no object
16      if((antWorldRef_->GetObjectAt(position_) != NULL) &&
17        (carryObject_ == NULL)) {
18        // Pick item, so the ant is now carrying the object
19        if (this->PickingDecision())
20          PickObject(antWorldRef_->PickObjectAt(position_));
21      }
22
23      // 2. case: there is no object at the current position
24      // and our agent carries an object
25      else if((antWorldRef_->GetObjectAt(position_) == NULL) &&
26        (carryObject_ != NULL)) {
27        if(this->DroppingDecision())
28        {
29          antWorldRef_->DropAntObject(carryObject_,position_);
30          dynamicPheromoneForDecision = 0.5;
31        }
32      }
33    }
```

Im letzten Abschnitt des Iterationsdurchlaufs werden die Pheromone auf dem aktuellen Feld des Agenten ausgesondert: Zuerst die Abgabe des statischen Pheromones in jedem Iterationsdurchlauf (`P_ETA`), dann die dynamische Abgabe von Pheromonen entsprechend der Übereinstimmung der Objekte in der Nachbarschaft (mit Gewichtung `P_DPD_SIMIL`) und schließlich eine dynamische Abgabe von Pheromon bei erfolgreicher Ablage eines Objektes (mit Gewichtung `P_DPD_DECISION`).
Die letzte Berechnung des Iterationsschrittes ist die Bestimmung eines neuen Richtungsvektors für den Agenten sowie eine Bewegung in diese Richtung (mit der entsprechenden Schrittweite des Agenten, in der Regel 1 - 6 Felder).

```
1     // Deposit static pheromones
2     antWorldRef_->AddPheromoneAt(parameter_[P_ETA], position_);
3
4     // Add dynamic pheromone for similarity of objects
5     antWorldRef_->AddPheromoneAt((dynamicPheromoneForSimilarity_ *
6       parameter_[P_DPD_SIMIL]), position_);
7
8     // Add dynamic pheromone for a dropping decision
9     antWorldRef_->AddPheromoneAt((dynamicPheromoneForDecision *
10      parameter_[P_DPD_DECISION]), position_);
11
12    // Find a new direction and do a step to that direction
13    this->MoveAntToNextLocation();
14  }
```

6.2.6 Interaktion und Interface

Da mit dem Simulationssystem insbesondere zahlreiche Experimente mit den unterschiedlichen Algorithmen durchgeführt werden sollten war die Entwicklung eines Interface notwendig, mit dem sich die zahlreichen Parameter der Berechnungen übersichtlich verändern lassen. Das Control-Panel (mit dem GLUI Toolkit entworfen) sammelt alle Steuerelemente in Gruppen zusammengefasst. Die folgenden Gruppierungen wurden angelegt:

- **Simulation Config**
  Einstellungen der Welt- und Pheromonansicht, Sichtbarkeit der Agenten, Iterationsschritte ausführen, Refreshrate der OpenGl Fenster und die maximale Iterationsanzahl.

- **Pheromone-Map Settings**
  Es ist möglich, sich durch die einzelnen Ebenen der Pheromon-Karte zu bewegen (nicht im Volume Display). Die Ebenen können mit der Anzeige der Welt-Darstellung verknüpft werden (Match Display Functions) und einzelne Ebenen ein- aund ausgeblendet werden (Display Axis). In der Volumen-Darstellung der Pheromone (Volume Display), in denen die Pheromon-Voxel mit Transparenz dargestellt werden, kann ein Cutoff-Value angegeben werden, so dass alle Pheromon-Werte unterhalb des Cutoff-Values ausgeblendet werden.

- **Agents (Create, Modify)**
  Hier können neue Agenten in der Welt angelegt werden. Zur Verfügung stehen fünf Grundkonfigurationen, sowie die Angabe der maximalen Schrittweite der Agenten.

- **Objects (Parameter-Space)**
  Die vordefinierten Presets der Objekttypen lassen sich abrufen und in der Welt anlegen, sowie die Objekte nach der Gauss-Verteilung (die Parameter-Textdatei wird ausgelesen) können erstellt werden. Es ist hier auch möglich, die Initialisierungsdatei des Gauss-Algorithmus anzugeben.

- **Algorithm Configuration**
  Hier kann die Algorithmus-Auswahl der Agenten geändert werden: Lumer und Faieta, Ramos, Dense Packing Algorithm, Simple Algorithm.

- **Pheromone configuration**
  Alle Parameter für die Berechnungen der Pheromone lassen sich hier modifizieren

- **Picking and dropping**
  Sowohl die Schwellenwerte für den Lumer und Faieta Algorithmus als auch für den

Algorithmus nach Vittorino Ramos lassen sich verändern.

- **Weighting, Wayfinding**
  Für die Wegwahl der Agenten kann der Koeffizient für den Einfluss der Drehung eingestellt werden (je höher der Wert, desto unwahrscheinlicher ist die Drehung des Agenten von seiner Richtung weg; am unwahrscheinlichsten ist die Drehung um 180°).

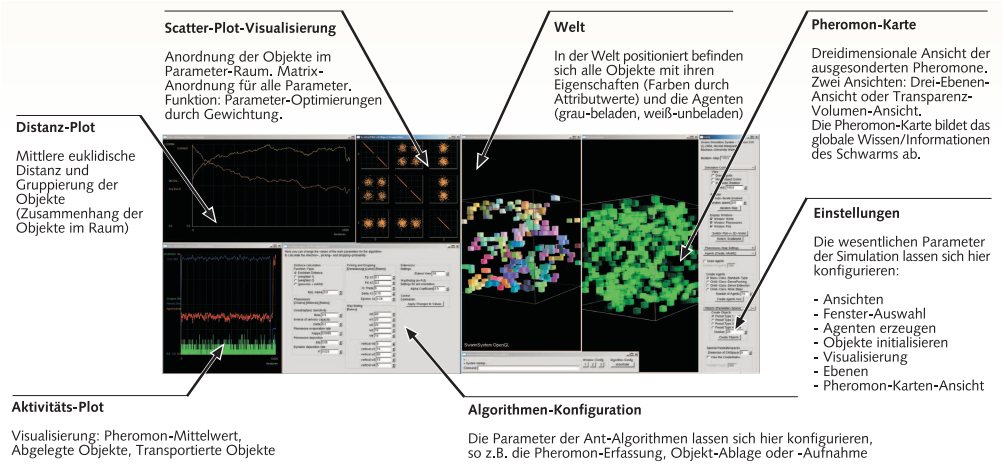

Abbildung 6.8: Interface des 3D-Simulations-Programmes

6.2.7 Daten

- **Testwerte**
  Diese in World.cpp vorgegebenen Testwerte, die sich als Preset abrufen lassen, sind nach verschiedenen Mustern aufgebaut (farbliche Gruppierungen) und bilden einfache Konfigurationen der Objekt-Parameter zu Testzwecken. Im häufigsten Fall werden drei Parameter für Objekte definiert, da sich diese direkt auf RGB-Farbwerte abbilden lassen und somit eine farb-codierte Darstellung der Objekte in der drei dimensionalen Welt ermöglichen.

- **Kugelpunkte**
  Für die Erstellung dieser Textdaten wird eine Kugel im Parameter-Raum erzeugt und anschließend in einem vorgegebenen Raster bei dieser Kugel einzelne Abschnitte (3D-Blöcke) generiert. Auf die drei Parameterwerte jedes einzelnen Blocks werden die Koordinaten der drei Achsen abgebildet, so dass sich hier Farbverläufe der RGB-Werte ergeben. Anschließend werden diese Blöcke zufällig im Raum plaziert. Ziel der Clustering Verfahren bei Anwendung auf diese Datensätze ist eine Gruppierung der Objekte, die möglichst exakt die ursprüngliche Form der Kugel dar-

stellt. Wie ein drei dimensionales Puzzle müssten die Agenten die Objekte zusammenfügen, und dabei zwischen den Kugelbestandteilen die minimalen euklidischen Distanzen ihrer drei Parameter finden (siehe Abbildung 6.9).

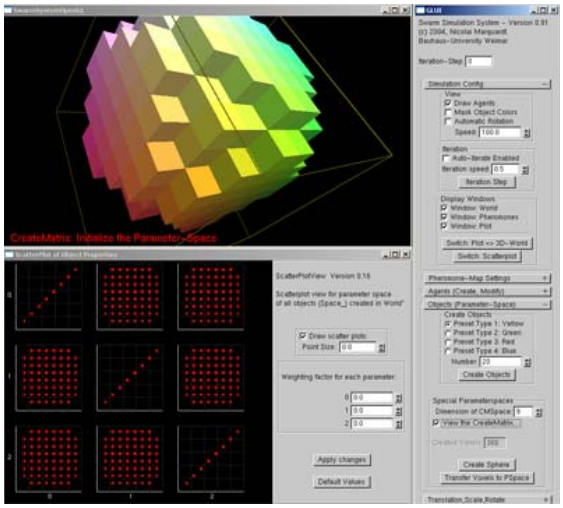

Abbildung 6.9: Scatterplot der Initial-Daten der Farbwert-RGB-Kugel

- **Gauss-Wolken**
  Nach dem Algorithmus der Gaussverteilung werden Punkt-Wolken im n-dimensionalen Raum erzeugt; verwendet wurden insbesondere Verteilungen der Wolken im dreidimensionalen Raum (Abbildung 6.10), der Grund liegt wiederum in der Farbdarstellung als RGB-Werte. Aber auch höher dimensionale Verteilungen der Gauss-Punktwolken wurden zum Clustering eingesetzt, so zu sehen in Abbildung 6.11.

6.3 Simulationen, Tests und Auswertungen

6.3.1 Grundlegendes zu den Simulationen

Mit Hilfe des entwickelten Simulations-Frameworks wurden zahlreiche Tests mit denen im vorangegangenen Kapitel aufgeführten Datensätzen und Algorithmen durchgeführt. Bei der Erstellung einer Simulation ist auf eine geeignete Wahl der zahlreichen Parameter zu achten, da ungünstige Parameter sich negativ auf das Laufzeitverhalten und die Sortierung auswirken (siehe auch [Handl et al. 2003b]). Für den Ablauf der Simulation ist es notwendig, genügend freie Zellen neben den bereits von Objekten

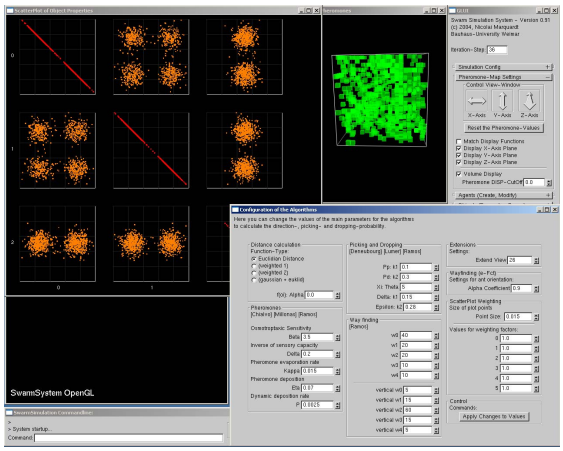

Abbildung 6.10: Die gauss-verteilten Punkt-Wolken

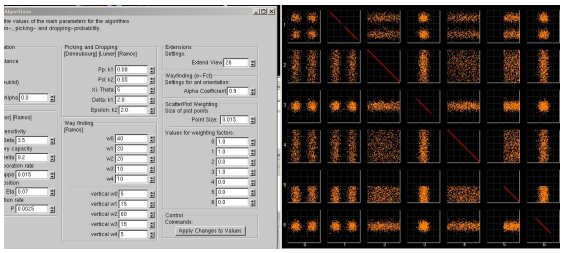

Abbildung 6.11: Höher dimensionale Erstellung der gauss-verteilten Daten

belegten Zellen zu reservieren. Dabei sollte das Verhältnis $r_{occupied} = \frac{N_{items}}{N_{cells}}$ konstant bleiben; wobei sich als optimales Verhältnis herausgestellt hat: $r_{occupied} = \frac{1}{10}$ (nach Erkenntnis von J. Handel [Handl et al. 2003b] und V. Ramos). Dieses Ergebnis konnte durch unsere Experimente bestätigt werden, und gilt sowohl für 2D als auch für 3D (wobei in diesem Fall aufgrund der visuellen Darstellung und der allgemein wesentlich kürzeren Verbindungswege das Verhältnis bis zu $r_{occupied} = \frac{1}{20}$ betragen kann). In den meisten der Simulationen werden Welt-Strukturen mit 8000 Feldern eingesetzt, und dementsprechend $\approx 800 - 1000$ Objekte erstellt. Für die Anzahl der Agenten gilt weiterhin: $\frac{N_{agents}}{N_{items}} = \frac{1}{10}$, so dass i.d.R. ≈ 100 Agenten eingesetzt werden.

6.3.2 Simulation: Gauss-verteilte Daten

Für die Simulation der gauss-verteilten Daten werden im Parameter-Raum vier Punktwolken erzeugt, von denen jede Punktwolke 200 Objekte umfasst. Diese Datensätze wurden bereits in zahlreichen vorhergehenden Experimenten von V. Ramos et al.

[Ramos et al. 1999], Lumer und Faieta [Lumer & Faieta 1994] sowie einigen anderen eingesetzt; eine dieser Verteilungen ist in Abbildung 6.12 zu sehen.

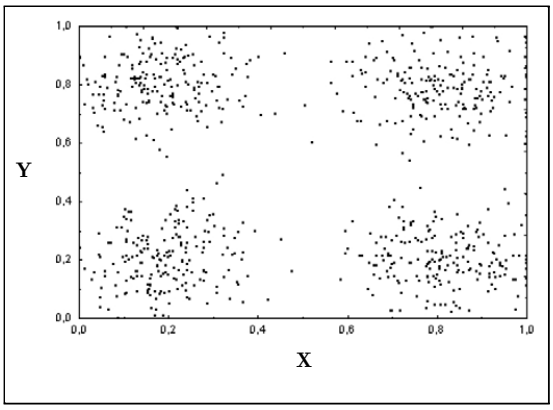

Abbildung 6.12: Punkteverteilung der Testdaten nach Gauss-Algorithmus, 800 Elemente. Grafik von V. Ramos et al. [Ramos et al. 1999]

In Anwendung des Lumer & Faieta Algorithmus ist es möglich, bereits nach 50.000 Iterationen eine Aufteilung in exakt die vier ursprünglichen Cluster zu erreichen (siehe Abbildung 6.13). Dies findet unter Verwendung der L&F-Berechnung in Kombination mit der Erweiterung der unterschiedlichen Laufweiten der Agenten statt (Schrittweite bis zu 6 Felder). Das von Lumer & Faieta empfohlene Kurzzeitgedächtnis der letzten abgelegten Objekte und Verhaltensänderungen (Modifikation der Schwellen-Parameter) sind dabei interessanter Weise nicht notwendig gewesen.

Mit dem Algorithmus von V. Ramos et al. [Ramos & Abraham 2003] ist eine Anwendung auf diesen Datensatz ebenfalls möglich, die Ergebnisse von bis zu 10^6 Iterationen sind in Abbildung 6.14 zu sehen. Mit den 3D-Experimenten wurden die Simulationen ebenfalls mit diesem Algorithmus durchgeführt, es waren jedoch keine Ergebnisse bis zur Aufteilung in 4 Cluster zu erreichen. Prinzipiell erwies sich die Anwendung des Ramos-Algorithmus als äußerst Anfällig von der Wahl der Parametern, insbesondere der für die Pheromon-Umgebung (u.a. δ, β, γ, p, κ).

Stadien eines Simulations-Ablaufs:
Die Durchläufe der Simulationen lassen Reglmäßigkeiten und Muster im Verlauf der Clusterung erkennen. Eine Zusammenfassung der wichtigen Abschnitte ist in Abbildung 6.15 zu sehen, jeweils zu den Zeitpunkten nach 5.000, 25.000 und 50.000 Iterationen (Weltgröße: 8000, Objekte: 800, Agenten: 100).
Anmerkungen zu den Grafiken:

1. Aktivitäten der Agenten: Beladene Agenten (rot), Pheromon-Dichte (orange) und

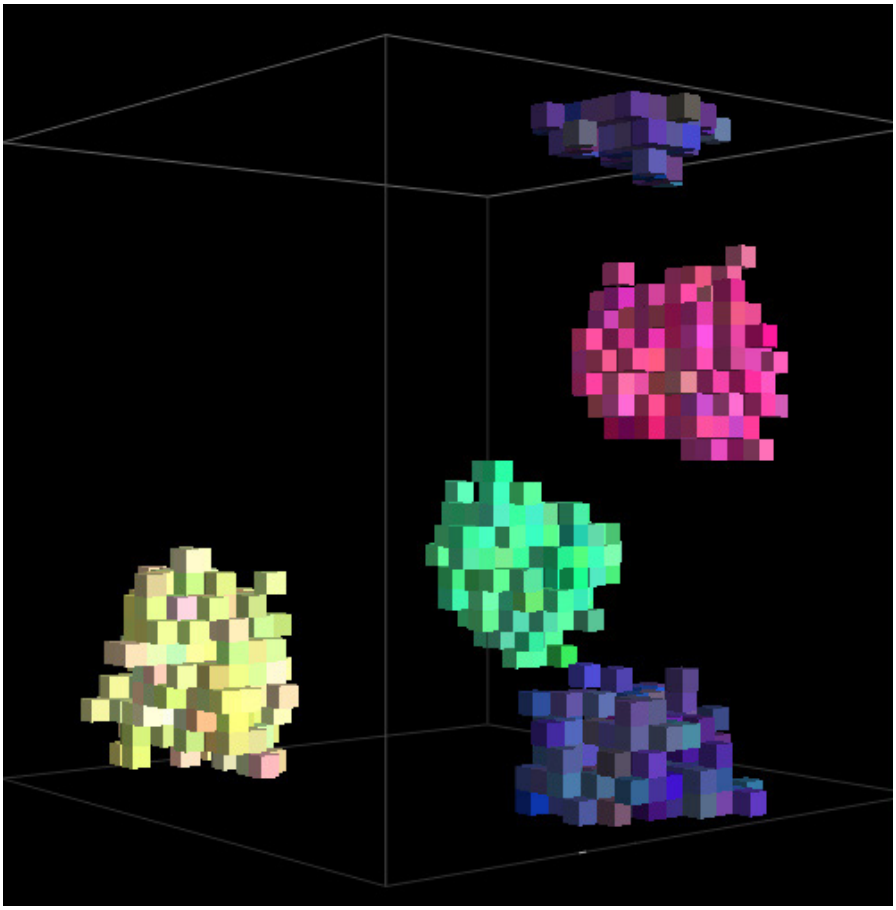

Abbildung 6.13: Simulation nach Algorithmus von Lumer und Faieta. Aufteilung in 4 Cluster bereits nach 50.000 Iterationen

abgelegte Objekte (gelb). Zu erkennen ist der rasche Anstieg der Pheromon-Dichte, bis sich dieser Wert auf einem konstanten Level einpendelt. Der Wert der beladenen Agenten (rot) steigt zu Beginn sprungartig auf die maximale Anzahl (nahezu alle Agenten nehmen das erste angetroffene Objekt), und sinkt innerhalb der ersten 5.000 Iterationen auf etwa die häfte der Agenten. Die Ablage der Agenten steigt innerhalb der ersten 2.000 Iterationen steil an.

2. In dieser Abbildung ist zu erkennen, wie die Aktivität des Ablegens durch die Agenten nach dem großen Anstieg zu Beginn der Simulation sich stetig um ein geringes Maß verkleinert.

3. Während den ersten 2.000 - 4.000 Iterationsschritten finden die meisten Aktivitäten zur Erhöhung des Sortierungswertes statt (Graph: hellblau und hellgrün). Der maximal erreichbare Sortierungsgrad ist insbesondere vom Wert α abhängig (i.d.R.

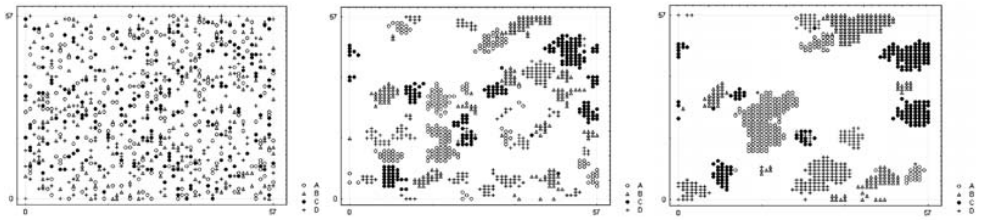

Abbildung 6.14: Ergebnisse der Clusterung nach Vitorino Ramos (gauss-verteilte Daten)

$\alpha = 0.15$). Ist α jedoch zu niedrig eingestellt, sinkt die Aktivität der Agenten drastisch und es werden wesentlich längere Simulations-Laufzeiten benötigt.

4. In dieser Abbildung wurde der Verlauf des Zusammenhalts der Objekte (Connectivity, bzw. average Neighbours) unterteilt: der größte Anstieg findet zeitgleich mit dem Anstieg der Sortierung innerhalb der ersten 2.000 - 4.000 Iterationen statt. Danach jedoch steigt der Wert des Zusammenhalts weiter, jedoch mit geringerer Steigung (a). Ab etwa 30.000 Iterationsschritten findet kaum noch eine Verbesserung des Zusammenhalts der Objekte statt. In den Abbildungen der Simulation ganz rechts ist zu sehen, dass sehr schnell ein Zusammenschluss zu kleinen Clustern stattfindet, die Gruppierung zu den finalen 4 Clustern jedoch noch sehr viele Iterationsschritte benötigt.

6.3.3 Simulation: Sphere-Clustering

Auch der Zusammenbau der in Voxel zerlegten 3D-Kugel wurde mit dem 3D-Framework durchgeführt. Dazu wurden die rund 1000 Voxel der Kugel im 3D-Raum zufällig verteilt und die Simulation mit 100 Agenten gestartet. Es sind für erste sichtbare Ergebnisse wesentlich mehr Iterationen notwendig als bei den gauss-verteilten Daten. Dennoch ist in den Grafiken der Abbildung 6.16 zu erkennen, wie in den bis zu 150.000 Iterationen ein schrittweiser Aufbau einzelner Teile der 3D-Kugel stattfindet. Der Status nach der letzten Iteration ist rechts in der Abbildung vergrößert dargestellt, wobei sich hier die gebildeten Farbverlaufs-Flächen besser erkennen lassen (dies sind die zwei beschriebenen Cluster).
Eine vollständige Rekonstruktion wurde mit diesen Algorithmen nicht möglich, eine bessere Sortierung als die der abgebildeten Flächen war nicht möglich. Dies liegt sicher zum einen daran, dass der Algorithmus nie terminiert, sondern durch die integrierten Zufallsvariablen stetig weitere Objekte aufgehoben und abgelegt werden, zum anderen aber sich auch an der großen Stabilität einmal gebildeter Cluster (und damit einer erschwerten Auflösung zusammenhängender Cluster-Flächen).

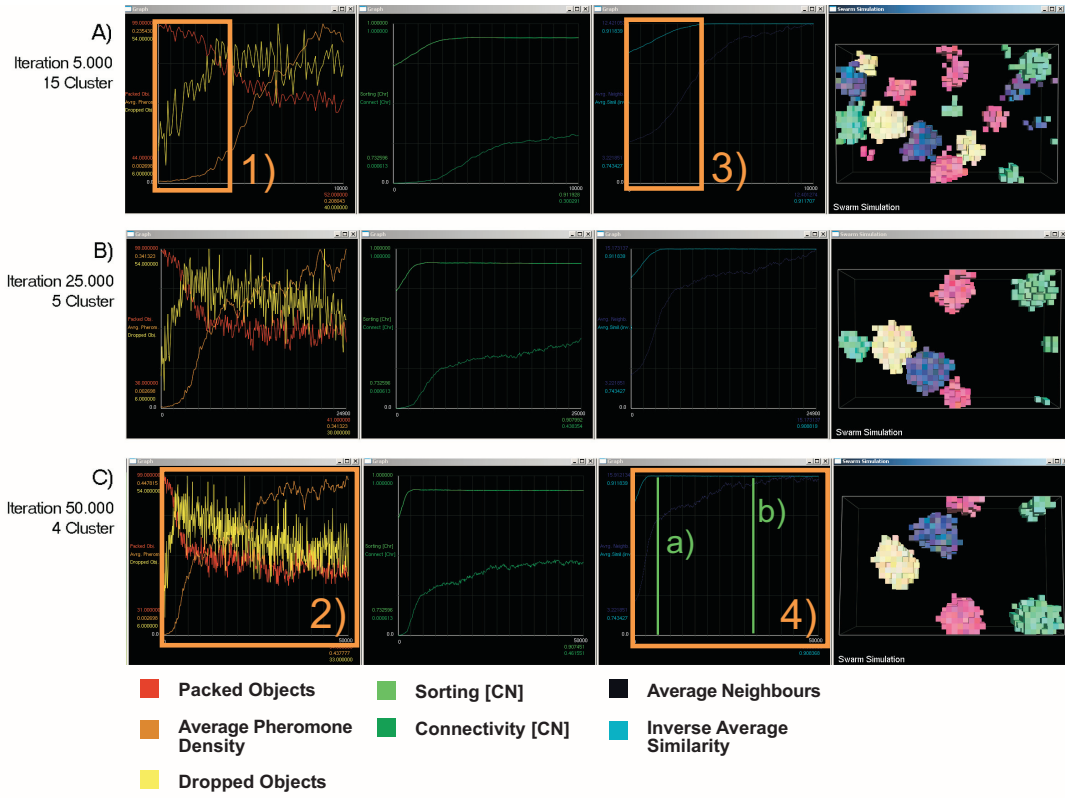

Abbildung 6.15: Auswertung der Simulations-Etappen.

6.3.4 Abstimmung der Pheromon-Anwendung

Die Bildung der Pheromon-Bahnen beeinflusst insbesondere die Wahl der Agenten für die einzuschlagende Richtung. Die ausgesonderten Pheromon-Konzentrationen sollen ein Geflecht von Wegen und Pfaden durch den Worldspace ermöglichen, das vor allem die größten Cluster verbindet.

Die Wahl der geeigneten Parameter gestaltet sich sehr schwierig, denn es haben sich einige nachteilige Auswirkungen falscher Parameter-Wahl während den Simulations-Anwendungen herausgestellt. So führt ein zu hoher Wert der osmotropotaxischen Sensitivität ($\beta > 4.5$) dazu, dass schon nach wenigen Iterationsdurchläufen die Bildung von Pheromonpfaden stattfindet, ohne dass sich bereits Cluster-Häufungen gebildet haben. Alle Agenten bewegen sich nun mit sehr hohen Wahrschinlichkeiten auf den Pheromonbahnen und nur zu sehr geringem Teil abseits der gebildeten Wege. Dies führt dazu, dass es wesentlich mehr Zeit benötigt, bis noch zu gruppierende Objekte

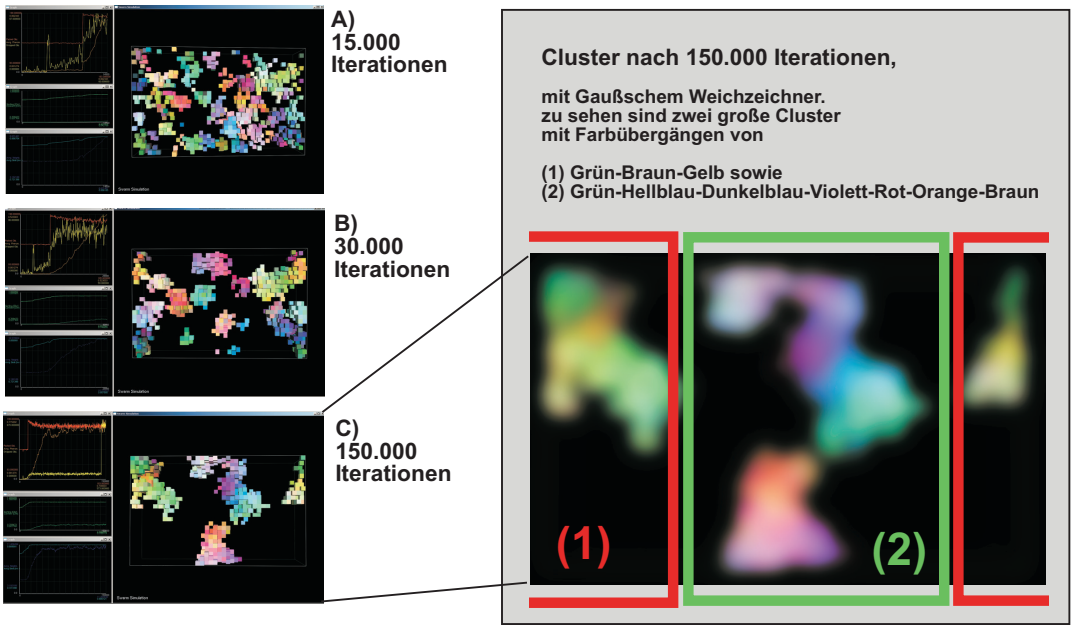

Abbildung 6.16: Ergebnisse der Simulation zur Clusterung einer in Voxel zerlegten 3D-Kugel)

überhaupt von den Agenten erfasst werden. Der Zeitpunkt einer Pfadbildung ist insgesamt sehr schwer zu bestimmen, und hängt neben den Variablenwerten auch dem bis zu diesem Zeitpunkt gebildeten Pheromonfeld ab. Eine Konfiguration der Variablen zur Laufzeit ist zwar möglich, jedoch ist der Einfluß kleiner Änderungen schwer einzuschätzen und kann im schlechtesten Fall eine Zerstörung des bis dahin gebildeten Pheromonnetzes bedeuten (wie beispielsweise die Wahl der Verdampfungsrate κ, der Aussonderung η oder auch des Wertes der osmtropotaxischen Sensitivität β unter 2.0).

6.3.5 Suboptimale Lösungen

Eines bei vielen der Testreihen aufgetretenes Problem ist die Bildung von nicht-optimalen Clustern. So gibt es sehr häufig Zusammenschlüsse von Mini-Clustern, die nahe der Start-Phase des Systems gebildet werden und nur nach vielen Iterationsschritten wieder aufgelöst werden. Häufig sind somit in ihren gruppierten Objekt-Daten gleiche Cluster über den Worldspace verteilt. Als Beispiel sind in Abbildung 6.19 die Gruppierungen von vier Objektarten in einige kleinere Cluster abgebildet (frühes Simulationsstadium). Cluster der Größenordnung von bis zu 30 Objekten werden sehr schnell durch den Zufallsvariablen-Einfluss bei der Picking-Entscheidung aufgelöst; sobald die Cluster je-

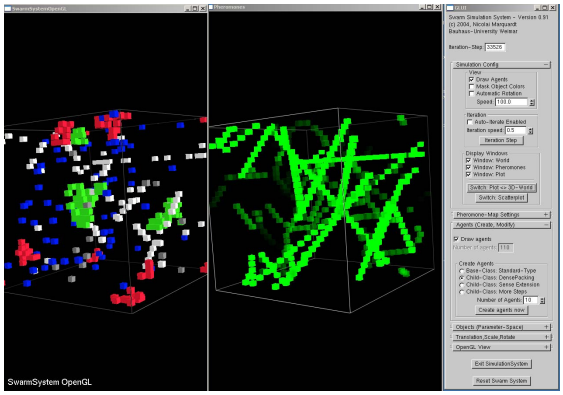

Abbildung 6.17: Pheromonpfade verbinden insbesondere Bereiche mit hoher Objektdichte.

doch mehr Objekte umfassen, findet eine solche Auflösung der Cluster erst bei sehr langen Iterationszeiten statt, da während dem Abbau des Clusters auch gleichzeitig wieder Elemente abgelegt werden.
Auch wenn viele der Mini-Cluster nach vielen Iterationen aufgelöst werden können, ist das große Problem dabei sicher der Verlauf der gesammten Clusterung: zu Beginn kann das System (Algorithmus nach V. Ramos) sehr schnell die sich ähnlichen Objekte gruppieren (mittlere euklidische Distanz sinkt schnell und Gruppierungs-Wert steigt), jedoch wird nach etwa 25.000 Iterationen die mittlere euklidische Distanz nicht wesentlich weiter verbessert, während jedoch die Gruppierung der Objekte durch die Auflösung vieler der kleinen Cluster schrittweise erhöht werden kann.

6.3.6 Variierende Schrittweiten

Agenten in der Grundkonfiguration bewegen sich immer mit der Schrittweite von einem Feld. Werden nun Agenten mit größeren Schrittweiten eingesetzt, so bewegen sich diese während den Iterationen direkt aus kleinen Clustern heraus; sie überspringen diese quasi. Schrittweiten von bis zu 10 Feldern wurden in Experimenten verwendet und immer gemeinsam mit etwa gleichvielen Agenten der Standard-Schrittweite eingesetzt (oder die verschiedenen Schrittweiten in Gruppen aufgeteilt). Auch eine Gleichvertielung der Schrittweiten in einem Intervall wurde verwendet, so dass zu jeder der möglichen Schrittweiten etwa gleichviele Agenten in der Welt agieren.
Der Einsatz unterschiedlicher Schrittweiten in Verbindung mit dem Verfahren nach Lumer und Faieta [Lumer & Faieta 1994] führt zu einer schnelleren Formierung der Cluster, wie dies auch schon bereits in [Handl et al. 2003a] festgestellt wurde. Die schnelleren Agenten haben den Vorteil, schneller aus den Minimal-Clustern herauszuspringen, was nach der Aufnahme eines Objektes aus dem Cluster den Vorteil bietet, sich unmittelbar nach dem Iterationsschritt bereits außerhalb der Felder zu

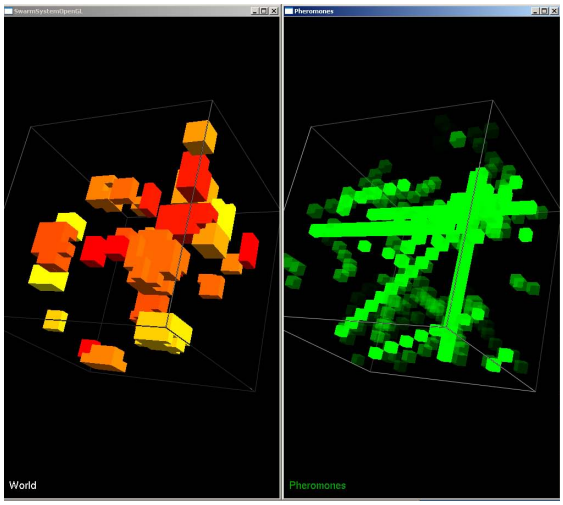

Abbildung 6.18: Bildung der Pfade beim gleichzeitgen Anwenden der Clustering Algorithmen. Führen vereinzelt zu suboptimalen Lösungen der Simulation.

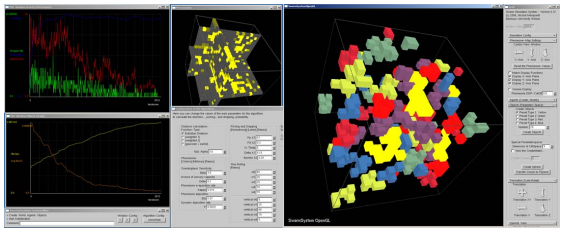

Abbildung 6.19: Bildung der kleinen Cluster-Gruppen

befinden, bei denen Aufgrund der minimalen euklidischen Distanz eine hohe Ablege-Wahrscheinlichkeit aufgetreten wäre.
Im folgenden einige der Daten aus den Experimenten zum Einsatz der variierenden Schrittweiten die eine leichte Verbesserung dieser Erweiterung belegen. Dabei wurden jeweils die Mittelwerte aus zehn Durchläufen gebildet, mit der folgenden Konfiguration: Weltgröße (8000 Felder): 40x20x10, 100.000 Iterationsschritte, 800 Objekte (Gauss Verteilung), 80 Agenten, Parameter: $\alpha \equiv 0.05, \beta \equiv 1.75, k_1 = 0.1, k_2 = 0.15$

6.3.7 3D-Visualisierungen

Wie in den vorherigen Abschnitten ausführlich erläutert bietet das Simulationssystem die zwei- und dreidimensionale Darstellung sowohl der Welt-Umgebung als auch des Pheromonfeldes. Auch wenn wir die beschriebene Beschleunigung des Clusterings bei dreidimensionalen Räumen beobachten konnten gestaltet sich deren Visualisierung doch

Simulationsergebnisse, je Simulation 10 Durchläufe und Mittelwertbildung			
max. Schrittweite	1	3	6
Sorting	0.907	0.905	0.918
Connectivity	0.446	0.438	0.470
Average Neighbours	15.35	15.19	16.11
Average Similarity of Objects (inv.)	0.908	0.908	0.911
No. of Cluster	6.39	6.12	4.71

Tabelle 6.1: Ergebnisse nach Simulationen mit Agenten unterschiedlicher Schrittweiten.

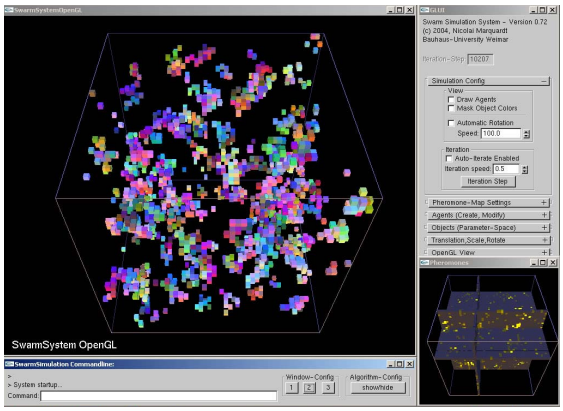

Abbildung 6.20: Worldspace von 80x80x80

sehr schwierig.
Das sicher größte Problem ist die Verdeckung von Bereichen der Simulation, die es mitunter unmöglich macht, dem Simulationsverlauf zu folgen oder auch die exakte Gruppierung der Objekte zu erkennen. Auch die Exploration der Pheromonkarte gestaltet sich schwierig, auch wenn die Anzeige von Layern oder die Transparenz-Darstellung eine Möglichkeit bieten, auch hier einen Überblick zu erhalten. Eine Durchsicht der Ebenen gestaltet sich wesentlich schwieriger als eine Übersicht der 2D-Karte. Nur sehr schwer lassen sich Objekte lokalisieren, die innerhalb der 3D-Strukturen eingeschlossen sind. Somit wird es bei de Clusterung komplexerer Eingangsdaten als unserer Testwerte schwierig, die Simulation zu verfolgen sowie das Ergebnis zu bewerten.

6.3.8 Aufbau der Welt-Dimensionen

Mit der Konfiguration aller drei Dimensionen zu einer beliebigen Form der virtuellen Welt stellte sich auch die Frage, ob der Aufbau der Welt eine Auswirkung auf den Verlauf der Simulation hat und wie die entsprechenden Formen Auswirkungen auf

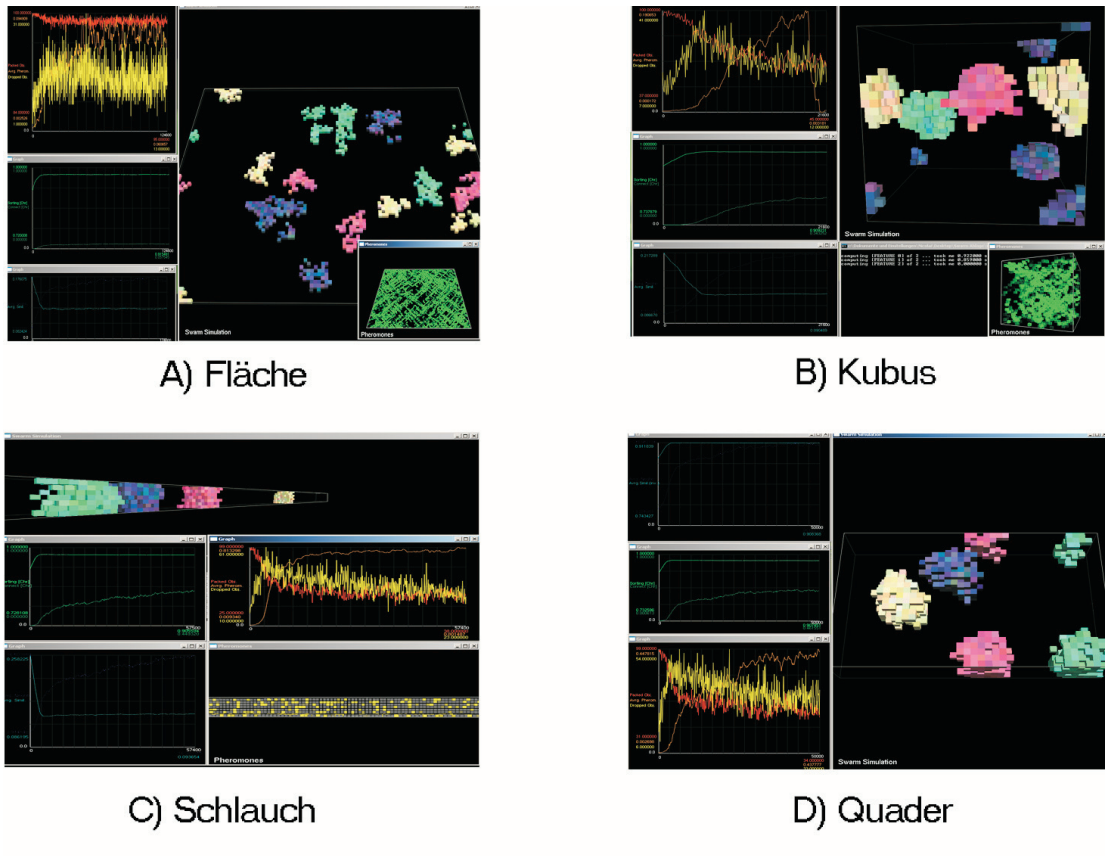

Abbildung 6.21: Anwendung verschiedener Dimensions-Größen für den Aufbau der Virtuellen Welt.

die Übersicht über die Simulation haben. Mögliche Strkturen dabei sind u.a. die zweidimensionale Fläche (wie sie auch in früheren Forschungen eingesetzt wird, u.a. Handel, Ramos, Lumer), 3D Kubus (gleiche Kantenlänge), schlauchartige Struktur oder auch eine quaderförmige Struktur.
Als Einschränkung für die Erfassung der Agenten in ihrer Umwelt war es notwendig, bei Welt-Dimensionen mit einer Größe kleiner als 3 die Erfassung auf zwei Dimensionen zu beschränken. Wegen der toroiden Struktur der Welt würden die Agenten sonst mehrfach das gleiche Objekt erfassen und es würden sich somit falsche Berechnungen für euklidische Distanzen ihres Nachbarschaftsbereiches ergeben.
Nachfolgend wurde mit vier möglichen Welt-Strukturen Simulationen bei gleicher Parameter-Wahl durchgeführt. Die Welt-Dimensionen der getesteten Strukturen ergeben immer die Größe von insgesamt 8000 Zellen (bis auf eine Ausnahme von 8036 Feldern), Testdaten sind die 4 Gauss-Punktwolken mit je 200 Objekten und somit 800 Objekte insgesamt. Für jede der möglichen Welt-Strukturen wurden zehn Simulationsdurchläufe mit jeweils 50.000 Iterationsschritten berechnet und anschließend der Mittelwert der Ergebnisse berechnet. Weitere Parameter der Simulation:

$Agenten = 100$, $\beta = 3.5$, $\delta = 0.2$, $k_1 = 0.1$, $k_2 = 0.15$.

Simulationsergebnisse, je Struktur 10 Durchläufe und Mittelwertbildung				
	A	B	C	D
Struktur	Fläche	Kubus	Schlauch	Quader
Dimension	80x100x1	20x20x20	7x7x164	40x25x8
Felder	8000	8000	8036	8000
Sorting	0.914	0.907	0.906	0.907
Connectivity	0.590	0.4251	0.428	0.455
Average Neighbours	5.071	14.98	16.79	15.33
Average Similarity of Objects (inv.)	0.916	0.909	0.908	0.909
No. of Cluster	11.25	4.42	5.30	4.89

Tabelle 6.2: Ergebnisse nach Simulationen in verschiedenen Strukturen der virtuellen Welt.

Anmerkungen:
Die Berechnungen der Algorithmen wurden für zwei- und dreidimensionale Strukturen verwendet. Dabei sind einige Merkmale bei den Berechnungen zu erkennen:

1. Der Werte der Connectivity ist bei der 2D Fläche signifikant höher als bei den drei Volumen-Strukturen. Dies liegt an der geringeren Oberfläche/Ränder, die die Strukturen im zweidimensionalen gegenüber dem dreidimensionalen besitzen. Festgestellt werden konnte bei den 3D-Strukturen auch Lücken innerhalb der Formen, die bei 2D-Anordnungen wesentlich seltener auftreten.

2. Der Durchschnittliche Wert der Nachbarn ist eine absolute Angabe der Nachbarn; in dem Wert der Connectivity wurde mit der jeweiligen maximalen Anzahl der Nachbarn normiert. Auffällig jedoch ist die höchste Anzahl der Nachbarn bei der Schlauch-Struktur, die jedoch auch gleichzeitig eine höhere durchschnittliche Anzahl der Cluster aufweist (es liegen mehr Cluster entlang der Struktur vor; die Agenten haben hier im schlechtesten Fall den längsten Weg zwischen zwei Punkten der Struktur).

3. Die Anzahl der gebildeten Cluster liegt bei der Kubus-Struktur am niedrigsten, was sicher damit zusammenhängt dass hier die Agenten die kürzesten Wege (aufgrund der Randbedingungen) zurücklegen müssen. Die Diagonale beträgt im „worst case“ 20 Felder, bei der Schlauch-Struktur dagegen 164 Felder. Die Kubus-Struktur hat jedoch den Nachteil der größten Schwierigkeiten durch Verdeckung.

4. Alle der 3D-Strukturen erreichen wesentlich schneller eine Anordnung der gauss-verteilten Objekte in ihrer ursprünglichen Gruppierung zu 4 Punktwolken als die 2D-Fläche. Die 3D-Strukturen bieten den Agenten mehr Möglichkeiten, sich innerhalb der virtuellen Welt zu bewegen und die Objekte zu Clustern zusammenzufügen.

5. Die Form als Quader verbindet mehrere Vorteile: durch ihre 3D-Struktur wird eine Clusterung in 4 Gruppen ähnlich schnell wie bei den anderen 3D-Strukturen erreicht. Sie bietet daneben den Vorteil, für eine Exploration der 3D-Welt eine Dimension mit der Größe 4 zu besitzen, die eine Ebenen-Darstellung sehr gut ermöglicht (wie z.B. mit den Layern bei der Anzeige der Pheromon-Bahnen). Somit ist die Quaderstruktur ein guter Mittelweg zwischen der zweidimensionalen Fläche und dem Kubus.

6.4 Zusammenfassung

Die Implementierung dieses Swarm-Intelligence-Systems ermöglichte es, die zu Grunde liegenden Methoden und Ideen dieser Verfahren zu verstehen und testen zu können. Ein zu Beginn sehr überschaubarer Umfang einzelner Komponenten mit ihren sehr eingeschränkten Interaktionsmöglichkeiten offenbart sich nach näherer Beschäftigung damit als durchaus komplexes System, mit teils unvorherzusehenden Auswirkungen kleiner Veränderungen. Insbesondere die Wahl geeigneter Parameter (und deren vielfältigen Kombinationen) stellte eine großer Herausforderung dar. Aber auch die Erweiterungen und Veränderungen der bestehenden Algorithmen nach Lumer, Faieta, Deneubourg und Ramos erweis sich als spannendes Forschungsfeld.
Die Wahl von OpenGL als Visualisierungskomponente erwies sich als ideal um die dreidimensionale Darstellung der virtuellen Welt mit allen Agenten und Objekten bewerten zu können. Neben einigen bereits erwähnten Vorteilen bereitete jedoch der Umgang mit der 3D-Darstellung auch einige Probleme. Der Test mehrerer möglicher Welt-Dimensionen erwies sich ebenfalls als sehr interessant und zeigte die Möglichkeit, schneller zu Clustering Ergebnissen zu gelangen, auch wenn Verdeckung von Objekten und Gruppen ein Problem darstellten.
Neben den fortwährend modifizierten und neuentwickelten Modulen des C++-Frameworks stellten sicher die zahlreichen Durchläufe und Evaluierungen der Simulationen einen der größten Arbeitsschwerpunkte dieses Forschungsprojektes dar. So bedeutete die Untersuchung der Verfahren und der zahllosen Parameter-Einstellungen doch eine enorme Anzahl der Durchläufe im System mit je bis zu mehreren tausend Iterationen.

7 Schluss

7.1 Fazit

In diesem Projekt konnten wir uns dem Forschungsgebiet der *Swarm Intelligence* nicht nur theoretisch sondern auch praktisch nähern. Es entstanden drei unterschiedliche Implementierungen der Simulations-Systeme, die uns ermöglichten, die Funktionsweise der Systeme besser zu verstehen. Dabei haben wir nicht nur bestehende Systeme nachgebildet, sondern auch eigene Ideen zur Verbesserung mit einbringen können. Es wurden eine Vielzahl von Simulationen durchgeführt und anschließend evaluiert, um Komponenten der System verstehen und beurteilen zu können.
Wir haben festgestellt, dass ein zunächst einfaches System zu einem komplexen System wächst und das Finden der günstigsten Parameter eine große Herausforderung darstellt und maßgeblich zur erfolgreichen Druchführung der Simulationen beiträgt. Viele Erkenntnisse zum Einsatz der unterschiedlichen Algorithmen nach Ramos, Handl, Deneubourg, Dorigo und anderen, wurden in den vorherigen Kapiteln ausführlich erläutert. Wir sehen noch zahlreiche spannende Forschungsfelder im Bereich der *Swarm Intelligence*, die wir im folgenden Abschnitt noch kurz erläutern möchten.

7.2 Ausblick

Bereits in den drei vorherigen Kapiteln der detaillierten Beschreibungen unserer entwickelten *Swarm-Intelligence-Systeme* wurde vereinzelt auf die Überlegungen für zukünftige Entwicklungen der Systeme verwiesen. Nachfolgend möchten wir noch einen zusammenfassenden Überblick über mögliche weitere Forschungsschwerpunkte geben:

- **Strukturen bauen**
  Die Verwendung einer dreidimensionalen Abbildung der Daten wurde bereits in den vorherigen Kapiteln diskutiert. Ein weiterer interesanter Ansatz der bisher nur wenig untersucht wurde ist dabei die Bildung von Objektstrukturen, die sich durch die abgelegten Objekte der Agenten ergeben können. Erst zum Ende der Durchführung dieses Forschungsprojektes stießen wir auf die Idee von Bonabeau et al. [Bonabeau et al. 1994], nach der die Agenten nicht mehr Objekte aufnehmen

und ablegen können (und somit eine Gruppierung der Objekte erreichen), sondern neue Objekte und somit auch Strukturen erzeugen können. Diese Erzeugung richtet sich nach der Erfassung vorhandener Objekte in der Umgebung (aufbauend auf einer initialen Struktur, i.d.R. ein einzelnes Element).
Damit ist es möglich, durch einfach Vorgaben an die Agenten komplexe Strukturen zu entwickeln. Die Vorgaben würden dabei einem Set an Mustern entsprechen (Belegungs-Tabellen der umliegenden Felder), nach denen die Agenten entsprechend neue Teil-Objekte anbauen können. Auch unterschiedliche Arten der generierbaren Objekte sind denkbar.
Zu Beginn enthält der World-Space ausschließlich ein einzelnes Initialobjekt, zu dem die Agenten nacheinander Strukturbauteile anbauen können. Das Prinzip und der Ablauf ist ähnlich dem Vorgehen bei nicht-deterministischen Automaten. Der Aufbau des Regelwerkes ist dabei die große Schwierigkeit, ebenso wie die unterschiedlichen zu vergebenden Wahrscheinlichkeiten für die Anwendung der unterschiedlichen Regeln [Theraulaz & Bonabeau 1995a] [Theraulaz & Bonabeau 1995b]. Zu sehen sind solche Strukturen in den Abbildungen 7.1 und 7.2.

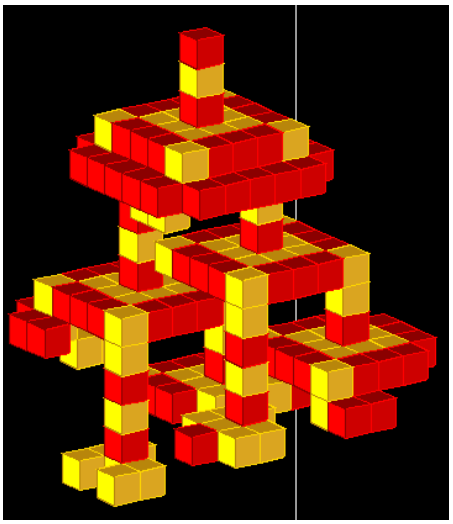

Abbildung 7.1: Bildung von 3D-Strukturen nach [Theraulaz & Bonabeau 1995a]

- **Vergleich von Sorting-Algorithmen**
  Ein Performance-Vergleich zwischen den zahlreichen Ant-Algorithmen und anderen Verfahren hat bisher nur in wenigen Forschungsarbeiten stattgefunden. Eine dieser bisher seltenen Forschungsarbeiten zur Evaluierung ist sicher die Studie von Julia Handl [Handl et al. 2003b]. Allerdings gibt es zur Evaluierung und dem Vergleich der agenten-basierten Verfahren noch viele offene Fragen.
  Wichtig wäre hier sicher, in entsprechenden Performance-Tests den SI-Algorithmen mehrere traditionelle Clusterung-Algorithmen gegenüber zu stellen (z.B. k-Means), ähnlich wie in der Studie von J.Handl. Gleichzeitig sollten die Vor- und Nachteile

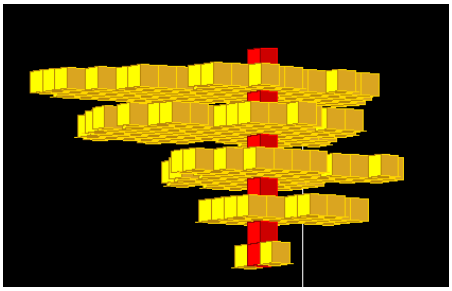

Abbildung 7.2: Eine weitere mögliche Struktur, die sich nach Anwendung der Regeln ergibt [Theraulaz & Bonabeau 1995a]

der Swarm-Systeme gegenüber den anderen Verfahren in den Tests ausgelotet werden, um somit für unterschiedliche Anforderungen und Datenmengen ein ideales und möglichst flexibles Verfahren aussuchen zu können. Dafür wäre ein Selektions-Algorithmus denkbar, der Anhand einer vorhandenen Datenbasis und Bedingungen („continous data“) den am besten anwendbaren Clustering-Algorithmus bestimmt und anwendet.

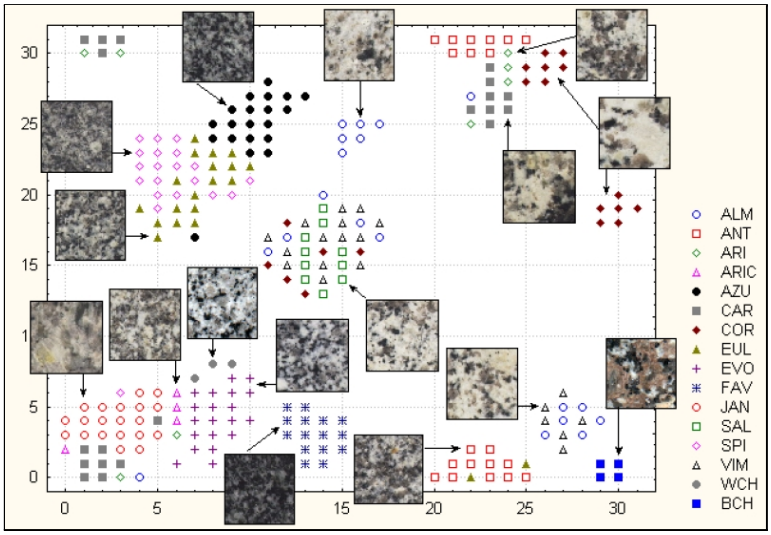

Abbildung 7.3: SI-Algorithmen im Einsatz zur Clusterung von Granit-Sorten [Ramos et al. 2002]

- **Performance-Steigerung**
  In diesem Projekt lag der Schwerpunkt nicht auf der Steigerung der Effizienz der Algorithmen, sondern auf der Untersuchung möglichst vieler verschiedener Verfahren und der Implementierung eines Frameworks. Somit wurde ein Überblick über unterschiedliche Vorgehensweisen bei der Problemlösung mittels SI-Verfahren möglich.

Nach der Implementierung und Untersuchtung der verschiedenen Verfahren ist es jedoch eine Herausforderung, die Algorithmen möglichst performant in einem kompakten Programm zu implementieren um zu erkennen, mit welchen minimalen Berechnungszeiten solch ein SI-System angewendet werden kann. Denkbar wäre auch die bereits angesprochene Implementierung der Verfahren auf GPUs.

- **Daten**
  Eine weitere Frage die nach dem Ende dieses Projektes bestehen bleibt: welche Datensätze eignen sich für das Clustering mit den SI Algorithmen. Zwar haben wir zahlreiche Durchläufe der Simulationen mit generierten Testdatensätzen durchgeführt, allerdings wurden noch keine Echt-Daten eingesetzt. Somit bleibt noch die Herausforderung, welche Datensätze mit welcher Anzahl an Parametern für eine Clusterung und Visualisierung in den entwickelten SI Systemen geeignet sind. Szenarien könnten ähnlich den Anwendungen von V. Ramos, A. Abraham und anderen sein [Ramos et al. 1999] [Abraham & Ramos 2004] [Ramos & Almedia 2000], in denen SI-Algorithmen zur Clusterung von Text-Inhalten, Internet-Nutzungs-Daten oder auch unterschiedlicher Granit-Arten (mit Textur-Vergleich, siehe Abbildung 7.3) [Ramos et al. 2002] eingesetzt wurden.

Dies sind vier mögliche Herausforderungen mit denen sich zukünftige Studierende in Forschungsprojekten befassen können, wobei sicher noch weitere Ergänzungen denkbar sind. Es ergeben sich hier zahlreiche Schwerpunkte für zukünftige Forschungen in dem Bereich der *Swarm Intelligence*.

Literaturverzeichnis

[Abraham & Ramos 2004] A. Abraham, V. Ramos: Web Usage Mining Using Artificial Ant Colony Clustering and Linear Genetic Programming
Department of Computer Schience, Oklahoma State University; 2004, CVRM, Technical University of Lisbon 143

[Abramowitz 1965] M. Abramowitz and I. A. Stegun. Handbook of Mathematical Functions. Dover Publications, Inc., New York, 1965. 1046 pp. 55

[Aron, Deneubourg, Goss and Pasteels 1990] S. Aron, J.-L. Deneubourg, S. Goss, and J. M. Pasteels. Functional Self-Organisation Illustrated by Inter-Nest Traffic in the Argentine Ant Iridomyrmex humilis. In Biological Motion, edited by W. Alt and G. Hoffman, 533-547. Berlin: Springer-Verlag, 1990 19

[Bay 1995] J. S. Bay. Design of the Army-Ant Cooperative Lifting Robot. IEEE Robotics and Automation Mag. 2 (1995): 36-43. 51

[Beckers, Holland and Deneubourg 1994] R. Beckers, O. E. Holland and J.-L. Deneubourg. From Local Actions to Global Tasks: Stigmergy and Collective Robotics. In Artificial Life IV, edited by R. Brooks and P. Maes, 181-189. Cambridge, MA: MIT Press, 1994. 36

[Bilchev and Parmee 1995] G. Bilchev und I. C. Parmee. The Ant Colony Metaphor for Searching Continous Design Spaces. On Proceedings of AISB Workshop on Evolutionary Computing Lecture Notes in Computer Science 993, edited by T. C. Fogarty, 25-39. Berlin: Springer-Verlag, 1995. 23

[Bonabeau, Guérin, Snyers, Kuntz, Théraulaz and Cogne 1998] E. Bonabeau, S. Guérin, D. Snyers, P. Kuntz, G. Théraulaz and F. Cogne. Complex Three-Dimensional Architectures Grown by Simple Agents: An Exploration with a Genetic Algorithm. Evol. Comp. (1998): submitted. 49

[Bonabeau, Dorigo and Theraulaz 1999] E. Bonabeau, M. Dorigo and G. Theraulaz.

Swarm Intelligence - From Natural to Artificial Systems, Oxford University Press, 1999, New York 9, 17, 19, 20, 21, 22, 26, 27, 30, 31, 32, 33, 34, 35, 36, 44, 45, 46, 48

[Bonabeau, Théraulaz, Arpin and Sardet 1994] E. Bonabeau, G. Théraulaz, E. Arpin and E. Sardet. The Building Behavior of Lattice Swarms. In Artificial Live IV, edited by R. Brooks and P. Maes, 307-312. Cambridge, MA: MIT Press, 1994. 47

[Bonabeau, Theraulaz and Deneubourg 1996] E. Bonabeau, G. Theraulaz and J.-L. Deneubourg. Quantitative Study of the Fixed Threshold Model for the Regulation of Division of Labour in Insect Societies. Proceedings Roy. Soc. London B 263 (1996): 1565-1569. 25

[Bonabeau, Theraulaz and Deneubourg 1997] E. Bonabeau, G. Theraulaz and J.-L. Deneubourg. Adaptive Task Allocation Inspired by a Model of Division of Labor in Social Insects. In Bio-Computation and Emergent Computing, edited by D. Lundh, B. Olsson and A. Narayanan, 36-45. Singapore: World Scientific, 1997. 47

[Bonabeau and Théraulaz 2000] E. Bonabeau and G. Théraulaz. Swarm smarts. Scientific American, pages 72-79, 2000. http://www.gjalc.org/soci196/swarm_ smarts.pdf. 9, 52

[Bonabeau et al. 1994] E. Bonabeau, G. Théraulaz, E. Arpin and E. Sardet: The building behavior of lattice swarms. In Artificial Life IV, Brooks, R. and Maes, P. eds., pp. 307-312, MIT Press 1994. 109, 140

[Bonabeau, Sobkowski, Theraulaz and Deneubourg 1997] E. Bonabeau, A. Sobkowski, G. Theraulaz and J.-L. Deneubourg. Adaptive Task Allocation Inspired by a Model of Division of Labour in Social Insects. In Bio-Computation and Emergent Computing, edited by D. Lundh, B. Olsson, and A. Naranyanan, 36-45. Singapore: World Scientific, 1997. 28

[Bowden, Terfort, Carbeck and Whitesides 1997] N. Bowden, A. Terfort, J. Carbeck and G. M. Whitesides. Self-Assembly of Mesoscale Objects Into Ordered Two-Dimensional Arrays. Science 276 (1997): 233-235. 49

[Brian 1983] M. V. Brian. Social Insects: Ecology and Behavioural Biology. Chapman & Hall, 1983. 44

[Broughton, Tan and Coates 1997] T. Broughton, A. Tan and P. S. Coates. The USe of Genetic Programming in Exploring 3D Design Worlds. In Proceedings of CAAD Futures, edited by R. Junge. München: Kluwer Academic Publishers, 1997. 50

[Bullnheimer, Hartl and Strauss 1997] B. Bullnheimer, F. Hartl and C. Strauss. A New

Rank Based Version of the Ant System: A Computational Study. Working Paper # 1, Adaptive Information Systems and Modelling in Economics and Management Science, Vienna, 1997. 21

[Caloud, Choi, Latombe, Pape and Yim 1990] P. Caloud, W. Choi, J.-C. Latombe, C. Le Pape and M. Yim. Indoor Automation With Many Mobile Robots. In Proceedings 1990 IEEE/RSJ International Conference on Intelligent Robots and Systems, 67-72. Los Alamitos, CA: IEEE Computer Society Press, 1990. 51

[Chialvo & Millonas 1995] D. Chialvo, M. Millonas: How Swarms build Cognitive Maps The Santa Fe Institute, Neural Systems, Memory and Aging, University of Arizona In Luc Steels (Ed.): The Biology and Technology of Intelligent Autonomous Agents, pp. 439-450, NATO ASI Series, 1995 11, 40, 58, 59, 117, 118, 119, 120

[Calderone and Page 1996] N. W. Calderone and R. E. Page. Temporal Polyethism and Behavioral Canalization in the Honey Bee, Apis mellifera. Anim. Behav. 51 (1996): 631-643. 27

[Ceusters 1980] R. Ceusters. Etude de degré de Couverture du ciel par löa Végétation au-dessus des nids de Formica polystena Foerst. Biol. Ecol. Médit. VII (3) (1980): 187-188. 44

[Ceusters 1986] R. Ceusters. Simulation du nid Naturel des Fourmis par de nids Artificiel Placé sur un Gradient de Température. Actes Coll. Insect. Soc. 3 (1986): 235-241. 44

[Chrétien 1996] L.Chrétien. Organisation Spatiale du Mateériel Provenant de l'excavation du nid chez Messor Barbarus et des Cadavres d'ouvrières chez Lasius niger (Hymenopterae: Formicidae). Ph.D. dissertation, Université Libre de Bruxelles, 1996. 29

[Coates, Healy, Lamb and Voon 1996] P. S. Coates, N. Healy, C. Lamb and W. L. Voon. The Use of Cellular Automata to Explore Bottom Up Architectonic Rules. Paper presented at Eurographics UK Chapter, 14th Annual Conference, 26-28 March 1996. 50

[Collier 2003] N. Collier: RePast, An Extensible Framework for Agent Simulation, http://repast.sourceforge.net/docs/repast_intro_final.doc, Last visited 20.09.2004 112

[Deneubourg, Goss, Franks and Pasteels 1990] J.-L. Deneubourg, S. Goss, N. Franks, and J.-M. Pasteels. The Self-Organizing Exploratory Pattern of the Argentine Ant. J. Insect Behavior 3 (1990): 159-168. 18

[Deneubourg and Goss 1989] J.-L. Deneubourg and S. Goss. Collective Patterns and Decision Making. Ethol. Ecol. & Evol. 1 (1989): 295-311. 47

[Deneubourg et al. 1991] J. Deneubourg, S. Gross, et al.: The Dynamic of Collective Sorting Robot-like Ants and Ant-Like Robots,
SAB ,90, 1st Conference On Simulation of Adaptive Behaviour: From Animals to Animats. Cambridge, MA: MIT Press, 1991 29, 30, 31, 32, 33, 65, 67, 84, 109, 114, 116

[Di Caro and Dorigo 1998] G. Di Caro and M. Dorigo. AntNet: Distributed Stigmergetic Control for Communications Networks. J. Art. Int. res. 9 (1998): 317-365 23

[Di Caro and Dorigo 1997a] G. Di Caro and M. Dorigo. AntNet: A Mobile Agents Approach to Adaptive Routing. Technical Report IRIDIA/97-12, université Libre de Bruxelles, Belgium, 1997. 23

[Di Caro and Dorigo 1997b] G. Di Caro and M. Dorigo. A Study of Distributed Stigmergetic Control for Packet-Switched Communications Networks. Technical Report IRIDIA/97-20, Université Libre de Bruxelles, Belgium, 1997. 23

[Donald, Jennings and Ris 1994] B. Donald, J. Jennings and D. Rus. Analyzing Teams of Cooperating Mobile Robots. In Proceedings 1994 IEEE International Conference on Robotics and Automation, 1896-1903. Los Alamitos, CA: IEEE Computer Society Press, 1994. 51

[Dorigo and Gambardella 1997a] M. Dorigo and L. M. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computing 1 (1997): 53-66 21

[Dorigo and Gambardella 1997b] M. Dorigo and L. M. Gambardella. Ant Colonies for the Traveling Salesman Problem. BioSystems 43 (1997): 73-81 21

[Dorigo, Maniezzo and Colorni 1991] M. Dorigo, V. Maniezzo and A. Colorni. The ant system: An autocatalytic optimizing process. technical report no. 91-016 revised. Technical report, Politecnico di Milano, Italy, 1991. ftp://iridia.ulb.ac.be/pub/mdorigo/tec.reps/TR.02-ANTS-91-016REV.ps.gz. 18

[Dorigo, Maniezzo and Colorni 1996a] M. Dorigo, V. Maniezzo and A. Colorni. The ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics - Part B, 26(1):29-41, 1996. ftp://iridia.ulb.ac.be/pub/mdorigo/journals/IJ.10-SMC96.ps.gz. 21

[Dorigo, Maniezzo and Colorni 1991b] M. Dorigo, V. Maniezzo and A. Colorni. Positive

Feedback as a Search Strategy. Tech. Rep. No. 91-016, Politecnico di Milano, Italy, 1991 21

[Dorigo 1992] M. Dorigo. Ottimizzatione, Apprendimento Automatico, ed Algoritmi Basati su Metafora Naturale. Ph.D. Dissertation, Politecnico di Milano, Italy, 1992 21

[Dorigo 2004] M. Dorigo: „Ant Colony Optimization"-Project, Université Libre des Bruxelles http://iridia.ulb.ac.be/~mdorigo/ACO/index.html, Stand Januar 2004 18, 109

[Doty and van Aken 1993] K. L. Doty and R. E. van Aken. Swarm-Robot Materials Handling Paradigm for a Manufacturing Workcell. In Proceedings 1993 IEEE International Conference on Robotics and Automation, 778-782. Los Alamitos, CA: IEEE Computer Society Press, 1993. 51

[Franks 1989] N. R. Franks. Army ants: A collective intelligence. American Scientist, 77, mar/apr 1989 17

[Franks and Sendova-Franks 1992] N. R. Franks and A. B. Sendova-Franks. Brood Sorting by Ants: Distributing the Workload Over The Work Surface. Behav. Ecol. Sociobiol. 30 (1992): 109-123. 30, 31

[Franks, Wilby, Silverman and Tofts 1992] N. R. Franks, A. Wilby, V. W. Silverman and C. Tofts. Self-Organizing Nest Construction in Ants: Sophisticated Building by Blind Buldozing. Anim. Behav. 44 (1992): 357-375. 44

[Franks and Deneubourg 1997] N. R. Franks and J.-L. Deneubourg. Selft-Organizing Nest Construction in Ants: Individual Worker Behaviour and the Nest's Dynamics. Anim. Behav. 54 (1997): 779-796. 44

[Freisleben and Merz 1996] B. Freisleben and P. Merz. New Genetic Local Search Operators for the Travelling Salesman Problem. In Proceedings 4th International Conference Parallel Problem Solving from Nature, PPSN IV, edited by H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-S. Schwefel, 890-899. Berlin: Springer-Verlag, 1996 21

[Gambardella, Taillard and Dorigo 1997] L. M. Gambardella, E. D. Taillard and M. Dorigo. Ant Colonies for the QAP. Technical Report IDSIA 4-97, IDSIA, Lugano, Switzerland, 1997. Published in J. Oper. Resh. Soc. 50(2) (1999): 167-176 21

[Gaussier and Zrehen 1994] P. Gaussier and S. Zrehen. Avoiding the World Model Trap: An Acting Robot Does Not Need to be Smart. Robotics and Computer-Integrated Manufacturing 11 (1994): 279-286. 36

[Gaussier and Zrehen 1994] P. Gaussier and S. Zrehen. A Constructivist Approach for Autonomous Agents. In Artificial Life and Virtual Reality, edited by N. Magnenat Thalmann and D. Thalmann. 97-113. New York, NY: Wiley & Sons, 1994. 37

[GLUI 2004] GLUI Programmers Reference, 2004, Version 2.2, Written by Paul Rademacher and Nigel Stewart, http://www.nigels.com/glt/glui/glui_manual_v2_beta.pdf 92, 121

[Goldberg 1989] D. E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley, 1989. 52

[Grassé 1959] P.-P. Grassé. La Reconstruction du nid et les Coordinations Inter-Individuelles chez Bellicositermes Natalensis et Cubitermes sp. La théorie de la Stigmergie: Essai d'interprétation du Comportement des Termites Constructeurs. Insect Soc. 6 (1959): 41-80. 14, 25, 46

[Grassé 1984] P.-P. Grassé. Termitologia, Tome II. Fondation des Societés. Construction. Patris: Masson, 1984. 25, 44, 47

[Handl et al. 2003a] J. Handl, J. Knowles, M. Dorigo: Strategies for the increased robustness of ant-based clustering. In: Engineering Self-Organising Systems, Lecture Notes in Computer Science, Volume 2977, Springer Verlag, 2004, IRIDIA, Université Libre de Bruxelles 12, 13, 83, 84, 86, 87, 92, 134

[Handl et al. 2003b] J. Handl and J. Knowles and M. Dorigo: Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and 1d-som. Technical Report TR/IRIDIA/2003-24, IRIDIA, Université Libre de Bruxelles, July 2003. http://www.handl.julia.de. 37, 127, 128, 141

[Hölldobler and Wilson 1978] B. Hölldobler and E. O. Wilson. The Multiple Recruitment Systems of the African Weaver ant Oecophylla longinoda (Latreille). Behav. Ecol. Sociobiol. 3 (1978): 19-60. 51

[Hölldobler 1990] B. Hölldobler. Territorial Behavior in the Green Tree Ant (Oecophylla smaragdina). Biotropica 15 (1983): 241-250. 51

[Holden 1997] C. Holden. On the Scent of a Data Trail. Science 278 (1997): 1407. 24

[Holland 1975] J. H. Holland. Adaptation in Natural and Artificial Systems. Ann Arbor, MI: The University of Michigan Press, 1975. 52

[Hosokawa, Shimoyama and Miura 1995] K. Hosokawa, I. Shimoyama and H. Miura.

Dynamics of Self-Assembling Systems. Analogy with Chemical Kinetics. Artificial Life 1 (1995): 413-427. 49

[Hosokawa, Shimoyama and Miura 1996] K. Hosokawa, I. Shimoyama and H. Miura. Two-Dimensional Micro-Self-Assembly Using the Surface Tension of Water. Sensors and Actuators A 57 (1996): 117-125. 49

[Hosokawa, Shimoyama and Miura 1997] K. Hosokawa, I. Shimoyama and H. Miura. Self-Assembling Microstructures. In Artificial Life V, edited by C. G. Langton and K. Shimohara, 362-369. Cambridge, MA: MIT Press, 1997. 49

[Oster and Wilson 1978] G. Oster and E. O. Wilson. Caste and Ecology in the Social Insects. Princeton, NJ: Princeton University Press, 1978. 24

[Johnson and McGeoch 1997] D. S. Johnson and L. A. McGeoch. The Travelling Salesman Problem: A Case Study in Local Optimization. In Local Search in Combinatorial Optimization, edited by E. H. L. Aarts and J. K. Lenstra. New York, NY: Wiley & Sons, 1997 21

[Klein 2002] J. Klein: breve - a 3D simulation environment for the simulation of decentralized systems and artificial life. Proceedings of Artificial Life VIII, the 8th International Conference on the Simulation and Synthesis of Living Systems, The MIT Press, 2002 112

[Krink and Vollrath 1997] T. Krink and F. Vollrath. Analysing Spider Web-Building Behaviour with Rule-Based Simulations, and Genetic Algorithms. J. Theor. Biol. 185 (1997): 321-331. 49

[Kube and Zhang 1994] R. C. Kube and H. Zhang. Collective Robotics: From Social Insects to Robots. Adaptive Behavior 2 (1994): 189-218. 51

[Kube and Zhang 1995] R. C. Kube and H. Zhang. Stagnation Recovery Behaviors for Collective Robotics. In Proceedings 1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, 1883-1890. Los Alamitos, CA: IEEE Computer Society Press, 1995. 51

[Kube and Zhang 1997] R. C. Kube and H. Zhang. Task Modelling in Collective Robotics. Auton. Robots 4 (1997): 53-72. 51

[Kuntz and Layzell 1995] P. Kuntz and P. Layzell. A New Stochastic Approach to Find Clusters in Vertex Set of Large Graphs with Applications Partitioning in VLSI Technology. Tech. Rep. LIASC, Ecole Nationale Supérieure des Télécommunications de Bretagne, 1995. 37

[Kuntz, Layzell and Snyers 1997] P. Kuntz, P. Layzell and D. Snyers. A Colony of Ant-Like Agents for Partitioning in VLSI Technology. In Proceedings Fourth European Conference on Artificial Life, edited by P. Husbands and I. Harvey, 417-424. Cambridge, MA: MIT Press, 1997. 37

[Lumer & Faieta 1994] E.D. Lumer, B. Faieta: Diversity and Adaption in Populations of Clustering Ants
in Cliff, D., et. Al: From Animals to Animats 3, Proc. Of the 3rd Int. Conference on the Simulations of Adaptive Behaviour, MIT Press, Cambridge, MA, 1994 13, 32, 84, 90, 109, 115, 116, 129, 134

[Lumer and Faieta 1995] E. Lumer and B. Faieta. Exploratory Database Analysis via Self-Organization. (1995): Unpublished manuscript. 36

[Marquardt 2003] Nicolai Marquardt: Intelligenz von Schwärmen: Grundlagen, Theorien und Anwendungen. Ausarbeitung zum VR Seminar im WS 03/04, Bauhaus-Universität Weimar, Fakultät Medien, 2003 9, 11, 58, 116, 118, 120

[Martinoli, Ijspeert and Mondada 1999] A. Martinoli, A. J. Ispeert and F. Mondada. Understanding Collective Aggregation Mechanisms: From Probabilistic Modelling to Experiments with Real Robots. Robotics and Autonomous Systems 29 (1999): 51-63. 37

[Moffet 1988] M. W. Moffet. Cooperative Food Transport By an Asiatic Ant. National Geog. Res. 4 (1988): 386-394. 50

[Morley 1996] R. Morley. Painting Trucks at General Motors: The Effectiveness of a Complexity-Based Approach. In Embracing Complexity: Exploring the Application of Complex Adaptive systems to Business, 53-58. Cambridge, MA: The Ernst & Young Center for Business Innovation, 1996. 28

[Morley and Ekberg 1998] R. Morley and G. Ekberg. Cases in Chaos: Complexity-Based Approaches to Manufacturing. In Embracing Complexity: A Colloquium on the Application of Complex Adaptive Systems to Business, 97-702. Cambridge, MA: The Ernst & Young Center for Business Innovation, 1998. 28

[Ramos & Abraham 2003] V. Ramos, A. Abraham: Swarms On Continous Data, Technical University of Lisbon, Portugal, 2003, http://alfa.ist.utl.pt/~cvrm/staff/vramos/ 109, 115, 129

[Ramos & Almedia 2000] V. Ramos, F. Almedia: Artificial Ant Colonies in Digital Image Habitats A Mass Behaviour Effect Study on Pattern Recognition (2000)

http://alfa.ist.utl.pt/~cvrm/staff/vramos/ 2nd Int. Workshop on Ant Algorithms, pp. 113-116, Brussels, Belgium, 2000 143

[Ramos and Merelo 2002] V. Ramos and J. J. Merelo. Self-Organized Stigmergic Document Maps: Environment as a Mechanism for Context Learning, in AEB´2002 - 1st Spanish Conference on Evolutionary and Bio-Inspired Algorithms, E. Alba, F. Herrera, J.J. Merelo et al. (Eds.), pp. 284-293, Centro Univ. de Mérida, Mérida, Spain, 6-8 Feb. 2002. 11, 12, 40, 41, 60, 84, 87, 119

[Ramos et al. 1999] V. Ramos, P. Pina, F. Muge: From Feature Extraction to Classification: A Multidisciplinary Approach applied to Portuguese Granites, 11th Scan. Conf. On Image Analysis, pp. 817-824, Greenland, 1999 11, 129, 143

[Ramos et al. 2002] V. Ramos, P. Pina, F. Muge: Self-Organized Data and Image Retrieval as a Consequence of Inter-Dynamic Relationships in Artificial Ant Colonies, CVRM-Geo Systems Centre, Tech. Univ. of Lisbon (IST), 2002 11, 13, 142, 143

[Rauch 1999] E. Rauch: How Swarms build Cognitive Maps;
Datenmaterial zum veroeffentlichten Paper und weitere Anmerkungen,
Weblink: http://cnls.lanl.gov/~rauch/swarmweb, Stand der Website vom 15.01.2004
117, 118, 119

[RePast 2004] University of Chicago: RePast - REcursive Porous Agent Simulation Toolkit. Agent based modelling toolkit for Java, Reference: http://repast.sourceforge.net, Last visited 20.09.2004 111

[Robson and Traniello 1998] S. K. Robson and J. F. A. Traniello. Resource Assesment, Recruitment Behavior, and Organization of Cooperative Prey Retrieval in the Ant Formica schaufussi (Hymenoptera: Formicidae). J. Insect Behav. 11 (1998): 1-22. 50

[Robinson 1992] G. E. Robinson. Regulation of Vision of Labor in Insect Societies. Annu. Rev. Entomol 37 (1992): 637-665. 24

[Robinson, Page and Huang 1994] G. E. Robinson, R. E. Page and Z.-Y. Huang. Temporal Polyethism in Social Insects is a Developmental Process. Anim Behav. 48 (1994): 467-469. 27

[Schoonderwoerd, Holland, Bruten and Rothkrantz 1996] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-Based Load Balancing in Telecommunications Networks. Adaptive Behavior 5 (1996): 169-207 23

[Schoonderwoerd, Holland and Bruten 1997] R. Schoonderwoerd, O. Holland and J.

Bruten. Ant-like agents for load balancing in telecommunications networks. Agents, pages 209-216, 1997. http://citeseer.nj.nec.com/schoonderwoerd96antbased.html. 18

[Sims 1991] K. Sims. Artificial Evolution for Computer Graphics. Comp. Graph. 25 (1991): 319-328. 50

[Stuetzle and Hoos 1997] T. Stuetzle and H. Hoos. The MAX-MIN Ant System and Local Search for the Traveling Salesman Problem. In Proceedings IEEE International Conference on Evolutionary Computation, ICEC'97, 309-314. Los Alamitos, CA: IEEE Computer Society Press, 1997. 21

[Stuetzle and Hoos 1997] T. Stuetzle and H. Hoos. Improvements on the Ant System: Introducing MAX-MIN Ant System. In Proceedings International Conference on Artificial Neural Networks and Genetic Algorithms. Vienna: Springer Verlag, 1997. 21

[Sudd 1960] J. H. Sudd. The Transport of Prey by an Ant Pheidole crassinoda. Behaviour 16 (1960): 295-308. 51

[Sudd 1963] J. H. Sudd. How Insects Work in Groups. Discovery 24 (1963): 15-19. 50

[Sudd 1965] H. J. Sudd. The Transport of Prey by Ant. Behaviour 25 (1965): 234-271. 51

[Sugawara and Sano 1997] K. Sugawara and M. Sano. Cooperative Acceleration of Task Performance: Foraging Behavior of Interacting Multi-Robot Systems. Physica D 100 (1997): 343-354. 37

[Swarm.org 2004] Santa Fe Insitute: Swarm.org Toolkit. Reference: http://www.swarm.org, Wiki: http://wiki.swarm.org, Last visited 20.09.2004 111

[Tamassia, Battista and Battini] R. Tamassia, G. Battista and C. Battini. Automatic Graph Drawing and Readability of Diagrams. IEEE Trans. Syst. Man Cybern. 18 (1988): 61-79. 37

[Theraulaz & Bonabeau 1995a] G. Théraulaz and E. Bonabeau: Coordination in distributed building. Science, 269, pp. 686-688., 1995 11, 47, 141, 142

[Theraulaz & Bonabeau 1995b] G. Théraulaz and E. Bonabeau: Modelling the collective building of complex architectures in social insects with lattice swarms, Journal of theor. Biology, 177, 381., 1995 47, 141

[Theraulaz, Goss, Gervet and Deneubourg 1991] G. Theraulaz, S. Goss, J. Gervet and

J.-L. Deneubourg. Task Differentiation in Polistes Wasp Colonies: A Model for Self-Organizing Groups of Robots. In Proceedings First International Conference on Simulation of Adaptive Behavior: From Animals to Animats, edited by J.-A. Meyer and S. W. Wilson, 346-355. Cambridge, MA: MIT Press, 1991. 28

[Theraulaz, Bonabeau and Deneubourg 1998] G. Theraulaz, E. Bonabeau and J.-L. Deneubourg. Threshold Reinforcement and The Regulation of Division of Labour in Insect Societies. Proceedings Roy. Soc. London B 265 (1998): 327-335 28

[Traniello and Beshers 1991] J. F. A. Traniello and S. N. Beshers. Maximization of Foraging Efficiency and Resource Defense By Group Retrieval in the Ant Formica schaufussi. Behav. Ecol. Sociobiol. 29 (1991): 283-289). 50

[Waldspurger, Hogg, Huberman and Kephart 1992] C. A. Waldspurger, T. Hogg, B. A. Huberman and J. O. Kephart. Spawn: A Distributed Computational Economy. IEEE Trans. Softw. Engineer. 18 (1992): 103-117. 28

[White 2000] T. White. Swarm intelligence and logistics and general informations. Web. http://www.scs.carleton.ca/ arpwhite/courses/95590Y/. 18

[White, Pagurek and Oppacher 1998] T. White, B. Pagurek and F. Oppacher. Connection Management Using Adaptive Mobile Agents. In Proceedings of teh International Conference on Parallel Distributed Processing Techniques and Applications (PDPTA'98), 802-809. CSREA Press, 1998 23

[Wikipedia, Normalverteilung] http://de.wikipedia.org/wiki/Normalverteilung 55

[Wilson 1984] E. O. Wilson. The Relation Between Caste Ratios and Division of Labor in the Ant Genus Pheidole (Hymenoptera: Formicidae). Behav. Ecol. Sociobiol. 16 (1984): 89-98. 25

[Wilson 1990] Edward O. Wilson, Bert Hölldobler: The Ants, Springer Verlag, 1998 (Cambridge, Belknap, 1990) 117

[Wodrich 1996] M. Wodrich. Ant Colony optimization. B.Sc. Thesis, Department of Electrical and Electronic Engineering, University of Cape Town, South Africa, 1996 23