Untersuchungen zur Dynamik der Massenrekrutierung bei Ameisen

S. Pielström

Lehrstuhl für Verhaltensphysiologie und Soziobiologie (Zoologie II), Julius-Maximilians-Universität Würzburg, Biozentrum, Am Hubland, 97074 Würzburg, Deutschland

Zusammenfassung

Eine Reihe von theoretischen Arbeiten beschäftigt sich mit den Mechanismen kollektiver Entscheidungen einer Ameisenkolonie für einen von zwei angebotenen Wegen in einem Versuchsystem. In der vorliegenden Arbeit wurden einige dieser Modelle eingehend diskutiert und dahingehend verallgemeinert, daß sie auf eine beliebige Anzahl von Entscheidungsmöglichkeiten anwendbar sind. Das erweiterte Modell konnte die früheren Ergebnisse reproduzieren. Darüber hinaus wurde berücksichtigt, daß vermutlich viele Ameisenarten nicht beliebig viel Spurpheromon auf einem Weg akkumulieren, sondern bei einem bestimmten Maximalwert aufhören, die Spur zu verstärken. Es konnte gezeigt werden, daß über die Festlegung einer solchen Maximalkonzentration die Anzahl aus einer Auswahl gewählter Ressourcen reguliert werden kann. Hierin besteht ein möglicher Mechanismus der Regulierung des Verhaltens einer Gruppe, ohne daß die Individuen ihr Verhalten ändern, oder auch nur Informationen über die Notwendigkeit einer Anpassung erhalten müssen.

Einleitung

Ein wesentliches Problem bei der Brutpflege besteht oftmals in der verhältnismäßig geringen Mobilität juveniler Organismen. Diese eingeschränkte Mobilität erfordert in der Regel das Verbleiben der Jungtiere an einem mehr oder weniger sicheren Ort, der in der Regel als Nest bezeichnet wird und für die Jungtiere geeignete Lebensbedingungen bietet. Hier verbleiben die Organismen in den frühen Phasen ihrer Ontogenese, in der Regel bis sie die Geschlechtsreife erreicht haben. Ist die betreffende Art auf Ressource angewiesen, die direkt am Nest nicht in ausreichender Menge zur Verfügung steht, so müssen die „pflegenden“ Individuen diesen Bereich regelmäßig verlassen, um in der Umgebung Ressourcen zu sammeln. Diese Art der Ressourcennutzung bezeichnet man als *central place foraging*.

Je nach Art, Größe und Verfügbarkeit der Ressource, Räuber und Konkurrenzdruck, Bewegungsart- und Energetik des betreffenden Tieres, sind hierbei unterschiedliche Strategien unter verhaltensökologischen Gesichtspunkten optimal. Ein wichtiger Aspekt ist hierbei oftmals der Kompromiß zwischen der Investition von Zeit und Energie in das auffinden von Ressourcen und dem ausbeuten bzw. abtransportieren derselben. Ein Individuum, das sich voll und ganz auf die Ausbeutung einer Ressource konzentriert, läuft Gefahr, eine andere zu ignorieren, selbst wenn die andere profitabler sein sollte. Ein Individuum, das sich zu sehr auf die Suche nach den besten Ressourcen konzentriert vernachlässigt dabei möglicherweise die Nutzung der erreichbaren. Eine optimale Sammelstrategie erfordert somit ein ausgewogenes Verhältnis der Investitionen in die Suche nach Ressourcen und die Nutzung derselben.

Soziale Organismen haben diverse Möglichkeiten, ihre Effizienz beim Sammeln von Ressourcen gegenüber solitären Organismen zu steigern. Allein schon das Zusammentragen der Erträge der einzelnen Individuen in einer Art „Ressourcenpool“ der Gruppe minimiert das Risiko einer temporären Ressourcenunterversorgung. Während der Ertrag eines Individuums einer großen Varianz unterliegt, profitiert der soziale Organismus vom mittleren Ertrag der Gruppe, der einer geringeren Varianz unterliegt.

Zudem kann der Ertrag einer sozialen Gruppe auch noch deutlich über die maximale Summe der Individualerträge hinaus gesteigert werden. Das einfachste Mittel hierzu ist die Rekrutierung zu bereits gefundenen Ressourcen. Sie ermöglicht es, Investitionen in die Suche nach Ressourcen zu Gunsten der Transportaktivität deutlich zu verringern und somit den Gesamtertrag der Gruppe zu steigern. Ein hoch entwickeltes Kommunikationssystem ermöglicht hierbei zudem den Austausch von Informationen über verschiedene Bekannte Ressourcen. In diesem Fall kann die Gruppe sogar gegebenenfalls all ihre Transportbemühungen auf die profitabelste erreichbare Ressource konzentrieren. Der Zugewinn, den Rekrutierung gegenüber rein individueller Sammeltätigkeit ermöglicht, ist rein quantitativ, d.h. eine Leistung, die auch ein Individuum erbringen könnte wird durch die Kooperation in der Gruppe gesteigert.

Ein qualitativer Unterschied zwischen solitären und sozialen Organismen tritt dann auf, wenn die Kooperation in der Gruppe den Zugang zu Ressourcen erschließt, die dem Individuum gar nicht zugänglich wären. So ermöglicht zum Beispiel kooperatives Jagen einem sozialen Prädatoren das Überwältigen einer Beute, die für ein Einzeltier nicht erreichbar wäre. Gerät eine Art im Laufe der Generationen in Abhängigkeit von einer solchen, nur für die Gruppe erschließbaren Ressource, so sollte dies eine bedeutende evolutionäre Schwelle darstellen, da

ein Rückfall in die solitäre Lebensweise nun, unabhängig vom Grad genetischer Verwandtschaft, nicht mehr evolutionsstabil sein kann.

In der Familie der Ameisen (Hymenoptera: Formicidae) zeigt sich die volle Bandbreite all dieser Entwicklungsmöglichkeiten sozialer Kooperation beim Sammeln von Ressourcen. Im Zuge ihrer Radiation und Anpassung unterschiedlichste Lebensräume- und Bedingungen hat diese hoch soziale Arthropodengruppe eine Vielzahl von Sammelstrategien entwickelt. Während Arten, die eine eher ursprüngliche Lebensweise pflegen, oftmals immer noch individuell fouragieren, verfügen viele Arten über hochentwickelte, meist chemische Rekrutierungsmechanismen. Hierbei folgen rekrutierte Arbeiterinnen entweder einem chemischen Gradienten leicht flüchtiger Botenstoffe in der Luft oder aber einer Spur weniger flüchtiger Substanzen am Boden, um eine Nahrungsquelle zu finden, die von einer Nestgenossin markiert wurde. Viele Arten haben sich durch Kooperation auch Nahrungsquellen erschlossen, die für ein solitäres Insekt nicht nutzbar wären. Bespiele hierfür sind die Jagdschwärme der Heeresameisen bzw. Treiberameisen der Gattungen *Eciton* und *Dorylus*, sowie die aufwendigen Arbeitsketten der Blattschneiderameisen der Gattungen *Atta* und *Acromyrmex*. Erstere sind durch Koordination ihrer Aktionen bei der Jagd in der Lage, Beutetiere überwältigen und abtransportieren, die deutlich zu groß sind, um für einem Individuum als Beute in Frage zu kommen. Bemerkenswert ist in diesem Zusammenhang auch die Fähigkeit dieser Tiere, die Bewegung der Gruppe derart zu koordinieren, daß die Ameisen in breiter Front eine Fläche lückenlos nach Beutetieren absuchen können. Blattschneiderameisen haben im Laufe ihrer Evolution eine Reihe von sequentiellen Arbeitsschritten entwickelt, die ihnen Blattmaterial als Nahrungsressource zugänglich zu machen. Da sie, wie alle Ameisen, prinzipiell eigentlich nicht über die physiologischen Anpassungen verfügen, die zur Verdauung von Cellulose notwendig sind, muß daß Blattmaterial zuvor in einem aufwendigen Prozeß zerkleinert und mit Hilfe eines symbiotischen Pilzes vorprozessiert werden. Vor allem aber zeigen die Individuen dieser Arten bereits morphologische Anpassungen an die Ausführung einer bestimmten Teilaufgabe, was sich vor allem in extremem Größenpolymorphismus äußert (Hölldobler und Wilson, 1990).

Neben zahlreichen experimentellen Untersuchungen in Feld und Labor wurden in der Vergangenheit auch einige theoretische Studien zum Fouragierverhalten von Ameisen durchgeführt. Während viele dieser Arbeiten sich mit konkreten ökologischen Fragestellungen beschäftigen, gibt es auch eine ganze Reihe von Untersuchungen, die sich vor allem auf die proximaten Mechanismen und ihre systemtheoretischen Aspekte konzentrieren.

Die Erkenntnisse zahlreicher Einzelstudien zu den proximaten Aspekten des Sammelverhaltens bei Bienen und Ameisen wurden von Sumpter und Pratt (2003) in einem Modell von Differentialgleichungen zusammengefaßt. In diesem Modell werden die Tätigkeitsgruppen, also die Menge der Individuen, die aktuell eine bestimmte Tätigkeit ausüben, als Populationen betrachtet, die miteinander in Wechselwirkung stehen.

Sammelstrategien im Umweltkontext

Die Frage, welche Sammelstrategie unter bestimmten Umweltbedingungen den maximalen Ertrag ermöglicht, haben Jaffé und Deneubourg (1992) in einem idividuenbasierten Simulationsmodell eingehend untersucht. Sie verglichen zum einen individuelle mit sozialen Sammelstrategien, zum anderen die Wirkung dreier unterschiedlicher Rekrutierungssysteme auf den Ertrag. Zudem analysierten sie den Einfluß der Umwelt auf das optimale Verhältnis der Investitionen in Suche und Transport. In ihrem Modell simulierten sie Ameisenkolonien unterschiedlicher Größe, die auf einer Fläche *patches* von Ressourcen suchen und einsammeln. Diese *patches* treten dabei in variierbarer räumlicher Dichte und Größe auf.
Es zeigte sich, daß rein individuelle Sammelstrategien in der Simulation durch drei Faktoren begünstigt werden: Geringe Koloniegröße, große Individuen und geringe Größe der *patches*. Wenn ein gefundener *patch* sofort von einem einzelnen Tier vollständig ausgebeutet werden kann, ist jede Form von Rekrutierung überflüssig. Der optimale Anteil von *scouts*, also Tieren, die mit der Suche nach neuen *patches* beschäftigt sind, verhielt sich im Modell umgekehrt proportional zu Ressourcendichte und Koloniegröße. Die verglichenen Rekrutierungsmethoden waren Gruppenrekrutierung (d.h. die rekrutierende Ameise rekrutiert immer eine konstante Anzahl von Nestgenossinen), autokratische Massenrekrutierung (d.h. es werden immer genau so viele Nestgenossinen rekrutiert, wie zur vollständigen Ausbeutung der gefundenen Ressource notwendig sind) und demokratische Massenrekrutierung (d.h. die Arbeiterinnen folgen einer Duftspur, die Wahrscheinlichkeit dieser zu folgen hängt von ihrer Konzentration ab).
Die Gruppenrekrutierung erwies sich vor allem bei kleinen *patches*, die in großer Dichte vorkommen, als besonders erfolgreiche Strategie. In einer Umwelt, in der eher große Nahrungsquellen in geringer Dichte vorkommen, zeigte sich die demokratische Massenrekrutierung der der Gruppenrekrutierung überlegen. In allen getesteten Szenarien schnitt die autokratische Massenrekrutierung besonders gut ab, allerdings ist fraglich,

inwiefern diese Variante in der Natur überhaupt umzusetzen ist, da sie dem rekrutierenden Individuum eine Einschätzung der Lage abverlangt, die in den meisten Fällen die sensitiven Fähigkeiten einer Ameise überfordern dürfte.

Ladungsgrößen im sozialen Kontext

Ein etwas spezielleres Problem in der Verhaltensökologie der Sammelstrategien sozialer Insekten stellt die Wahl der Ladungsgrößen durch Individuen dar. Aus dem Einfluß der Ladungsgröße auf die Bewegungsgeschwindigkeit des transportierenden Tieres läßt sich eine theoretische, optimale Ladungsgröße ableiten, bei der die Gesamttransportrate ihr Maximum erreicht. Bei Honigbienen und Blattschneidermeisen zeigen experimentelle Untersuchungen jedoch, daß die Individuen oftmals Ladungen transportieren, die deutlich unter diesem theoretischen Optimum liegen. Eine mögliche Erklärung dieses Phänomens bietet die Informationstransferhypothese. Nach dieser Hypothese nehmen Individuen eine Minderung ihrer Transportleistung in Kauf, um, vor allem dank der nun höheren Bewegungsgeschwindigkeit, effizienter rekrutieren zu können. Der Materialtransfer wird also vernachlässigt zugunsten des Informationstransfers, so daß der Nettoertrag der Gruppe, die nun schnellere Entscheidungen mit größerer Genauigkeit treffen kann, größer ist als der einer Gruppe, in der lediglich die individuellen Transportleitungen optimiert sind (Núñez, 1982; Roces und Núñez, 1993). Gestützt wird diese These durch ein analytisches Modell von Dornhaus et al. (2006).
In Bezug auf Blattschneiderameisen könnte auch die Verarbeitung des eingetragenen Materials eine Rolle spielen. Eingetragenes Blattmaterial muß von diesen Tieren in einem aufwendigen Prozeß schrittweise weiter zerkleinert werden. Burd und Howard (2005a) vermuten, daß die nachgeschalteten Arbeitsschritte der Blattzerkleinerung und –Verarbeitung den limitierenden Faktor darstellen. Anhand eines Modells, das die Weiterverarbeitung der eingetragenen Fragmente simuliert, konnte gezeigt werden, daß die unter diesem Gesichtspunkt optimale Fragmentgröße kleiner ist als die zur Maximierung der Transportrate optimale Größe (Burd und Howard, 2005b).

Mechanismen der kollektiven Entscheidung

Ein viel beachtetes Problem auf proximater Ebene ist die kollektive Entscheidungsfindung. Bei Ameisen besteht die Notwendigkeit zur Findung einer kollektiven Entscheidung vor allem dann, wenn die *scouts*, also diejenigen Tiere, die die Umgebung des Nestes nach Nahrungsquellen absuchen, mehrere Alternativen gefunden haben und zu diesen hin rekrutieren. Letztlich ist die Ameise, die das Nest verläßt also mit mehr als nur einer Duftspur konfrontiert und muß entscheiden, welcher der Spuren sie folgt. Interessant erscheint dabei, wie sich die gesamte Kolonie auf die alternativen Wege verteilt.

Zuerst wurde dieses Problem von Goss *et al.* (1989) untersucht. Sie beschäftigten sich mit der Frage, ob eine Ameisenkolonie in der Lage ist, den kürzesten Weg zu einer Nahrungsquelle zu finden, und welche Mechanismen dabei eine Rolle spielen. In einem Experiment boten sie einer Laborkolonie die Wegameise *Lasius niger* eine Futterquelle an, die von den Ameisen über zwei getrennte, unterschiedlich lange Holzbrücken erreicht werden konnte. Das Experiment zeigte, daß sich die Arbeiterinnen zunächst gleichmäßig auf beide Wege verteilen, nach einer gewissen Vorlaufzeit jedoch fast ausschließlich den kürzeren Weg nutzen.

Die Autoren vermuteten, daß diese Fähigkeit nicht auf individueller Wahrnehmung der Ameisen basiert, sondern durch die spezielle Dynamik ihres Spurlegeverhaltens von selbst zustande kommt. Sie entwickelten eine Monte-Carlo-Simulation (Anhang 1.) basierend auf der Annahme, daß eine Ameise, die an einer Weggabelung steht, ihre Entscheidung einzig und allein von den Pheromonkonzentrationen auf beiden Wegen abhängig macht, also anhand lokaler Informationen unabhängig von vorangegangener Erfahrung entscheidet. In diesem Modell bewegen sich Ameisen mit einer festgelegten Flußrate in beide Richtungen auf einem Weg, der sich in der Mitte in einen kürzeren und einen längeren Abschnitt teilt. Jede Ameise hinterläßt dabei eine Mengeneinheit Spurpheromon. Erreicht eine Ameise eine der beiden Weggabelungen, so ist die Wahrscheinlichkeit, einen bestimmten Weg auszuwählen eine Funktion der Pheromonkonzentrationen beider Wege:

$$p_1 = \frac{(20 + c_1)^2}{(20 + c_1)^2 + (20 + c_2)^2}$$

(p_1 = Wahrscheinlichkeit, Weg 1 auszuwählen; c_1 = Pheromonkonzentration auf Weg 1; c_2 = Pheromonkonzentration auf Weg 2)

Die Wahrscheinlichkeit, einen bestimmten Weg zu wählen verhält sich also nicht linear zum Konzentrationsunterschied, wodurch schon ein geringer Überschuß an Duftstoff auf einer Seite sehr großen Einfluß auf die Präferenz der Ameisen hat. Geringe Unterschiede werden quasi autokatalytisch verstärkt. Ein geringerer Überschuß auf dem kürzeren Weg ergibt sich automatisch, da, selbst bei gleichmäßiger Verteilung der Ameisen an der Weggabelung, der kürzere Weg schneller zurückgelegt und somit von mehr Ameisen pro Zeiteinheit passiert und markiert wird.

Weiterführende Experimente (Beckers *et al*., 1990) zeigten einen ähnlichen Effekt bei der Entscheidung zwischen zwei prinzipiell gleichwertigen Alternativen. Auch wenn die Kolonie über gleich lange Wege mit zwei gleichwertigen Futterquellen verbunden wird, konzentriert sich der Verkehr nach einer Weile auf einen der beiden Wege, während der andere weitgehend ignoriert wird. Die Entscheidung für eine der beiden Alternativen ist dabei zufällig. Diese Beobachtung untermauert die Hypothese der autokatalytischen Verstärkung von Konzentrationsunterschieden, da in diesem Fall bereits zufällige Schwankungen so starke Unterschiede in der Auswahlwahrscheinlichkeit bewirken, daß die ganze Kolonie sich letztlich für nur eine von zwei gleichwertigen Alternativen entscheidet (Abb. 1).

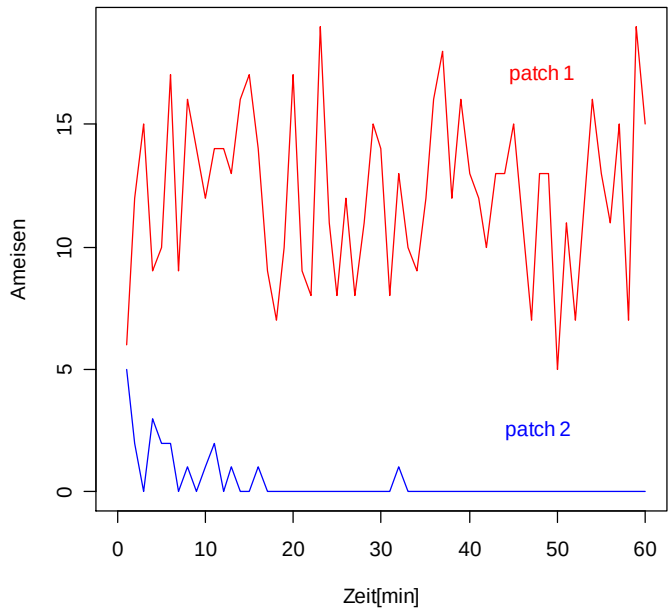

Abb. 1: Beispiel für einen Simulationslauf bei zwei gleichwertigen Futterquellen (nach Deneubourg 2003). Dargestellt ist die jeweilige Anzahl von Ameisen an beiden *patches* über die Zeit. (Anhang 2.)

Bietet man an einer der beiden Futterquellen eine Zuckerlösung höherer Konzentration an, dann fällt die Entscheidung überwiegend zugunsten der besseren Alternative aus. Vermutet wurde in diesem Zusammenhang, daß die Arbeiterinnen in Abhängigkeit von der Qualität einer Futterquelle stärker oder schwächer den Weg markieren. Eine eindeutige Entscheidung für die bessere Futterquelle zeigt sich jedoch nur dann, wenn beide Alternativen zeitgleich angeboten werden. Wenn die Ameisen zunächst nur zu einer Futterquelle Zugang haben sind sie nicht in der Lage, sich auf eine nachträglich zusätzlich angebotene, bessere Futterquelle umzustellen, was ebenfalls darauf hindeutet, daß sich die Arbeiterinnen an der Weggabelung eher von lokalen Pheromonkonzentrationen leiten lassen als von individueller Erfahrung. Der Konzentrationsunterschied zwischen einem frischen und einem bereits etablierten Weg ist so groß, daß auch ein deutlich attraktiveres Futterangebot die Entscheidung für die andere Alternative nicht mehr rückgängig machen kann.

Weitere Untersuchungen zeigten, daß eine Reihe weiterer Faktoren in diesem Versuchssystem eine Rolle spielen. Zunächst einmal konnte gezeigt werden, daß die Stärke der hinterlassenen Duftsputen tatsächlich von der Qualität der besuchten Futterquelle beeinflußt wird (Beckers *et al.*, 1992a; Beckers *et al.*, 1993) und daß die Tiere, wie schon im Modell von Goss *et al.* (1989) vorrausgesetzt, auch auf dem Weg vom Nest zu einer Futterquelle bereits aktiv Spuren legen, was die kollektive Entscheidung für einen der beiden Wege beschleunigt (Beckers *et al.*, 1992a). In der Regel spuren Arbeiterinnen auf dem Weg zu einer Futterquelle mit vergleichbarer Intensität wie auf dem Weg von der Futterquelle zum Nest (Beckers *et al.*, 1992b). Die Auswahl des kürzesten Weges wird zusätzlich dadurch unterstützt, daß die Ameisen mit einer gewissen Wahrscheinlichkeit auf dem Weg umdrehen, und zur letzten Weggabelung zurücklaufen. Auf einem längeren Weg passiert das häufiger. Davon abgesehen steigt die Wahrscheinlichkeit einer Kehrtwende, wenn der Winkel des Weges stark von der kürzesten Verbindungslinie zwischen Nest und Nahrung abweicht (Beckers *et al.*, 1992a).

Zudem konnte gezeigt werden, daß Kolonien von *L. niger* bei der Auswahl einer Holzbrücke in einem solchen Experiment auch die Breite des angebotenen Weges berücksichtigen. Ist einer der beiden Wege breiter als der andere wird dieser bei hohen Verkehrsdichten bevorzugt (Dussutour *et al.*, 2006).

Gemeinsam ist all diesen Untersuchungen, daß sie Verhalten einer bestimmten Art unter ähnlichen, verhältnismäßig artifiziellen Laborbedingungen beschreiben. Aron *et al.* (1993) konnten in einer vergleichenden Studie auch an *Linepithema humile* (früher *Iridomyrmex humilis*) viele der an *Lasius niger* gewonnenen Erkenntnisse reproduzieren. Allerdings ergab

dieser Versuch auch, daß gerade bei *Lasius niger* auch die individuelle Orientierung anhand optischer Reize noch eher eine Rolle spielt als bei *Linepithema humile*.

Das sich ein System, in dem die Ameisen nicht über Brücken laufen, sondern sich frei auf einer Fläche bewegen können, ähnlich verhält, konnten Sumpter und Beekman (2003) an *Monomorium pharaonis* zeigen.

Die Modelle, die zu diesem Themenkomplex entwickelt wurden (Goss *et al.* 1989; Beckers *et al.* 1990; Beckers *et al.* 1992a; Beckers *et al.* 1992b; Beckers *et al.* 1993) dienten alle dazu, die Mechanismen, die hinter dem Verhalten eines bestimmten Versuchssystems vermutet wurden zu simulieren und dabei die experimentellen Ergebnisse so gut wie möglich zu reproduzieren um die Hypothesen über die Gesetzmäßigkeiten in dem jeweiligen Versuchssystem zu bestätigen. Der nächste Schritt sollte nun in der Verallgemeinerung dieser Modelle unabhängig von einem speziellen Versuchsdesign liegen, mit dem Ziel, daß Sammelverhalten von Ameisenkolonien auch in einer komplexeren, naturnahen Umgebung erklären zu können. Dies ist die Vorraussetzung für das Verständnis der Rolle dieser proximaten Mechanismen bei der Anpassung an verschiedene ökologische Bedingungen und somit der Brückenschlag von den proximaten zu den ultimaten Fouragiermodellen.

Zur Verallgemeinerung der proximaten Modelle müssen zunächst Aspekte berücksichtigt werden, die in den sehr speziellen Versuchssystemen, die bisher simuliert wurden, keine besondere Rolle gespielt haben aber in einer komplexeren Umwelt durchaus von Bedeutung sein könnten. So ist z.B. in der ersten Monte-Carlo-Simulation von Goss *et al.* (1989) die Pheromonkonzentration rein additiv modelliert. Sie wird niemals geringer und kann praktisch unendlich große Werte annehmen, wenn die Simulation unendlich lange läuft. Bereits Beckers *et al.* (1992a) berücksichtigten in ihrem Modell die Verdunstungsrate der Pheromone auf Ameisenstraßen, einen Effekt von vermutlich großer ökologischer Bedeutung. Die Verdunstung von Duftstoffen verhindert zum einen, daß eine Straße noch ewig weiter belaufen wird, auch wenn die Futterquelle schon längst erschöpft ist. Zum anderen begrenzt die Verdunstungsrate die potentielle Länge einer Spur, da ab einer bestimmten Laufentfernung mehr Pheromon verdunstet, als von den Ameisen in der gleichen Zeit abgelegt werden kann, die Etablierung einer Ameisenstraße ist somit nur auf begrenzte Entfernungen möglich. Empirische Beobachtungen zeigen außerdem, daß Arbeiterinnen im Laufe eines Versuches auf einer etablierten Straße immer weniger Spurverhalten zeigen (Beckers et al. 1992b; Geißler, unveröffentlicht). Eine denkbare Erklärung für dieses Phänomen wäre, daß die Ameisen selbst aufhören, die Spur zu verstärken, wenn diese eine bestimmte Maximalkonzentration Erreicht hat.

In den bisherigen Modellen wurden die Parameter der Entscheidungsfunktion immer so festgelegt, daß die Ergebnisse der Simulationsläufe den Versuchsergebnissen besonders ähnlich waren. Eine verallgemeinerte Version der immer wieder auftauchenden Entscheidungsfunktion legte Deneubourg (2003) vor:

$$p_1 = \frac{(k + c_1)^n}{(k + c_1)^n + (k + c_2)^n}$$

(p_1 = Wahrscheinlichkeit, Weg 1 auszuwählen; c_1 = Pheromonkonzentration auf Weg 1; c_2 = Pheromonkonzentration auf Weg 2; k, n = konstante Parameter)

Da eine weitere Beschränkung bisheriger Modelle darin besteht, daß sie ausschließlich Systeme mit einem Nest und zwei Futterquellen beschreiben, wird in der vorliegenden Arbeit ein Modell beschrieben, das eine Ameisenkolonie und eine beliebige Anzahl von Futterquellen simuliert und dabei eine Begrenzung der maximal möglichen Pheromonkonzentration durch die Ameisen annimmt. Ziel der Untersuchung ist dabei das Verständnis des Einflusses einer solchen Konzentrationsbeschränkung auf das kollektive Sammelverhalten eines Insektenstaates.

Modell

Das verwendete Modell (Anhang 3.) simuliert das Sammelverhalten einer Ameisenkolonie auf einer Fläche von 1000 x 1000 Längeneinheiten über 10000 Zeitschritte. Die Simulation umfaßt ein Nest und eine beliebige Anzahl von Futterquellen, im folgenden als *patches* bezeichnet, sowie eine festgelegte Anzahl von Ameisen, die sich auf der Fläche mit einer Geschwindigkeit von 10 Längeneinheiten pro Zeitschritt bewegen. Jedem Objekt werden eine räumliche Position und ein Radius zugeordnet. Jeder virtuellen Ameise wird zudem eine aktuelle Tätigkeit zugeordnet, von der das Verhalten dieser Ameise abhängt. Zu Beginn der Simulation sind alle Ameisen noch ohne Tätigkeit und befinden sich im Nest. In jedem Zeitschritt besteht für jede Ameise eine Wahrscheinlichkeit von 0.1, daß diese das Nest verläßt und nach Nahrung sucht. Die möglichen Tätigkeiten waren im einzelnen:

- *suchen*: Die Ameise bewegt sich über die Simulationsfläche und ändert dabei zufällig die Richtung nach folgender Formel:

  $$\Delta\alpha = 2 \cdot \pi \cdot \tau \cdot \rho$$

  ($\Delta\alpha$ = Änderung des Bewegungswinkels in Bogenmaß; τ = Drehkonstante, entspricht der maximalen Richtungsänderung geteilt durch π; ρ = Zufallszahl zwischen -0.5 und 0.5) Die zufälligen Richtungsänderungen sind, wie bei der Brown'schen Molekülbewegung, normalverteilt um den Mittelwert 0.
- *beladen*: Die Ameise befindet sich an einem Futter*patch*.
- *tragen*: Die Ameise bewegt sich auf dem kürzesten Weg zum Nest.
- *entladen*: Die Ameise befindet sich im Nest.
- *Spur folgen*: Die Ameise bewegt sich auf dem kürzesten Weg zu einem *patch*.

Die Ameise wechselt dabei nach bestimmten Regeln ihre Tätigkeit. Hierbei ist nur eine begrenzte von Tätigkeitsübergängen möglich. Welche Tätigkeit eine Ameise als nächstes ausüben kann, hängt zunächst von der vorangegangenen ab. Eine „*suchende*" Arbeiterin wechselt zu „*beladen*". Nach „*beladen*" folgt „*tragen*", anschließend „*entladen*". Vom „*entladen*" kann die Ameise entweder wieder zu „*suchen*" wechseln, oder zu „*Spur folgen*". Eine Ameise mit der Tätigkeit „*Spur folgen*" macht weiter mit „*beladen*". Alle in der Simulation möglichen Übergänge sind in Abb. 2 dargestellt.

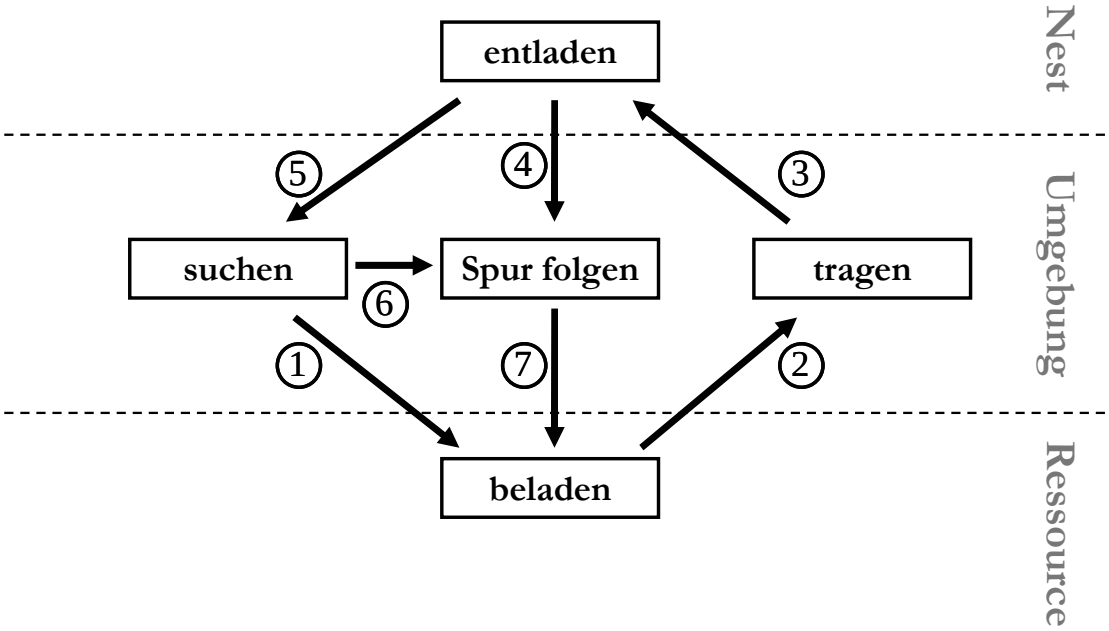

Abb. 2: Flußdiagramm der simulierten Tätigkeiten: Die Tätigkeitszustände werden in den Kästen dargestellt, die numerierten Pfeile stellen die möglichen Übergänge da. Die räumliche Position während der jeweiligen Tätigkeiten wird durch die gestrichelten Linien dargestellt.

patches und trails

Zu jedem Futter*patch* gibt es einen Weg, im foglgenden *trail* genannt, der die direkte Verbindungslinie zwischen *patch* und Nest darstellt. Dieser *trail* ist in Abschnitte unterteilt, deren Länge jeweils der von den Ameisen in einem Zeitschritt zurückgelegten Distanz entspricht, d.h. 10 Längeneinheiten. Jeder dieser Abschnitte hat eine bestimmte Pheromonkonzentration. Am Ende eines jeden Zeitschrittes verringern sich die Konzentrationen auf allen Abschnitten, wodurch die Verdunstung der Duftstoffe simuliert wird. Die Konzentration nimmt dabei in jedem Schritt um einen festen Prozentsatz ab. Die Abnahme errechnet sich nach folgender Funktion:

$$\Delta c = c \cdot (1 - r)$$

(c = Pheromonkonzentration; r = Verdunstungsrate) Zu Begin der Simulation ist diese Konzentration überall null. Jede Ameise, die sich auf einem *trail* bewegt, also im Zustand *„tragen“* oder *„Spur folgen“* befindet, erhöht die Pheromonkonzentration in dem Abschnitt, in dem sie sich gerade befindet, um eine Einheit, es sei denn, die Konzentration hat eine festgelegtes Maximum erreicht.

Tätigkeitsübergänge

Der Übergang von einer Tätigkeit in die nächste erfolgte unter folgenden Bedingungen:

- *1. suchen – beladen*: Die Ameise hat einen *patch* gefunden.
- *2. beladen – tragen*: Eine vorgegebene Verweildauer ($t_{beladen}$) ist verstrichen.
- *3. tragen – entladen*: Die Ameise hat das Nest erreicht.
- *4. entladen – Spur folgen*: Eine vorgegebene Verweildauer ($t_{entladen}$) ist verstrichen, es gibt wenigstens einen *trail* mit einer Konzentration > 0. Die Ameise folgt dem *trail* nun mit einer festgelegten Wahrscheinlichkeit ($1 - p_{scout}$).
- *5. entladen – suchen*: Eine vorgegebne Verweildauer ($t_{entladen}$) ist Verstrichen und die Ameise folgt keinem *trail*.

- *6. suchen – Spur folgen*: Die Ameise ist auf einen *trail* gestoßen. Die Wahrscheinlichkeit p, daß die Ameise diesem *trail* folgt ist nun:

$$p = p_{max} \cdot \left(1 - e^{p_{inc} \cdot c}\right)$$

(p_{max} = maximale Folgewahrscheinlichkeit; p_{inc} = Anstieg der Folgewahrscheinlichkeit in Abhängigkeit von der Pheromonkonzentration; c = Pheromonkonzentration)

- *7. Spur folgen – beladen*: Die Ameise hat den *patch* erreicht.

Die Entscheidung zwischen mehreren Wegen

Wenn mehrere *trails* vom Nest weg zu verschiedenen *patches* führen, muß sich eine Ameise, die einem *trail* folgen will, für eine der Alternativen entscheiden. Die Wahrscheinlichkeit, einem bestimmten von mehreren Wegen zu folgen hängt von nicht linear von der relativen Pheromonkonzentration ab. Der Funktion zur Bestimmung dieser Wahrscheinlichkeit liegt die von Deneubourg (2003) beschriebene Entscheidungsfunktion zugrunde, sie ist jedoch dahingehend erweitert, daß sie ein System mit beliebig vielen Entscheidungsalternativen beschreiben kann:

$$p_i = \frac{(k + c_i)^n}{\sum_{j=1}^{N} (k + c_j)^n}$$

(p_i = Wahrscheinlichkeit, dem *trail* i zu folgen; N = Anzahl der *trails*; c_i = Pheromonkonzentration auf *trail* i; k = Wendepunkt der Entscheidungsfunktion; n = Steigung der Entscheidungsfunktion)

konstante Parameter

In der vorliegenden Untersuchung wurden die Modellparameter wie folgt festgelegt:

- Nest:
  - Position: x = 500; y = 500
  - Radius: 50 Längeneinheiten

- *patches*:
    - Radius: 30 Längeneinheiten
- *trails*:
    - Verdunstungsrate: 1% pro Zeitschritt
- Ameisen:
    - Radius: 10 Längeneinheiten
    - τ: 0.2
    - p_{max}: 1
    - p_{inc}: 0.1
    - p_{scout}: 0.01
- Ablauf:
    - $t_{beladen}$: 1
    - $t_{entladen}$: 1
    - n: 2
    - Simulationsdauer: 10 000 Zeitschritte

Ergebnisse

Zunächst einmal konnte das vorliegende Modell die Ergebnisse früherer Simulationen reproduzieren. Bei standardisierten Simulationsbedingungen (100 Ameisen; k=5; Nest: x = 500, y = 500; 2 *patches*)ohne Beschränkung der maximalen Pheromonkonzentration auf den *trails* wählte die virtuelle Kolonie von zwei gleich weit entfernten, gleichwertigen *patches* einen aus und ignorierte den anderen weitgehend. Die Auswahl erfolgte dabei zufällig. Anhand der Pheromonkonzentrationen beider *trails* am Ende des Simulationslaufes läßt sich zeigen, daß eine der beiden Alternativen signifikant häufiger gewählt wurde als die andere, obwohl beiden Wahlmöglichkeiten gleichwertig waren (gepaarter t-Test: $t = 352$; $p < 0.001$). (Abb. 3)

Unter der Annahme, daß ein qualitativer Unterschied zwischen den Futterquellen die Ameisen, die von *patch* A veranlaßte, die doppelte Menge Spurpheromon zu produzieren, kam es zu deutlich mehr Entscheidungen für *patch* A (Wilcoxon Paarvergleichstest: $V = 4522$; $p < 0.001$) (Abb. 4). Noch deutlicher fiel die kollektive Entscheidung dann aus, wenn einer von zwei gleichwertigen *patches* nur halb so weit vom Nest entfernt lag wie der andere.

In diesem Fall wurde fast ausschließlich der näher gelegene *patch* ausgewählt, die Pheromonkonzentration auf den Wegen zu näher gelegenen *patches* waren signifikant höher (Wilcoxon Paarvergleichstest: $V = 5047$, $p < 0.001$) (Abb. 5).

Die Simulation einer ähnlichen Situation mit vier verschiedenen *patches*, von denen drei gleich weit vom Nest entfernt lagen, eine jedoch nur halb so weit, zeigte, daß die Kolonie auch unter mehr als zwei Futterquellen die nächstliegende finden kann (Abb. 6).

Die Auswirkung einer Maximalkonzentration, d.h. eines Grenzwertes, den die Pheromonkonzentration auf einem *trail* nicht übersteigen kann, auf das Verhalten des Systems hing von der Höhe dieser Begrenzung ab. In Simulationen mit vier gleichwertigen, gleich weit entfernten *patches* (100 Ameisen; Nest: $x = 500$, $y = 500$; *patch* A: $x = 100$, $y = 100$; *patch* B: $x = 100$, $y = 900$; *patch* C: $x = 900$, $y = 900$; *patch* D: $x = 900$, $y = 100$) konzentrierten sich die Kolonien wie in den vorherigen versuchen auf nur eine Alternative, wenn die Konzentrationen auf den *trail* hohe Werte annehmen konnten. Wurde die Maximalkonzentration jedoch sehr niedrig angesetzt, so kam es zu keiner kollektiven Entscheidung für eine Auswahlmöglichkeit, sondern die Ameisen verteilten sich gleichmäßig auf alle *patches*. Bei $k > 0$ zeigten sich Wertebereiche für die Maximalkonzentration, in denen die Ameisenkolonien eine stabile Entscheidung für eine begrenzte Zahl von Alternativen > 1 zeigten. Sie verteilten sich in diesem Fall gleichmäßig auf zwei bzw. drei von vier *patches* (Abb. 7). Bei einer kürzeren Laufstrecke, die der Hälfte der Laufstrecke im vorherigen Simulationslauf entsprach, besuchten die Ameisen auch bei höheren Maximalkonzentrationen mehr als einen *patch*. Die Bereitschaft, mehrere Alternativen gleichzeitig auszuwählen sank jedoch, wenn die Anzahl der beteiligten Individuen halbiert wurde (Abb. 8). Auch bei konstanter Maximalkonzentration stieg die Anzahl ausgewählter *patches* mit zunehmender Koloniegröße stufenweise an (Abb 9).

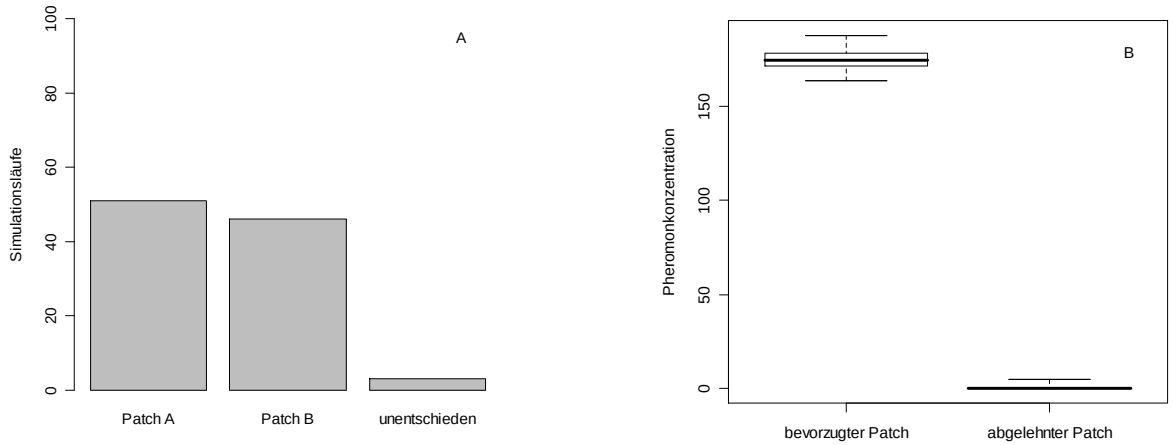

Abb. 3: A: Kollektive Entscheidungen bei der Wahl zwischen zwei gleichwertigen *patches* (*patch* A: x = 100, y = 100; *patch* B: x = 900, y = 900) in 100 Simulationsdurchläufen. Ein *patch* wurde dann als „bevorzugt“ gewertet, wenn seine Pheromonkonzentration größer war als ein 100faches der Pheromonkonzentration am konkurrierenden *patch*. B: Der Graph zeigt die Pheromonkonzentrationen der jeweils bevorzugten und die der abgelehnten Alternativen gegeneinander aufgetragen. (Box-Whisker-Plot: Der Mittelbalken repräsentiert den Median, der Kasten das Intervall vom unteren bis zum oberen Quartil. Die gestrichelten Arme enden am Minimal- bzw. Maximalwert. Ausreißer werden durch Punkte dargestellt.)

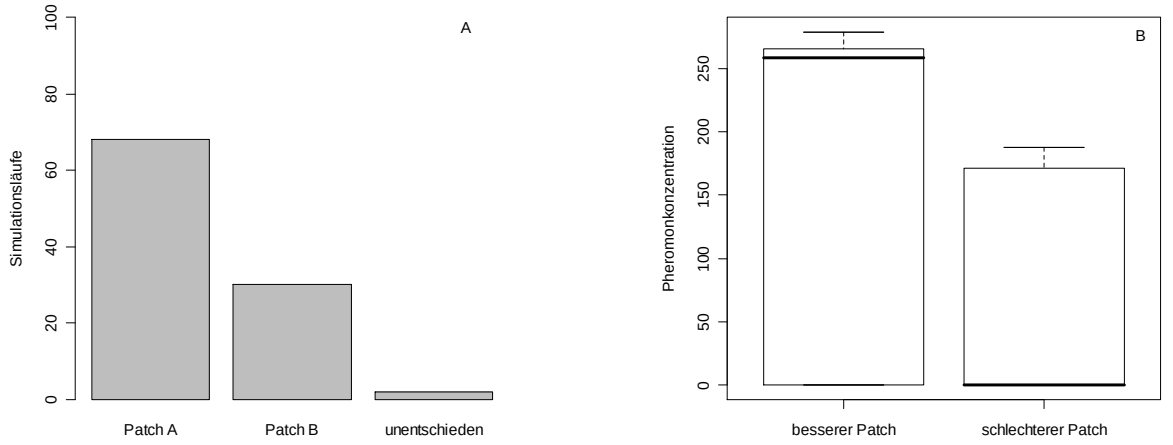

Abb. 4: A: Kollektive Entscheidungen bei der Wahl zwischen zwei *patches* (*patch* A: x = 100, y = 100; *patch* B: x = 900, y = 900) unterschiedlicher Qualität in 100 Simulationsdurchläufen. Ein *patch* wurde dann als „bevorzugt“ gewertet, wenn seine Pheromonkonzentration größer war als ein 100faches der Pheromonkonzentration am konkurrierenden *patch*. B: Der Graph zeigt die Pheromonkonzentrationen der jeweils besseren und die der schlechteren Alternativen gegeneinander aufgetragen. (Box-Whisker-Plot: Der Mittelbalken repräsentiert den Median, der Kasten das Intervall vom unteren bis zum oberen Quartil. Die gestrichelten Arme enden am Minimal- bzw. Maximalwert. Ausreißer werden durch Punkte dargestellt.)

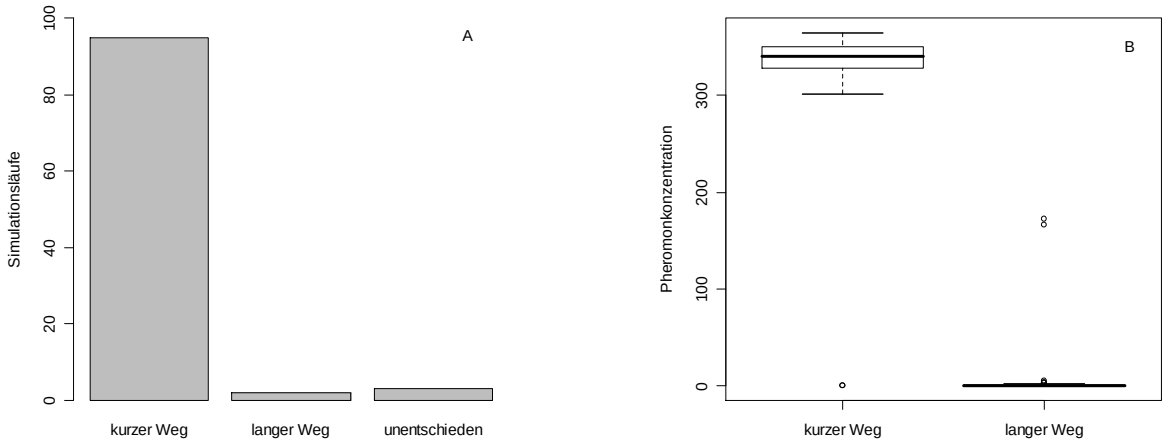

Abb. 5: A: Kollektive Entscheidungen bei der Wahl zwischen zwei gleichwertigen *patches* mit unterschiedlicher Entfernung zum Nest (*patch* A: x = 100, y = 100; *patch* B: x = 700, y = 700; kurzer Weg: 282.8 Längeneinheiten; langer Weg: 565.7 Längeneinheiten) in 100 Simulationsdurchläufen. Ein *patch* wurde dann als „bevorzugt“ gewertet, wenn seine Pheromonkonzentration größer war als ein 100faches der Pheromonkonzentration am konkurrierenden *patch*. B: Der Graph zeigt die Pheromonkonzentrationen am kurzen und am langen Weg gegeneinander aufgetragen. (Box-Whisker-Plot: Der Mittelbalken repräsentiert den Median, der Kasten das Intervall vom unteren bis zum oberen Quartil. Die gestrichelten Arme enden am Minimal- bzw. Maximalwert. Ausreißer werden durch Punkte dargestellt.)

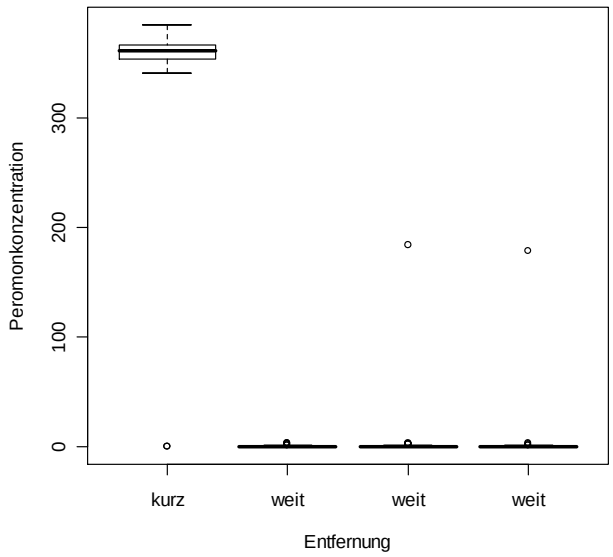

Abb: 6: Pheromonkonzentrationen an vier simultan dargebotenen *patches*. Die kurze Entfernung entspricht dabei dem 0.5fachen der weiten Entfernung zum Nest (*patch* A: x = 300, y = 300; *patch* B: x = 100, y = 900; *patch* C: x = 900, y = 900; *patch* D: x = 900, y = 100). (Box-Whisker-Plot: Der Mittelbalken repräsentiert den Median, der Kasten das Intervall vom unteren bis zum oberen Quartil. Die gestrichelten Arme enden am Minimal- bzw. Maximalwert. Ausreißer werden durch Punkte dargestellt.)

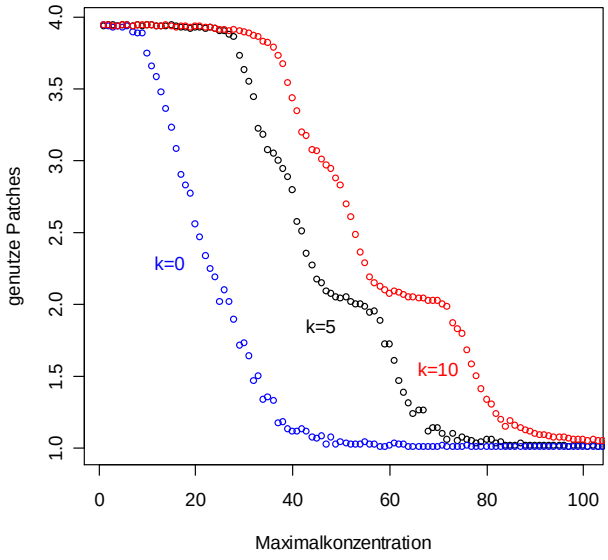

Abb: 7: Anzahl der von den simulierten Ameisenkolonien ausgewählten *patches* aufgetragen gegen die Maximalkonzentration der Pheromonspuren für drei verschiedene k. Um die Anzahl genutzter *patches* zu bestimmen, wurden die Pheromonkonzentrationen aller vier Alternativen am Ende des Simulationslaufes in relative Werte umgerechnet. Da die Werte bei besuchten *patches* etwa gleich waren, entspricht der Kehrwert der höchsten, relativen Konzentration der Anzahl der *patches*, die besucht wurden. Jeder Datenpunkt repräsentiert den Mittelwert von 100 Simulationsdurchläufen.

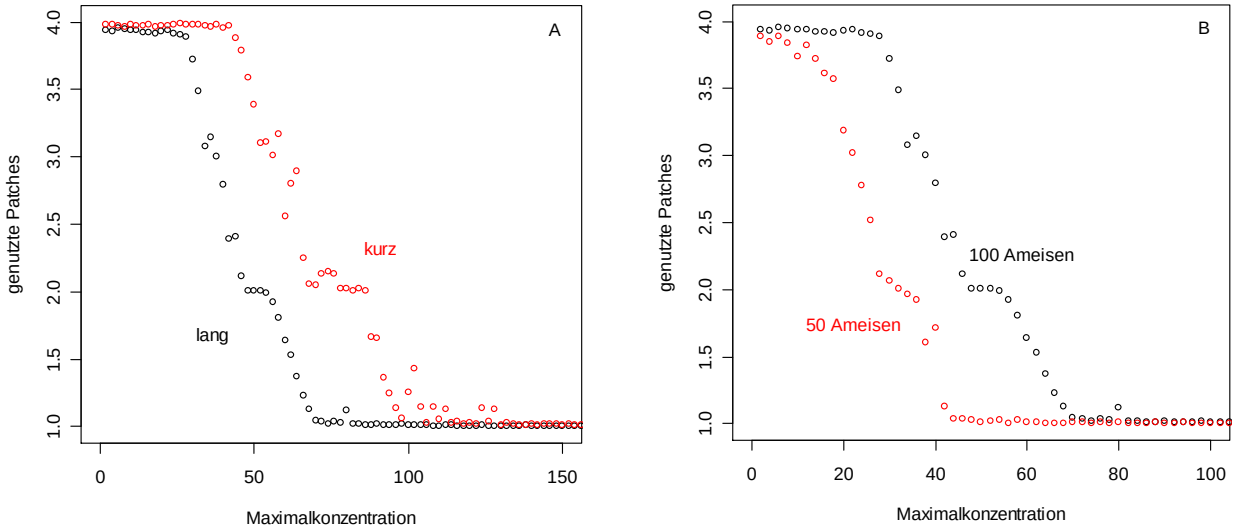

Abb. 8: Anzahl der von den simulierten Ameisenkolonien ausgewählten *patches* aufgetragen gegen die Maximalkonzentration der Pheromonspuren bei k = 5. Um die Anzahl genutzter *patches* zu bestimmen, wurden die Pheromonkonzentrationen aller vier Alternativen am Ende des Simulationslaufes in relative Werte umgerechnet. Da die Werte bei besuchten *patches* etwa gleich waren, entspricht der Kehrwert der höchsten, relativen Konzentration der Anzahl der *patches*, die besucht wurden. Jeder Datenpunkt repräsentiert den Mittelwert von 10 Simulationsdurchläufen.
A: Auswirkung der Streckenlänge bei konstanter Individuenzahl: In der Simulationsreihe „lang“ (100 Ameisen; Nest: x = 500, y = 500; *patch* A: x = 100, y = 100; *patch* B: x = 100, y = 900; *patch* C: x = 900, y = 900; *patch* D: x = 900, y = 100) waren die vier *patches* doppelt

so weit vom Nest entfernt wie in der Simulationsreihe „kurz“ (100 Ameisen; Nest: x = 500, y = 500; *patch* A: x = 300, y = 300; *patch* B: x = 300, y = 700; *patch* C: x = 700, y = 700; *patch* D: x = 300, y = 300).
B: Auswirkung der Individuenzahl bei konstanter Streckenlänge: Hier ist eine Simulationsreihe mit 100 Individuen aufgetragen gegen eine Simulationsreihe mit nur 50 Individuen (Nest: x = 500, y = 500; *patch* A: x = 100, y = 100; *patch* B: x = 100, y = 900; *patch* C: x = 900, y = 900; *patch* D: x = 900, y = 100).

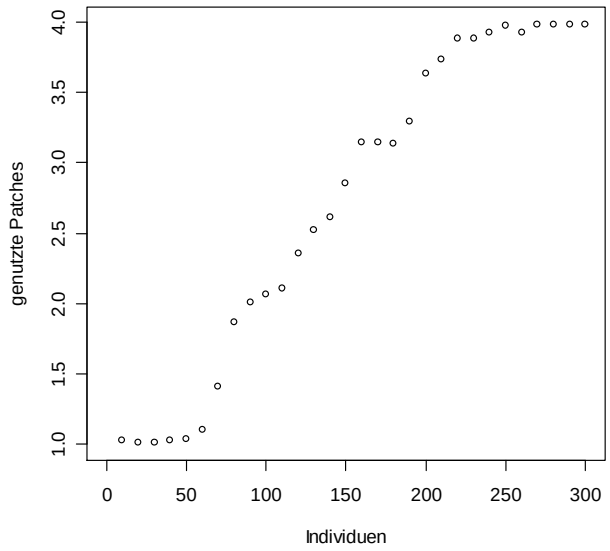

Abb: 9: Anzahl der von den simulierten Ameisenkolonien ausgewählten *patches* aufgetragen gegen die Koloniegröße bei einer Maximalkonzentration von 50 Einheiten und k = 5. Um die Anzahl genutzter *patches* zu bestimmen, wurden die Pheromonkonzentrationen aller vier Alternativen am Ende des Simulationslaufes in relative Werte umgerechnet. Da die Werte bei besuchten *patches* etwa gleich waren, entspricht der Kehrwert der höchsten, relativen Konzentration der Anzahl der *patches*, die besucht wurden. Jeder Datenpunkt repräsentiert den Mittelwert von 20 Simulationsdurchläufen. (Nest: x = 500, y = 500; *patch* A: x = 100, y = 100; *patch* B: x = 100, y = 900; *patch* C: x = 900, y = 900; *patch* D: x = 900, y = 100

Diskussion

Es konnte gezeigt werden, daß das vorliegende, theoretische Modell des kollektiven Fouragierverhaltens von Ameisenkolonien in der Lage ist, die Ergebnisse vorangegangener theoretischer und praktischer Untersuchungen zu reproduzieren und in Hinblick auf die Anzahl der wählbaren Alternativen zu verallgemeinern. Weiterhin konnte gezeigt werden, daß eine Begrenzung der Pheromonkonzentration, die aufgrund empirischer Beobachtungen anzunehmen ist, unter bestimmten Bedingungen einen entscheidenden Einfluß auf das Verhalten eines solchen Systems haben kann. Erreichen die Konzentrationen auf einer Ameisenstraße niemals hohe Werte, so bleibt die kollektive Entscheidung der Gruppe für eine

Auswahlmöglichkeit aus und die Individuen verteilen sich gleichmäßig auf alle Alternativen. Ein solches Verhalten läßt sich tatsächlich bei zahlreichen Ameisenarten in der Natur beobachten, vor allem bei solchen Arten, die bestimmte Ressourcen über lange Zeiträume nutzen. Vertreter der *Attini* z.B. nutzen komplexe, weit verzweigte Straßensysteme, die teilweise über Monate unverändert bleiben und es kommt niemals zu einer Konzentration des Verkehrs auf nur eine Straße. Unter diesen Bedingungen stellt sich natürlich zudem die Frage, inwiefern die Produktion von Pheromonen auch als Kostenfaktor eine Rolle spielt, ein Problem, das bei Arten, die eine kurzfristig auftretende Ressource schnell ausbeuten müssen vielleicht weniger ins Gewicht fällt. Bei den Ameisenarten, die in früheren Experimenten eine starke Tendenz zu klaren Entscheidungen für eine Alternative gezeigt haben, *Lasius niger*, *Linepithema humile* und *Monomorium pharaonis*, handelt es sich durchaus um Ameisen, die einen großen Teil, möglicherweise sogar manchmal ihren gesamten Nahrungsbedarf als opportunistische Detritusverwerter decken. Sie suchen einen bestimmten Laufbereich um ihr Nest nach toten Insekten und ähnlichen, profitablen aber nur kurzfristig zugänglichen Ressourcen ab.

Unter bestimmten Umständen ist eine Gruppe, die streng stereotyp dem Verhalten des Modells folgt sogar in der Lage, eine bestimmte Anzahl von Alternativen aus dem Angebot auszuwählen, anstatt sich auf eine zu konzentrieren, oder einfach alle zu nutzen. Hierbei wählen größere Gruppen mehr Futterquellen aus, als kleinere Gruppen, auch wenn die Individuen dabei nach den gleichen Verhaltensregeln agieren. Interessant ist hierbei vor allem die Tatsache, daß sich das Verhalten der Gruppe auf diese Weise in Abhängigkeit von einem bestimmten Parameter, in diesem Fall der Gruppengröße, ändern kann, obwohl sich am Verhalten der Individuen nichts ändert. Sollte diese Verhaltensmodifikation auf Gruppenebene tatsächlich eine Verhaltensanpassung darstellen, also einen adaptiven Wert haben, so wäre diese Anpassung des Gruppenverhaltens interessanterweise möglich, ohne daß auch nur ein Individuum eine Information über den unabhängigen Parameter, die Größe der Gruppe, hat.

Wie sich Veränderungen der unterschiedlichen Parameter nun genau auf das Verhalten des Systems auswirken bleibt zunächst noch unklar. Es ist aber zu vermuten, daß die theoretische Gleichgewichtskonzentration für eine bestimmte Strecke, in die eine Reihe von Parametern einfließt, dabei eine Rolle spielt. Denn die Einführung einer Verdunstungsrate für das Spurpheromon in ein Modell hat zwangsläufig zur Folge, daß die Pheromonkonzentrationen auf den Straßen einen bestimmten Wert nicht übersteigen können, auch dann nicht, wenn die Tiere selbst ungeachtet der schon vorhandenen Duftstoffmenge immer mehr Spuren legen.

Bei einem bestimmten Konzentrationswert liegen bei permanenter Erneuerung der Spur durch die Ameisen der Zuwachs an Pheromon und die Verlust durch Verdunstung im Gleichgewicht. Diese theoretische Gleichgewichtskonzentration (c_{eq}) läßt sich für eine bestimmte Strecke und eine bestimmte Zahl von beteiligten Individuen berechnen:

$$c_{eq} = \frac{v \cdot n}{d \cdot r}$$

(c_{eq} = Gleichgewichtskonzentration; v = Laufgeschwindigkeit der Individuen; n = Gesamtzahl der beteiligten Individuen; d = Länge des Weges; r = Verdunstungsrate des Pheromons) Bemerkenswert ist hierbei, daß sich c_{eq} proportional zum Kehrwert der Strecke verhält, also für lange Strecken, schnell sehr kleine Werte annimmt. Hierin könnte ein zusätzlicher Mechanismus gesehen werden, der die Entscheidung für eine näher gelegene Ressource erleichtert, möglicherweise auch dann, wenn diese erst später entdeckt wird. Eine sehr weit entfernte Ressource kann durch die geringe Gleichgewichtskonzentration auch nur eine relativ geringe maximale Attraktivität für die Individuen erreichen.

Um die Wirkung der verschiedenen Parameter in einem verallgemeinerten Modell zu untersuchen und das Verhalten des Systems unter variablen Bedingungen hinreichend zu verstehen, bedarf es also noch weiterer Analysen. Wünschenswert wäre dann, den adaptiven Wert der beobachteten Phänomene genau zu untersuchen. Hierzu sollten Simulationen durchgeführt werden, die unter variablen Umweltbedingungen bestimmte Kosten- und Nutzenparameter betrachten. Zu berücksichtigen wären dabei der Ressourceneintrag in einer bestimmten Zeit sowie die Varianz desselben, zurückgelegte Laufstrecken und auch die benötigte Pheromonmenge. Die Strategieparameter, insbesondere die Entscheidungskonstante k und die maximale Pheromonkonzentration, sollten unter verschiedenen Umweltbedingungen anhand der Kosten- und Nutzenparameter optimiert werden, so daß das Modell tatsächliche Vorraussagen über das Verhalten einer Ameisenkolonie im ökologischen Kontext erlaubt und die Lücke zur ultimaten Fragestellung nach dem eigentlichen Sinn der beobachteten Mechanismen schließt.

Literatur

Aron, S., Beckers, R., Deneubourg, J.-L., Pasteels, J.M., 1993. Memory and chemical communication in the orientation of two mass-recruiting ant species. *Insectes Sociaux 40*: 369-380.

Beckers, R., Deneubourg, J.-L., Goss, S., Pasteels, J.M., 1990. Collective decision making through food recruitment. *Insectes Sociaux 37(3)*:258-267.

Beckers, R., Deneubourg, J.-L., Goss, S., 1992a. Trails and U-turns in the selection of a path by the ant *Lasius niger*. *Journal of theoretical Biology 159*: 397-415.

Beckers, R., Deneubourg, J.-L., Goss, S., 1992b. Trail laying behaviour during food recruitment in the ant *Lasius niger* (L.*). Insectes Sociaux 39*: 59-72.

Beckers, R., Deneubourg, J.-L., Goss, S., 1993. Modulation of trail laying in the ant *Lasius niger* (Hymenoptera: Formicidae) and its role in the collective selection of a food source. *Journal of Insect Behaviour 6(6)*: 751-759.

Burd, M., Howard, J.J., 2005a. Central-place foraging continues beyond the nest entrance: underground performance of leaf cutting ants. *Animal Behaviour 70*:737-744.

Burd, M., Howard, J.J., 2005b. Global optimization from suboptimal parts: foraging sensu latu by leaf-cutting ants. *Animal Behaviour 70*:737-744.

Deneubourg, J.-L., 2003. Trail formation in ants**.** *Self organization in biological systems*. University Press of CA

Dornhaus, A., Collins, E.J., Dechaume-Moncharmont, F.-X., Houston, A.I., Franks, N.R., McNamara, J.M., 2006. Paying for information: partial loads in central place foragers. *Behav Ecol Sociobiol 61*: 151-161.

Dussutour, A., Nicolis, S.C., Deneubourg, J.-L., Fourcassié, V., 2006. Collective decisions in ants when foraging under crowded conditions. *Behav Ecol Sociobiol 61*: 17-30.

Goss, S., Aron, S., Deneubourg, J.-L., Pasteels, J.M., 1989. Self-organized shortcuts in the argentine ant. *Naturwissenschaften 76*: 579-581.

Hölldobler, B., Wilson, E.O., 1990. *The Ants*. Havard University Press, Cambridge, MA

Jaffe, K., 1980. Theoretical analysis of the communication systems for chemical mass recruitment in ants. *Journal of theoretical Biology 84*: 589-609. **?**

Jaffe, K., Deneubourg, J.-L., 1992. On foraging, recruitment systems and optimum number of scouts in eusocial colonies. *Insectes Sociaux 39*: 201-213.

Núñez, J.A., 1982. Honeybee foraging strategies at a foof source in relation to ist distance from the hive and the rate of sugar flow. *Journal of Apicultural research 21(3)*:139-150

Roces, F., Núñez, J.A., 1993. Information about food quality influences load-size selection in recruited leaf-cutting ants. *Anim Behav 45*:135-143

Sumpter, D.J.T., Beekman, M., 2003. From nonlinearity to optimality: pheromone trail foraging by ants. *Animal Behaviour 66*: 273-280.

Sumpter, D.J.T., Pratt, S.C., 2003. A modelling framework for understanding social insect foraging. *Behav Ecol Sociobiol 53*: 131-144.

Anhang: Quellcodes

1. Modell von Goss et al., 1989. Reproduziert als R-Skript (www.r-project.org)

```
#Model by Goss etal 1989: short and long trail experiment

#parameters:

time  = 30      #[min] simulation time
step  = 60      #[s] lenght of one time step
flux  = 0.2     #[probability/s] that one ant leaves the nest
q     = 1       #quantity of pheromone deposited
T     = 20      #time needed to walk the short trail
r     = 2       #ratio long/short
k     = 20      #constant decision parameter
n     = 2       #constant decision parameter

#initialize:

q_s_1 = 1
q_s_2 = 1
q_l_1 = 1
q_l_2 = 1
p_1   = numeric(time) #probabilitie to schoose the short trail at first
point
p_2   = numeric(time) #...
x     = c(1:time)     #x-values for plot

#simulate:

for (t in 1:time)
 {
 p_1[t] = (k+q_s_1)^n/((k+q_s_1)^n+(k+q_l_1)^n)
 p_2[t] = (k+q_s_2)^n/((k+q_s_2)^n+(k+q_l_2)^n)
 q_s_1  = q_s_1 + rpois(1,lambda=p_1[t]*flux*step)
 q_s_2  = q_s_2 + rpois(1,lambda=p_2[t]*flux*step)
 q_l_1  = q_l_1 + rpois(1,lambda=(1-p_1[t])*flux*step)
 q_l_2  = q_l_2 + rpois(1,lambda=(1-p_2[t])*flux*step)
 if (t > T)
  {
  q_s_1 = q_s_1 + rpois(1,lambda=p_1[t-T]*flux*step)
  q_s_2 = q_s_2 + rpois(1,lambda=p_2[t-T]*flux*step)
  }
 if (t > T*r)
  {
  q_l_1 = q_l_1 + rpois(1,lambda=(1-p_1[t-T*r])*flux*step)
  q_l_2 = q_l_2 + rpois(1,lambda=(1-p_2[t-T*r])*flux*step)
  }
 }
```

```
#plot:

plot(x,p_1,xlim=c(0,time),ylim=c(0,1),xlab="time[min]",ylab="ants",col="red
",type="l")
lines(x,1-p_1,col="blue")
```

2. Modell von Deneubourg, 2003. Reproduziert als R-Skript (www.r-project.org)

```
#simple trail laying choice model (Deneuboug 2003)
#parameters:

time = 60      #[min] simulation time
step = 60      #[s] lenght of one time step
flux = 0.2     #[probability/s] that one ant leaves the nest
q_1  = 1       #quantity of pheromone deposited by an ant returning from
patch 1
q_2  = 1       #...
k    = 5       #parameter in choice-function
n    = 2       #
f    = 0.0005  #[1/s] 1/mean lifetime of pheromone

#initialize:

trail_1   = 0           #pheromone concentration = C, trail 1
trail_2   = 0             #...
results_1 = numeric(time) #output vector for patch 1
results_2 = numeric(time) #...
x         = c(1:time)     #x-values for plot

#simulate:

for (i in 1:time)
 {
 p_1 = ((k+trail_1)^n) / ((k+trail_1)^n + (k+trail_2)^n) #probabilities
 p_2 = ((k+trail_2)^n) / ((k+trail_1)^n + (k+trail_2)^n)
 ants_1 = rpois(1,lambda=p_1*flux*step)                  #ant numbers
 ants_2 = rpois(1,lambda=p_2*flux*step)
 trail_1 = trail_1*(exp(-step*f))+ants_1*q_1             #new
concentrations
 trail_2 = trail_2*(exp(-step*f))+ants_2*q_2
 results_1[i] = ants_1
 results_2[i] = ants_2
 }

#plot:

plot(x,results_1,xlim=c(0,time),ylim=c(0,max(results_1,results_2)),xlab="Ze
it[min]",ylab="Ameisen",col="red",type="l")
lines(x,results_2,col="blue")
```

3. Delphi-Code zum Modell der vorliegenden Arbeit

```
unit AntSim60Unit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, TeEngine, Series, BubbleCh, ExtCtrls, TeeProcs, Chart, Math;

const

  MAX_X         = 1000;
  MAX_Y         = 1000;
  MAX_TIME      = 10000;
  MAX_PATCHES   = 100;
  PATCH_RADIUS  = 30;
  NEST_RADIUS   = 50;
  TRAIL_STEPS   = 200;        //max. Länge eines Trails
  MAX_ANTS      = 10000;
  TWIST         = 0.2;        //p(Richtungsänderung)
  ANT_RADIUS    = 10;         //Wahrnehmungsradius

type
  TForm1 = class(TForm)
    CloseButton: TButton;
    RunButton: TButton;
    chart1: TChart;
    Series1: TBubbleSeries;
    Series2: TBubbleSeries;
    Series3: TBubbleSeries;
    StopButton: TButton;
    Chart2: TChart;
    Series4: TLineSeries;
    Chart3: TChart;
    Series5: TLineSeries;
    Chart4: TChart;
    Series6: TLineSeries;
    Series7: TLineSeries;
    Series8: TLineSeries;
    AddPatchButton: TButton;
    SaveButton: TButton;
    n_AntsEdit: TEdit;
    n_AntsLabel: TLabel;
    PaintFreqEdit: TEdit;
    PaintFreqLabel: TLabel;
    PatchesEdit: TEdit;
    PatchesLabel: TLabel;
    LoadingTimeLabel: TLabel;
    LoadingTimeEdit: TEdit;
    UnloadingTimeLabel: TLabel;
    UnloadingTimeEdit: TEdit;
    SpeedLabel: TLabel;
    SpeedEdit: TEdit;
    c_MarkLabel: TLabel;
    c_MarkEdit: TEdit;
    EvapLabel: TLabel;
    EvapEdit: TEdit;
    Nest_xEdit: TEdit;
    Nest_xLabel: TLabel;
```

```
  Nest_yLabel: TLabel;
  Nest_yEdit: TEdit;
  f_ScoutsLabel: TLabel;
  p_Max_ScoutEdit: TEdit;
  p_Inc_ScoutEdit: TEdit;
  p_BoredEdit: TEdit;
  Label1: TLabel;
  Label2: TLabel;
  Label6: TLabel;
  PatchesCheckBox: TCheckBox;
  Label7: TLabel;
  Patch1Label: TLabel;
  Patch2Label: TLabel;
  Patch_yLabel: TLabel;
  Patch_xLabel: TLabel;
  Patch2_yEdit: TEdit;
  Patch2_xEdit: TEdit;
  Patch1_yEdit: TEdit;
  Patch1_xEdit: TEdit;
  Delay2Edit: TEdit;
  DelayLabel: TLabel;
  p_PlotChart: TChart;
  Series9: TLineSeries;
  PherCheckBox: TCheckBox;
  Label8: TLabel;
  Patch_QualityCheckBox: TCheckBox;
  Patch1_qEdit: TEdit;
  Patch2_qEdit: TEdit;
  Label9: TLabel;
  FeedBackCheckBox: TCheckBox;
  Label10: TLabel;
  CMaxEdit: TEdit;
  Label11: TLabel;
  flowEdit: TEdit;
  Label12: TLabel;
  Label13: TLabel;
  nEdit: TEdit;
  kEdit: TEdit;
  BidirectionalCheckBox: TCheckBox;
  degPatchCheckBox: TCheckBox;
  PatchLoadEdit: TEdit;
  Label3: TLabel;
  procedure RunButtonClick(Sender: TObject);        //Anwendung starten
  procedure PatchesCheckBoxClick(Sender: TObject);  //Patches zufällig?
  procedure PatchesEditChange(Sender: TObject);     //nicht zu viele Patches ?
  procedure AddPatchButtonClick(Sender: TObject);   //Patch hinzufügen
  procedure StopButtonClick(Sender: TObject);       //Programm anhalten
  procedure SaveButtonClick(Sender: TObject);       //Ergebnisse speichern
  procedure CloseButtonClick(Sender: TObject);      //Programm schließen
  procedure p_Max_ScoutEditChange(Sender: TObject);
  procedure p_Inc_ScoutEditChange(Sender: TObject);
  procedure Patch_QualityCheckBoxClick(Sender: TObject);
  procedure FeedBackCheckBoxClick(Sender: TObject);
  procedure degPatchCheckBoxClick(Sender: TObject);

private
  { Private-Deklarationen}
public
  { Public-Deklarationen}
end;

TNest = record
```

```
  x,y    : double; //Koordinaten
  radius : word;
  ants   : word;//Ameisen vor Ort (Unload nicht vergessen)
  end; //Nest

TTrail = array[1..TRAIL_STEPS] of double;

TNext_Trail = record
  x,y    : double;      //nächster Punkt auf dem Trail
  index  : byte;        //welcher Trail?
  end; //nächstliegender Trail aus der Sicht einer bestimmten Ameise

TPatch = record
  x,y    : double;
  radius : word;
  ants   : word;        //Amesien vor Ort
  load   : double;      //Futtermenge
  trail  : TTrail;      //Pheromonkonzentrationen
  end; //Futterpatch

TAnt = record //Ameise
  x,y,speed,
  motiv,                //Motivation,Begeisterung von der Futterquelle
  alpha,load : double; //Koordinaten, Geschwindigkeit und Bewegungswinkel,
  radius,           //Radius und momentane Ladung an Futter,
  patch_index,          //Patch, den die Ameise zur Zeit besucht
  patch_dist : word;    //und die Entfernung zum besagten Patch in Schritten.
  counter,              //Zeitzähler, für 'loading-' und 'unloading time'
  status     : byte;   //aktueller Zustand/Tätigkeit
  // 0 = home
  // 1 = scout
  // 2 = load
  // 3 = nestbound
  // 4 = unload
  // 5 = patchbound
  end; //eine Ameise und ihre Eigenschaften

Procedure Simulate();
Procedure ReadValues();                         //Parameter einlesen
Procedure Initialize(var paint_time : word);       //Simulation starten
  Procedure SetNest(var s : TNest);               //Nest einsetzen
  Procedure SetPatch(index : word; var s : TPatch);//Futter einsetzen
  Procedure SetAnt(var s : TAnt);                 //Ameise einsetzen

Procedure Move(var m : TAnt);                     //Aktionen der Ameisen
  Procedure Home(var h : TAnt);                   //rausgehen oder nicht?
  Procedure Scout(var s : TAnt);                  //Scouts bewegen
    Procedure Search(var s : TAnt);               //zufällige Bewegung
    Procedure HitTrail(index : word;
      var h : TAnt; var t : TNext_trail);         //Spur in der Nähe?
    Procedure EvalCon(index : word;
      var e : TAnt; var t : TNext_trail);         //Konzentration?
  Procedure Load(var l : TAnt);                   //Ladung aufnehmen
  Procedure Nestbound(var n : TAnt);              //Futter wegtragen
  Procedure Unload(var u : TAnt);                 //Ladung abgeben
  Procedure Patchbound(var p : TAnt);             //Auf der Spur laufen

Procedure Go(var g : TAnt; dest_x,dest_y : double);//auf einen Punkt zu laufen
Procedure PatchDist(patch : word; var p : TAnt);   //Entfernung zum Patch
Procedure ChooseTrail(var c : TAnt);//Trail auswählen
Procedure Evaporation(var trail : TTrail);         //Spur verfliegt
Procedure DeletePatch(index : word);
```

```
  Procedure ReadCapWord(caption : string; var value : word);//Editierfeld lesen
  Procedure ReadCapDouble(caption : string; var value : double);//s.o.für double
  Procedure CheckDelay(time : word);              //Patch erscheint?
  Procedure PGraph(p_max,p_inc : double);         //Plot p(c)

  Procedure Graph(time : word);                   //Graphik erstellen

  Procedure Save();                               //Ergebnisse speichern

  Function Distance(ax,ay,bx,by : double): double;   //Distanz 2er Punkte
  Function Lot_x(ax,ay,bx,by,px,py:double) : double; //x(Oli'scher Schnittpunkt)
  Function Lot_y(ax,ay,bx,by,px,py:double) : double; //y(Oli'scher Schnittpunkt)
  Function Lot_denom(ax,ay,bx,by:double) : double;   //Nenner von Olivers Formel
  Function P_Follow(p_max,p_inc,c : double): double; //p(Ameise folgt dem Trail)

var
  Form1     : TForm1;
  Nest      : TNest;
  intake    : double;  //ins Nest eingetragenes Futter
  stop,add  : boolean; //Abfragevariablen für 'stop' und die 'add patch'-Button

  n_patches,        //Anzahl der Patches
  n_ants,           //Anzahl der Ameisen
  paint_freq,       //Zeitschritte, nach denen die Graphik aktualisiert wird
  loading_time,     //benötigte Zeit für das Aufnehmen der Ladung
  unloading_time,   //benötigte Zeit für das Ablegen der Ladung
  patch_delay,      //Zeitverzögerung für das Erscheinen von Patches
  patch_load,       //mittlere Nahrungsmenge am Patch
  speed : word;     //zurückgelegte Entfernung/Zeitschritt


  trail_evap,       //Anteil an Pheromon, der pro Zeitschritt verdunstet
  nest_x,nest_y,    //Nestkoordinaten
  patch1_q,
  patch2_q,         //Patchqualitäten
  patch1_x,
  patch1_y,
  patch2_x,
  patch2_y,         //Koordinaten für Patches
  move_sum,         //Summe aller gelaufenen Entfernungen
  c_mark,           //abgegebene Pheromonmenge bei einer Markierung
  c_max,            //Maximale Pheromonkonzetration bei Qualität = 1
  p_max_scout,      //Die Wahrscheinlichkeit einem Trail zu folgen ist eine..
  p_inc_scout,      //..Funktion der Pheromonkonzentration an der jeweiligen..
                    //..Stelle. Es handelt sich dabei um eine begrenzte Wachs..
                    //..tumsfunktion mit einer horizontalen Asymptote (p_max)..
                    //..und einer Steigung (p_inc). Beide Parameter können in..
                    //..Abhängigkeit vom jew. Status festgelegt werden.
  f_scouts,         //Anteil an Scouts unter den aktiven Tieren
  k,n,              //Parameter für sigmoide p-Funktion nach Deneubourg
  int_mean_old,     //intake Gleitmittelwert für Graphik
  flow : double;    //Wahrscheinlichkeit, das ein Tier in einem Zeitschritt..
                    //..das Nest verlässt.

  Patches   : array[1..MAX_PATCHES] of TPatch; //alle Futterquellen
  Ants      : array[1..MAX_ANTS] of TAnt;      //alle Ameisen

  output    : textfile;                  //Textdatei zum Daten ablegen

implementation

{$R *.DFM}
```

```
//Funktionen_____________________________________________________________________
//_______________________________________________________________________________

Function Distance(ax,ay,bx,by:double): double;   //Entfernungen
  begin
  result := sqrt(sqr(ax-bx) + sqr(ay-by));
  end; //Koordinaten 2er Punkte -> Entfernung

Function Lot_x(ax,ay,bx,by,px,py:double) : double; //x(Oli'scher Schnittpunkt)
  begin
  result := (ax-((bx-ax)*(bx*ax-sqr(ax)+px*(-bx+ax)-py*by+py*ay+by*ay-sqr(ay)))
    /(sqr(bx)-2*bx*ax+sqr(ax)+sqr(by-ay)));
  end; //Olivers Formel: Koordinaten 3er Punkte -> x-Wert des Schnittpunktes..
       //..der Geraden AB mit dem Lot von C auf selbige Gerade.

Function Lot_y(ax,ay,bx,by,px,py:double) : double; //y(Oli'scher Schnittpunkt)
  begin
  result := (ay-((by-ay)*(bx*ax-sqr(ax)+px*(-bx+ax)-py*by+py*ay+by*ay-sqr(ay)))
    /(sqr(bx)-2*bx*ax+sqr(ax)+sqr(by-ay)));
  end; //Olivers Formel: Koordinaten 3er Punkte -> x-Wert des Schnittpunktes..
       //..der Geraden AB mit dem Lot von C auf selbige Gerade.

Function Lot_denom(ax,ay,bx,by:double) : double; //Nenner von Olis Formel
  begin
  result := (sqr(bx)-2*bx*ax+sqr(ax)+sqr(by-ay));
  end; //wenn sich die Ameise genau auf der Geraden befindet, ergibt der Nen-..
       //..ner 0, das vorher muss geprüft werden.

Function P_Follow(p_max,p_inc,c : double): double; //p(Ameise folgt dem Trail)
  begin
  result := p_max * (1- exp(- p_inc * c));
  end; //Wahrscheinlichkeit in Abhängigkeit von der Konzentration


//Simulation_____________________________________________________________________

Procedure Simulate();
  var time, paint_time, ant, patch : word;    //Zählvariablen
  begin
  Initialize(paint_time);               //Objekte einsetzen
  for time:=1 to MAX_TIME do            //Zeitschleife
    begin
    CheckDelay(time);
    for ant:=1 to n_ants do             //alle Ameisen..
      begin
      Move(Ants[ant]);                  //Aktionen der Ameisen
      if Ants[ant].counter > 0 then       //Zeitzähler prüfen
        dec(Ants[ant].counter);
      end; //for ant:=...
    if paint_time = 0 then              //Graphik erzeugen
      begin
      Graph(time);                      //Punkte zeichnen
      application.processmessages;        //Auf neue Eingaben prüfen
      paint_time := paint_freq;           //Zeitzähler zurücksetzen
      end; //if paint_time...
    for patch:=1 to n_patches do        //alle Trails...
      begin
      Evaporation(Patches[patch].trail);  //Trail verdunstet
      if (Patches[patch].load = 0) and (sum(Patches[patch].trail) = 0) then
        begin
        DeletePatch(patch);
```

```
      end;
    end; //for patch:=...
  paint_time := paint_time - 1;       //Zeit herunterzählen
  if stop=true then break;            //Abbruch?
  if add=true then                    //wurde 'add patch' angeklickt?
    begin
    inc(n_patches);                   //Anzahl der Patches erhöhen
    SetPatch(0,Patches[n_patches]);   //neuen Patch generieren
    add := false;                     //Schalter wieder deaktivieren
    end; //if add..
  end; //for time:=...
 Form1.StopButton.Visible  := false;
 Form1.SaveButton.visible  := true;
 Form1.CloseButton.Visible := true;
 end;

//_____________________________________________________________________________

procedure TForm1.RunButtonClick(Sender: TObject);
 begin
 randomize;                          //Zufallsgenerator
 ReadValues();                       //Parameter einlesen
 Simulate();
 end; //procedure

procedure TForm1.PatchesCheckBoxClick(Sender: TObject); //Patches zufällig?
 begin
 if Form1.AddPatchButton.Visible = true then //d.h. Patches vorher zufällig
   begin
   Form1.AddPatchButton.Visible := false;
   if not (Form1.PatchesEdit.text = '1') or (Form1.PatchesEdit.text = '2') then
   Form1.PatchesEdit.text := '2';    //nicht mehr als 2 Patches
   Form1.Patch1Label.Visible    := true;
   Form1.Patch_xLabel.Visible   := true;
   Form1.Patch_yLabel.Visible   := true;
   Form1.Patch1_xEdit.Visible   := true;
   Form1.Patch1_yEdit.Visible   := true;
   Form1.Label9.Visible         := true;
   Form1.Patch1_qEdit.Visible   := true;
   if Form1.PatchesEdit.text = '2' then
     begin
     Form1.Patch2_xEdit.Visible   := true;
     Form1.Patch2_yEdit.Visible   := true;
     Form1.Patch2Label.Visible    := true;
     Form1.DelayLabel.visible     := true;
     Form1.Delay2Edit.Visible     := true;
     Form1.Patch2_qEdit.Visible   := true;
     end;
   end //if Form1.Add...
 else
   begin
   Form1.AddPatchButton.Visible := true; //Button einblenden
   Form1.Patch1Label.Visible    := false;
   Form1.Patch2Label.Visible    := false;
   Form1.Patch_xLabel.Visible   := false;
   Form1.Patch_yLabel.Visible   := false;
   Form1.Patch1_xEdit.Visible   := false;
   Form1.Patch1_yEdit.Visible   := false;
   Form1.Patch2_xEdit.Visible   := false;
   Form1.Patch2_yEdit.Visible   := false;
   Form1.DelayLabel.visible     := false;
   Form1.Delay2Edit.Visible     := false;
```

```
    Form1.Label9.Visible         := false;
    Form1.Patch1_qEdit.Visible   := false;
    Form1.Patch2_qEdit.Visible   := false;
    end; //else...
  end;

procedure TForm1.PatchesEditChange(Sender: TObject); //nicht mehr als 2 Patches
  begin
  if Form1.PatchesCheckBox.checked = false then
    begin
    if not (Form1.PatchesEdit.text = '1') or (Form1.PatchesEdit.text = '2') then
    Form1.PatchesEdit.text := '2';     //nicht mehr als 2 Patches
    if Form1.PatchesEdit.text = '2' then
      begin
      Form1.Patch2_xEdit.Visible   := true;
      Form1.Patch2_yEdit.Visible   := true;
      Form1.Patch2Label.Visible    := true;
      Form1.DelayLabel.visible     := true;
      Form1.Delay2Edit.Visible     := true;
      Form1.Patch2_qEdit.Visible   := true;
      end; //if Delay...
    end; //if CheckBox...
  end;

procedure TForm1.Patch_QualityCheckBoxClick(Sender: TObject);
  begin
  if Form1.Patch_QualityCheckBox.checked = true then
    begin
    Form1.Patch1_qEdit.enabled := true;
    Form1.Patch2_qEdit.enabled := true;
    end
  else
    begin
    Form1.Patch1_qEdit.enabled := false;
    Form1.Patch2_qEdit.enabled := false;
    end;
  end;

procedure TForm1.FeedBackCheckBoxClick(Sender: TObject);
  begin
  if Form1.FeedBackCheckBox.checked = true then
    begin
    Form1.Label10.visible := true;
    Form1.CMaxEdit.visible := true;
    end
  else
    begin
    Form1.Label10.visible := false;
    Form1.CMaxEdit.visible := false;
    end;
  end;

procedure TForm1.degPatchCheckBoxClick(Sender: TObject);
  begin
  if Form1.degPatchCheckBox.checked = true then
    begin
    Form1.Label3.visible := true;
    Form1.PatchLoadEdit.visible := true;
    end
  else
    begin
    Form1.Label3.visible := false;
```

```
   Form1.PatchLoadEdit.visible := false;
   end;
  end;

procedure TForm1.AddPatchButtonClick(Sender: TObject); //neuen Patch hinzufügen
  begin
  add := true;
  end;


procedure TForm1.StopButtonClick(Sender: TObject); //Programm anhalten
  begin
  Form1.SaveButton.visible  := true;   //Save und Closebutton ausblenden
  Form1.CloseButton.Visible := true;
  Form1.StopButton.Visible  := false;
  Form1.AddPatchButton.enabled := false;
  stop := true;
  end;

procedure TForm1.SaveButtonClick(Sender: TObject); //Daten in Textfile speichern
  begin
  Save();
  end;

procedure TForm1.CloseButtonClick(Sender: TObject); //Programm schließen
  begin
  close;
  end;

//Initialisierung_____________________________________________________________

Procedure ReadValues(); //Parameter einlesen
  begin
  ReadCapWord(Form1.n_AntsEdit.text,n_ants);
  ReadCapWord(Form1.PaintFreqEdit.text,paint_freq);
  ReadCapWord(Form1.PatchesEdit.text,n_patches);
  ReadCapWord(Form1.LoadingTimeEdit.text,loading_time);
  ReadCapWord(Form1.UnloadingTimeEdit.text,unloading_time);
  ReadCapWord(Form1.SpeedEdit.text,speed);
  ReadCapWord(Form1.PatchLoadEdit.text,patch_load);
  ReadCapDouble(Form1.c_MarkEdit.text,c_mark);
  ReadCapDouble(Form1.flowEdit.text,flow);
  if Form1.FeedBackCheckBox.checked = true then
    begin
    ReadCapDouble(Form1.cMaxEdit.text ,c_max);
    end;
  if Form1.PatchesCheckBox.checked = false then //nur, wenn Patches nicht zu-..
    begin                                 //..fällig positioniert werden.
    ReadCapWord(Form1.Delay2Edit.text ,patch_delay);
    if patch_delay > 0 then n_patches := 1;     //hier nur mit einem anfangen
    end;
  ReadCapDouble(Form1.Patch1_qEdit.text ,patch1_q);
  ReadCapDouble(Form1.Patch2_qEdit.text ,patch2_q);
  ReadCapDouble(Form1.EvapEdit.text,trail_evap);
  ReadCapDouble(Form1.Nest_xEdit.text,nest_x);
  ReadCapDouble(Form1.Nest_yEdit.text,nest_y);
  ReadCapDouble(Form1.Patch1_xEdit.text,patch1_x);
  ReadCapDouble(Form1.Patch1_yEdit.text,patch1_y);
  ReadCapDouble(Form1.Patch2_xEdit.text,patch2_x);
  ReadCapDouble(Form1.Patch2_yEdit.text,patch2_y);
  ReadCapDouble(Form1.p_Max_ScoutEdit.text,p_max_scout);
  ReadCapDouble(Form1.p_Inc_ScoutEdit.text,p_inc_scout);
```

```
 ReadCapDouble(Form1.p_BoredEdit.text,f_scouts);
 ReadCapDouble(Form1.nEdit.text,n);
 ReadCapDouble(Form1.kEdit.text,k);
 end; //diese Prozedur liest die Parameter, die in die Editierfenster einge-..
     //..geben wurden. All diese Parameter sind oben als globale Variablen..
     //..deklariert.

Procedure Initialize(var paint_time : word); //Objekte einsetzen
 var patch,ant : word;
 begin
 Form1.SaveButton.visible  := false;   //Save und Closebutton ausblenden
 Form1.CloseButton.Visible := false;
 Form1.StopButton.visible  := true;
 Form1.AddPatchButton.enabled := true;
 stop := false;              //Abbruchkriterium
 add  := false;              //Patch hinzufügen
 paint_time := 0;            //Zähler für die Graphikausgabe
 intake := 0;                //Futtereintrag
 move_sum := 0;              //Kostenvariable Bewegung
 Form1.series2.clear;        //evtl. alte Graphiken löschen
 Form1.series3.clear;
 Form1.series4.clear;
 Form1.series5.clear;
 Form1.series6.clear;
 Form1.series7.clear;
 Form1.series8.clear;
 PGraph(p_max_scout,p_inc_scout);
 SetNest(Nest);              //Nest erzeugen
 for patch:=1 to n_patches do //alle Patches erzeugen
  begin
  SetPatch(patch,Patches[patch]);
  end; //for patch:=...
 for ant:=1 to n_ants do      //alle Ameisen erzeugen
  begin
  SetAnt(Ants[ant]);
  end; //for ant:=...
 end; //Alle Variablen werden auf ihre Startwerte gesetzt, Nest, Futter-..
     //quellen und Ameisen werden erzeugt.

Procedure SetNest(var s : TNest); //Nest erzeugen
 begin
 s.x      := nest_x;       //Koordinaten des Nestes, werden im Editierfen-..
 s.y      := nest_y;       //..ster festgelegt.
 s.radius := NEST_RADIUS;  //der Radius ist konstant.
 end; //procedure

Procedure SetPatch(index : word; var s : TPatch); //Futterquelle erzeugen
 var
 i : word;
 begin
 if Form1.PatchesCheckBox.checked = true then //Patches zufällig verteilen?
  begin
  s.x := random*MAX_X;     //Die Positionierung der Futterquelle erfolgt..
  s.y := random*MAX_Y;     //..hier zufällig
  s.load := patch_load;
  end //if CheckBox...
 else
  begin
  if index = 1 then  //patch1
   begin
   s.x := patch1_x;
   s.y := patch1_y;
```

```
    s.load := patch1_q*patch_load;
    end  //if index...
   else              //patch2
    begin
    s.x := patch2_x;
    s.y := patch2_y;
    s.load := patch2_q*patch_load;
    end
   end; //else...
  s.radius := PATCH_RADIUS;     //Der Radius ist konstant.
  s.ants   := 0;               //keine Ameisen vor Ort
  for i:=1 to TRAIL_STEPS do    //alle Abschnitte auf dem zugehörigen Trail:
   begin
   s.trail[i] := 0;             //noch kein Pheromon vorhanden
   end; //for i:=1..
  end; //procedure

Procedure SetAnt(var s : TAnt); //Ameise erzeugen
  begin
  s.x       := Nest.x;      //Zu beginn der Simulation befinden sich alle..
  s.y       := Nest.y;      //..Ameisen im Nest.
  s.speed   := speed;       //wird in einem Editierfenster festgelegt.
  s.alpha   := 2*pi*random; //die Laufrichtung ist zu Beginn zufällig
  s.radius  := ANT_RADIUS;  //der Wahrnehmungsradius ist konstant
  s.load    := 0;           //noch keine Ladung
  s.counter := 0;           //Zeitzähler zurückgesetzt
  s.status  := 0;           //,da die Ameise ja noch im Nest ist.
  end; //procedure

//Tätigkeiten_____________________________________________________________

Procedure Move(var m : TAnt); //Was tut die Ameise?
  begin
  if m.status = 0 then Home(m);        //je nachdem, welchen Wert die Status-..
  if m.status = 1 then Scout(m);       //..variable eiener Ameise hat, wird..
  if m.status = 2 then Load(m);        //..an eine bestimmte Tätigkeitsproze-..
  if m.status = 3 then Nestbound(m);   //..dur übergeben.
  if m.status = 4 then Unload(m);
  if m.status = 5 then Patchbound(m);
  end; //procedure

Procedure Home(var h : TAnt);//0
  begin
  h.x       := Nest.x;                  //Ameise ist im Nest
  h.y       := Nest.y;
  if flow > random then
    begin                             //..neue Futterquellen suchen?
    ChooseTrail(h);                   //Trail folgen?
    end; //if (h.status...
  end; //Die Prozedur prüft, ob eine Ameise von einem Trail angelockt wird,..
      //.. spontan das Nest verlässt, um Nahrung zu suchen oder im Nest bleibt.

Procedure Scout(var s : TAnt);//1
  var
  next_trail : TNext_trail;           //der nächstliegende Trail
  patch : word;                       //Zählvariable
  begin
  s.patch_index := 0;                 //noch kein Patch ausgewählt
  Search(s);                          //Bewegung im Raum
  for patch:=1 to n_patches do        //Schleife über alle Patches
    begin
    if (Distance(s.x,s.y,Patches[patch].x,Patches[patch].y) //Patch gefunden?
```

```
    < Patches[patch].radius) and (Patches[patch].load > 0) then
    begin
    s.patch_index := patch;                    //index anpassen
    s.status := 2;                             //Ameise beginnt zu laden
    s.counter := loading_time;                 //Zeit wird gezählt
    Patches[patch].ants := Patches[patch].ants + 1; //Ameisenzahl
    end; //if Distance...
  if s.patch_index = 0 then                    //Trail gefunden?
    begin
    HitTrail(patch,s,next_trail);              //Trail gefunden?
    EvalCon(patch,s,next_trail);               //Trail stark genug?
    end;//if s.patch_index...
  end;//for patch:=1...
 end; //Diese Prozedur bewegt die Ameise und überprüft, ob sie dabei auf Pat-..
     //..ches oder die dazugehörigen Trails gestoßen ist.

Procedure Load(var l : TAnt);//2
 begin
 if l.counter = 0 then              //fertig geladen?
  begin
  l.status     := 3;                //zum Nest laufen
  l.patch_dist := 0;                //Dinstanz zum Patch
  l.motiv      := c_mark;
  if Patches[l.patch_index].load > 0 then
   begin
   l.load := 1;                     //Ladung aufnehmen
   if Form1.degPatchCheckBox.Checked = true then
    Patches[l.patch_index].load := Patches[l.patch_index].load - 1;
   end
  else
   begin
   l.status := 1;
   end;
  if Form1.PherCheckBox.checked = true then
   begin
   l.motiv := l.motiv*exp(- 0.01*Distance(Patches[l.patch_index].x,
   Patches[l.patch_index].y,Nest.x,Nest.y));
   end; //Markieren abhängig von der Entfernung?
  if Form1.Patch_QualityCheckbox.checked = true then
   l.motiv := l.motiv*Patches[l.patch_index].load; //...oder der Qualität?
  dec(Patches[l.patch_index].ants);  //eine Ameise weniger am Patch
  end; //if l.counter...
 end; //Die Prozedur prüft, ob eine Ameise lange genug am Patch war und..
     //..schickt sie dann auf den Rückweg. Wenn erwünscht wird die Motiva-..
     //..tion der Ameise an Entfernung und Qualitätsfaktor des Patches ange-..
     //..passt, von diesem Motivationswert hängt dann die Stärke der Mar-..
     //..kierung ab.

Procedure Nestbound(var n : TAnt);//3
 begin
 if not (Distance(n.x,n.y,Nest.x,Nest.y) < n.radius) then //Nest erreicht?
  begin //nein, dann:
  Go(n,Nest.x,Nest.y);        //Richtung Nest bewegen
  inc(n.patch_dist);          //neue Position auf dem Trail
  Patches[n.patch_index].trail[n.patch_dist]
  := Patches[n.patch_index].trail[n.patch_dist] + n.motiv; //Trail markieren
  if (Form1.FeedBackCheckBox.checked = true)
  and (Patches[n.patch_index].trail[n.patch_dist] > c_max*n.motiv) then
   Patches[n.patch_index].trail[n.patch_dist] := c_max*n.motiv;
  end //if not (Dinstance...
 else    //ja, Nest erreicht:
  begin
```

```
   n.x := Nest.x;                    //Ameise genau ins Nest setzen,..
   n.y := Nest.y;                    //..um Komplikationen zu vermeiden.
   Patches[n.patch_index].trail[n.patch_dist]
   := Patches[n.patch_index].trail[n.patch_dist] + n.motiv; //letzte Markierung
   if (Form1.FeedBackCheckBox.checked = true)
   and (Patches[n.patch_index].trail[n.patch_dist] > c_max*n.motiv) then
     Patches[n.patch_index].trail[n.patch_dist] := c_max*n.motiv;
   n.status := 4;                    //Entladen
   n.counter := unloading_time;      //Zähler einstellen
   end; //else...
 end; //Die Prozedur bewegt eine Ameise zum Nest, lässt sie dabei markieren..
     //..und ändert ihren Status in 'unload', wenn sie das Nest erreicht hat.

Procedure Unload(var u : TAnt);//4
 begin
 if u.counter = 0 then   //nötige Zeit verstrichen?
   begin
   intake   := intake + u.load; //Ladung abgeben
   u.load   := 0;               //eigene Ladung wieder auf 0 setzen
   u.patch_index := 0;          //kein Memory-Effect
   ChooseTrail(u);              //wieder auf den Trail?
   end; //if u.counter...
 end;   //Ist die nötige Zeit vergangen, entläd diese Prozedur die Ameise und..
      //..prüft, ob sie zum Trail zurückkehrt oder im Nest bleibt.

Procedure Patchbound(var p : TAnt);//5
 begin
 if not (Distance(p.x,p.y,Patches[p.patch_index].x,Patches[p.patch_index].y)
   < Patches[p.patch_index].radius) then     //schon angekommen?
   begin //nein, dann:
   if Form1.BidirectionalCheckBox.checked = true then //bidir. trail laying?
     begin
     PatchDist(p.patch_index,p);
     p.motiv := 1; //Patches[p.patch_index].trail[p.patch_dist];          //_______________
     Patches[p.patch_index].trail[p.patch_dist]
     := Patches[p.patch_index].trail[p.patch_dist] + p.motiv;
     if (Form1.FeedBackCheckBox.checked = true)
     and (Patches[p.patch_index].trail[p.patch_dist] > c_max*p.motiv) then
       Patches[p.patch_index].trail[p.patch_dist] := c_max*p.motiv;
     end;
   Go(p,Patches[p.patch_index].x,Patches[p.patch_index].y); //zum Futter laufen
   end //if not...
 else //angekommen, dann:
   begin
   p.status := 2;                    //Patch erreicht, beladen beginnen
   p.counter := loading_time;        //Zähler eingestellt
   inc(Patches[p.patch_index].ants); //mehr Ameisen am Patch
   end; //else...
 end; //Eine Ameise wird den Trail entlang Richtung Patch bewegt, hat sie..
     //..diesen erreicht, so beginnt sie mit dem beladen.

//allgemeine Prozesse_____________________________________________________

Procedure Go(var g : TAnt; dest_x,dest_y : double); //auf einen Punkt zu laufen
 begin
 g.speed := SPEED;
 if not (sqrt(sqr(g.x-dest_x) + sqr(g.y-dest_y)) = 0) then //div 0 vermeiden
   begin
   g.x := g.x-(g.speed*(g.x-dest_x))/sqrt(sqr(g.x-dest_x)+sqr(g.y-dest_y));
   g.y := g.y-(g.speed*(g.y-dest_y))/sqrt(sqr(g.x-dest_x)+sqr(g.y-dest_y));
   end
 else
```

```
   if g.x = dest_x then
     if g.y < dest_y then g.y := g.y + g.speed
     else                 g.y := g.y - g.speed
   else
     if g.x < dest_x then g.x := g.x + g.speed
     else                 g.x := g.x - g.speed;
  move_sum := move_sum+distance(g.x,g.y,dest_x,dest_y);
  end; //Ameise, Zielpunkt -> Ameise mit neuen Koordinaten

Procedure PatchDist(patch : word; var p : TAnt); //Enferung zum Patch berechnen
  begin
  p.patch_dist := trunc((Distance(p.x,p.y,
  Patches[patch].x,Patches[patch].y)+1)/SPEED);
  end; //Nummer des Patches, Ameise -> Distanz zum Patch in SCHRITTEN (ent-..
       //..spricht Feld in dem Array, in dem die Pheromkonzentrationen gespei-..
       //..werden).

Procedure ChooseTrail(var c : TAnt); //Trail auswählen
  var
  i,j     : word; //2 Zählvariablen für 2 Schleifen über alle Patches
  ran,p,sum,conc : double; //Zwischenwerte
  term    : array[1..MAX_PATCHES] of double;
  begin
  sum := 0;            //Summe der Wahrscheinlichkeiten
  p := 0;             //Summenvariable für die relativen Wahrscheinlichkeiten
  conc := 0;           //Abfragevariable, wird größer 0 wenn Spur vorhanden
  for i:=1 to n_patches do //1.Schleife, Zwischenwerte berechnen
    begin
    PatchDist(i,c);                    //aktuelle Entfernung
    if Patches[i].trail[c.patch_dist] > 0 then //sonst hat auch ein leerer..
      term[i] := Power((k + Patches[i].trail[c.patch_dist]),n)//..Trail p > 0
    else term[i] := 0;
    sum := sum + term[i];              //Summe berechnen
    conc := conc + Patches[i].trail[c.patch_dist];
    end;  //for i:=...
  if (conc > 0) and (f_scouts < random) then  //wenn alle Trails stark sind...
    begin
    ran := random;                     //eine Zufallszahl ziehen
    j   := 0;                         //Zählvariable zurücksetzen
    while j < n_patches do             //Erwählen eines Trails
      begin
      j := j+1;                       //In dieser Schleife werden die..
      PatchDist(j,c);                 //..relativen Wahrsch. der einzel-..
      p := p + term[j]/sum;           //..nen Patches/Trail aufaddiert..
      if p > ran then                 //..und in jedem Schritt mit einer..
        begin                         //..Zufallszahl(ran) vergleicht.
        c.patch_index := j;
        c.status := 5;
        j := n_patches;
        end;   //if p...
      end; //for j:=...
    end //if (sum...
  else c.status := 1;                  //ansonsten geht die Ameise Scouten
  end;  //Zunächst wird geprüftm ob eine Ameise überhaupt irgend einem Trail..
       //..folgen will. Wenn ja, wird sie mit Hilfe der relativen Wahrschein-..
       //..lichkeiten einem bestimmten Trail zugeordnet.

Procedure Evaporation(var trail : TTrail); //Trail verdunsten
  var j : word;
  begin
  for j:=1 to TRAIL_STEPS do
    begin
```

```
    trail[j] := trail[j]*(1-trail_evap);
    if trail[j] < 0.1 then trail[j] := 0;
    end;
  end; //Alle Schritte auf dem Trail werden durchgegangen und die neuen Kon-..
       //..zentrationen werden berechnet.

Procedure DeletePatch(index : word);
  var patch, ant : word;
  begin
  for patch:=index to (MAX_PATCHES-1) do
    begin
    Patches[patch] := Patches[patch+1];
    end;
  for ant:=1 to n_ants do
    begin
    if Ants[ant].patch_index = index then
      begin
      Ants[ant].status := 1;
      Ants[ant].patch_index  := 0;
      end;
    if Ants[ant].patch_index > index then
      dec(Ants[ant].patch_index);
    end;
  dec(n_patches);
  end;

Procedure ReadCapWord(caption : string; var value : word); //parameter lesen
  var code : integer; //Fehlercode für "val"
  begin
  val(caption,value,code);
  end; //Ein Editierfenstereintrag wird als 'word'-Variable eingelesen

Procedure ReadCapDouble(caption : string; var value : double); //parameter lesen
  var code : integer;  //Fehlercode for "val"
  begin
  val(caption,value,code);
  end; //Ein Editierfenstereintrag wird als 'double'-Variable eingelesen

Procedure CheckDelay(time : word); //neuer Patch erscheint?
  begin
  if (Form1.PatchesEdit.text = '2') and (time = patch_delay) then
    begin
    inc(n_patches);
    SetPatch(2,Patches[n_patches]);
    end;
  end;

Procedure PGraph(p_max,p_inc : double);
  var c,p : double;  //Konzentration,Wahrscheinlichkeit
  begin
  Form1.Series9.clear;
  c := 0;
  p := 0;
  if not (p_inc = 0) then
    begin
    while p < p_max*0.99 do
      begin
      p := P_Follow(p_max,p_inc,c);
      Form1.Series9.AddXY(c,p,'',clred);
      c := c + p_inc*0.1;
      end;
    end;
```

```
  end;

//Unterprozeduren_____________________________________________________________

Procedure Search(var s : TAnt); //Fläche absuchen
  begin
  s.speed := SPEED;
  s.alpha := s.alpha + 2*TWIST*pi*(random-0.5);  //Richtung
  s.x := s.x+s.speed*cos(s.alpha);              //neue Koordinaten
  s.y := s.y+s.speed*sin(s.alpha);
  if s.x > MAX_X then s.x := MAX_X;             //am Rand?
  if s.x < 0     then s.x := 0;
  if s.y > MAX_Y then s.y := MAX_Y;
  if s.y < 0     then s.y := 0;
  end; //eine Ameise bewegt sich im Raum und ändert zufällig die Richtung. Bei..
       //..Überschreiten der Begrenzung wird auf den Grenzwert zurückgesetzt.

Procedure HitTrail(index : word; var h : TAnt; var t : TNext_trail);
  begin
  if not(Lot_denom(Nest.x,Nest.y,Patches[index].x,Patches[index].y) = 0) then
    begin  //Nenner=0?
    t.x := Lot_x(Nest.x,Nest.y,Patches[index].x,Patches[index].y,h.x,h.y);
    t.y := Lot_y(Nest.x,Nest.y,Patches[index].x,Patches[index].y,h.x,h.y);
    end //Die Koordinaten des Oli'schen Punktes werden berechnet..
  else
    begin
    t.x := h.x; //falls Ameise genau auf der Linie und Nenner = 0
    t.y := h.y;
    end;
  end; //Berechnet den nächstliegenden Punkt/das Lot auf einem Trail für eine..
       //..Ameise.

Procedure EvalCon(index : word; var e : TAnt; var t : TNext_trail);//Dist.und c?
  begin
  PatchDist(index,e);  //Entferung zum Patch?
  if (Distance(e.x,e.y,Patches[index].x,Patches[index].y)//hinter dem Nest?
    < Distance(Nest.x,Nest.y,Patches[index].x,Patches[index].y)
    - Nest.radius)
    and (Distance(e.x,e.y,Nest.x,Nest.y)               //hinter dem Patch?
    < Distance(e.x,e.y,Patches[index].x,Patches[index].y))
    and (Distance(e.x,e.y,t.x,t.y) < e.radius)         //Entfernung zum Trail?
    and (P_Follow(P_MAX_SCOUT,P_INC_SCOUT,Patches[index].trail[e.patch_dist])
    > random) then                                     //genug Pheromon?
    begin
    e.patch_index := index;  //Patch auswählen
    e.status := 5;           //zum Patch bewegen
    end; //end if
  end; //Diese Prozedur prüft, ob sich eine Ameise auf der Gerade zwischen dem..
       //..NEst und einem Patch befindet. Wenn ja, und wenn die Wahrschein-..
       //..lichkeit, diesem Trail zu folgen, hoch ist, folgt sie dem Trail.

//____________________________________________________________________________

Procedure Graph(time : word); //wir malen...
  var patch, ant : word;
  begin
  if time = 1 then int_mean_old := 0;
  Form1.series1.clear;                             //alte Ameisen löschen
  for ant:=1 to N_ANTS do                          //Ameisen einzeichnen
    begin
    Form1.series1.AddBubble(Ants[ant].x,Ants[ant].y,Ants[ant].radius,",clred);
    end;
```

```
  for patch:=1 to N_PATCHES do                    //Patches einzeichnen
    begin
    if Patches[patch].load > 0 then
      Form1.series2.AddBubble(Patches[patch].x,Patches[patch].y,
      Patches[patch].radius,'P',clgreen)
    else Form1.series2.clear;
    end;
  Form1.Series3.AddBubble(Nest.x,Nest.y,Nest.radius,'',clblack);//Nest -"-
  Form1.Series4.AddXY(time,(intake/(n_ants*time+int_mean_old)),'',clred);
  int_mean_old := intake/(n_ants*time);
  Form1.Series5.AddXY(time,Patches[1].ants,'',clred);     //Ameisen Patch 1
  Form1.Series7.AddXY(time,Patches[1].trail[1],'',clred); //Trail 1
  if n_patches > 1 then
    begin
    Form1.Series6.AddXY(time,Patches[2].ants,'',clgreen); //Ameisen Patch 2
    Form1.Series8.AddXY(time,Patches[2].trail[1],'',clgreen);//Trail 2
    end;
  end; //Diese Prozedur zeichnet die aktuellen Positionen aller Objekte in die..
      //..Landschaft ein und aktualisiert alle Graphiken.

//_____________________________________________________________________________

Procedure Save(); //Ergebnisse speichern
  var patch : word;
  begin
  assignfile(output,'output.txt');                //Datei zuordnen
  append(output);                                 //Datei öffnen
  writeln(output,'parameters:');                  //vorgegebene Parameter
  writeln(output,'MAX_TIME:',MAX_TIME);
  writeln(output,'n_patches:',n_patches);
  writeln(output,'PATCH_RADIUS:',PATCH_RADIUS);
  writeln(output,'LOADING_TIME:',LOADING_TIME);
  writeln(output,'TRAIL_EVAP:',TRAIL_EVAP);
  writeln(output,'NEST_RADIUS:',NEST_RADIUS);
  writeln(output,'UNLOADING_TIME:',UNLOADING_TIME);
  writeln(output,'N_ANTS:',N_ANTS);
  writeln(output,'SPEED:',SPEED);
  writeln(output,'TWIST:',TWIST);
  writeln(output,'ANT_RADIUS:',ANT_RADIUS);
  writeln(output,'C_MARK:',C_MARK);
  writeln(output,'P_MAX_SCOUT:',P_MAX_SCOUT);
  writeln(output,'P_INC_SCOUT:',P_INC_SCOUT);
  writeln(output,'fraction of scouts:',f_scouts);
  writeln(output,'flow:',flow);
  writeln(output,'n:',n);
  writeln(output,'k:',k);
  writeln(output,'___________');
  writeln(output,'results:');                     //Ergebnisse
  writeln(output,'');
  writeln(output,'intake',intake:7:0);
  writeln(output,'sum of moves',move_sum:7:0);
  writeln(output,'');
  for patch:=1 to n_patches do
    begin
    writeln(output,'patch:',patch);
    writeln(output,'distance:',
      Distance(Patches[patch].x,Patches[patch].y,Nest.x,Nest.y):3:1);
    if n_patches = 2 then writeln('time delay:',patch_delay);
    writeln(output,'ants:',Patches[patch].ants);
    writeln(output,'ants:',Patches[patch].trail[1]:5:0);
    writeln(output,'');
    end;
```

```
  closefile(output);                    //Textfile schließen
  end;

//_____________________________________________________________________________

procedure TForm1.p_Max_ScoutEditChange(Sender: TObject);
  var max,inc : double;
  begin
  ReadCapDouble(Form1.p_Max_ScoutEdit.text,max);
  if max > 1 then
    begin
    Form1.p_Max_ScoutEdit.text := '1';
    ReadCapDouble(Form1.p_Max_ScoutEdit.text,max);
    end;
  ReadCapDouble(Form1.p_Inc_ScoutEdit.text,inc);
  PGraph(max,inc);
  end;

procedure TForm1.p_Inc_ScoutEditChange(Sender: TObject);
  var max,inc : double;
  begin
  ReadCapDouble(Form1.p_Max_ScoutEdit.text,max);
  ReadCapDouble(Form1.p_Inc_ScoutEdit.text,inc);
  PGraph(max,inc);
  end;

//_____________________________________________________________________________


end.
```