Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Praktikumsbericht

Parallelisierung einer Populationsdynamik-Simulation

Eingereicht von: Nils Petersen (6237766)
Gärtnerstraße 115
20253 Hamburg

am: 31.10.2012

Studiengang: Master of Science / Bioinformatics

Advisor: Nathanael Hübbe, Julian Kunkel, Petra Nerge

Kurzfassung

Hier wird die Parallelisierung der Simulation einer Organismenpopulation vorgestellt. Simuliert wurden bewegliche Organismen in einer zweidimensionalen Gitterwelt, welche sich vermehren und nach Mendel'schen Regeln Allele vererben. Um das Programm zu parallelisieren, wurde die Gitterwelt in gleich große Sektionen eingeteilt. Die Implementierung fand in C statt und die Parallelisierung wurde mit MPI-Funktionen realisiert. Aufgrund einer zu teuren Funktion in der Simulationsimplementation wurde bei der Parallelisierung eine superlinearere Laufzeitbeschleunigung erreicht.

Inhaltsverzeichnis

Kapitel 1

1 Einleitung

1.1 Problemstellung und Idee des Projekts

Es sollte die Populationsdynamik einer Gruppe beweglicher Organismen simuliert werden, um diese Simulation anschließend zu parallelisieren.

1.2 Mendel'sche Vererbung

Die klassische Genetik nach Mendel beschreibt die Vererbung von dominanten und rezessiven Allelen und die Ausprägung dieser im Phänotypen. Höhere Organismen besitzen einen diploiden Chromosomensatz, wobei auf jedem homologen Chromosom jeweils ein Allel des entsprechenden Gens liegt. Welches Allel ausgeprägt wird, hängt davon ab, ob es sich um dominante oder rezessive Allele handelt. Dominante Gene überschatten dabei immer rezessive. Während der Meiose wird der Chromosomensatz halbiert und jeweils eine Hälfte an die Nachkommen weitervererbt [1].

1.3 Vererbung Rezessiver Gene mit erhöhter Fitness

Nach den Mendel'schen Gesetzen sollte bei einer Gleichverteilung der Allele in der Elterngeneration und einer Vererbung ohne Selektion oder Genkopplung eine Population mit 75% Phänotypen des dominanten und 25% Phänotypen des rezessiven Typs entstehen. Diese Verteilung kann jedoch

anders aussehen, wenn einer der beiden Phänotypen einen Fitnessvorteil mit sich bringt.

1.4 Modellierung in einer Simulation der Populationsdynamik von beweglichen Organismen mit Mendel'scher Vererbung

Die Population wurde in einer zweidimensionalen Gitterwelt, in welcher sich an den verschiedenen Koordinatenpunkten mehrere Organismen befinden, modelliert. Die Organismen sind dabei diskrete Objekte mit Eigenschaften wie Alter, Geschlecht und Genotyp. In jedem Zug bewegt sich jeder Organismus auf eines der acht Nachbarfelder (Abbildung 1.1). Weibliche Organismen wählen zudem - sofern vorhanden - einen Partner aus dem gleichen Feld und bringen nach einer festgelegten Rundenzahl Nachwuchs zur Welt. Für jeden Nachwuchsorganismus werden die Chromosomen der Eltern zufällig rekombiniert, also nach Mendel'schen Regeln vererbt. Desweiteren wird für jeden Organismus in jeder Runde geprüft, ob er überlebt. Das Überleben der Organismen ist in Abhängigkeit vom Alter und der lokalen Populationsdichte mit Sigmoidalfunktionen modelliert.

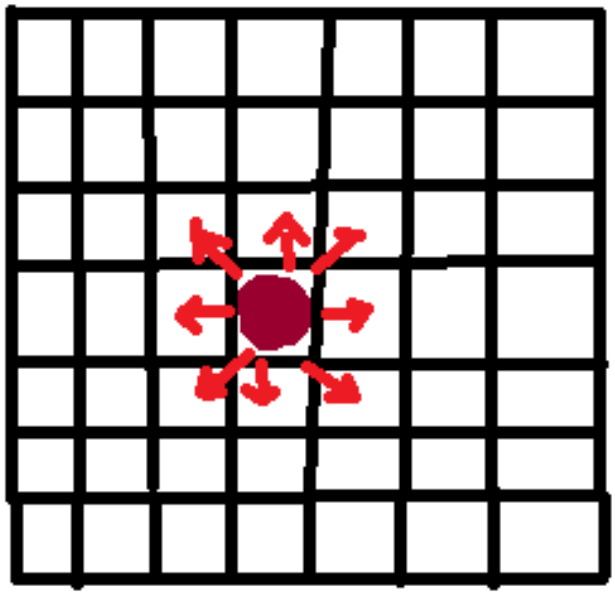

Abbildung 1.1: Der Organismus (hier als roter Punkt dargestellt) bewegt sich in jeder Runde zufällig auf einer der acht Nachbarfelder in der ihn umgebenden Gitterwelt.

Kapitel 2

2 Implementierung der Simulation

2.1 Progammiersprache

Die Simulation wurde in ANSI-C geschrieben und mit dem GNU C-Compiler gcc bzw. mpicc kompiliert.

2.2 Die modellierte Welt

Die Welt, in welcher sich die Organismen befinden, ist ein quadratisches Gitter, an deren Koordinatenpunkten sich eine Menge von Organismen befinden kann. Die Kanten dieser Welt sind miteinander verbunden, so dass eine torusförmige Karte entsteht. Sie ist als übergeordnete Datenstruktur implementiert und ihr Kern besteht aus zwei Listen mit den Organismen und einer Matrix mit Information über die Populationsdichte. Wie später näher erläutert, ist die Ordnung der Organismen in Listen ungünstig für die Partnersuche implementiert und führt zu einer sehr uneffizienten Laufzeit der Simulation. Darüber hinaus sind in der Welt-Struktur Informationen wie die Länge der Achsen und das aktuelle Jahr enthalten. Global sind außerdem (in Form von Makros) einige Parameter wie die Kapazität eines Koordinatenpunktes definiert.

2.3 Der Organismus

Die in Listen angeordneten Organismen sind ebenfalls als Strukturen implementiert. Diese enthalten die Information über das Alter, die Position auf der Karte sowie ob ein weiblicher Organismus trägt und - falls ja - wie lange. Darüber hinaus sind die Chromosomen in Form von Integer-Arrays gespeichert. Jede Position in diesem Array ist dabei ein Allel. Ist dieses Allel dominant, steht an dieser Position eine 1 - sonst eine 0. Darüber hinaus sind für tragende Organismen die Genotypen des Partners gespeichert, da diese bei der Geburt des Nachwuchs benötigt werden. Wird ein neuer Organismus geboren, wird entsprechend Speicher geholt, stirbt er, wird er aus der Liste entfernt und der Speicher wird befreit.

2.4 Die initiale Bevölkerung

In die Welt wird zu Beginn der Simulation eine initiale Bevölkerung aus einer festgelegten Zahl Organismen eingesetzt. Die Starpositionen werden mittels ganzer Zahlen über die Funktion rand() aus der C-Standardbibliothek vergeben. Mit derselben Funktion werden die Allele sowie das Alter jedes Organismus initialisiert. Bei der Vergabe der Allele werden die dominanten Allele mit einer Wahrscheinlichkeit von 0,7 und die rezessiven Allele mit einer Wahrscheinlichkeit von 0,3 gewählt.

2.5 Iterativer Ablauf

Die Simulation erfolgt in Runden, in denen immer dieselbe Abfolge von Funktionen für jeden Organismus aufgerufen werden.

2.5.1 Bewegung, Sterben etc.

Um die zufälligen Ereignisse in der modellierten Umwelt zu simulieren, wurde die Funktion rand() der C-Standartbibliothek verwendet um Integer Zufallszahlen und Zufallszahlen zwischen 0 und 1 zu generieren. Zu Beginn jeder Runde wird für jeden Organismus geprüft, ob er diese Runde überlebt. Die Wahrscheinlichkeit zu sterben wurde abhängig von

der Populationsdichte als Sigmoidfunktion und abhängig vom Alter als Exponentialfunktion modelliert. Form und Position der Kurven können über Parameter eingestellt werden und auf diese Weise auch die durchschnittliche Bevölkerungskapazität der einzelnen Felder festgelegt werden. Außerdem gab es zwei Gene, die, wenn beide Chromosomen das rezessive Allel trugen, für eine Änderung der Sterblichkeit sorgten. Eines führte zu einer höheren Lebenserwartung bezüglich des Höchstalters, das andere verbesserte die Überlebenschancen bei hoher Bevölkerungsdichte. Überlebt der Organismus bewegt er sich auf eines der Nachbarfelder, wo er einen Partner suchen oder Nachwuchs bekommen kann. Die Anzahl der Nachkommen wird ebenfalls zufällig bestimmt, in der in den Ergebnissen ausgewählten Implementierung wurden zwischen 0 und 2 Nachkommen geboren. Auch hier gab es rezessive Phänotypen mit einem Vorteil. Diese konnten bis zu vier Nachkommen haben.

2.5.2 Anmerkung zur Partnersuche

In jeder Runde wird für jeden weiblichen Organismus geprüft, ob er geschlechtsreif und nicht bereits befruchtet ist. Ist dies der Fall, wird die gesamte Liste der männlichen Organismen nach möglichen Partnern durchsucht und aus diesen einer ausgewählt und dessen Genotyp beim weiblichen Organismus für den später folgenden Nachwuchs gespeichert. Da in der Implementierung nur Partner aus dem gleichen Feld relevant sind, aber die gesamte Liste durchsucht wird, ist diese Methode sehr ineffizient und führt zu sehr hohen Laufzeiten - und wie an späterer Stelle diskutiert zu einer hohen Beschleunigungsrate bei der Parallelisierung.

2.6 Visualisierung

Nach einer festgelegten Zahl von Runden wird ein Bild der Karte erstellt. Je nach gewünschter Auflösung wird eine Anzahl von Feldern zusammen als ein Pixel dargestellt. Für diese Felder werden die Koordinaten, die Gesamtpopulation sowie das Vorkommen jedes möglichen Phänotyps in einer Struktur gespeichert. Diese Strukturen werden gesammelt und an eine Visulisierungsfunktion übergeben. Diese nutzt Funktionen aus dem Modul bmpfile [2], um eine Bilddatei im Bitmap-Format zu erstellen. Der Funktion wird auch ein Wert übergeben, mit wievielen Pixel jeder Punkt in

der Visualisierung dargestellt werden soll. Das entstandene Bild ist in vier Teile unterteilt, welche jeweils die gesamte Welt zeigen. Das Vorkommen der Organismen ist relativ zur möglichen Gesamtzahl in Farben von Weiß (kein Vorkommen) über Grün nach Blau dargestellt und ermöglicht so eine intuitive Erfassung der Populationsdynamik. Im Bild oben rechts ist die gesamte Population dargestellt. Oben links sind Vorkommen eines rezessiven Phänotyps ohne weitere Auswirkungen gezeigt, während das Vorkommen des im unten links dargestellten Phänotyps einen Überlebensvorteil bezüglich der Populationsdichte haben und die unten rechts eine bessere Überlebenswahrscheinlichkeit im hohen Alter besitzen. Die Bitmap-Dateien wurden nach der Ausführung der Simulation mit dem Programm ffmpeg [3] zu einer Videosequenz zusammengefügt.

2.7 Optionen

Sowohl der seriellen als auch der parallelen Version des Simulationsprogramms können Rundenzahl, Weltgröße, initiale Populationsgröße, Auflösung und Pixelgröße des Bildes sowie das Verzeichnis zum Speichern der Bilddateien als Kommandozeilenparameter mitgegeben werden. Die Optionen -h und -? geben eine Beschreibung der möglichen Optionen.

Kapitel 3

3 Parallelisierung

3.1 Bibliothek

Für die Parallelisierung wurde die MPICH Implementierung der MPI-Bibliothek genutzt [**?**].

3.2 Aufteilen der Welt

Zur Parallelisierung der Simulation wurde die quadratische Karte in kleinere quadaratische Karten zerteilt. Da die Ahnzahl auch immer quadratisch zu Zerteilung entlang der Achsen ist, ist die Zahl der verwendeten Prozesse auf Quadratzahlen beschränkt. Übertritt ein Organismus die Grenzen seiner Welt wird er zu dem entsprechenden Prozess geschickt, auf welchem die benachbarte Karte simuliert wird (Abbildung 3.1). Dafür wird zunächst die Anzahl der zu erwartenden Organismen als Datentyp MPI_INT mit den Funktionen MPI_Isend(..) und MPI_Irecv(..) übermittelt. Mit der Funktion MPI_Wait(..) wird gewartet bis diese Information übermittelt ist, so dass genügend Speicher für die ankommenden Organismen bereit gestellt wird. Zum Versenden der Organismen wird bei Progammstart die Organismen-Datenstruktur mittels MPI_Type_create_struct(..) und anschließendem MPI_Type_commit(..) versendbar gemacht. In jeder Iteration werden dann die grenzübertretenden Organismen an den entsprechenden Prozess versendet. Nach der Ankunft aller Organismen (MPI_Wait(..)) kann anschließend die nächste Runde beginnen.

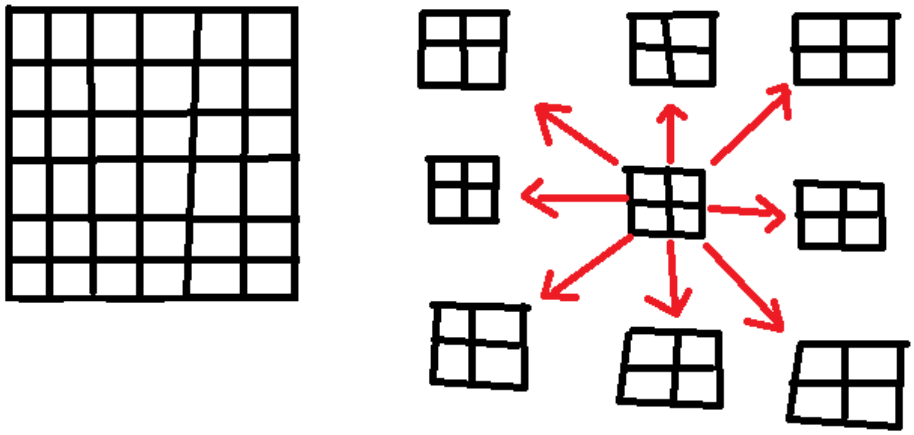

Abbildung 3.1: Die quadratische Welt wird in kleinere quadratische Teilwelten zerlegt. Organismen, welche die Grenzen der sie umschließenden Teilwelt übertreten, werden an den entsprechenden Prozess gesendet, an der sich der benachbarte Weltteil befindet.

3.3 Visualisierung der zerteilten Welt

Um die zerlegte Karte zu visualisieren, werden die bereits beschriebenen Populationsinformationen in jedem Prozess in Strukturen gesammelt. Die Information wird anschließend von jedem Prozess an einen Visualisierungsprozess gesendet, welcher die Informationen für alle Prozesse - und damit für alle Teilwelten - sammelt und daraus wie auch in der seriellen Version eine Bilddatei erzeugt (Abbildung 3.2). Die verwendete Struktur wird mit denselben Funktionen initialisiert und versendet wie bereits die Organismen. Der Visualisierungsprozess wartet mit MPI_Wait(..) auf die Ankunft der Information aller Prozesse bevor die Datei erzeugt wird. Die Simulationsprozesse müssen hingegen erst warten wenn sie das nächste Mal Information verschicken wollen und können erst dann weiter simulieren.

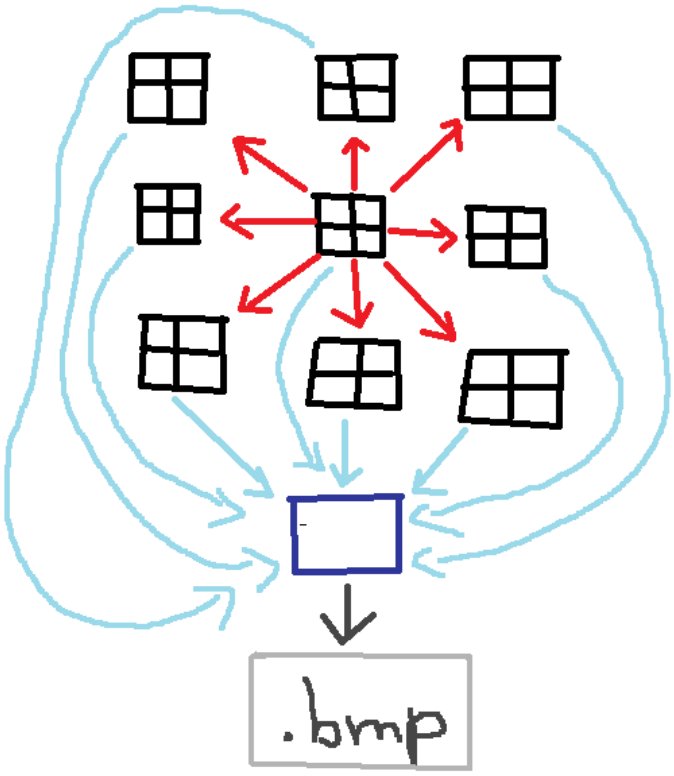

Abbildung 3.2: Die Information über die Populationsdichte und die Verteilung der Phänotypen wird in jedem Prozess, welcher einen Teil der Welt simuliert gesammelt und an einen Simulationsprozess versendet, welcher die Information zusammenfügt und eine Bilddatei im Bitmap-Format erzeugt.

Kapitel 4

4 Ergebnisse

4.1 Laufzeitbeschleunigung

Die Simulation einer 240*240 Felder großen Welt zeigt eine superlineare Beschleunigung von 1 bis 17 Prozesse. Allein mit fünf Prozessen wird eine 17.5-fache Beschleunigung erreicht (Abbildungen 4.1 und 4.2). Dieser enorme Geschwindigkeitsgewinn ist durch die ungünstige Implementierung der Partnersuche zu erklären. Da für jeden paarungsbereiten weiblichen Organismus jeweils die gesamte Liste der männlichen Organismen durchsucht wird, steigt die Zahl der Operationen quadratisch mit der Weltgröße. Durch die Aufteilung der Welt werden auch weniger Organismen betrachtet. Zählt man nun alle Prozesse zusammen, tut der Algorithmus also weniger, liefert aber das gleiche Ergebnis. Von 17 bis 101 Prozesse gibt es keinen Effizienzgewinn mehr. Dies ist dadurch zu erklären, dass bereits bei 17 Prozessen die Teilwelt recht klein (60*60 Felder) ist und ein Schritt schnell berechnet werden kann. Eine weitere Aufteilung bringt mehr Sendeoperationen mit sich, die bei 25 Prozessen dann auch bereits über einen Knoten hinaus geht, bei 65 bereits über zwei. Erhöht man die Kartengröße auf 480*480 Felder, erhält man auch mit höheren Prozesszahlen noch eine Beschleunigung (Abbildung 4.3).

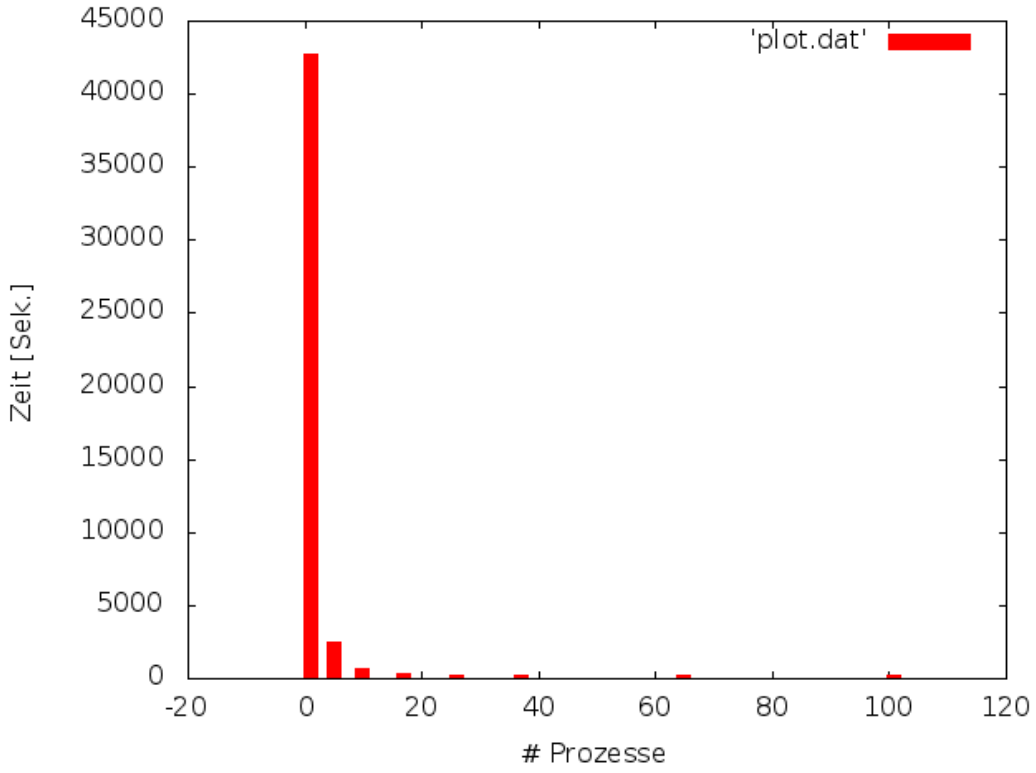

Abbildung 4.1: Bei einer 240*240 Felder großen Welt ist der Geschwindigkeitsgewinn zunächst enorm, nimmt aber bei größeren Prozesszahlen ab.

4.2 Tracing

4.2.1 Kommunikation zwischen den simulierenden Prozessen

Zu Beginn der Simulation ist die simulierte Welt noch wenig bevölkert. Das führt dazu, dass der Rechenaufwand in einer Iteration gering ist und die Zeit in der die Prozesse aufeinander warten relativ dazu wesentlich größer ist (Abbildung 4.4). Mit zunehmender Bevölkerung nimmt auch der Rechenaufwand zu und die zum versenden und empfangen verwendete Zeit zwischen den Simulationsrunden ist relativ zur Simulationszeit nur noch sehr gering (Abbildung 4.4). Dies zeigt auch, dass bei einer stark zerteilten Welt, wie sie bei einer Simulation mit vielen Prozessen vorkommt, die Teilwelten so klein sind, dass durch weitere Zerteilungen kein großer Zeitgewinn mehr erreicht werden kann, da mehr Zeit zum Versenden und

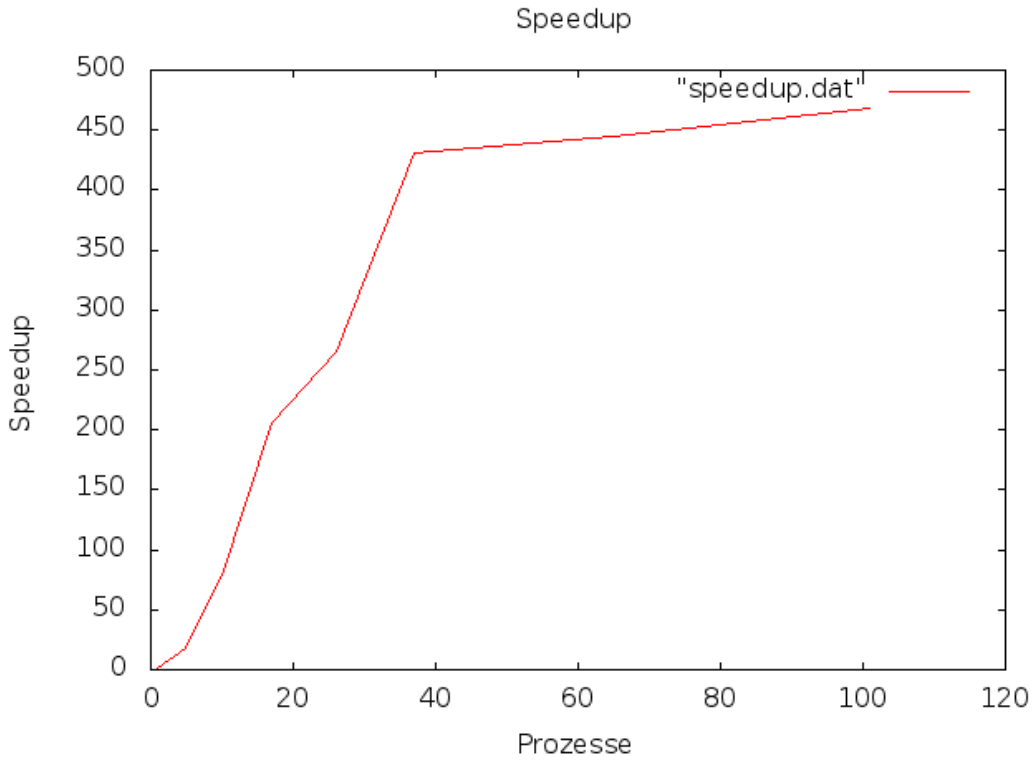

Abbildung 4.2: Die superlineare Laufzeitbeschleunigung der Simulation einer 240*240 Felder großen Welt ist durch die ungünstige Implementierung der Partnersuche zu erklären, welche bei kleineren Teilwelten wesentlich weniger Schritte arbeiten muss.

Warten verwendet wird.

4.2.2 Kommunikation zwischen Simulations- und Visualisierungsprozess

Auch hier ist es so, dass zunächst die benötigte Zeit für die Simulationsiterationen gering ist, so dass die Simulationsprozesse auf den Visualisierungsprozess warten müssen (Abbildung 4.4). Später ist es dann umgekehrt, der Visualisierungsprozess wartet auf die Simulationsprozesse (Abbildung 4.4).

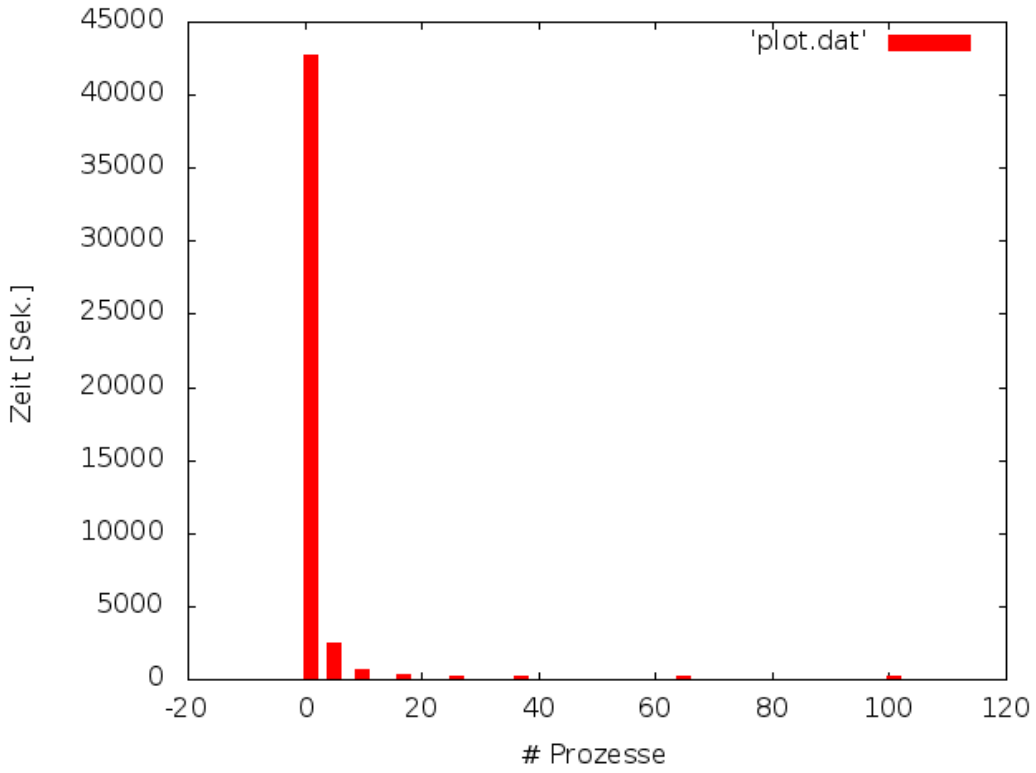

Abbildung 4.3: Bei der Simulation einer 480*480 Felder großen Welt ist auch noch bei der Erhöhung von 65 auf 101 Prozessen eine Geschwindigkeitszunahme zu beobachten.

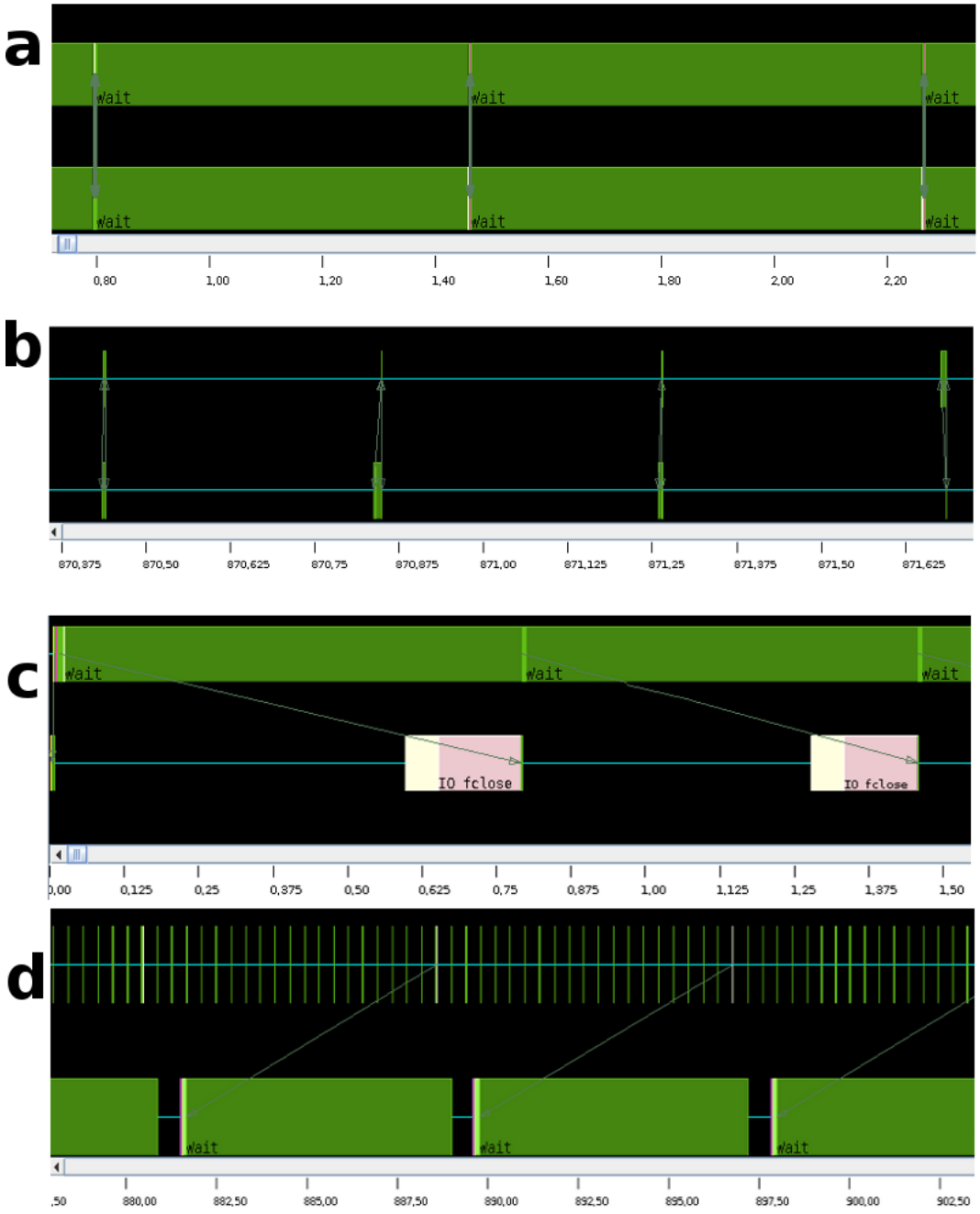

Abbildung 4.4: Gezeigt sind die Traces einer Simulation mit fünf Prozessen. Bilder wurden jeweils nach 20 Iterationen erzeugt. Die Pfeile stellen Sendeopertionen dar, die grünen Balken sind Wartezeit in der Funktion MPI_Wait(..). **a** Zwei Simulationsprozesse zu Beginn der Simulation **b** Zwei Simulationsprozesse gegen Ende der Simulation **c** Simulations- und Visualisierungsprozess zu Beginn und **d** Ende der Simulation.

Kapitel 5

5 Anmerkungen und Ausblick

Hier werden einige Schwächen der Implementierung und mögliche Verbesserungen diskutiert.

5.1 Die Partnersuche

Die Implementierung der Partnersuche ist sehr ineffizient und führt zu hoher Laufzeitbeschleunigung bei mehr verwendeten Prozessen. Eine einfache Möglichkeit, diese Methode zu ändern, wäre eine Matrix entsprechend der Weltgröße bzw. der Teilweltgröße anzulegen, in welcher Listen mit Zeigern auf die in diesem Feld befindlichen männlichen Organismen sind. Die Aktualisierung dieser Matrix bei Bewegung oder Sterben von Organismen hätte lineare Laufzeitkomplexität. Das gleiche gilt für die Partnersuche eines weiblichen Organismus, da die Zahl der Organismen pro Feld praktisch nach oben hin beschränkt ist.

5.2 Aufteilung der Weltkarte

Die Welt ist bei der Parallelisierung insofern ungünstig aufgeteilt, als dass nur eine Quadratzahl einer ganzen Zahl an Prozessoren für die Simulation verwendet werden und dazu immer ein Visualisierungsprozess verwendet wird. So können beispielsweise Knoten nicht vollständig genutzt werden und es gehen Ressourcen verloren. Alternative Aufteilungen der Weltkarte sind möglich. Darüber hinaus ist es sinnvoll, eine Mindestgröße für jede

Teilwelt festzulegen, da - wie im letzten Kapitel erläutert - eine Steigerung der verwendete Prozesse nicht zu einer schnelleren Laufzeit führt.

5.3 Visualierungsprozesse

Eine Aufteilung der Arbeit des Visualisierungsprozesses wäre leicht zu implementieren. Abwechselnd könnten mehrere Prozesse die Information erhalten und eine Bilddatei erzeugen. Dies wäre allerdings nur sinnvoll, wenn Bilder im Abstand weniger Simulationsiterationen gemacht werden sollen und wenn diese Iterationen schneller berechnet sind als die Bilddatei erstellt ist.

In der aktuellen Implementierung wartet der Visualisierungsprozess mit dem Erstellen des Bildobjektes bis alle Information eingetroffen ist. Stattdessen könnte bereits mit dem Erzeugen des Objektes und dem Eintragen von Pixelwerten begonnen werden, sobald überhaupt Information da ist. Dies brächte global betrachtet allerdings nur einen minimalen Laufzeitgewinn.

5.4 Anschließende Visualisierung

Eine mögliche Alternative zur aktuellen Implementierung ist, die Information für das Erzeugen der Bilder zunächst parallel in Dateien zu schreiben und anschließend die Dateien parallel von mehreren Prozessen in Bilddateien umzuwandeln. Dies würde einen Visualisierungsprozess überflüssig machen, macht aber auch nur in Fällen Sinn, in denen die Visualisierung laufzeitkritisch ist, also in Fällen, in denen die Bilderzeugung länger dauert als die dazwischenliegenden Simulationsiterationen.

5.5 Speicher für Organismen

Bisher wird für jeden einzelnen Organismus Speicher im Heap reserviert. Eine Datenstruktur, in der für mehrere Organismen Speicher geholt wird, wäre dankbar.

5.6 Versenden der zu erwartenden Organismenzahl

Statt in jeder Iteration zunächst die zu erwartende Zahl an Organismen zu schicken, wäre es möglich Speicher bereitzustellen, der in den meisten Fällen ausreichen sollte. Sollten doch einmal mehr Organismen geschickt werden, könnte mit der ersten versendeten Organismenmenge die Information geschickt werden, wieviele weitere Organismen noch auf diesem Weg versendet werden müssen.

Literaturverzeichnis

[1] Neil A. Campbell, Jane B. Reece, and Jürgen Markl. *Biologie*. Pearson Studium, München.

[2] libbmp The BMP library for read and write BMP file. `http://code.google.com/p/libbmp/`.

[3] ffmpeg. `http://ffmpeg.org/`.